A Testbed for Investigating C3I Security

Prithviraj Janardhan Kurapothula

Abstract*— Command, Control, Communication, and Intelligence (C3I) system is a system-of-system that integrates computing machines, sensors, and dynamic communication networks. C3I systems are increasingly used in critical civil and military operations for achieving information superiority, assurance, and operational efficacy. E-health is a prime application for C3I systems where patients' data must be collected, processed, and voraciously transmitted to doctors. However, both C3I systems and e-healthcare units face widespread cyber threats, which makes their security a critical concern. Any anomaly in healthcare operations can lead to the loss of precious lives. Therefore, there is a need for a holistic security mechanism to secure the healthcare operations based on C3I systems. For this purpose, we have designed a comprehensive research plan for enhancing the security level of C3I systems. The plan aims to develop a C3I healthcare testbed to perform vulnerability analysis, design and evaluate defence strategies for securing C3I systems.

Keywords – Command, Control, C3I, C4I, C4ISR, Body Sensor Network, Healthcare, Testbed

I. INTRODUCTION

A Command, Control Communication, and Intelligence (C3I) system is an integration of data-gathering sensors, computing machines, and a communication network to constitute the Information and Communication Technology (ICT) infrastructure for storing, analyzing, and transmitting information [15]. With the help of novel intelligence (i.e., Artificial Intelligence (AI) techniques), C3I systems offer unconventional assistance in data collecting, storing, processing, and transmitting among heterogeneous systems in a dynamic tactical environment [16]. These unique characteristics of C3I systems enable organizations to attain and sustain information superiority, operational efficacy, increased situational awareness, real-time decision support, swift communication, and enhanced collaboration among heterogeneous systems in their operations [17], [18]. Moreover, the C3I command unit ensures strict compliance with the organizational chain-of-command, which prevents a C3I asset from violating an informed course of action in organizational C3I operations. Therefore, C3I systems are increasingly used in sensitive civil and military tactical domains, such as search-and-rescue missions, emergency aids, transportation, fireguard, battlefield, airfield, and many other applications [15], [19], where timely data transmission and plan execution are of prime concern (Fig 1). Hospitals and healthcare units need to have robust monitoring and responsive systems to take care of patients efficiently. Patients need to be monitored under constant supervision and should timely be treated with proper medications. Therefore, e-healthcare systems are getting prevalent in healthcare domains where patients health record and data is systematically gathered and transmitted across different healthcare settings in real-time.

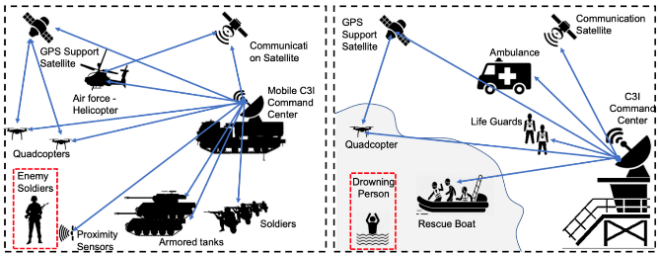


Fig. 1. C3I opertional scenarios for (a) military and (b)rescue domains [12]

Any anomaly in data collection, processing, transmission, and visualization would lead to inaccurate decisions by doctors (i.e., unfair treatment), which can eventually result in the loss of precious lives. Considering the sensitivity of the healthcare operations, we recommend the usage of C3I systems in the healthcare domain. As described above, C3I systems offer operational efficacy in data collection, information processing, and transmission. Moreover, C3I systems increase information assurance with a strict chain of command. Therefore, we assert that e-healthcare applications based on C3I systems can significantly improve healthcare operations.

Considering the sensitive nature of C3I application domains, a single security breach or malfunctioning of a C3I asset can lead to the collapse of the entire C3I operation. Historical events indicate that a cyberattack on critical military C3I instalments has detrimental consequences on national security. For example, Germans lost world war II mainly due to the use of compromised Enigma machines for communication [20]. Moreover, United States Ship (USS) stark was attacked by an Iraqi jet because the stark's radar "maliciously" confused the attacking Iraqi jet for a friendly Iranian jet, and the stark neither fired a weapon nor employed a countermeasure against the attack [21]. Similarly, in 1988, USS Vincennes "mistakenly" identified an Iranian passenger flight as a target and attacked it, which led to the loss of 290 innocent lives [21].

Moreover, an increased number of cyberattacks on C3I systems rings an alarming bell. For example, the USA Department of Defense faces around 250,000 attacks per year (*Defense Information Systems Agency, US DoD).* Similarly, around 1 million suspicious cyber incidents are daily logged by Britain's military systems (*Financial Times (2015)).* Furthermore, cyberattacks on the healthcare industry further deteriorate healthcare operations and degrade the confidentiality, integrity, and availability of patients' data. For example, U.S. Department of Health and Human Services Office for Civil Rights Breach Report revealed that cyberattacks exposed 38 million healthcare sector records in 2019 [22]. Therefore, there is a dire need to secure the C3I

healthcare operations by implementing defence strategies against malicious cyberattacks.

Considering the dire need to secure the C3I based e-health domain, we have designed a comprehensive research plan to increase the security level of C3I systems. As a part of the research project, we develop a small-scaled healthcare testbed based on C3I infrastructure in this report. For this purpose, we use biomedical sensors to collect patients' data and micro-controllers to process and transmit the health information to corresponding doctors. As a part of the C3I command centre, a healthcare dashboard is also designed to analyse the patients' health conditions in real-time. As a part of the project, we intend to perform vulnerability analysis on the testbed and develop defence strategies to increase the protection level of C3I systems.

The rest of the report is structured as follows. Section 2 presents the aims and challenges of our research plan. Section 3 provides the background of C3I systems and e-health infrastructure; it also reports the existing literature related to building a small-scaled testbed, models used in security assessment, methods used in penetration testing, and security requirement for C3I healthcare system. Section 4 briefly describes the research methodology for our research plan. Section 5 delineates the phases involved in constructing the C3I healthcare testbed. Section 6 explains construction of the real-time security interface. Section 7 penetration testing methods. Section 8 demonstrates the results.

II. AIMS AND CHALLENGES

Since C3I systems are being operated in sensitive environments (e.g., healthcare, military operations), it is cumbersome and expensive to evaluate security mechanisms for such systems in real-time. It is also observed that researchers have conducted simulation studies for the security evaluation of C3I systems [23]. Having realized the need for an experimental security evaluation, we devise a comprehensive research plan to evaluate the security of C3I systems experimentally. We aim at developing a small-scaled C3I testbed in our lab under a controlled environment to test the security of C3I systems. The research plan is carried out in the following stages:

During part 1 of the project

1. Develop a real-time C3I healthcare testbed based on necessary body sensors, intelligent processing units, command centre, and communication protocols.

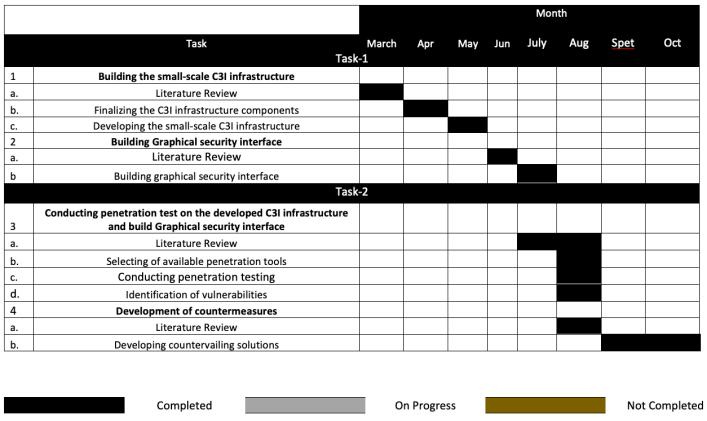
2. Develop a graphical security interface to assess security.

During part 2 of the project

3. Perform security tests to exploit the C3I testbed vulnerabilities. Also analyse the execution and impacts of cyber-attacks on the developed testbed infrastructure.

4. Develop and evaluate the defence strategies form security requirement learn from literature study and also based on the results from penetration testing conducted on the developed testbed to enhance the security of C3I systems.

Table 1: Progress Timeline



III. BACKGROUND AND RELATED WORK

In this section, we present brief details of C3I systems, and the components used to construct a C3I healthcare testbed such as body sensors, Arduino and raspberry pi. we explain the relevant literature corresponding to the small-scaled testbed construction.

A. C3I systems

C3I (Command, Control, Communications, and Intelligence) systems are information systems that provide prediction, prevention, and intervention for a situation. These systems have complex interactions compared to the "neural arc." that helps make decisions. Execution of C3I systems occurs by receiving data from the same or different systems or a combination of systems, then the situation estimation, option generation, decision making, and finally planning and instructing takes place [11]. The below figure shows C3I systems in context of military, and emergency rescue systems.

The following sections explain parts of C3I system.

1) Command and Control Unit

In any given scenario, the control unit is responsible for processing data, and the processing takes places in three stages; First, from the input data, and based on the C3I system it performs analysis, and estimation of situation takes place. Second, it creates options on the based on the given situation, and finally, using AI and machine learning technologies plan of action is generated. After the analysis, the control unit sends the data to command unit, and decision-makers in the command unit device a course of action and instruct control unit [11].

2) Communication unit

Communication in C3I systems takes place through rea-time or near real-time operations, mobility of systems, nematicity for clients and servers, and finally, working in a heterogeneous environment with different kinds of data. These features are achieved through modern technologies like High-speed LANs based on FDDI, Fast Ethernet, HIPPI or Fibre channel ATM standard which provide real-time isochronous commination [12].

3) Intelligence Unit

Intelligence unit is a part of control unit, which assists in decision making in the absence of a commander, or it assist the commander to make decision. This decision-making process is made possible using modern technologies like Machine Learning (ML), and Artificial Intelligence (AI).

B. Body Sensor Network

BSN is a form of wireless sensor network, It is also known as BAN, MBAN, or WBAN based on the context. BSN are used in any environment to measure physical, physiological, or biochemical parameters without the restriction on the activity the person is performing [9]. To illustrate, BSN can be used during activities like sports, training, or even daily base activities, and it is used to collect biomedical data like Electrocardiogram (ECG), glucose levels, temperature, pulse, blood pressure, and many other during the activity.
The characteristics of such a system are biocompatibility, long-term deployment, wearable, low powered wireless communication, autonomous sensing, and integrated systems. To achieve the above characteristics, and the reason for the existence of the system is because of the technical advancements in the field of micro and nano-electro-mechanical systems, and wireless networks technologies [10].

From the World Health Organisation stats, we can see that seven out of ten leading causes for deaths are non-communicable diseases like respiratory infections, neonatal conditions, heart diseases, and several other. BSN technology give us the characteristic and features for a useful and needed wide range of applications both in medical and not medical scenarios, and the applications are as shown in the figure above [15].

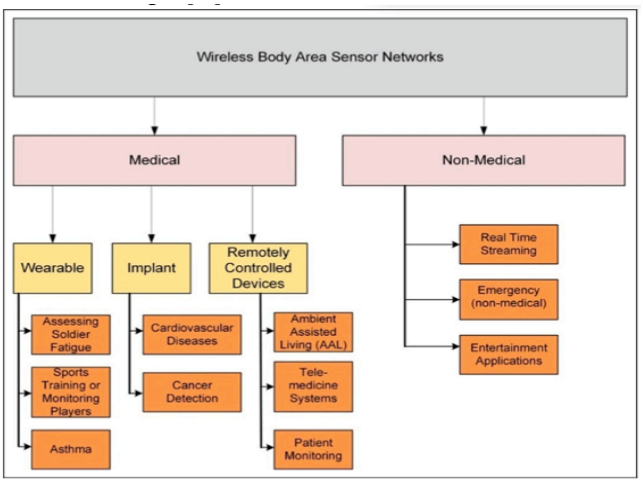


Fig. 2. Application of BSN [10]

C. Related Work

a. C3I System Infrastructure

The goal of the literature review was to comprehend and learn how to develop a small-scaled C3I infrastructure system for analysing C3I system vulnerabilities, executing and evaluating potential attacks, and implementing and evaluating defence tactics for safeguarding C3I systems.

