applied sciences

MDPI

Article

Cybersecurity and Medical Imaging: A Simulation-Based Approach to DICOM Communication

Stylianos Karagiannis [1,2,*,†], Emmanouil Magkos [2,†], Christoforos Ntantogian [2], Ricardo Cabecinha [3] and Theofanis Fotis [4]

[1] PDMFC, R. Fradesso da Silveira, 4-1B, 1300-609 Lisboa, Portugal
[2] Department of Informatics, Ionian University, Plateia Tsirigoti 7, 49100 Corfu, Greece; emagos@ionio.gr (E.M.); dadoyan@ionio.gr (C.N.)
[3] Hospital do Espírito Santo de Évora, EPE, Largo Senhor da Pobreza, 7000-811 Évora, Portugal; rjcabecinha@hevora.min-saude.pt
[4] Centre for Secure, Intelligent and Usable Systems (CSIUS), School of Sport & Health Sciences, University of Brighton, Brighton BN2 4AT, UK; t.fotis@brighton.ac.uk
* Correspondence: stylianos.karagiannis@pdmfc.com or skaragiannis@ionio.gr
† These authors contributed equally to this work.

Abstract: Medical imaging plays a crucial role in modern healthcare, providing essential information for accurate diagnosis and treatment planning. The Digital Imaging and Communications in Medicine (DICOM) standard has revolutionized the storage, transmission, and sharing of medical images and related data. Despite its advantages, implementation and deployment of the DICOM protocol often suffers from incomplete understanding, leading to vulnerabilities within the healthcare ecosystem. This research paper presents an implementation of DICOM communication and the development of a practical demonstration for simulation purposes The simulation can be used for conducting cybersecurity tests in the context of DICOM communication. Overall, the simulation provides a digital environment that can help in retrieving valuable insights into the practical aspects of DICOM communication and PACS integration, serving as a valuable resource for medical imaging professionals, researchers, and developers. These research results provide practical insights, and the DICOM simulation can be used in realistic contexts to showcase a variety of security scenarios.

Keywords: medical imaging; DICOM; PACS; eHealth; simulation

check for updates

Citation: Karagiannis, S.; Magkos, E.; Ntantogian, C.; Cabecinha, R.; Fotis, T. Cybersecurity and Medical Imaging: A Simulation-Based Approach to DICOM Communication. *Appl. Sci.* **2023**, *13*, 10072. https://doi.org/10.3390/app131810072

Academic Editor: Christos Bouras

Received: 26 July 2023
Revised: 31 August 2023
Accepted: 4 September 2023
Published: 6 September 2023

1. Introduction

Medical imaging is of the utmost importance in modern healthcare for accurate diagnosis and treatment planning [1–7]. DICOM, a widely used standard, has revolutionized the storage, transmission, and sharing of medical images and associated data [8–11]. DICOM stands for "Digital Imaging and Communications in Medicine". It is a widely used standard for the communication, storage, and management of medical images [12]. DICOM enables interoperability among various medical imaging devices and systems, ensuring that medical images and associated data can be easily exchanged and interpreted across different vendors and institutions. As part of the DICOM infrastructure, the Picture Archiving and Communication System (PACS) plays a key role in enabling the storage, retrieval, and distribution of medical images and associated information [13,14].

Past attacks targeting DICOM systems have exposed vulnerabilities in the security infrastructure of medical imaging technology [15]. These attacks highlight the need for heightened security measures around the safeguarding of sensitive patient data to maintain the integrity of medical images. For example, in 2017 researchers from Massachusetts General Hospital identified thousands of unprotected DICOM servers globally and hundreds in the United States, potentially revealing patient information [16]. Similarly, in 2018 a security researcher employed an internet scanning tool to access unprotected DICOM servers, and even produced a 3D model of a single individual's anatomy [17]. These incidents underscore the urgency of implementing robust security protocols, including encryption, authentication, network segmentation, and regular security audits, to mitigate risks and protect patient privacy within the DICOM ecosystem.

Despite the indispensability of DICOM and PACS, there is often a lack of comprehensive understanding and implementation in healthcare organizations [18,19]. This lack of comprehension and implementation can lead to various security challenges and vulnerabilities [20,21]. The existing literature highlights the security vulnerabilities in these systems, such as unencrypted communications, weak authentication mechanisms, and inadequate access controls [15,22–24]. Nevertheless, healthcare organizations often fail to implement robust security measures or conduct comprehensive security testing, leaving them vulnerable to cyberattacks [25,26].

This research paper proposes a DICOM simulation with the aim of improving understanding of the DICOM network protocol. The simulation serves as a practical demonstration that can enhance comprehension and implementation. By simulating data transfer using the DICOM protocol from a healthcare modality, the practical aspects of DICOM communication and PACS integration can be further investigated. The simulation involves the replication of the network layer for DICOM transfers and establishing the necessary connections to a PACS server.

1.1. Contribution

This research paper makes significant contributions to the field of medical imaging by providing a practical demonstration of DICOM data transfer in a medical imaging environment. The code has been uploaded to GitHub [27]. The key contributions of this research paper are as follows:

- We present a practical demonstration of data transfer in a medical imaging environment by simulating DICOM data transfer in a virtual environment. This demonstration provides an environment that can facilitate better understanding of DICOM communication and the integration of PACS in a realistic setting.
- We provide a step-by-step implementation and code snippets that serve as a valuable resource for education and learning in order to confer a better understanding of DICOM communication and PACS integration.
- The presented simulation offers a digital environment that enables hands-on experience of a realistic implementation for DICOM communication and PACS integration and allows interaction with the simulated environment.
- The simulated environment can be utilized as a valuable asset for security and privacy tests. By simulating DICOM transfers and PACS integration, common and critical vulnerabilities can be uncovered and comprehensive security testing conducted to address cybersecurity concerns in medical imaging systems.

1.2. Related Works

Several works have contributed to understanding DICOM communications and PACS integration in medical imaging. Zhou et al. [28] identified a lack of tools for PACS training and developed a Radiology Information System (RIS) simulator. However, advancements in medical imaging, DICOM communication, PACS integration, and cybersecurity have occurred subsequently. Coutinho et al. [21] introduced a cybersecurity methodology and tools for healthcare operational information systems with a focus on cybersecurity, for which they used Orthanc [29] and ONIS [30]. Other researchers have provided extensively details on the advantages of the Fast Healthcare Interoperability Resources (FHIR) protocol while outlining a user-friendly approach for individuals who are new to these concepts [31].

In the present research, a DICOM simulator was developed in a virtual lab context for use in conducting security tests. This study emphasizes in-depth analysis and learning aspects related to the fundamental attributes of the DICOM protocol, and the approach remains flexible for subsequent extensions, providing openings for further utilization by researchers according to their particular requirements.