Abrahamsson. et al. [1] research on a low-cost and energy-efficient cloud computing system showcased the power of single board computers. They built a Pi cluster consisting of 300 nodes. Many challenges of cost effectiveness and low power consumption was overcome using clusters of Raspberries but in terms of computational power, it was lacking. This first live system off the ground was set up at the Faculty of Computer Science in Free University of Bozen-Bolzano consisting of 300 Raspberry Pi. Configuring the hardware and the software was explained along with monitoring and maintaining the system. Being a green inexpensive testbed, it was a mobile and a robust data centre for operation in unfavourable environments.

Cox et al. [2] reported on the Iridis-pi cluster to demonstrate cheap and compact clusters in 2013. Due to its compact built it was readily portable. Other advantages included affordability, low-power consumption, and passive, ambient cooling, unlike conventional data-centre clusters that made it stand out. To tackle complex engineering and scientific challenges they inspired, and enabled students to apply high-performance computing and data handling by providing an affordable starting point, making the Iridisssss-pi cluster educationally applicable. Both its network performance and computational power proved to be of benchmark standards. The use of the ARM CPU also proved to be apt and in line with the new trend of low power, non-PC-compatible architectures in high performing clusters.

Fung Po Tso et al. [3] supported Cloud services that is usually housed by thousands of networking machines using Data Centres (DC) under a single roof. The major hurdle was the huge capital involved in setting up such an infrastructure causing a concern in evaluation of research and in its practical implementation. As the research on cloud computing often depends on a handful of software simulation and use of a testbed environment which are limited by only a few machines, they used cheap, power efficient single board computers and Raspberry Pi, to make the construction of the mini sized Cloud DCs cost effective. A scale model of a DC with clustered Raspberry Pi devices was demonstrated at the Glasgow Raspberry Pi Cloud (iCloud). It mimics the Cloud stack in every layer from resource virtualisation to network behaviour, to providing a full-featured Cloud Computing research for educational purposes.

One of the least-explored options in solving ML problems in edge computing, is the use of clusters of low-resource devices, like the Raspberry Pi. Though many hardware configurations have been looked at in the past, their performance remained unexplored in terms of ML tasks. The performance of a

Raspberry Pi micro-cluster was shown by Komninos et al. [4] coupled with industry-standard platforms like Spark for ML and Hadoop for distributed file storage. The end results looked promising for both local training and execution of ML based predictions when the latest Raspberry Pi 4 model was used. The major concern was to serve tourism applications using big data analysis in a distributed architecture with such computing resources.

Arunachalam & Andreasson [5] suggested a new smart AGRO IoT system, using the Raspberry-Arduino (RPA) powered smart mirrored and reconfigurable IoT facility for plant science research. It has capabilities of a high data collection mechanism with two software component nodes (i.e., Mirroring and Reconfiguration) running the IoT facility as one whole instance. While one minimised the downtime the other was towards leveraging the cores available for better resource utilisation. They suggested that this system could be deployed with ease at growth chambers, CNC farming test-bed setup, greenhouses and can also be scaled to large-farms with either using many standalone setup as heterogeneous instances of the facility, or as a single homogeneous instance by extending it as master-slave cluster configuration for communication. Using RaspberryPi-Arduino (RPA) ensures easy stability and scalability for continuous environment monitoring.

Bed et al. [6] and their research developed a Raspberry Pi based Smart Bed and Battery Driven Auto Warm Bolster, which was aimed at assisting better sleep quality by providing comfort based on individual need using the developed Smart Bed. The comfort settings were adjusted as needed using the Arduino and the fingerprint sensor which served the input-output controller. The whole set up can be assembled and customised as it comes as a smart module completely controlled by the Raspberry Pi. A warm bolster that was battery driven was attached as an accessory for an optimal level of warmth. It achieved a satisfaction of 83% when tested on 10 users.

The SWaT(Secure Water Treatment Testbed) testbed was studied by Mathur & Tippenhauer [7]. Being a modern Industrial Control System (ICS), the SWaT testbed is used to understand the impacts of physical and cyber-attacks on water treatment system. It also assesses the effectiveness of the detection and defence mechanism when under such an attack and also learn the effects of failure that cascading affect the other ICS. Made up of 6 stages, the water treatment process is controlled independently using local PLC. Alternative wired and wireless channels form the local fieldbus communications between the sensors, PCLs and actuators. The test bed indicates it's value in a realistic and active environment while conducting research, and also shows the design shortcomings that make the attack identification and detection difficult.

The development of a replicable and a cheap cyber-physical testbed mostly for research and training, was discussed by Rubio-Hernan el al. [8]. Its architecture is similar to the cyber-physical scenarios directed by the SCADA (Supervisory Control and Data Acquisition) technologies. It focused on the two protocols namely, Modbus and DNP3. They developed some unfavourable scenarios and detection strategies to correctly evaluate the testbed.

By extensive reading and understanding different testbed system, the following components were chosen for the functionality and resourcefulness. Arduino as Local Processing Unit for reading and transmitting sensors data and Raspberry pi as our Control Unit that is responsible for data processing, and hosting Command Unit Interface. Thus, in this project, we aim to develop a small-scaled C3I healthcare infrastructure for analysing the C3I systems vulnerabilities, execution and impacts of potential attack vectors.

b. Graphical Security Assessment

The purpose of the literature review here is to understand and build a graphical security assessment interface.

Hong et al. [25] proposed a framework with phases in this framework: data processing, security model generation, visualisation, security analysis, and model updates. They built an IoT Generator, a Security Model Generator, and a Security Evaluator. On the basis of the specified IoT network, the Security Model Generator builds the extended HARM; the Security Evaluator assesses the network's security using multiple security metrics. They examined the framework using three types of situations: smart home, health care monitoring, and environmental sensing. The expanded HARM calculated all possible attack pathways and selected security metrics during the security analysis. The study results allow the security decision maker to identify the most vulnerable network components, evaluate the efficiency of various defence methods, and choose the most effective network protection strategy.

Bayesian networks were utilised by Poolsappasit N, Dewri R and Ray [26] I to create a risk management framework called Bayesian Attack Graph (BAG), where administrators can use this framework to quantify the risk of network compromise at various levels. Threat analysis, risk assessment, loss expectancy, potential protections, and risk mitigation analysis are all part of BAG's security risk management. Administrators can use this component to do static and dynamic risk assessments, as well as risk mitigation analysis. BAG's security risk mitigation is modelled as a Multi-objective Optimisation Problem (MOOP) with a minimal level of optimisation complexity.

A study by Alhomidi M and Reed M [27] proposed a framework of risk assessment and optimisation to construct a graph using a genetic algorithm for generating attack paths. Six steps were described in the framework: attack graph generation, likelihood determination, loss estimation, risk determination, optimisation, and high-risk attack paths. Their proposed algorithm also identified the highest risk in order to construct a reduced attack tree.

c. Software Security Testing

The purpose of this literature research here is to have a better understanding of software penetration testing and to learn how to conduct software penetration tests.

According to the Arkin B, Stender S and McGraw G [28] penetration testing is the most generally utilised approach for evaluating software security, yet it is also the most misapplied method. They propose three approaches to performing penetration testing: the first is through static and dynamic pen-testing tools, which results in the early detection of bugs and vulnerabilities; the second approach discusses the benefits of conducting pen-testing at the unit, system, and integration level, which reduces deployment effort; and the third approach talks an about the advantages of integrating the pen-testing with the software development life cycle with mitigation strategies along the way

The Thompson [29] offers a five-step technique for performing application penetration testing. The first phase depicts the development of a threat model based on a tree, which supports security testers in breaking down an attack end-goal into testable subgoals that can be easily assessed. The second phase is a roadmap for developing a test plan, which includes logistics, deliverables, schedules, test cases, and security tools, so that the tester has a well-structured strategy to move forwards. The third phase discusses a structured approach for executing the constructed test cases, through dependency testing, user interface testing, design testing, and implementation testing. The tester now has all of the necessary information to create a problem report, which is the fifth phase. The final stage is to conduct a post-mortem to gain a better understanding of the flaws in the application software.

d. Security Requirement

Al-Janabi et al, [30] studied the main challenges in terms of security and privacy in a wireless body sensor network in a healthcare system. They have defined the Body area Networks (BAN) followed by emphasizes on the wider range of communication architecture and security and privacy requirements of BANs.

Table 2: Security threats and possible security solutions in WBAN [30]

Security threats	Security requirements	Possible security solutions
Unauthorized access	Key establishment and trust setup	Random key distribution and Public key cryptography
Message disclosure	Confidentiality and privacy	Link/network layer encryption and Access control
Message modification	Integrity and authenticity	Keyed secure hash function and Digital signature
Denial of Service (DOS)	Availability	Intrusion detection systems and redundancy
Compromised node	Resilience to node compromise	Inconsistency detection and node revocation and Tamper – proofing
Routing attacks	Secure routing	Secure routing protocols
Intrusions and malicious activities	Secure group management, Intrusion detection Systems and secure data aggregation	Secure group communication Intrusion detection systems

Thus, provide us with a valuable knowledge resource on privacy and security. Enabling us to find solutions to security requirements by looking into different literature.

The following research papers provide solutions to one or more security requirements mentioned in the figure Fig. 4.

Gope, Prosanta, and Tzonelih Hwang [31] study talks about the security requirements that must be present in a BSN healthcare system and they propose a security system namely BNS-care which provide solution to integrity, confidentiality or privacy and mutual authentication. The security requirements are focused on data privacy, data integrity, data freshness, authentication, anonymity, and secure localization. Therefore, they enforce the above securities into the BNS-care system primarily by dividing the securities into two parts namely network security and data security. They proposed a lightweight anonymous authentication protocol in the means of a solution to network securities. The protocol provides an effective means to authenticate a Local Processer Unit (LPU) in order to prevent external system access and data transfer thus, identifying the main system as an end user. The objectives of a lightweight anonymous authentication protocol are to achieve mutual authentication, anonymity, secure localization properties, to defeat forgery attacks and to reduce computation overhead. And they provided data security in BSN-care system by adopting an authenticated encryption scheme named offset codebook (OCB). The OCB provides integrity, privacy and data freshness to the system.

Tan et al. [32] addresses security and privacy issues in Body sensor networks (BNS) by deriving security requirements that include 1. Privacy at the storage site, 2. Tolerate cyber effected sensors, 3. Prevent unauthorized access and 4. scalability in granting permissions from general assumptions. Tan et al. [32] implements an Elliptic Curve Cryptography (ECC) based IBE-light scheme and developed protocols to satisfy the above requirements. Looking at the derived requirements gives us a broader view of the possible requirements needed in our testbed and also provides an efficient to encrypt data before sending it to the storage site.

The security, and the integrity of the medical data became big challenges for healthcare services applications. Elhoseny et al.[33] proposes a hybrid security model for securing the diagnostic text data in medical images. The proposed model starts by encrypting the secret data; then it hides the result in an image using steganography techniques. Both color and gray-scale images are used as cover images to conceal different text sizes. Compared with the state-of-the-art methods, the proposed model proved its ability to hide the confidential patient's data with high imperceptibility, capacity, and minimal deterioration. The proposed model efficiently provides confidentiality of data, but it is not feasible in scenarios where there is a frequent transmission of data because it overwhelms the network. In Prasanalakshmi et al.[34] study solves the overwhelming problem on the network but increases the computational requirement at the edges of the system.