Potter et al. [32] discussed the significance of the DICOM Validation Toolkit (DVTk) [33], which offers potential for individuals working with the DICOM standard. Other researchers [34] have utilized DVTk and highlighted the benefits of HIS, RIS, and PACS systems in healthcare management. Additionally, Oemig et al. [35] presented a collection of tools, including DVTk [33] and HAPI [36], to explore their development and integration with PACS. Mantri et al. [37] conducted a comparative study of DICOM API libraries, while NIST [38] identified software and managed risks within the PACS ecosystem. Mileva et al. [39] introduced covert channels for DICOM transport mechanisms, offering possibilities for covert communications, data exfiltration, and privacy leaking attacks.

Table 1 showcases the comprehensive differentiations and additions from the existing pertinent research. Within this study, a DICOM simulation and a topology are introduced to showcase the interaction between DICOM and PACS. This research is valuable to a broader audience interested in DICOM communication and for the practical application of DICOM simulations in real-world scenarios for learning and testing purposes.

Table 1. Comparison between related works and presented DICOM simulation.

Paper	Main Focus	Contribution of DICOM Simulation
[28]	Lack of tools for practical PACS training.	Comprehensive demonstration of DICOM data transfer within a medical imaging environment. It offers step-by-step implementation guidance and code snippets, enabling a better understanding of DICOM communication and integration with PACS systems.
[21]	Focus primarily on cybersecurity methodology.	The simulation serves as a controlled environment for conducting security assessments related to DICOM data transfer, offering a unique contribution by merging simulation with cybersecurity considerations.
[32]	Emphasis on the DICOM Validation Toolkit (DVTk).	Offers detailed Python library implementations that focus on fundamental DICOM communication concepts. Notably, it provides adaptable code that can be tailored to specific use-case requirements.
[35]	Exploration of DICOM software development and integration.	Enhances the practicality of training for medical imaging professionals, offering a contribution that bridges the gap between software development and medical imaging.
[37]	Comparative study of DICOM API libraries.	Practical understanding of data exchange, adding value by examining and by using DICOM APIs in a simulated environment.
[38]	Identification of risks within the PACS ecosystem.	Establish a practical baseline environment for comprehending and addressing security concerns in DICOM systems, offering valuable insights into risk identification.
[39]	Introduction of covert channels for DICOM transport mechanisms.	Extends the understanding of potential vulnerabilities and hidden pathways within DICOM communication.

1.3. Paper Structure

The rest of this paper is structured as follows. Section 3 presents the methodology employed to develop the DICOM simulation software. Section 4 provides details on the implementation and explanation of the Python code. Simulation results, validation, and analysis reports are provided in Section 5, along with a further explanation of the DICOM protocol. Finally, the paper concludes in Section 6 with a discussion of potential future research avenues.

2. Background

DICOM defines a set of entities representing different aspects of medical imaging and patient information [40]:

- Patients (P) encapsulates individual patient data, including attributes such as name, ID, and birthdate. A patient p can be associated with one or more studies ($P \to [S_1, S_2, \ldots]$).
- Study and Study Series (S and Se) includes related medical images and metadata, while Series groups images within a study based on common characteristics. A study s can include one or more series ($Se = [S_1, S_2, \ldots]$).
- Image (I) refers to a set of DICOM images I; Series Se groups related images ($Se \to [I_1, I_2, \ldots]$).
- Physician (Ph) represents healthcare professionals; images I are interpreted by a physician Ph ($Ph \to [I_1, I_2, \ldots]$).
- Modalities (M) refers to imaging-related equipment or healthcare modalities, including imaging devices, specifying manufacturer details and software version. Images I are generated by a specific modality M ($M \to [I_1, I_2, \ldots]$).

DICOM is designed for the management of medical images and associated data within a healthcare environment.

2.1. Picture Archiving and Communication System (PACS)

PACS is a crucial system utilized in the medical field for the storage, management, and distribution of medical images and associated patient information. It replaces traditional film-based systems by enabling healthcare providers to capture, store, retrieve, and view medical images electronically. With PACS, the cumbersome task of managing physical film records is replaced by a digital infrastructure that facilitates easy image access and promotes efficient collaboration among healthcare professionals. Orthanc [29,41], on the other hand, is an open-source implementation of a PACS server. It provides a lightweight, vendor-neutral, and standards-compliant solution for managing medical images and associated data. Orthanc serves as a software platform that can be utilized to establish a PACS server infrastructure. Its open-source nature offers flexibility and customization options, making it an appealing choice for institutions and developers seeking to create their own PACS environment.

2.2. Encoding and Compression Techniques in DICOM

The Transfer Syntax [42] in DICOM refers to the encoding and compression techniques used to represent and transmit DICOM data. It determines how DICOM objects, including images, are serialized into a byte stream that can be transmitted or stored. DICOM supports multiple Transfer Syntaxes to provide flexibility, interoperability, and efficient handling of various data formats. Commonly used Transfer Syntaxes in DICOM include the following [42]:

- **Implicit VR Little Endian:** a Transfer Syntax that uses a combination of implicit value representation encoding and little-endian byte ordering. It is widely used for uncompressed DICOM images, and supports a variety of data types. This is the default Transfer Syntax for most DICOM files.
- **Explicit VR Little Endian:** a Transfer Syntax that uses explicit VR encoding and little-endian byte ordering. Commonly used for uncompressed DICOM images, it provides explicit VR tags, which make it easier to parse and understand the data.
- **JPEG Lossless:** a Transfer Syntax that employs the Joint Photographic Experts Group (JPEG) Baseline algorithm for lossless compression of DICOM images, JPEG Lossless [12,42] is utilized when a balance between compression and preservation of image quality is required. It achieves significant compression ratios while maintaining the integrity of the image data. JPEG Lossless is particularly useful for storing or transmitting DICOM images where lossy compression is not acceptable.

- **JPEG Baseline:** a specific Transfer Syntax that utilizes the JPEG Baseline algorithm for compressing DICOM images, JPEG Baseline [42,43] is a lossy compression technique that achieves significant compression ratios while maintaining acceptable image quality for many medical imaging applications. JPEG Baseline is often used for transmitting DICOM images over networks with limited bandwidth or storage capacity, as it helps reduce the size of image data.

By supporting various Transfer Syntaxes, DICOM enables interoperability between different systems, ensuring that medical imaging data can be exchanged and interpreted correctly across diverse platforms and applications.