Zhu at el.[35] use the ZSS signature, a novel approach for data integrity verification based on a short signature. They

offer a technique for checking data integrity at a remote site while reducing the client's computing expense. This method employs a third party to evaluate the data, as well as masking measures to ensure data privacy at the third party. Scheme is capable to effectively avoiding forgery attacks. During the performance evaluation, the author compared the suggested model (ZSS-based signature mechanism) to traditional signature mechanisms (BLS-based signature mechanism). When compared to the previous approach, the comparison result indicated a significantly shorter signature time for the same number of data blocks.

Liu et al. [36] provides a security model for cloud data integrity verification using identity-based RSA signature that supports variable-length blocks with pubic auditing. Their construction of the security was based on the RSA having a big public exponent in the random oracle model, and thus showing the effectiveness through a developed prototype.

Kothmayr et al. [37] presented a security architecture with two-way authentication for the Internet of Things. Authentication occurs during a fully authenticated Datagram Transport Layer Security (DTLS) handshake and is based on the exchange of X.509 certificates with RSA keys that they have created. Extensive testing on IoT systems demonstrates that their suggested architecture maintains message integrity, secrecy, and authenticity while costing little energy, having minimal end-to-end latency, and having little memory overhead, making it a viable security solution for the booming IoT.

Table 3: Security solutions

papers	Integrity during data transfer	Integrity at storage	Confidentiality or privacy	Authentication	Mutual Authentication between node and server
Gope, Prosanta, and Tzonelih Hwang [31]			✓		✓
Tan et al. [32]			✓		
Elhoseny et al.[33]	✓		✓		
Prasanalakshmi et al.[34]	✓		✓		
Zhu at el.[35]		✓			
Liu et al. [36]		✓			
Kothmayr et al. [37]	✓		✓	✓	
Our work	✓	✓	✓	✓	✓

IV. RESEARCH METHODOLOGY

In this section, we describe the research methodology to conduct our research plan for protecting C3I systems. The research plan is carried in three phases. Each phase is explained as follows.

1. C3I healthcare testbed development

Literature Review: Review the existing literature related to development of small-scaled testbed to analyze the requirements for implementing the testbed such as components, communication protocols etc.

C3I infrastructure components: Identify the necessary and feasible C3I components required for developing the testbed. Besides the availability, cost, power, size, and weight of C3I components are crucial factors in selecting the components.

Small-scale C3I testbed development: Use the selected components and develop a feasible C3I healthcare testbed

2. Penetration Testing

Literature Review: Study existing graphical security interfaces, and also do an extensive research on penetration testing techniques.

Build graphical security interface: Use the selected components and develop a feasible C3I healthcare testbed

Penetration testing tool selection: Select a feasible penetration testing tools that required to perform cyber-attacks on C3I testbed. Moreover, select potential cyber-attacks that can be carried out on the developed C3I healthcare testbed.

Identification of vulnerabilities: Exhaustively check for loopholes in the system after the performing the penetration testing to identify the potential vulnerabilities.

Investigation of adverse cyber impacts: Analyse the impacts of the vulnerabilities and cyber-attacks; relate the impacts to a real world set up to understand the scale of damage caused by the cyber-attacks.

3. Development of Defense Strategies countermeasures

Literature Review: Analyse existing corrective actions for detecting, mitigating, and preventing the cyber-attacks on C3I systems.

Developing countervailing solutions: Develop C3I testbed can be used for developing, evaluating, and validating countermeasures for protecting C3I systems.

V. DELVELOPMENT OF C3I HEALTHCARE TESTBED

A. C3I Architecture for healthcare testbed

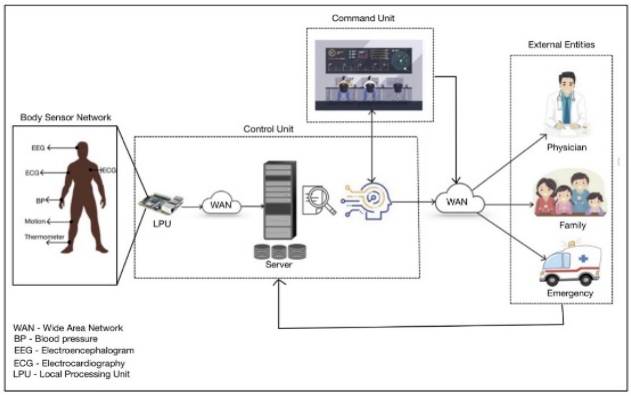


Fig. 3. C3I architecture for healthcare system.

The following subsection explains the C3I infrastructure in detail.

1. Body Sensor Network

The Body Sensor Network contains biometric sensors, and Local Processing unit that receives the data sent by the sensors, and performs two operations:

a. Initial processing of data takes place, and upon which if there are any abnormalities, it alerts the control unit.
b. Sends the sensors data to a central server which is the control unit, for further analysis.

2. Command, and Control unit

A healthcare server is a centralized control centre, which receives data from several Body Sensor Networks (BSN), and performs the following operation:

a. Analyse and interpret the data received from each BSN and find the degree of abnormality.
b. Sends the data to command centre for visualization, and decision making.
c. Based on the severity of the abnormality, and the instruction for the command centre, the control unit interacts with the external entities. The external entities in our case are local physicians, family members, or the emergency units.

3. Intelligence Unit

Intelligence unit is a part of control unit, which assists in decision making in the absence of a commander, or it assist the commander to make decision. This decision-making process is made possible using modern technologies like Machine learning, and Artificial Intelligence.

4. Communication System

Body area network (BAN):

a. BAN is a wired connection network that connects sensors to the Local processing unit and has a range of few meters.

Metropolitan area network (WAN):

b. The network provides connectivity for data and information between, the healthcare server, command centre and external entities.

B. C3I compenent selection for constructing healthcare testbed

C3I systems and their capabilities with body network systems guarantee desirable success in terms of real-life scenarios, whereas the security aspect can be tested. This was the main inspiration behind the development of the current system. As the Command center is a part of many operating situations, the control center is responsible for data processing, communications between the systems must be fast and secure, and intelligence must be insightful and useful. More importantly, the C3I capabilities and operations must be appropriate for all kinds of healthcare operations and support. Extensive literature reading was carried out to look for a better security system. Thus, by referencing to the latest research papers namely [7] and [8] in the terms of test security and reading extensively on the related articles written by Shachar Siboni, Abu Waraga and Siboni, and look at the existing research on security became the foundation for building a C3I real-world system to test security. Keeping the current world situation in mind and zeal to help Healthcare System, we wanted to build a C3I real-world system in the healthcare system using a body sensor network.

Finalization on the components for the system such as sensors, local processing unit Arduino as the microcontroller, and raspberry pi was chosen as the server as it is a portable and high performance, which could be observed in [1],[2],[3],[4] and [6]. The need for a simple, small, less yield, and cost-effective sensor system is at most important as keeping in mind the scale and size of the existing healthcare system. A complex sensory system would require intensive knowledge in the field of medicine and electronics. It would incur an additional cost, and as with complexity, the size and power requirement also increase. The sensory system would require a compatible local processing unit with easy use and portability, and connectivity with the wi-fi system. Thus, the requirement of powerful processing small, portable servers is essential, thus being used for healthcare testbed.

C. C3I testbed Implementation

The following section will explain the implementation methodology of all the different parts of the C3I healthcare testbed.

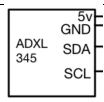
1. Data Collection Sensors

When selecting the sensor for building a testbed, an ideal plan would be to minimize the work around the sensors and concentrate on the working interactions between the entire system. To perform this, the chosen sensors followed two requirements. Firstly, every sensor had to be compatible with a single microprocessor, and secondly, the chosen sensors had to have pre-built libraries.

The following table describes the hardware components using in the system.

Table 4. Sensors

Name	***Circuit diagram***	***Description***
MAX30105		MAX30105 is a powerful Heart rate, Oximeter, and Smoke/ Partial sensor. It uses photoplethysmography (PPG) technique, where LED photodetectors are used to detect the amount of light reflecting ro the sensor. Using the reflected light and surrounding it determines the heart rate, oxygen level, or Smoke (core electronics).

Name	*Circuit diagram*	*Description*
MyoWare Muscle Sensor		MyoWare Muscle Sensor is to measure the Electromyography (EMG) of a muscle. The analog signal generated by the sensor can be read using a microcontroller that converts the analog signal to digital.
AD8232 Sensor Cable Biomedical Sensor pad		AD8232 is a heart monitor that is used to measure ECG of a person, it uses biomedical sensor pad, and sensor cables to read the electrical signals for the heart. AD8232 converts these signals and displays the graphs.
ADXL345		The ADXL345 is a small and low powered sensor that measures static acceleration, and dynamic acceleration. It measures with high resolution by using 3-axis accelerometer.
U-blox NEO-6M GPS Module		NEO-6M is a lowered power GPS sensor that can track up to twenty-two satellites on fifty channels, to locate the device. It is also inexpensive compare to other GPS sensors.

1. Local Processing Unit

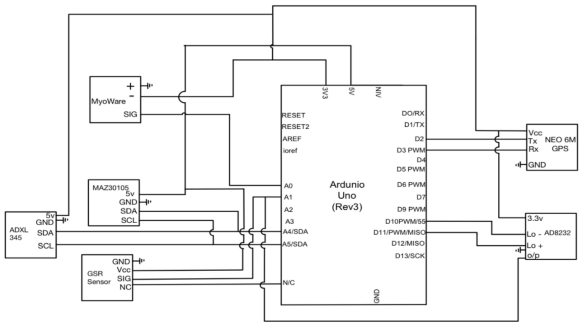


Fig. 4. Circuit diagram of BSN system

The above diagram, which consists of a logically connected circuit, shows the Body Sensor Network (BSN) containing biometric sensors that read biomedical data and a WIFI enabled Local Processing Unit (LPU) that receives data from all the sensors, which is then sent to a Control Unit using a TCP connection. To build the LPU that houses the data collection unit and sensors. The following steps were taken; Step-1: Required libraries were downloaded to run the sensor. Step-2: A code was Programmed to connect the LPU to the nearest Wi-Fi. Step-3: The output code was placed for the sensors, one after the other in a serial manner. Step-4: Designed a message format and built a TCP connection between the Control Unit and LPU.

One major drawback was the use of Arduino, which prohibited the system from reading and sending all the sensors' data at the same time since the device would not support asynchronous code.

2. Control Center

The Control Centre is the central entity that provides services to make the entire system run. It consists of three servers, namely, the LPU server responsible for services to LPU, Frontend, and Backend servers, providing interface features to the command centre to support a dynamic real-time website. The following subsection provides a detailed explanation of the servers.

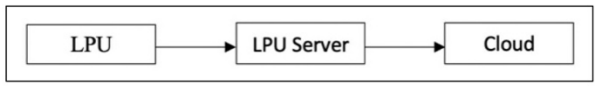


Fig. 5. Communication instance of LPU Sever.

The LPU server responsible for the LPU operations was designed to accept connections from multiple processing TCP connections using a python socket program. It also analyses each incoming data packet, based on its message format to determine the state of the sensor, whether alive or dead, and all this analysed info along with the data packet is stored on cloud.

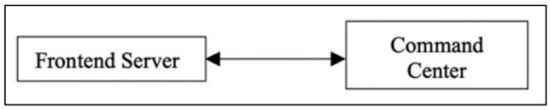


Fig. 6. Communication instance of Frontend Sever.

The Frontend Server is built using react, which is responsible for rendering every visual content that is seen by the command centre client, which includes every type of data from text, buttons, graphs, menus, etc. It is built using react framework on node and bootstrap. The framework was used to design the web-interface. The main objective of the frontend server is, 1. To renders received data from the backend server to the healthcare and security interface, 2. Providing support to an external mail API called MailGun, which is used for two-way authentication and emergency alerts to external entities, and 3. Effectively implementing the dynamic functionality using react state components.

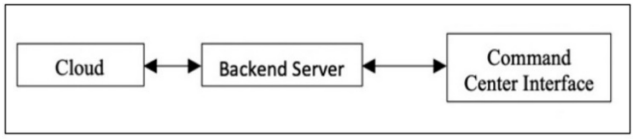


Fig. 7. Communication instance of Backend Sever.

The backend server was built using a node. To fulfill the functionality to act as a gateway between the frontend interface and the cloud. It communicates data with the Frontend server by sending and receiving HTTP requests.

As the Command centre interface being a dynamic real-time interface, it requires continuous monitoring of the database. Any latest data should be immediately rendered with respect to the frontend interface. In order to facilitate the above function backend server uses a wide range of HTTP requests. Thus, making the formats easy and efficient as to access the database.

3. *Command Centre*

The command centre is responsible for the visualization of healthcare data. The data which has been sent from remote LPU, and the security interface for the entire system. As command centre interface functionalities relies on frontend and backend server for rendering the visual contents. The command centre has been composed of three interfaces, namely, 1.login interface, which provides privacy and security to the command centre interface, and 2—Healthcare Dashboard, which renders the real-time healthcare data from the local processing.

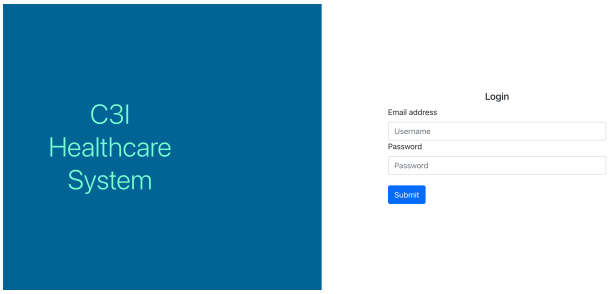


Fig. 8. Login Interface

The login module of the command centre will let clients into the system after recognizing their pre-stored passwords entered by them. As pre-stored password has a limited ability to protect the data and system. In order to strengthen the security, an added layer of security 2-way authentication is employed as it adds an additional layer of security to the authentication process by making it harder for attackers to gain access to a person's data. A 6-digit key will be sent to the users' email, and later the user will have to use the key for access into the system. As to prevent an attacker who might know the victim's password and knowing the password alone would not be enough to pass the authentication check

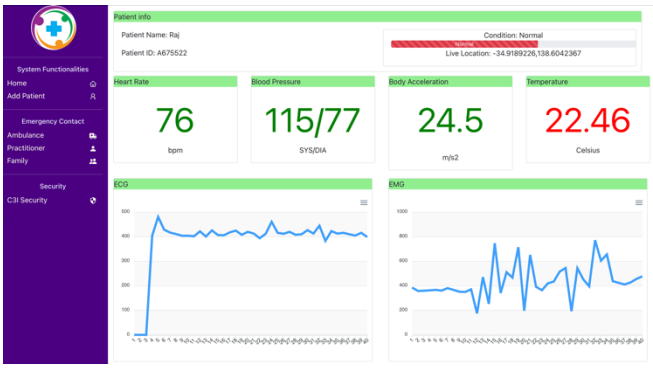


Fig. 9. Healthcare Interface

The medical interface offers a dashboard displaying six sensor data, namely GPS, Heart rate, Acceleration, Temperature, ECG, and ENG. It also offers a method of sending an emergency alert to an external entity who could be either a practitioner, an emergency unit, or a family member. On the left of the dashboard, a navigation Bar allows exploring the system completely. The Arduino sends the live sensor data to the cloud for storage. Every time new data is appended to the cloud, it gets rendered on the dashboard instantaneously. This data is continuously monitored to recognize the criticality and is rendered in a different color, if it hits those levels.

4. *Comunication System*

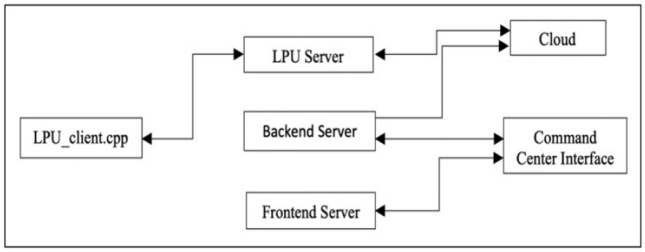


Fig. 10. Communication instance of the entire system.

The communication system that houses the Arduino runs a program to read data from the sensor. As it establishes a TCP connection with the LPU server in the Control Unit. The LPU server analyses the message packet to make sure the sensors are active, and then the data is then sent to the cloud for storage. The Backend Server uses HTTP Requests and Responds for any kind of data transfer associated with the cloud. The Frontend Server then renders this data to the Command Centre Clients.

5. *Data Storage*

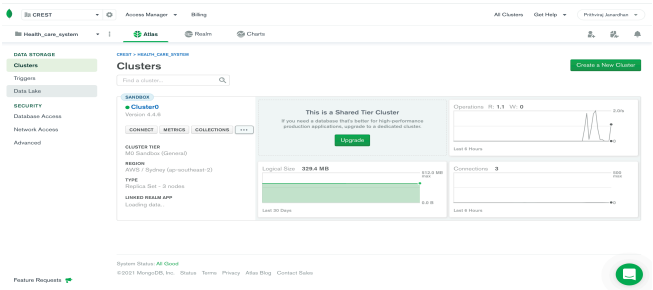


Fig. 11. MongoDB Atlas

The testbed uses the cloud as the fully manageable database which has been developed by MongoDB. The advantage of using such system is as following.

- It would reduce the strain on building and maintaining the database as it would be managed by an external company.
- Observation and study could be carried out on the security functionality as the cloud would be introduced to a real time dynamic system.
- Securities are provided by MongoDB by authenticating user's database. As they manage secrets with the Hashicorp vault, and they also use many technologies to maintain their security. Giving us a good credible source.

The need for the cloud in the testbed is primarily to store healthcare data that is being transmitted by the LPU server and stores the state and condition of all the available sensors and its system.

VI. SECURITY INTERFACE

Real-Time Security analysis of C3I healthcare system

The C3I healthcare system is a real-time health monitoring system comprised of healthcare sensors, servers, and web-based clients. Vulnerabilities in the system can occur at every component and exploiting these vulnerabilities might result in serious security risks such as unauthorized access, denial of services (Dos), compromised sensors, routing attacks, and healthcare information leaks [30]. As a result of its complexity, the ability to detect potential attack scenarios and restrict the impact of malicious attacks becomes a key problem. We created an autonomous near-real-time security web interface utilizing GrSM (Graphical Security Model) and the Common Vulnerability Score System to analyse the system's security at any stage in its lifespan (CVSS). The section that follows discusses security metrics and web security interface.

Security metrics

One method for assessing the security of a heterogeneous system is to divide it into vulnerability, node, path, and network levels and calculate them using metrics such as; 1) Attack Successful Probability: This is the likelihood that an attacker will succeed in his or her attack aim. 2) Attack Impact: This is the amount of harm that an attacker may cause if he or she achieves their aim. 3) Attack Risk: This is the level of risk to the system if an attacker succeeds in his aim, and CVSS: This is the vulnerability's base score ([40] and [41]). The sub-sections that follow provide a security evaluation at each level as well as the algorithm utilized at that level.

A. Vulnerability level: Gives an assessment of how a single vulnerability when exploited impacts the system. It is computed by calculating the attack successful probability (asp_v), potential loss (ai_v), risk(ar) and CVSS base score ($cvss_v$) of successful exploitation of the vulnerability by an attacker. The equations for calculating the above metrics are given below.

Equation:

$$asp_v = \frac{Exploilability\ score}{10}$$

$$ai_v = Attack\ impact$$

$$ar_v = asp_v + ai_v$$

$$cvss_v = CVSS\ base\ score$$

Algorithm:

Algorithm 1: The algorithm calculates vulnerability level metrics based on the equation given above.

Function vulnerabilityLevel(vulnaribilityName):
Input: vulnaribilityName
Output:$(asp_v), (ai_v), (ar_v), (cvss_v)$
$cvssBaseScore, attackImpact, explaitabilityScore \leftarrow$
$getthevaluesfromcloudtheusingthevulnerabilityID$
$cvss_v \leftarrow cvssBaseScore\ ai_v \leftarrow attackImpact\ asp_v \leftarrow \frac{explaitabilityScore}{10}$
$ar_v \leftarrow ai{*}asp$
return $asp_v, ai_v, ar_v, cvss_v$

B. Node Level: Provides an assessment of how a single node containing vulnerabilities can impact the system when exploited. It is computed by calculating of the attack successful probability (asp_n), potential loss (ai_n), risk(ar_n) and CVSS base score ($cvss_n$) of successful exploitation of the node by an attacker. The equations for calculating the above metrics are given below.

Equation:

$$asp_n = \begin{cases} \prod_{v \in n(v)} asp_v & if\ \text{AND} \\ 1 - \prod_{v \in n(v)} (1 - asp_v) & if\ \text{OR} \end{cases}$$

$$ai_n = \begin{cases} \sum_{v \in n(v)} ai_v & if\ \text{AND} \\ \underset{v \in n(v)}{\text{MAX}}\ (ai_v) & if\ \text{OR} \end{cases}$$

$$ar_n = \begin{cases} \sum_{v \in n(v)} ar_v & if\ \text{AND} \\ \underset{v \in n(v)}{\text{MAX}}\ (ar_v) & if\ \text{OR} \end{cases}$$

$$cvss_n = \begin{cases} \sum_{v \in n(v)} cvss_v & if\ \text{AND} \\ \underset{v \in n(v)}{\text{MAX}}\ (cvss_v) & if\ \text{OR} \end{cases}$$

Algorithm:

Algorithm 2: The algorithm calculates Node level metrics based on the equation given above.

Function nodeLevel(node, logic):
Input: node, logic
Output:$asp_n, ai_n, ar_n, andcvss_n$
$ai_n \leftarrow 0$
$ar_n \leftarrow 0$
$cvss_n \leftarrow 0$
$flag \leftarrow 0$
for *each vulnerability in node* **do**
  $asp_v, ai_v, ar_v, cvss_v \leftarrow vulnerbilityLevel(vulnerability)$;
  if *flag is 0* **then**
    $asp_n \leftarrow asp_v$
  else
    $asp_n \leftarrow asp_n * asp_v$
  end
  $flag \leftarrow 1$
  if *logic is* ***and*** **then**
    $ai_n \leftarrow ai_n + ai_n$
    $cvss_n \leftarrow cvss_n + cvss_v$
    $ar_n \leftarrow ar_n + ar_v$
  end
  if *logic is* ***or*** **then**
    if $ai_v > ai_n$ **then**
      $ai_n \leftarrow ai_v$
    end
    if $ar_v > ar_n$ **then**
      $ar_n \leftarrow ar_v$
    end
    if $cvss_v > cvss_n$ **then**
      $cvss_n \leftarrow cvss_v$
    end
  end
end
if *logic is* ***or*** **then**
  $asp_n \leftarrow 1–[1 - asp_n]$
end
return $asp_n, ai_n, ar_n, andcvss_n$

C. Path Level: Gives an assessment of how a single path containing nodes can impact the system when exploited. It is computed by calculating of the attack successful probability (asp_p), potential loss (ai_p), risk(ar_p) and CVSS base score ($cvss_p$) of successful exploitation of a target via an attack path by an attacker. The equations for calculating the above metrics are given below.

Equation:

$$asp_p = \prod_{n \in p(n)} asp_n$$

$$ai_p = \sum_{n \in p(n)} ai_n$$

$$ar_p = \sum_{n \in p(n)} ar_n$$

$$cvss_p = \sum_{n \in p(n)} cvss_n$$

Algorithm:

Algorithm 3: The algorithm calculates Path level metrics based on the equation given above.