2.3. Service–Object Pair (SOP) Class and Transfer Syntax

SOP Classes [42] are service objects that encapsulate DICOM image files. An SOP Class defines a specific type of medical imaging or non-imaging service that can be performed on a DICOM object. It represents a combination of a service and an object on which the service can be performed. Popular SOP Classes include, among others [42]:

- **CT Image Storage (1.2.840.10008.5.1.4.1.1.2):** stores Computed Tomography (CT) images, which are used for diagnostic purposes and provide detailed cross-sectional views of the body.
- **MR Image Storage (1.2.840.10008.5.1.4.1.1.4):** stores Magnetic Resonance Imaging (MRI) images, which are used to visualize internal structures of the body using magnetic fields and radio waves, providing detailed anatomical information.
- **Ultrasound Image Storage (1.2.840.10008.5.1.4.1.1.6.1):** stores ultrasound images; ultrasound imaging uses high-frequency sound waves to generate images of organs and tissues in real-time, and is often used in obstetrics, cardiology, and other medical applications.
- **X-ray Radiography Image Storage (1.2.840.10008.5.1.4.1.1.1):** stores X-ray images; widely used for various medical purposes, X-rays provide two-dimensional images that can help diagnose bone fractures, detect abnormalities, and visualize internal structures.

The above SOP Classes define the necessary metadata, attributes, and operations required to handle and interpret images of their respective modalities.

2.4. Presentation Contexts

Presentation Contexts in DICOM define the combination of a specific SOP Class and one or more supported Transfer Syntaxes that an AE can handle for communication purposes. When two DICOM devices establish a connection or association, they negotiate the set of services and data formats that they can support. This negotiation is accomplished using Presentation Contexts. A Presentation Context consists of the following components [42]:

- **Abstract Syntax:** this identifies the class or type of DICOM object being transmitted or requested, and represents a specific SOP Class, such as CT Image Storage or MR Image Storage.
- **Transfer Syntaxes:** these define the encoding and compression techniques used to represent and transmit the DICOM data, and specify how the DICOM object is serialized into a byte stream. Multiple Transfer Syntaxes can be associated with a single Abstract Syntax, allowing for flexibility in data formats and compression options.

The process of association negotiation in the context of medical imaging involves the exchange of information between two devices. During this negotiation, both devices share a list of supported Presentation Contexts. Each Presentation Context is composed of an Abstract Syntax along with one or more Transfer Syntaxes that the device is capable of supporting for that specific Abstract Syntax.

The two devices then compare the lists of Presentation Contexts they have shared and work towards finding a mutually supported Presentation Context. When this mutual agreement is reached, it signifies that the devices can effectively communicate and conduct operations involving DICOM objects. This is done using the designated Abstract Syntax, while the agreed-upon Transfer Syntax is employed for the communication.

For instance, consider a scenario in which two devices establish a Presentation Context with an Abstract Syntax. This agreement indicates that the devices are now equipped to exchange and manipulate magnetic resonance image data using the JPEG Baseline compression algorithm. This enables seamless and efficient communication between the devices while ensuring compatibility in handling medical image information.

2.5. Application Entities (AEs)

The AE concept [42] is fundamental to Digital Imaging and DICOM standards. It represents a node or device in a network that communicates using DICOM protocols. An AE can be a piece of medical imaging equipment, such as a CT scanner, MRI machine, or PACS server. In the context of DICOM, an AE is identified by a unique AE title, which serves as its network address. The AE title is used to establish connections and enable communication between different entities in a DICOM network. Each AE is responsible for sending and receiving DICOM objects, including medical images, patient information, and related metadata.

AEs provide services such as querying/retrieval, storage, and printing of DICOM objects. They can act as a Service Class Provider (SCP), which receives requests for services, or a Service Class User (SCU) [42], which initiates requests and interacts with other endpoints. The roles of SCPs and SCUs can be performed by the same AE or by different endpoints depending on the system architecture and requirements. An Application Entity represents a DICOM node or device in the network. In the DICOM simulator, an AE is created using the AE class from *Pynetdicom*. The AE title is set to *"MODALITY"*, which identifies the simulated modality in the network.

3. Methodology

This section provides an in-depth explanation of the methodology used to implement the DICOM simulator in Python. It outlines the key steps involved in establishing communication with the PACS server, configuring the necessary libraries, and sending DICOM images. To understand the interactions between DICOM modalities and PACS servers, a mathematical model can be formulated to describe the association between DICOM modalities and PACS servers.

The association between a DICOM modality and a PACS server involves several parameters that define the communication and data exchange. Let us consider a DICOM modality M and a $PACS$ server. The relationship between M and $PACS$ can be represented using Equation (1):

$$A(M,S) = \{AE_M, AE_{PACS}, SOP_{\text{class}}, TS\} \tag{1}$$

where $A(M,S)$ represents the association between the modality M and PACS server $PACS$, AE_M is the Application Entity representing the DICOM modality in the network, AE_S is the Application Entity representing the PACS server in the network, SOP_{class} denotes the SOP Class specifying the type of medical image or information being exchanged, and TS is the transfer syntax that defines how the data are encoded and decoded during transmission.

Application Entities (AE) can be represented as nodes in a network graph. Each node corresponds to an AE, which can be either the DICOM modality or the PACS server. The communication between these nodes is facilitated by the DICOM protocol, as presented in Equation (2). Equation (2) describes nodes and edges as the elements of a graph, where AE_M and AE_S are nodes representing entities within the network and edges (AE_M, AE_S) signify a connection or relationship between these nodes:

$$\text{Nodes: } N = \{AE_M, AE_S\} \text{ and Edges: } E = \{(AE_M, AE_S)\} \tag{2}$$

where N is the set of nodes representing the DICOM modality and PACS server and E is the set of edges indicating the communication links between the modality and the server.

SOP classes define the type of medical image or object being exchanged, while the Transfer Syntax specifies how the data are encoded and compressed for transmission. The association of SOP classes and transfer syntax is presented in Equation (3):

$$A(SOP_{\text{class}}, TS) = \{SOP_{\text{class}}, TS\} \tag{3}$$

where A is a function that takes two parameters, namely, SOP_{class} and TS, while SOP_{class} represents a parameter related to the class of SOP in the context of DICOM, TS similarly represents the Transfer Syntax in DICOM, SOP_{class} represents the specific type of medical image or information, TS denotes the transfer syntax used for encoding and decoding, and the function A in Equation (3) takes two DICOM-related parameters and constructs a set containing their values.

DICOM communication often involves sensitive patient data, necessitating secure protocols [44]. Transport Layer Security (TLS) and Secure Sockets Layer (SSL) ensure encrypted data transmission and authentication, while Internet Protocol Security (IPsec) offers secure network-layer communication. DICOM Secure Communication Profiles define secure communication guidelines for enhanced data privacy and integrity in medical imaging networks. The choice of protocol depends on security needs, regulatory compliance, and infrastructure compatibility. The dependency graph (Figure 1) depicts the DICOM architecture followed in this research. These concepts, although only briefly summarized in this section, are explored in greater depth later on, shedding light on their profound influence within the field of medical imaging.