Function pathLevel(path):
Input: path
Output:$asp_p, ai_p, ar_p, and cvss_p$
$ai_p \leftarrow 0$
$ar_p \leftarrow 0$
$cvss_p \leftarrow 0$
$flag \leftarrow 0$
for *each node in path* **do**
  $asp_n, ai_n, ar_n, cvss_n \leftarrow nodeLevel(node, logic)$;
  if *flag is 0* **then**
    $asp_p \leftarrow asp_n$
  else
    $asp_p \leftarrow asp_p * asp_n$
  end
  $flag \leftarrow 1$
  $ai_p \leftarrow ai_p + ai_n$
  $ar_p \leftarrow ar_p + ar_n$
  $cvss_p \leftarrow cvss_p + cvss_n$
end
return $asp_p, ai_p, ar_p, and cvss_p$

D. Network Level: Provides an assessment of how a network containing all the possible paths can impact the system when exploited. It is computed by calculating of the attack successful probability (asp_{nt}), potential loss (ai_{nt}), risk(ar_{nt}) and CVSS base score ($cvss_{nt}$) of successful exploitation of a target via an all the attack paths by an attacker. The equations for calculating the above metrics are given below.

Equation:

$$asp_{nt} = \sum_{p \in nt(p)} asp_p$$

$$ai_{nt} = \text{MAX}(ai_p)$$

$$ai_{nt} = \text{MAX}(ai_p)$$

$$cvss_{nt} = \text{MAX}(cvss_p)$$

Algorithm:

Algorithm 4: The algorithm calculates Network level metrics based on the equation given above.

Function pathLevel(network):
Input: path
Output:$asp_p, ai_p, ar_p, and cvss_p$
$ai_{nt} \leftarrow 0$
$ar_{nt} \leftarrow 0$
$cvss_{nt} \leftarrow 0$
for *each path in network* **do**
  $asp_p, ai_p, ar_p, cvss_p \leftarrow nodeLevel(path)$
  $asp_{nt} \leftarrow asp_{nt} + asp_p$
  if $ai_p > ai_{nt}$ **then**
    $ai_{nt} \leftarrow ai_p$
  end
  if $ar_p > ar_{nt}$ **then**
    $ar_{nt} \leftarrow ar_p$
  end
  if $cvss_p > cvss_{nt}$ **then**
    $cvss_{nt} \leftarrow cvss_p$
  end
end
return $asp_{nt}, ai_{nt}, ar_{nt}, and cvss_{nt}$

Web Security interface

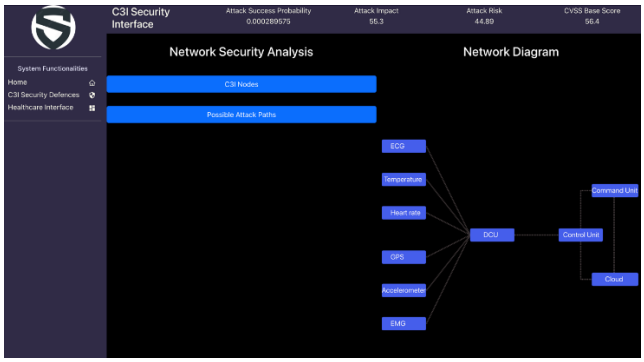


Fig. 12: Real-time web security interface

The security interface was developed to allow system administrators to obtain real-time security assessment at any stage in the system's lifecycle. The web security interface that is illustrated in Fig. 12 gives a visual understanding of security. In the middle of the interface, there lies an option to view the vulnerability, node and path level security assessment, and the network diagram which can be seen on the right side of the interface, which illustrates the diagrammatic view of assessment being performed.

VII. PENETRATION TESTING

Penetration testing is a process to identify and exploit flaws in the system in order to improve its overall security. The primary goal of doing a penetration test was to detect potential vulnerabilities in the system. After conducting a thorough review of the methods for performing tests, we decided to conduct the tests using the available static and dynamic tools. We performed the security tests in two phases, the first involving static security test tools and the second involving dynamic security test tools. The following two sections explain the two phases in detail.

Static security testing

White box testing, or static security testing, is a method of analysing source code in order to identify vulnerabilities. Static testing can is performed without executing the software giving it the ability to test the software throughout its development lifecycle. It also can help detect vulnerabilities in the early stages of software development and resolve faults simultaneously. The table 1 shows some of the characteristics of static security testing.

Table 5**:** the characteristics of static security testing

It's white Box Security testing
Requires Source code
Finds vulnerabilities from earlier development lifecycle
It cannot discover run-time and environment-related issues
Mostly supports all kinds of software

To perform static security the following tools were used.

1. Deepsocuce: It is a static analyser that is design to inspect and analysis source code in a repository, and it simultaneously track different metrics and issues. The analyser has low false positive rate, and comes with easy configuration and integration with Bitbucket, GitHub, GitLab workflows. The results after analysis are categorised the into bugs, performance issues, security issues, and type check issues. To perform the static analysis, the tool was integrated with our GitHub repository where all our code is present [38].

2. Nodejsscan: The tool is a static security code scanner for Node.js projects. It has a user interface with various dashboards that display an application's security status. It's used to detect weaknesses in online apps, web services, and serverless apps, and it scans mainly for remote code injection, open redirect, SQL injection, and XSS

vulnerabilities. To perform static analysis using the tool, we had to create a zip file containing all the server software files and feed the tool with the generated zip file [39].

Dynamic security testing

Dynamic security testing is a black box security testing. The application must be deployed before the test can be executed, and testing is done by simulating external attacks on the application. These tests that are being executed tries to break into the application from the outside by looking for vulnerabilities in exposed interfaces. A clear benefit of running a dynamic test is that it can detect runtime issues that a static test cannot. The table 2 shows some of the characteristics of dynamic security testing.

Table 6: The characteristics of dynamic security testing.

It's black Box Security testing
Requires a running application
Finds vulnerabilities towards the end of the software development lifecycle
More expensive to fix found vulnerabilities
Typically scans only applications like web apps and web servers

To perform static security the following tools were used.

1. Zap: Zed Attack Proxy, is a multi-platform, open-source online application security testing tool developed by OWASP (Open Web Application Security Project). It can be used during the development and testing phases of a web app, ZAP is used to uncover a variety of security flaws such as application error disclosures, cookie not HttpOnly flags, missing anti-CSRF tokens and security headers, private IP disclosures, SQL injections, and XSS injections vulnerabilities. Apart from being a scanner, ZAP may also be used as an intercept proxy or to test the webpage manually [40].

2. Arachni: It is a feature-rich, modular, and high-performance Ruby framework designed to aid penetration testers and administrators in assessing the security of web applications. It trains itself by seeing and learning from the behaviour of the web application during the scanning process, and it is designed to identify security issues within a web application. The open-source security testing tool can perform attacks like SQL injection, blind SQL injection using differential and timing attacks, NoSQL injection, cross-site request forgery, LDAP injection, OS command injection, and cross-site scripting [41].

3. Creashtest Security: It is a dynamic vulnerability scanner that conducts single-page, multi-page, and documentation-based scans to identify attack vectors across the web application that being scanned. It performs wide range of security tasks that include fingerprinting, fuzzing, deserialization, injection attacks, XSS attacks, CSRF attacks, and checks for SSL/TLS vulnerabilities [42].

To perform dynamic analysis, instated of deploying the application we used a tunnelling tool called ngrok to *expose* our web server running on our local machine to the internet. The generated HTTP link from ngrok was used as the input to perform dynamic testing using the tools described above.

VIII. DEFENSIVE STRATIGIES

External Integrity Verification

In our approach of using data centre as cloud benefits system to be elastic and scalable. With many practical and financial advantages of using cloud comes with strong concerns in data security and privacy [35]. In our C3I healthcare system, the data coming from every individual Local Processing Unit (LPU) is entirely outsourced to the MongoDB cloud service provider, which means our system does not store and manage the data locally. Generally, cloud providers deploy their own security mechanisms, but it is safe to have self-developed verification mechanism. The following section explains the data integrity verification mechanism deployed to the C3I healthcare system.

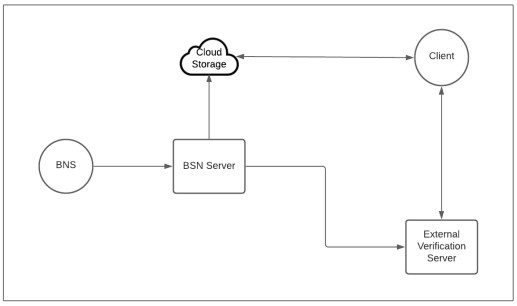


Fig. 13: Working context of External verification Sever

There are three main participants in the verification process.

1. BNS Sever: The sever receives messages from various LPUs that are a part of remote Body Sensor Network, stores sensor data form the message on cloud, and the server is also responsible for sending unique hash values to External Verification server.

2. External Verification Sever: The server is responsible for storing time stamped hash values from BSN sever and provides integrity verification check.

3. Clients (Commander): The client views the real-time healthcare data on the dashboard and the data being viewed is verified for integrity by the External Verification sever.

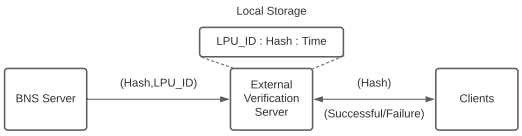


Fig. 14: The integrity verification process

We can view the working of integrity verification as three different independent processes taking place at BSN server, External verification server, and at frontend client respectively.

1. The first process takes place at BNS sever when a message received form a BSN and has the following steps:

   Step 1: Extracts the sensors data from the message.
   Step 2: Converts the sensor data into float values and add the values.
   Step 3: Generates a SHA256 hash using the resultant value from the above step.
   Step 4: Sends the respective BSN_ID and the SHA256 hash to the External server using storeHash API.

2. The second process has two APIs which are part of External Verification Server

   - storeHash API: It responsible for storing a time stamped hash value coming from BSN server in the local storage.
   - checkIntegrity API: The API takes hash and BSN_ID as the input and verifies the hash value with the stored hash and send the status back to the client.

3. The third process takes place when updating the dashboard at client side and has the following steps.

   Step 1: Converts the sensor data into float values that being retrieved form the cloud using the backend server.
   Step 2: Generates a SHA256 hash using the resultant value from the above step.
   Step 3: Send the respective BSN_ID and the SHA256 hash to the External server using checkIntegrity API.
   Step 4: If the status for the API is successful then it updates the dashboard otherwise show a warning message on the dashboard.

The building data integrity verification mechanism ensures data integrity and availability of data that stored in the cloud and provides data privacy at External Verification server.

Two Way Authentication

Today, Numerous software systems use traditional authentication techniques, which uses the username as the owner's identity and password to prove the identity of the account owner. Here the server saves the password's hash instead of the actual password to protect against any data leak that discloses the passwords. Securing the system in this way seems reliable. However, many passwords are still compromised due to users not using strong passwords and improvised new technological hardware that crack passwords at alarming speeds [47]. Adding a layer of security on top of the traditional authentication method using a one-time password increases the system's safety, known as multi-factor authentication.

This defensive strategy aims to build an authentication system on the client side that securely complements the one used today by utilising intelligent mobile devices and existing security standards to increase the effect of cracking the C3I healthcare system. The following explains TOTP based authentication system that was built to accommodate a secure way to authenticate the commander of the system.

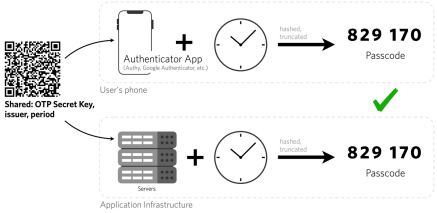


Fig. 15: Working of Two-Way Authentication [48]

The Time-Based One-Time Password (TOTP) algorithm implemented uses three components 1. Mutually shared secret 2. Time as the input and 3. signing function [49].

1. Mutually shared secret: When the secret is generated, it is sent to the backend server to be stored, and the client uses google authenticator or a similar app to keep the secret.
2. Time as input: Both the generator (Googe authenticator) and the validator (C3I server) use time as the input. The time must be synchronised to effectively generate the correct token.
3. Sign in Function: It is a hash function that uses HMAC-SHA1, the algorithm signs a value using a secure one-way hash function. We may check authenticity by using the algorithm where only those who know the secret and have a synchronised time input can create the same code which are used for validation.