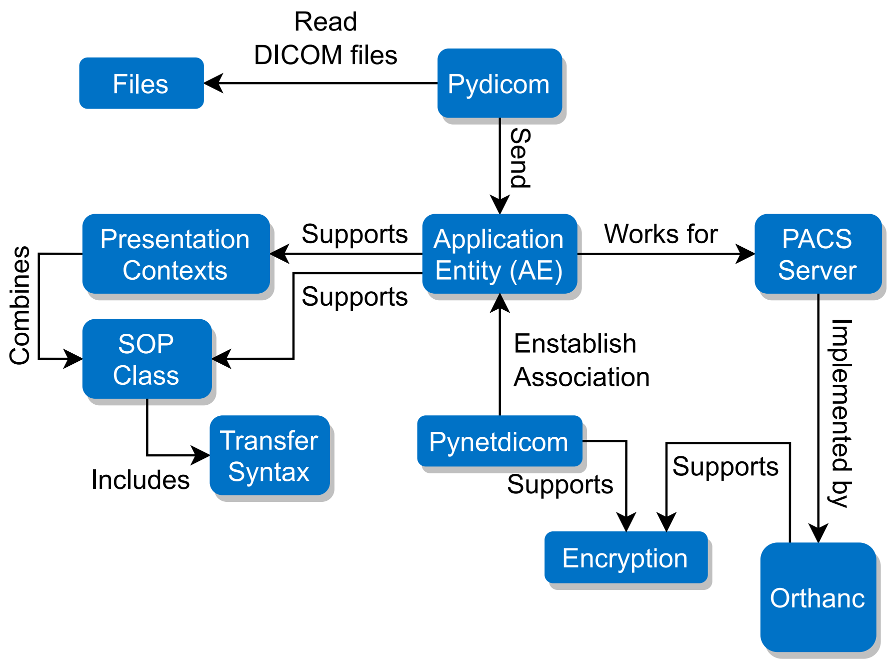

Figure 1. Dependency graph for DICOM/PACS communication.

At the core of DICOM is the AE, which designates network-connected devices such as imaging systems, workstations (Figure 1), or healthcare modalities. Playing a crucial role in simplifying accessibility for healthcare professionals, PACS servers are responsible for the storage, retrieval, and orchestration of medical images and their associated data.

The SOP Classes play a categorical role in delineating various types of medical studies or objects, each tailored to a specific imaging modality such as CT scanners or MRI machines. They detail the required attributes and structural elements crucial for accurate representation of data. Intimately connected to this are the concepts around Transfer Syntaxes, which encompass the encoding principles governing the compression and formatting of DICOM data. These principles ensure seamless interoperability across a diverse array of devices.

During communication, Presentation Contexts come into play by establishing agreements between devices regarding the supported SOP Classes and Transfer Syntaxes for the duration of the communication session. These fundamental concepts, intricate in their complexity and significant in their role, are poised for a more profound exploration later in this section.

As presented in Figure 1, to establish the association between the AE and the PACS, in this research we employed *Pynetdicom* [45]. *Pynetdicom* is a Python implementation of the DICOM networking protocol that enables communication between different DICOM entities. It provides the necessary functionalities to establish association, transfer DICOM data, and manage network connections. In terms of data security during transmission, both Orthanc and *Pynetdicom* support encryption. Encryption is a crucial aspect of securing sensitive medical data during transmission to ensure privacy and prevent unauthorized access. The use of encryption adds an extra layer of protection to the data being transferred between the AE and the PACS, reducing the risk of potential security breaches.

3.1. Pydicom: Modify and Write DICOM Files with Python

In this research, *Pydicom* [46] was used as a tool to read DICOM files and send them to the Application Entity (AE). The AE supports Presentation Contexts, which consist of SOP Classes and Transfer Syntaxes [39,42,46]. The integration as presented in Figure 1 focuses on the integration between the AE and the PACS server.

Pydicom is a widely-used Python library that offers a range of functionalities for working with DICOM files. It is specifically designed to handle DICOM data, allowing users to read, manipulate, and write DICOM files effectively. In the context of our presented DICOM simulator, *Pydicom* is utilized to read the DICOM files from the local folder and prepare them for transmission to the PACS server. The library provides the necessary tools and functions to parse the DICOM files, extracting relevant information and organizing it into a structured format.

One of the key functions provided by the *Pydicom* library is *dcmread()*. This function is used to read and parse DICOM files in Python. It accepts the path or a file-like object of the DICOM file as input and returns a Dataset object. The Dataset object represents the parsed data and contains various attributes and methods to access and manipulate the DICOM information. By utilizing the *dcmread()* function, the DICOM simulator can read the DICOM files and extract the required information, such as patient data, imaging parameters, and study details. This information can then be utilized to simulate the transmission of DICOM data to the PACS server.

3.2. Pynetdicom: A Python Implementation of the DICOM Networking Protocol

Pynetdicom is a Python library that implements the DICOM network protocol and enables communication between DICOM applications. It provides functionalities to establish associations between DICOM nodes, and facilitates the sending and receiving of DICOM data. In the context of the DICOM simulator, *Pynetdicom* is utilized to establish a connection with the PACS server and transmit DICOM images. The library offers an AE class, which is employed to create an Application Entity representing the local node or device in the simulator. To define the supported presentation contexts, which determine the combination of SOP Classes and Transfer Syntaxes that the AE can handle, the *supported_contexts* attribute of the AE class is set. This attribute specifies the supported DICOM services and protocols.

The *add_requested_context()* method is used to specify additional requested contexts, allowing the AE to indicate the SOP Classes and Transfer Syntaxes that it wants to use for communication with the remote DICOM entity. The *associate()* method establishes an association with the PACS server, enabling subsequent data exchange. This method initiates the DICOM association negotiation process, which involves negotiating supported presentation contexts, transferring the DICOM application context, and verifying the compatibility between the local and remote DICOM entities.

To transmit DICOM datasets to the connected PACS server, the *send_c_store()* method is utilized. This method ensures proper encoding and delivery of the DICOM data. It handles the DICOM C-STORE service, which is responsible for storing DICOM datasets on the PACS server. The *send_c_store()* method provides status codes to indicate the success or failure of the storage request. Finally, the *release()* method is used to terminate the association between DICOM entities. This method initiates a graceful closure of the communication session by sending release request messages to both the local and remote entities.

3.3. Association Establishment between AE and PACS

In the diagram provided in Figure 2, the deployment of a simulation on Host 1 is illustrated. The main process, labeled *"Association"*, is associated with the AE within the simulation. The AE serves as a representation of a device or node within the network, denoted as AE. Similarly, the Orthanc PACS server is represented as $PACS$. The initiation of communication between these entities is facilitated by the following method, establishing a connection for seamless data exchange, as defined in Equation (4).

$$association\_successful = associate(AE, PACS) \tag{4}$$

The success of this initiation is captured by the variable association_successful, indicating whether or not the association process was successful. Data transmission, which involves the exchange of medical images and related information, can be visually represented as a flow using directional arrows. This abstraction aids in understanding the dynamic exchange between different components of the system. The connectivity and integrity of the communication link are monitored using the function presented in Equation (5), allowing the verification outcome to be defined as Equation (6).

$$status = check\_status(modality, PACS) \tag{5}$$

$$association\_successful = associate(AE, PACS) \tag{6}$$

Data processing and release encompasses a sequence of operations applied to DICOM data, and fulfills specific tasks within the system. The established communication link between the AE and PACS is vital for seamless data exchange and successful execution of communication operations. Figure 2 visualizes the connection between the simulation and the PACS server, showcasing the flow of DICOM data and communication within the simulated environment. This connection enables the transmission of DICOM images and associated information from the simulation to the PACS server, mimicking real-world scenarios and facilitating the exploration and understanding of DICOM communication and PACS integration.

The simulation involves processing a dataset and initiating a request. The Association process reads the dataset and performs various operations. A status check is applied to verify the connection of the DICOM modality to the PACS server hosted on Host 2. If the connectivity is confirmed, the Association process sends the processed data to the PACS server. After the data transmission is successful, the Association process releases the connection. Overall, the diagram showcases the flow of data processing and communication between Host 1, the Association process, and the PACS server on Host 2.

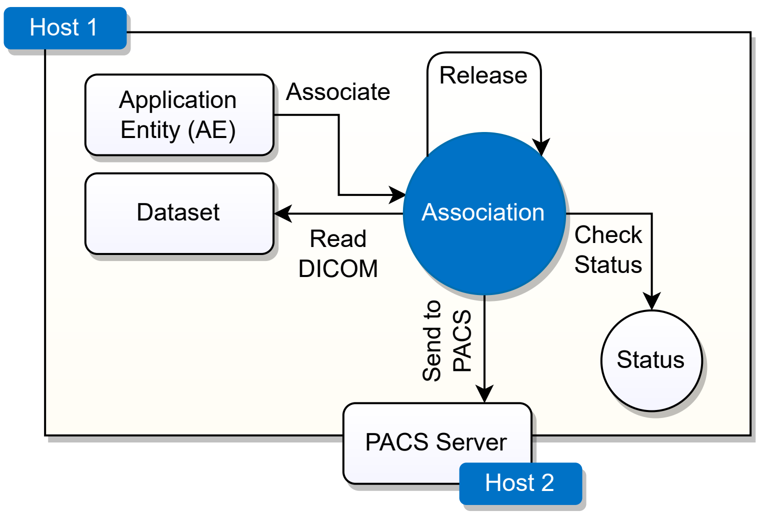

Figure 2. Establishment of the association between the AE and the PACS server.

To establish a connection with the PACS server, the AE utilizes the *associate()* method from the *Pynetdicom* library. This method serves as a way for the AE to initiate an association with the PACS server, enabling communication between them.

The *associate()* method requires several input parameters, including the IP address and port number of the PACS server, as well as the AE title associated with the PACS server. When this method is invoked, the AE triggers the association process. If the process is successful, it results in the creation of a communication link between the AE and the PACS server. When this association has been successfully established, the AE gains the ability to effectively interact with the PACS server.

This interaction encompasses various tasks, such as sending and retrieving medical images and the associated data. By leveraging the established connection, the AE can transmit DICOM data objects to the PACS server.

3.4. Encryption

The DICOM simulator provides an option to add a TLS layer for encryption, which is supported by Orthanc as well. Enabling TLS support in the DICOM simulator ensures that the transmission of DICOM data between the simulator and PACS server is encrypted, thereby enhancing data security. To enable TLS support in ORTHANC, the following steps should be completed:

- **Building a TLS Context.** A TLS context needs to be created within ORTHANC. The TLS context defines the configuration and settings for the TLS encryption.
- **Loading the Necessary Certificate and Private Key.** To establish secure communication, a valid TLS certificate, and its corresponding private key need to be provided. The TLS certificate is obtained from a trusted Certificate Authority (CA) and serves as proof of identity for the server.
- **Adding the Supported Context with the TLS Context to the AE.** In Orthanc, an AE represents a network entity that can communicate using the DICOM protocol. The TLS context is then added to the AE configuration in order to associate the TLS encryption capabilities with the AE.

By completing these steps and configuring the TLS settings correctly, the DICOM simulator with TLS support ensures that DICOM data transmitted between the simulator and the PACS server is encrypted. This encryption helps to protect the confidentiality and integrity of the data during transmission, mitigating the risk of unauthorized access or tampering.

4. Implementation

This section provides detailed insights into the implementation of the DICOM simulator, with a focus on the necessary configurations, setup of the environment, and the steps involved in establishing communication with the PACS server and sending DICOM images.

4.1. Setting Up the Python Libraries

Before implementing the DICOM simulator, it is essential to set up and install the required *Pydicom* and *Pynetdicom* libraries. The provided code is written in Python and demonstrates a function for sending DICOM files to a PACS server. Providing a pip requirements file or setting up an isolated environment can be streamlined using services such as Docker [47], which simplifies the environment setup process and ensures enhanced portability [27]. An extended explanation of the code follows in the code listing below.

In Listing 1, the necessary modules and classes required for the code are imported. *os* is imported to perform file operations, *dcmread* is imported from the *Pydicom* library for reading DICOM files, *JPEG Baseline* is imported from *pydicom.uid* for specifying the Transfer Syntax and AE, *StoragePresentationContexts* is imported from *Pynetdicom* for managing the application entity and storage presentation contexts, and *CTImageStorage* is imported from *pynetdicom.sop_class* for specifying the DICOM storage context.

Listing 1: Importing the libraries.

```
import os
from pydicom import dcmread
from pydicom.uid import JPEG Baseline
from pynetdicom import AE, StoragePresentationContexts
from pynetdicom.sop_class import CTImageStorage
```

The main function *send_dicom_to_pacs()* (Listing 2) receives as input four arguments: *dicom_folder* (the path to the folder containing DICOM files), *pacs_ip* (the IP address of the PACS server), *pacs_port* (the port number of the PACS server), and *pacs_ae_title* (the AE title of the PACS server). Inside the function, an instance of AE is created, with the AE title set to *"MODALITY"*. The supported storage presentation contexts are set to *StoragePresentationContexts*, and a requested context for X-ray Angiographic Image Storage is added with the UID *"1.2.840.10008.5.1.4.1.1.12.1"* and the transfer syntax *[JPEGBaseline]*.