Edge node Authentication

In the C3I healthcare system, there exists a BNS server which is the central entity, which receives, processes and stores information from the BNS nodes. To sustain the trustworthy accessibility and connectivity of the BSN node, it becomes essential to establish a secure end-to-end connection with sustainable authentication. Here we propose a two-phase authentication protocol that let BSN end nodes authenticate to the BNS server. The following section explains the working of the protocol.

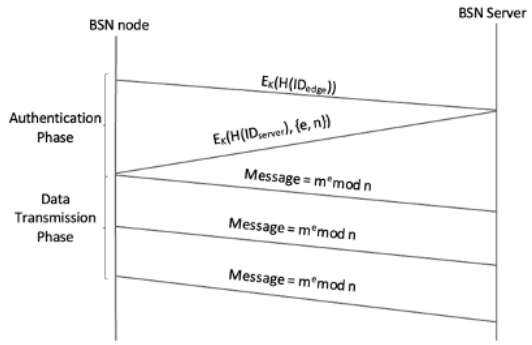


Fig. 16: Edge Node Authentication

The protocol has two phases

- Authentication Phase: The BSN node authenticates its identity using a hashed ID that is encrypted with the shared symmetric key.
- Data Transmission Phase: After successful identification of the node, the BSN server send a public key values (e,n) using with which the BSN node encrypts the message for secure transmission

The proposed protocol supports the authentication of nodes and a secure way to send messages to the BNS server.

IX. Results and Analysis

A. C3I Healthcare Testbed

1 Testbed Setup and Testing

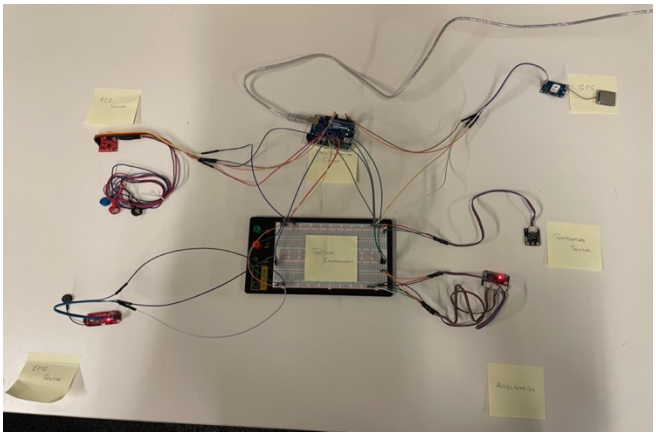


Fig. 17. Local Processing Unit

The six sensors, namely the GPS, Thermometer, Heart Rate, Acceleration, ECG, and EMG, are used for this model testbed. They are all connected via a breadboard to the Arduino that acts as a Data Collection Unit. The sensor data is read one after the other in our testbed, unlike a real system where all the sensors read data at the same time. This was the only shortcoming here. The rest of the functionality is very close to that of a real system.

Fig. 18. Testbed Setup

Later, the entire testbed was set up in the lab to simulate and represent a real medical healthcare system using the sensors, Arduino for a Local Processing Unit (LPU), and the Raspberry Pi for Servers above. To simulate the real scenarios, some noise was feed into the sensor setup. The LPU was tested for every sensor (Arduino), and its value was checked to make sure it was corresponding to the real values. The Arduino was also tested to see if it could establish well connectivity to the nearest WIFI, and a TCP connection with the Control unit was set up to observe the packets indicating synchronism. The observation led to a successful outcome and results.

In the Control Unit, the Backend Server was checked if it was actively connected with the cloud, and the HTTP requests were successfully set up with it for data transfer. The Frontend server was then tested to see if the data and the visual content were correctly rendered to the Command Unit.

The Command unit of the dashboard was checked for data integrity in real-time. Also, the visuals were closely monitored to make sure there were no discrepancies in tandem with the data.

B. Visualising data transfer between the systems

The healthcare data from all the sensors are being transferred from LPU to the healthcare interface in near real-time. To make the transfer possible, the communication happens in six stages 1. Sensors transfer data to LPU using an Ethernet connection 2. LPU sends the sensor data to the LPU server using WAN technology 3. LPU Server sends the received healthcare data to MongoDB cloud 4. The Frontend interface keeps requesting updated healthcare data to Backend server 5. The Backend server responds to every request sent by sending JSON format data to the Frontend server. 6. The Frontend server renders the healthcare data to the dashboard that is sent by the Backend server. We explain the above six stages in detail in the following subsection using a single healthcare sensor.

2. Data at Local Processing Unit

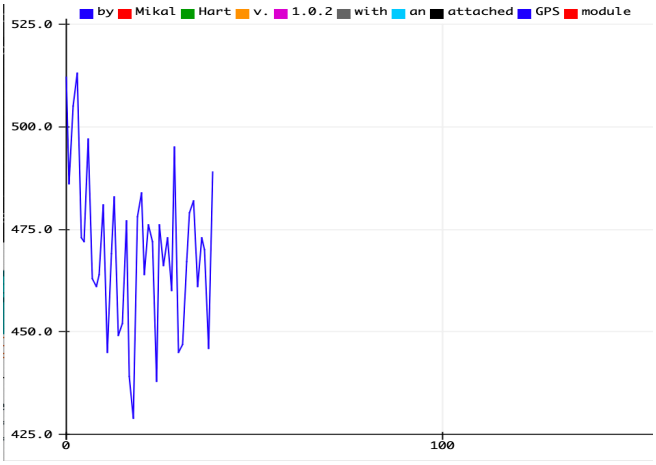


Fig. 19. EMG sensor values on serial ploter

Fig. 20. EMG sensor values on serial Moniter

We can visualize the sensor reading using serial monitor and serial Plotter, which are the per installed software tools present in Arduino IDE. The serial monitor shows the EMG sensor reading from the sensor and the Plotter, plots digital values on the line graph. The digital values coming from the sensor are placed in a message format containing sensor ID and associated digital value, which is then sent to an LPU server using a TCP socket connection.

3. Data at Control Unit

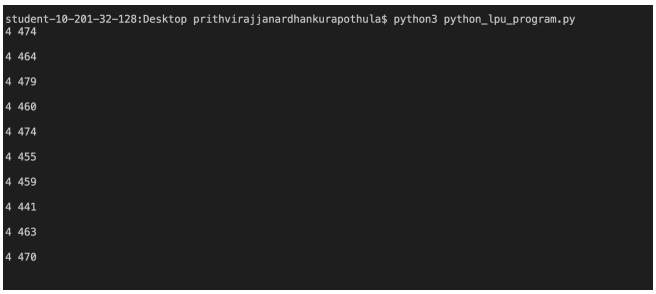


Fig. 21. EMG sensor values at LPU server

The above figure shows the EMG sensor data being received at the LPU server from the LPU(Arduino). The server analysis each packet and determines the sensor and its present state. After which, the analyzed data is sent to the cloud.

4. Data in cloud

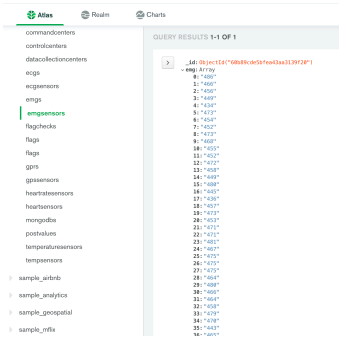


Fig. 22. EMG sensor values at cloud

The above shows the EMG values being updated on the cloud. The updated EMG values are accessed by the back-end server when the front-end interface requests updated values through an http request.

5. Data at Frontend Interface

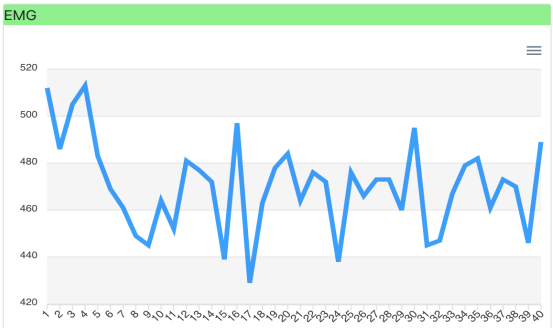


Fig. 23. EMG sensor values on the dashboard

The Above figure shows the near real-time EMG sensor values on the dashboard. To do this, the frontend server keeps requesting the backend server for the updated values in a continuous loop. When the backend server sends updated values, the frontend server instantly renders the content onto the dashboard.

C. C3I system assessment using built security interface

The following section talk about the security assessment being made from a lower level to the upper lever starting form vulnerability level and going up to network level.

Node and Vulnerability level:

Fig. 24. Dialogbox to select nodes at node level

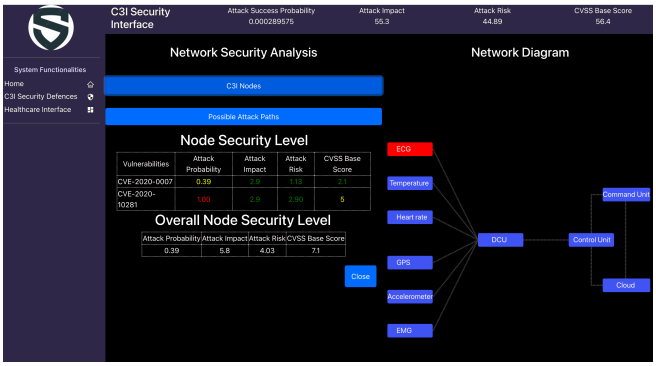


Fig. 25. Results for vulnerability and node level security assessment

The dialog box in the Fig. 24 is shown when the user clicks on the ‘C3I Nodes’ button option on the screen. The dialog box shows all the C3I nodes that are present in the system. Clicking on the nodes will show vulnerability level and node level security assessment.

In Fig. 25, the ECG component is highlighted red on the network diagram that shows the node being assessed. As one can observe on the left side of the network diagram, the vulnerability and node level security assessments are being shown. These values are calculated algorithmically, as presented in section 6.

Path level:

The dialog box in the Fig. 26 is seen when the user clicks on the ‘Possible attack paths.’ The dialog box shows all the possible targets that are present in the system. Clicking on them will show path level security assessment.

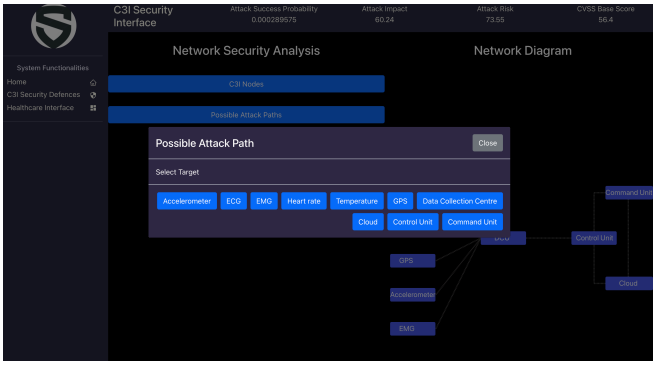


Fig. 26: Dialogbox to select target at path level

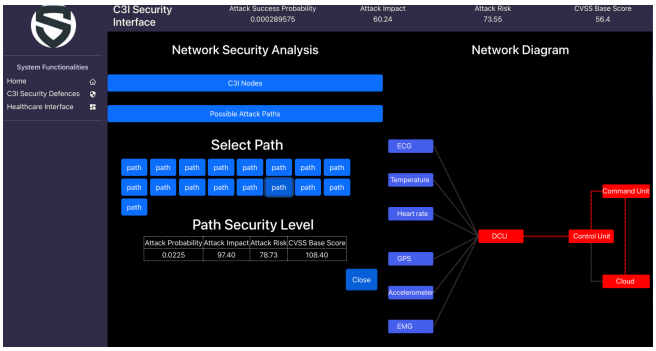


Fig. 27: Path level assessments

The Fig. 27 shows the change in appearance because the user has clicked on the ‘cloud’ button in the ‘selected target’ dialog box. In the figure we can observe that one of the paths is being assessed which is shown in red at the network diagram. As one can notice on the left side of the network diagram, the path level security assessment is being shown, and these values are calculated algorithmically as presented in the section 6.