Listing 2: The main function.

```
def send_dicom_to_pacs(dicom_folder,pacs_ip,pacs_port,pacs_ae_title)↩
    :
    # Create an Application Entity
    ae = AE(ae_title='MODALITY')
    ae.supported_contexts = StoragePresentationContexts

    # Add the requested transfer syntax for the X-ray Angiographic ↩
        Image Storage contextae.add_requested_context↩
        ('1.2.840.10008.5.1.4.1.1.12.1', [JPEGBaseline])
```

An association is established (Listing 3) with the PACS server using the associate method of the AE instance, binding the IP address of the PACS server, port number, and defined the AE title as arguments. Then, a message is printed indicating the successful connection.

Listing 3: Connecting to the PACS server.

```
    assoc = ae.associate(pacs_ip, pacs_port, ae_title=pacs_ae_title)
    if assoc.is_established:
        print('Connected to the PACS server.')
```

Inside the established association, the function, as presented in Listing 4, iterates over the files in the specified *dicom_folder*. If a file has the extension *.DCM*, then its full path is obtained using *os.path.join*. The DICOM file is read using *dcmread* and stored in the dataset variable. The *send_c_store* method of the association is called to send the DICOM dataset to the PACS server. The status of the storage request is checked, and appropriate messages are printed based on the success or failure of the operation.

Listing 4: Iterating over DICOM files in the folder.

```
        for filename in os.listdir(dicom_folder):
            if filename.endswith('.DCM'):
                dicom_file = os.path.join(dicom_folder, filename)
                # Read the DICOM file
                dataset = dcmread(dicom_file)
                # Send the DICOM image to the PACS server
                status = assoc.send_c_store(dataset)
                # Check the status of the storage request
                if status:
                    print(f'Successfully sent {dicom_file} to the ↩
                        PACS server.')
                else:
                    print(f'Failed to send {dicom_file} to the PACS↩
                         server.')
```

After processing all the DICOM files, the association is released (Listing 5) using the release method and a message is printed indicating the disconnection from the PACS server. If the initial association could not be established, a message is printed indicating the failure to connect to the PACS server.

Listing 5: Releasing the association.

```
        # if assoc.is_established=true then Release the association
        assoc.release()
        print('Disconnected from the PACS server.')
    else:
        print('Failed to connect to the PACS server.')
# The send_dicom_to_pacs main function finishes
```

Listing 6 demonstrates how to use the *send_dicom_to_pacs* function. The *dicom_folder*, *pacs_ip*, *pacs_port*, and *pacs_ae_title* variables are defined with sample values. These should be replaced with the actual values corresponding to the specific setup instance. The function is then called with these arguments to initiate the DICOM transfer process. The commented-out lines related to the TLS layer and certificate paths, indicating that while TLS encryption is not currently being used in this code, it can be implemented by uncommenting and modifying these lines.

Listing 6: Usage Example.

```python
dicom_folder = r'C:\Users\bloodraven\Desktop\02_Projects\DT\dicom'
pacs_ip = '192.168.0.162'  # Replace with the actual IP address
pacs_port = 4242  # Replace with the actual port number
pacs_ae_title = 'ORTHANC'  # Replace with the actual AE name/title

# Call the main function send_dicom_to_pacs
send_dicom_to_pacs(dicom_folder, pacs_ip, pacs_port, pacs_ae_title)
```

5. Simulation Results and Analysis

The diagram in Figure 3 represents the simulated environment and the integration between the DICOM simulation and the PACS server. It comprises three key components: a Windows 10 machine on which the DICOM simulation is executed, an Ubuntu Virtual Machine (VM) with a Docker container as a deployment for Orthanc, and a dedicated KALI Linux machine to enable packet sniffing. The simulation and the deployed topology can be used as a sandbox for DICOM communication security scenarios.

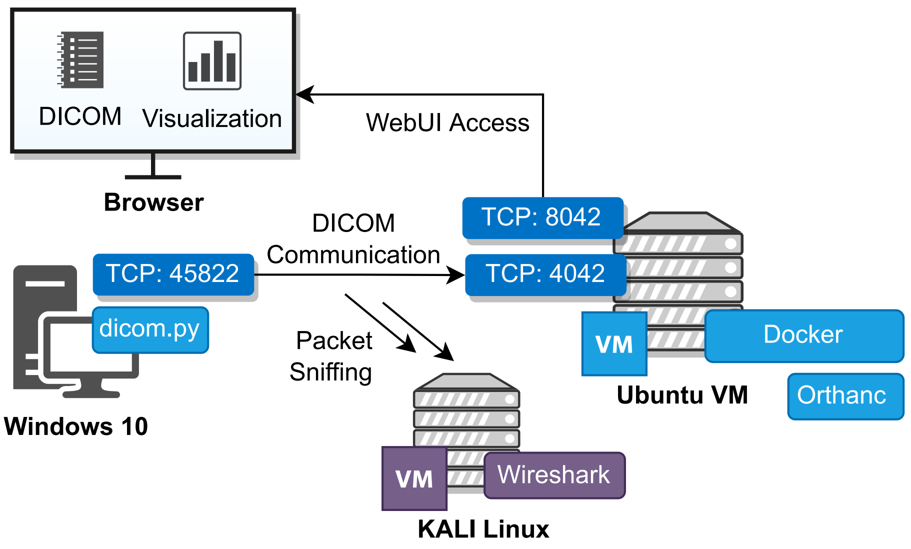

Figure 3. The DICOM/PACS simulated environment.

To ensure device identification within the local network (Figure 3), specific IP addresses are assigned for the VMs of Windows 10, Ubuntu VM, and KALI Linux. The reserved IP addresses facilitate communication and data exchange among the different components of the simulated environment.

5.1. Setting Up Orthanc to Accept Data from the Modality Simulator

Within the configuration of Orthanc (*/etc/orthahnc/orthanc.json*), the *"DicomModalities"* section should be changed to accept data from the specific modality. The example contains three parameters [41]:

- Application Entity Title (AET): this parameter represents the unique identifier of the remote modality. It cannot exceed 16 characters in length. In the example, *"MODALITY"* is used as the AET.
- Remote Network Address: this parameter specifies the IP address or hostname of the remote modality. In the example, *"192.168.0.105"* (Windows 10) is used as the address.
- TCP port number: this parameter defines the port number on which the DICOM protocol is running on the remote modality. The default port for DICOM is 104.

The configuration (Listing 7) suggests that multiple DICOM modalities can be listed under the *"DicomModalities"* section [41], each one identified by a unique AET and associated with a specific network address and port number for communication.

Listing 7: Configuration options for Orthanc: adding the DICOM modalities.

```
``DicomModalities'' : {
  /**
   * The first parameter is the
   * AET of the remote modality (cannot be longer than 16
   * characters), the second one is the remote network address,
   * and the third one is the TCP port number corresponding
   * to the DICOM protocol on the remote modality (usually 104).
   **/
  // ``sample'' : [ ``STORESCP'', ``127.0.0.1'', 2000 ]
  ``modality'' : [ ``MODALITY'', ``192.168.0.105'', 104 ]
}
```

5.2. Enabling SSL/TLS

Enabling TLS in Orthanc (Listing 8) involves several steps, as outlined below [41]:

1. SSL enabled: the *"SslEnabled"* option in the configuration file should be set to true. By default, this option is initially set to false; changing it to true is necessary to enable SSL/TLS encryption.
2. Specify the path to the SSL certificate: the value of *"SslCertificate"* needs to be updated with the file path of the SSL certificate. It is important that the certificate file is in the PEM format and contains both the certificate and the private key.
3. Set the minimum accepted SSL protocol version (optional): the introduction of the *"ssl_protocol_version"* option in Orthanc 1.8.2 allows for the specification of the minimum accepted SSL protocol version. This option can be added if desired, with the default requirement being for SSL 1.2. The available protocol versions and their corresponding values can be found in the documentation or configuration file.

Listing 8: SSL/TLS options for Orthanc.

```
/**
   * Security-related options for the HTTP server
   **/
  // Whether remote hosts can connect to the HTTP server
  ``RemoteAccessAllowed'' : true,
  // Whether or not SSL is enabled
  ``SslEnabled'' : false,
  // Path to the SSL certificate used by the HTTP server. The file
  // must be stored in the PEM format, and must contain both the
  // certificate and the private key. This option is only ↩
      meaningful
  // if ``SslEnabled'' is true.
  ``SslCertificate'' : ``certificate.pem'',
  // Sets the minimum accepted SSL protocol version
  // (cf. ``ssl_protocol_version'' option of civetweb). By default,
  // require SSL 1.2. This option is only meaningful if ``↩
      SslEnabled''
  // is true. (new in Orthanc 1.8.2)
**/
```

When these changes have been made, the configuration file must be saved and the Orthanc server restarted to implement the modifications. As a result, the HTTP server of Orthanc becomes accessible over a secure TLS connection.

5.3. Validation Tests: Scenario Test Execution

Two distinct scenarios are illustrated in Listing 9 from the script outputs. The scenarios depict different outcomes related to the transfer of data to a PACS server for validation purposes related to the present research.

Listing 9: Validation testing.

```
# Scenario 1: Successful transfer - Output of the successful ↩
    connection
C:\Users\bloodraven\Desktop\02_Projects\DT>python dicom.py
Connected to the PACS server.
Successfully sent C:\Users\bloodraven\Desktop\02_Projects\DT\dicom↩
    \0002.DCM to the PACS server.
Disconnected from the PACS server.

# Scenario 2: Incorrect IP or Port Number - Output of failed ↩
    connection
C:\Users\bloodraven\Desktop\02_Projects\DT>python dicom.py
Failed to connect to the PACS server.
```

Upon initiating the script execution (Listing 9), the command prompt confirms a successful connection establishment with the PACS server. This connection allows subsequent data transmission. Specifically, the output specifies that the file *"0002.DCM"* is located at the specified path. Similarly, in Scenario 2 the output reveals that there is an inability to establish the intended connection, indicating that an erroneous IP address or port number was specified in the script or that other network-related complications impeded the connection process.

5.4. Network Traffic Analysis: TCP/DICOM Communication Flow

This section provides the mathematical model that describes the progression of the TCP/DICOM communication flow of the DICOM Communication. Moreover, Listings 10–13 show the progression of the TCP/DICOM communication flow between the DICOM modality and the PACS server.

Listing 10: Network flow: association initialization.

```
# Time|Message Type|Information|(Port Numbers)
T_{1} 7.633204882|A-ASSOCIATE request|DICOM: A-ASSOCIATE request ↩
    MODALITY --> ORTHANC|(55822) --> (4242)
T_{2} 7.633597464|A-ASSOCIATE accept|DICOM: A-ASSOCIATE accept ↩
    MODALITY <-- ORTHANC|(55822) <-- (4242)
```

Listing 11: TCP segments in between the DICOM communication: data transmission.

```
# Time|Message Type|Information|(Port Numbers)
T_{3,4,...} **Multiple TCP flows [TCP segment of a reassembled PDU]
```

Listing 12: Data reassembly, storage, and command execution.

```
# Time|Message Type|Information|(Port Numbers)
**Multiple TCP flows [TCP segment of a reassembled PDU]

T_{6} 7.671284786|P-DATA, C-STORE-RQ I|DICOM: P-DATA, C-STORE-RQ ID↩
    =1|(55822) --> (4242)
T_{7} 7.671830832|P-DATA, X-ray Angiog|DICOM: P-DATA, X-ray ↩
    Angiographic Image Storage Fragment (reassembled in #1394)↩
    |(55822) --> (4242)
T_{8} 7.684974645|P-DATA, X-ray Angiog|DICOM: P-DATA, X-ray ↩
    Angiographic Image Storage|(55822) -->  (4242)
T_{9} 7.750206628|P-DATA, Command ID=1|DICOM: P-DATA, Command ID=1 (↩
    Success)|(55822) <-- (4242)
```

Listing 13: Association termination.

```
# Time|Message Type|Information|(Port Numbers)
T_{10} 7.755639129|A-RELEASE request|DICOM: A-RELEASE request↩
    |(55822) --> (4242)
T_{11} 7.755860145|A-RELEASE response|DICOM: A-RELEASE response↩
    |(55822) <-- (4242)
```

5.4.1. Association Initialization

At the start of the communication, the `MODALITY` entity initiates the process by sending an `A-ASSOCIATE request` to `Orthanc` indicating its intention to establish a DICOM association. This request occurs at time T_1, and is represented as Equation (7).

$$MODALITY \xrightarrow{\text{A-ASSOCIATE request}} PACS \quad \text{at time } T_1 \tag{7}$$

In response to the request, `Orthanc` acknowledges the establishment of the association by sending an `A-ASSOCIATE accept` message back to `MODALITY`. This acceptance message occurs at time T_2, and is defined by Equation (8).

$$MODALITY \xleftarrow{\text{A-ASSOCIATE accept}} PACS \quad \text{at time } T_2 \tag{8}$$

This includes the establishment of the association, data transmission, successful command execution, and the termination of the connection (Listing 10).

A-ASSOCIATE request. The A-ASSOCIATE request is the initial step in establishing a DICOM association between the MODALITY entity and the Orthanc entity. It is sent by the MODALITY device/system to the Orthanc device/system. The request contains information about the supported DICOM services, transfer syntaxes, and other negotiation parameters.