Network Level

Fig. 28: Network level Assessments

The figure illustrates the network level security assessment metrics values that were calculated for every constant interval of time.

Sensor availability check

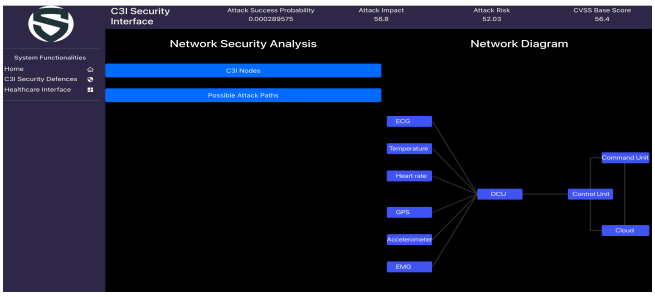


Fig 29. Interface when all nodes are present

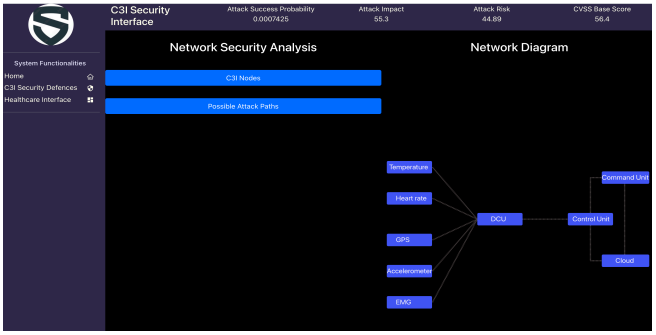


Fig 30. Interface when ECG node is removed

Sensor availability check is one feature that was added on top of the security assessment metrics. The Fig. 29 illustrates that all nodes in the system are present and working properly. But when you look at Fig 30, we see that, ECG node in the network diagram is not present. This is because sensor ECG was removed and was intentionally done for the purpose of testing and to show the working of the feature. Finally, we notice the change in the attack successful probability in both above figures. The change is because there is a change in the infrastructure which changes security impact on the system and working of the security interface indeed works in real-time.

D. Results form initial penetration testing

In this section, we examine the test results that were evaluated shortly after the establishment of a functionally working system. This is being done to understand the system's security state so that we can plan what security actions to take and security mechanisms to put in place to secure the system. The following section contain the results from the static and dynamic security testing.

Deepsource

Table 7: Results from deepsource

Vulnerability	**CWE**
Server Leaks information via "X-Powered-By" HTTP Response Header Field(s)	CWE-200
Style issues	-
Code related bug issues	-
Anti-patterns	-

The analyser identified four types of bugs in 158 places, nine anti-patterns in 331 places, 13 style-related issues in 67 places, three performance issues in 12 places, and one security issue. After carefully looking into the results, we realised that most of them were basic easy to solve code-related issues; for example, a few of the issues were related to code having console logs. We realised that the 'X-Powered-By' HTTP head was present that caused a single security issue. The problem with the header being present is that it leaks out the information of the technologies being used, leading to a successful attack when the attacker finds and exploits the vulnerabilities present in the technologies.

Nodejsscan

Table 8: Results from Nodejsscan

Vulnerability	***CWE***
Node insecure random generator	CWE-327
Express XSS	CWE-79
Security Misconfiguration	CWE-693

The tool found 16 different types of issues with a total of 17 issues, there is one warring, one error, and 15 information issues. The error issue is Express XSS, which occur when there is a input form that is not validating the user input, and the warring error was about a file having an random generator that not secure. The 15 information issues tell us about security misconfiguration in the files.

Arachni

Table 9: Results from Arachni

Vulnerability	***CWE***
Missing 'X-Frame-Options' header	CWE-693
Common Sensitive file	CWE-219
Private IP address disclosure	CWE-200
Allowed HTTP method	Informational

The tool found four issues; first is a missing 'X-Frame-Option' header means that the application is at risk of clickjacking attacks; the second issue was sensitive file exposure vulnerability; the third issue was disclosure of private IP address, and the last point is an information issue.

Zap

Table 10: Results from Zap

Vulnerability	**CWE**
Application Error Disclosure	CWE-200
Cross-Domain Misconfiguration	CWE-264
CSP: Wildcard Directive	CWE-693
X-Frame-Option Header Not Set	CWE-1021
Private IP Disclosure	CWE-200
Server Leaks information via "X-Powered-By" HTTP Response Header Field(s)	CWE-200
Information Disclosure – Suspicion Comments	Informational
Timestamp Disclosure – Unix	Informational

The zap scanning tool found nine different types of issues; first, there was one instance of application error disclosure; Second, cross-domain misconfiguration issue was found at 18 instances, and it occurs due to cross-origin resource sharing misconfiguration on the web server. CSP: wildcard directive, X-Frame-Options and private IP disclosure issues were present at only one instance. Two header-related issues were present; one was X-Powered-By' HTTP which leads to server leaks, and the other was a missing X-contend header. The last two issues are informational issues; one was about information disclosure through suspicious comments, and the other was a timestamp disclosure.

Creashtest Security

Table 11: Results for Creashtest

Vulnerability	CWE
The X-Frame-Options header is not set	CWE-693
The Strict-Transport-Security (HSTS) header is not set	CWE-523
The Content-Security header is not set	CWE-1021
The Referrer-Policy header is not set	CWE-293
The X-Content-Type-Options header is not set	CWE-200

The tool conducted the tests by sending 4,562 attack requests, and it found 51 issues. All five issues are missing header related to issues: namely, x-Frame-options, content-security-policy, Strict-transport-security (HSTS), referrer-policy, and x-content-type-options headers. And the tool found 46 port scanner related informational issues.

E. Secutity testing the system in continous pipeline

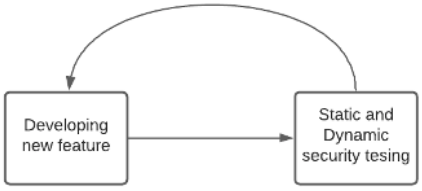


Fig 31. Security testing in continuous pipeline

Following the initial security testing, we used a pipeline technique to do additional security testing. Before going on to the next feature or module, we tested the security of the system for each new feature or module that was developed. This allows us to analyze the behavior of software at every stage of development, which decreases the complexity of testing and correcting vulnerabilities.

F. Defensive stratigies

For the C3I healthcare system, three defensive techniques were deployed in order to safeguard and improve its overall security. The following sections provide an explanation of how the system works as well as illustrations of the system's protected components.

Two-way Authenticaiton

Two-way authentication system is built using TOTP algorithm where the six-digit code is being generated for constant time at the client side. The mechanism works in two phases one is registration phase and the other is verification phase.

1. Registration Phase

Once the user has registered their username and password, they must use Google authenticator to store the secret generated by the QR code on their device, and the secret is sent to the backend server when the sign up is successful. This The information about the new client is saved in the cloud by the server and is used when verifying the client.

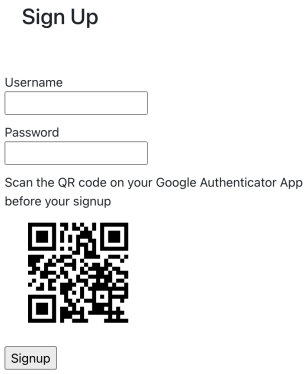


Fig 32. Sign Up page

2. Verification Phase

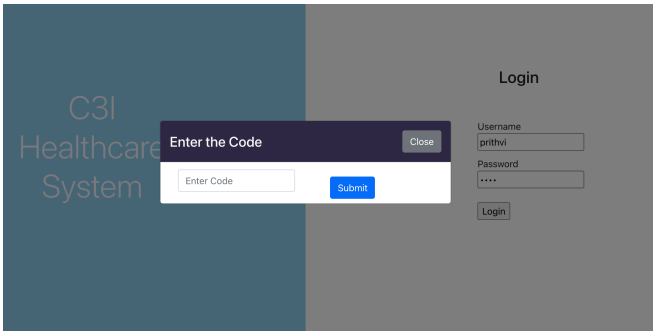


Fig 33. Dialog box to enter the code

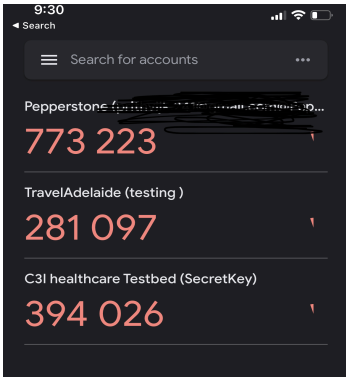


Fig 34. Google Authenticator

Following successful one-time authentication, the dialogue box depicted in Figure 1 appears, in which the client enters the code generated by Google Authenticator, which is forwarded the backend server for verification. The backend server generates the verification code using the time, and secret key as the input and compares the code with code that being sent by the frontend client. If the code is matches, then client is granted access to the web interface, where he or she can examine the dashboard and security interface. Otherwise, throughs an error.

Because the code is generated at the client's end, it is impossible to launch an attack aimed at stealing a short-term secret during the data transmission.

Edge Node Authenticaiton

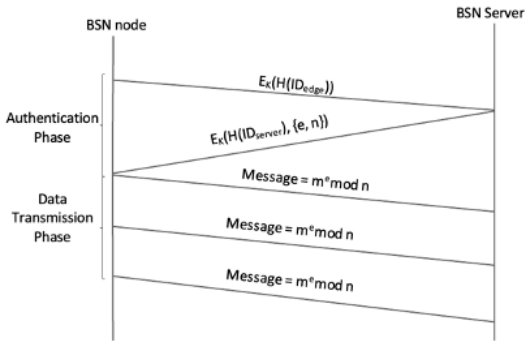


Fig 35. Sequence diagram of Edge Node Authentication

Edge Node Authentication was developed to authenticate the BSN nodes in the system. In order to communicate with the BSN server, the BSN node establishes a TCP connection with the BSN server, which provides reliable, ordered, and error-checked data transfer. However, how data privacy is maintained during transmission is up to the discretion of the application developer. Through the use of public key encryption and transfer of encrypted hash IDs using symmetry key, we were able to assure both the authentication of the BSN node and the preservation of privacy while the data was being sent.

External Verification Server

The server and the defensive mechanism were built to check integrity of the data when receive at the frontend client from the cloud storage. If the check fails, then the client does not update the sensor values on the healthcare dashboard and throws an error. To provide such an integrity verification check three components i.e., BNS sever, External Verification server and Client of C3I healthcare system play an important role.

```
jake@jake-ThinkPad-L440:~/Desktop/servers$ python3 manBnsServer.py
hash: 59e19706d51d39f66711c2653cd7eb1291c94d9b55eb14bda74ce4dc636d015a  ID:
BSN2837
hash: c2356069e9d1e79ca924378153cfbbfb4d4416b1f99d41a2940bfdb66c5319db  ID:
BSN2837
hash: e29c9c180c6279b0b02abd6a1801c7c04082cf486ec027aa13515e4f3884bb6b  ID:
BSN2837
hash: 785f3ec7eb32f30b90cd0fcf3657d388b5ff4297f2f9716ff66e9b69c05ddd09  ID:
BSN2837
hash: 5f9c4ab08cac7457e9111a30e4664920607ea2c115a1433d7be98e97e64244ca  ID:
BSN2837
```

Fig 36. Logs at BSN Sever

As shown in the fig 36, the console is logging unique hash values, which are then forwarded to the External Verification Server for integrity verification process. The hash values are generated when the BSN server receives data from the BNS node, and the uniqueness of hash value is leveraged by using the data that is being updated to the cloud during that time period. Specifically, this is accomplished because the frontend client uses the same hashing process to produce the hash, and this hash is then provided to the External Verification Server for verification.

```
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node verificationServer.js`
91e9240f415223982edc345532630710e94a7f52cd5f48f5ee1afc555078f0ab
Server listening on 8000
Form server
hash:59e19706d51d39f66711c2653cd7eb1291c94d9b55eb14bda74ce4dc636d015a ID:BS
N2937
Form server
hash:c2356069e9d1e79ca924378153cfbbfb4d4416b1f99d41a2940bfdb66c5319db ID:BS
N2937
Form server
hash:e29c9c180c6279b0b02abd6a1801c7c04082cf486ec027aa13515e4f3884bb6b ID:BS
N2937
Form server
hash:785f3ec7eb32f30b90cd0fcf3657d388b5ff4297f2f9716ff66e9b69c05ddd09 ID:BS
N2937
For client
hash:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ID:BS
N2937
For client
hash:e29c9c180c6279b0b02abd6a1801c7c04082cf486ec027aa13515e4f3884bb6b ID:BS
N2937
For client
hash:785f3ec7eb32f30b90cd0fcf3657d388b5ff4297f2f9716ff66e9b69c05ddd09 ID:BS
N2937
For client
hash:785f3ec7eb32f30b90cd0fcf3657d388b5ff4297f2f9716ff66e9b69c05ddd09 ID:BS
N2937
```

Fig 37. Logs at External Verification Server

The logs of hash values on the External Verification Server console, which are being sent by the BSN server and the frontend client, are depicted in the Fig.37 above. The External verification server saves hash values that are being sent by the BSN server in a Time:ID format, and this is done in order to verify the hash value that is being sent by the front-end client. As soon as the External Verification Server receives a hash, it compares it to the stored hash that was sent by the BNS server; if both hashes are identical, then the server sends a successful integrity verification message to the frontend client; if they are not, the server sends an unsuccessful message.

The top level design view of the implemented system

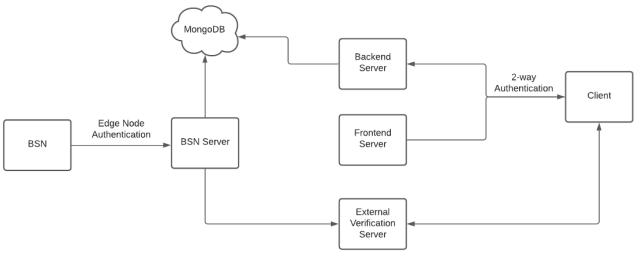


Fig 38. Complete view of C3I healthcare System

The fig 38 shows the top-level design diagram of the entire system with implemented security measures. The integration of external integrity verification, two-way authentication and edge node authentication makes the system as secure as possible from every aspect of the architecture.

CONCLUSION

Cybersecurity vulnerabilities are rife in any network. To analyse their impact in the healthcare systems we designed a C3I system integrated with a BSN. The security

vulnerabilities were assessed in real time using graphical user interface standards. Penetration tests were also conducted to perform an exhausted assessment of the developed testbed. Some code related bugs, misconfigurations and header related problems were identified and solved. With this research we learnt that the vulnerabilities are many and can never be eliminated completely. We also integrated external integrity verification, two-way authentication, and edge node authentication as the three defensive solutions suggested by various authors and try to make the system as cyber secure as possible.

REFERENCE

[1]. Abrahamsson, p., et al. (2013). "Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment." IEEE 5th International Conference on Cloud Computing Technology and Science, vol. 2, pp. 170-175.

[2]. Cox, S. J., et al. (2014). "Iridis-pi: a low-cost, compact demonstration cluster." Cluster Computing, vol. 17, no. 2, pp. 349–358.

[3]. Fung Po Tso., et al. (2013). "The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud

[4]. Computing Infrastructures." IEEE 33rd International Conference on Distributed Computing Systems Workshops, pp. 108–112.

[5]. Komninos, k., et al. (2019). "Performance of Raspberry Pi microclusters for Edge Machine Learning in Tourism." The 2019 European Conference on Ambient Intelligence.

[6]. Arunachalam, A & Andreasson, H (2021), "RaspberryPi-Arduino (RPA) powered smart mirrored and reconfigurable IoT facility for plant science research", Internet Technology Letters, vol. 1, no.1.

[7]. Bed, S., et al. (2019), "Smart Bed and Battery Driven Auto Warm Bolster based on Raspberry Pi and Arduino" Internetworking Indonesia Journal, vol.11 no.1. Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), pp. 31-36.

[8]. Mathur, A. P. & Tippenhauer, N. O (2016). "SWaT: a water treatment testbed for research and training on ICS security." 2016 International

[9]. Rubio-Hernan, J., el al (2017), 'Security of Cyber-Physical Systems: From Theory to Testbeds and Validation', in Security of Industrial Control Systems and Cyber-Physical Systems, Springer International Publishing, Cham, pp. 3–18.

[10]. Imperial College London, (2021), 'Body Sensor Network', viewed 5 May 2021, <https://www.imperial.ac.uk/hamlyn-centre/research/sensing/body-sensor-networks/>

[11]. Khan, RA & Pathan, A-SK (2018), 'The stare-of-the-art wireless bodyarea sensor network: A survey', International Journal of Distribution Sensor Networks, vol. 14, no. 4, p. 155014771876899–.

[12]. Djordjevi, S., et al., (1997), "Towards active c3i systems," in TELSIKS 1997.

[13]. Ahmad, H et al., (2021), "A Review on C3I Systems' Security: Vulnerabilities, Attacks, and Countermeasures" .

[14]. Lai, X., et al., (2013), "A survey of body sensor networks.", Sensors. Vol.13 (5), p.5406-5447.

[15]. Action J. M (2018), "Escalation through Entanglement: How the vulnerability of command-and-control system".

[16]. Shenoy, R.P., 1987. Command, control, communication and intelligence systems. A review. Defence science journal. Delhi, 37(4), pp.423-431.

[17]. Djordjevi, S., Kajan, E.K. and Mitrovi, D., Towards Active C3I Systems.

[18]. Shiner, D.V., 1989. The Military Approach to Competitive Advantage. Marketing Intelligence & Planning.

[19]. Mineo, J., 2004. Advanced Command, Control, Communications, & Intelligence (C3I) Systems Analysis and Trade-Offs. CACI TECHNOLOGIES INC CHANTILLY VA.

[20]. Feng, W., Ailiang, Z., Xiuluo, L., Jia, W., Ying, L., Jianjun, Q. and Min, W., 2020, December. Research on Space Launch Information Integration. In Journal of Physics: Conference Series (Vol. 1693, No. 1, p. 012052). IOP Publishing.

[21]. Ormrod, D., 2014, October. The coordination of cyber and kinetic deception for operational effect: attacking the C4ISR interface. In 2014 IEEE Military Communications Conference (pp. 117-122). IEEE.

[22]. Ormrod, D.G., 2014, June. A 'wicked problem'—Predicting sos behaviour in tactical land combat with compromised C4ISR. In 2014 9th International Conference on System of Systems Engineering (SOSE) (pp. 107-112). IEEE.

[23]. PMP, J.A.C. and CAP, C., 2021. Cybersecurity in the Healthcare Industry. The Journal of Medical Practice Management: MPM, 36(4), pp.229-231.

[24]. Agatino.., et al .., (2017) "A simulation study of C4I communication Network Under Cyber Attacks" Research Gate,

[25]. Ge M, Hong JB, Guttmann W, Kim DS. A framework for automating security analysis of the internet of things. Journal of Network and Computer Applications. 2017 Apr 1;83:12-27.

[26]. Poolsappasit N, Dewri R, Ray I. Dynamic security risk management using bayesian attack graphs. IEEE Transactions on Dependable and Secure Computing. 2011 Jun 30;9(1):61-74.

[27]. Alhomidi M, Reed M. Attack graph-based risk assessment and optimisation approach. International Journal of Network Security & Its Applications. 2014 May 1;6(3):31.

[28]. Arkin B, Stender S, McGraw G. Software penetration testing. IEEE Security & Privacy. 2005 Feb 14;3(1):84-7.

[29]. Thompson HH. Application penetration testing. IEEE Security & Privacy. 2005 Feb 14;3(1):66-9.

[30]. Al-Janabi S, Al-Shourbaji I, Shojafar M, Shamshirband S. Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications. Egyptian Informatics Journal. 2017 Jul 1;18(2):113-22.

[31]. Gope P, Hwang T. BSN-Care: A secure IoT-based modern healthcare system using body sensor network. IEEE sensors journal. 2015 Nov 20;16(5):1368-76.

[32]. Tan CC, Wang H, Zhong S, Li Q. Body sensor network security: an identity-based cryptography approach. InProceedings of the first ACM conference on Wireless network security 2008 Mar 31 (pp. 148-153).

[33]. Elhoseny M, Ramírez-González G, Abu-Elnasr OM, Shawkat SA, Arunkumar N, Farouk A. Secure medical data transmission model for IoT-based healthcare systems. Ieee Access. 2018 Mar 21;6:20596-608.

[34]. Prasanalakshmi B, Murugan K, Srinivasan K, Shridevi S, Shamsudheen S, Hu YC. Improved authentication and computation of medical data transmission in the secure IoT using hyperelliptic curve cryptography. The Journal of Supercomputing. 2021 May 26:1-8.

[35]. Zhu H, Yuan Y, Chen Y, Zha Y, Xi W, Jia B, Xin Y. A secure and efficient data integrity verification scheme for cloud-IoT based on short signature. IEEE Access. 2019 Jun 24;7:90036-44.

[36]. Liu C, Yang C, Zhang X, Chen J. External integrity verification for outsourced big data in cloud and IoT: A big picture. Future generation computer systems. 2015 Aug 1;49:58-67.

[37]. Kothmayr T, Schmitt C, Hu W, Brünig M, Carle G. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In37th Annual IEEE Conference on Local Computer Networks-Workshops 2012 Oct 22 (pp. 956-963). IEEE.

[38]. Al-Janabi S, Al-Shourbaji I, Shojafar M, Shamshirband S. Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications. Egyptian Informatics Journal. 2017 Jul 1;18(2):113-22.

[39]. Wadhawan Y, AlMajali A, Neuman C. A comprehensive analysis of smart grid systems against cyber-physical attacks. Electronics. 2018 Oct;7(10):249.

[40]. Ge M, Hong JB, Guttmann W, Kim DS. A framework for automating security analysis of the internet of things. Journal of Network and Computer Applications. 2017 Apr 1;83:12-27.

[41]. Eom T, Hong JB, An S, Park JS, Kim DS. A framework for real-time intrusion response in software defined networking using precomputed graphical security models. Security and Communication Networks. 2020 Feb 18;2020.

[42]. Welcome to DeepSource Documentation. DeepSource Docs. (n.d.) https://deepsource.io/docs/.

[43]. GitHub.2021. GitHub - ajinabraham/nodejsscan: nodejsscan is a static security code scanner for Node.js applications.. [online] Available at: https://github.com/ajinabraham/nodejsscan

[44]. "The ZAP Homepage." OWASP ZAP, 25 Aug. 2021, www.zaproxy.org/.

[45]. " Web Application Security Scanner Framework." Arachni, www.arachni-scanner.com/.

[46]. "Vulnerability Scanning Made Easy." *Crashtest Security*, 8 June 2021,crashtest-security.com/.

[47]. Plata IT, Calpito JL. Application Of Time-Based One Time Password (TOTP) Algorithm For Human Resource E-Leave Tracking Web App. International Journal of Scientific and Technology Research.

2020;9(3):4070-7.

[48]. www.twilio.com. (n.d.). What is a Time-based One-time Password (TOTP)? [online] Available at: https://www.twilio.com/docs/glossary/totp.

[49]. garbagecollected.org. (n.d.). How Google Authenticator Works. [online] Available at: https://garbagecollected.org/2014/09/14/how-google-authenticator-works/.