A-ASSOCIATE accept. The A-ASSOCIATE accept message is sent by Orthanc in response to the A-ASSOCIATE request from MODALITY. It indicates that Orthanc accepts the association. This message contains negotiated parameters, such as the agreed-upon DICOM services, transfer syntaxes, and other configuration details.

5.4.2. Data Transmission

Throughout the communication session, `MODALITY` sends multiple TCP segments, which are acknowledgments (ACKs), to `PACS`. These segments are transmitted at different times, denoted as T_3, T_4, etc., representing a series of communication events. as presented in Equation (9):

$$MODALITY \xrightarrow{\text{TCP segments}} PACS \quad \text{at times } T_3, T_4, \ldots, \tag{9}$$

where each T_i corresponds to a specific ACK transmitted by `MODALITY`. A sample of the data transmission in TCP segments is presented in Listing 11.

Multiple TCP flows in between (TCP segment of a reassembled PDU). The TCP segments (Listing 11) are acknowledgments (ACK) sent from the AE, namely, MODALITY to Orthanc acknowledging the successful reception of a previous PDU fragment. The *"S"* indicates a TCP segment of a reassembled PDU, suggesting that this is part of the reassembly process involving the fragmented data mentioned previously.

5.4.3. Data Reassembly and Storage

As the data transmission continues, `MODALITY` sends `P-DATA` messages, specifically, `C-STORE-RQ` (C-STORE request) messages, to `Orthanc` for the purpose of storing DICOM objects. `Orthanc` receives and reassembles the data fragments. The reassembly events occur at times T_6, T_7, T_8, etc., forming the sequence of reception events expressed in Equation (10):

$$PACS \xrightarrow{\text{Reassembled fragments}} MODALITY \quad \text{at times } T_6, T_7, T_8, \ldots, \tag{10}$$

where each T_i indicates the reception and reassembly of data fragments by `Orthanc`. Then, `MODALITY` notifies the successful execution of a command by transmitting a `P-DATA` message containing the relevant information, including `Command ID=1 (Success)`, to `PACS`. This message is received by `PACS` at time T_9, as presented in Equation (11).

$$MODALITY \xleftarrow{\text{P-DATA, Command ID=1 (Success)}} PACS \quad \text{at time } T_9 \tag{11}$$

P-DATA, C-STORE-RQ I. The P-DATA message is used to transfer DICOM data between the MODALITY and Orthanc entities; in this case, it is a C-STORE request (C-STORE-RQ) sent by MODALITY to Orthanc. Here, *"ID=1"* indicates that this is the first C-STORE request in the communication session. This request (Listing 12) includes information about the data to be stored, such as the DICOM object or image. A C-STORE request (C-STORE-RQ) is a DICOM message used to request the storage of DICOM objects, such as medical images, in a destination device or system. It contains information about the object to be stored, including its unique identifier and transfer syntax; C-STORE-RQ enables the standardized and reliable transfer of DICOM objects in healthcare environments.

P-DATA, X-ray Angiog (reassembled fragment). This message (Listing 12) represents a fragment of the X-ray Angiographic Image Storage data. The message continues as long as the transmission of the X-ray Angiographic Image Storage data continues, and is reassembled in a final P-DATA network packet. This signifies that the data transmission is ongoing and more fragments are being sent to complete the image or object.

PDATA, Command ID=1 (Success). This message (Listing 12) represents a complete DICOM P-DATA message containing the entire X-ray Angiographic Image Storage. It indicates that the data transmission for this particular object or image is complete.

5.4.4. Association Termination

To conclude the communication, `MODALITY` initiates the termination process by sending an `A-RELEASE request` to `Orthanc` at time T_{10}, as expressed in Equation (12).

$$MODALITY \xrightarrow{\text{A-RELEASE request}} PACS \quad \text{at time } T_{10} \tag{12}$$

`PACS` responds by acknowledging the request with an `A-RELEASE response`, which is received by `MODALITY` at time T_{11}, as presented in Equation (13).

$$MODALITY \xleftarrow{\text{A-RELEASE response}} PACS \quad \text{at time } T_{11} \tag{13}$$

A-RELEASE Request. The A-RELEASE request is sent by MODALITY to initiate the termination (Listing 13) of the DICOM association between MODALITY and Orthanc. This indicates the intention to close the connection gracefully. A-RELEASE is a message used in the DICOM protocol to initiate the termination of a DICOM association between two devices or systems. The A-RELEASE message is sent by one entity to the other to indicate an intention to gracefully close the connection. Upon receiving the A-RELEASE message, the recipient entity acknowledges the request by sending an A-RELEASE response. The A-RELEASE process allows for the orderly termination of the DICOM association, ensuring the proper release of resources and connections associated with the session. This is an essential step in maintaining the integrity and efficiency of DICOM communication between devices or systems in healthcare environments.

A-RELEASE Response. The A-RELEASE response is sent by Orthanc in response to the A-RELEASE request from MODALITY. It acknowledges the termination request and signifies the successful termination of the DICOM association.

5.5. Discussion

The simulated environment for DICOM communication serves as a platform for testing and addressing essential aspects of network security and related cybersecurity issues. The environment is designed to replicate DICOM communication scenarios, and facilitates a comprehensive understanding of potential threats as well as the actions needed to mitigate them. DICOM communication is susceptible to several cyberthreats, and the presented simulation can assist in conducting security assessments as well as in implementation and testing of mitigation actions in an isolated and realistic environment. Cyberthreats selected as important can be replicated within the simulation, including the following:

1. Insecure communication: without encryption, sensitive DICOM data transmitted over the network are vulnerable to eavesdropping and unauthorized access.
2. Weak authentication: insufficient user authentication mechanisms can lead to unauthorized users gaining access to DICOM systems, resulting in compromised data integrity.
3. Outdated software: running outdated DICOM software components can expose vulnerabilities that malicious actors can exploit for unauthorized access or data manipulation.
4. Insufficient training: lack of user awareness and training can result in successful social engineering attacks, leading to unauthorized data access.
5. Unprotected networks: failure to segment and secure the network exposes DICOM systems to potential breaches from unauthorized network traffic.
6. Data manipulation: without proper integrity checks, attackers might manipulate DICOM data during transmission, leading to compromised diagnoses or treatments.
7. Unauthorized access: poorly managed access control can result in unauthorized users gaining entry to DICOM resources.
8. Lack of intrusion detection: absence of intrusion detection mechanisms makes it challenging to detect and respond to unauthorized network activities.
9. Auditing and weakness identification (M1047): it is crucial to perform regular audits and scans on DICOM systems in order to identify weaknesses that could compromise communication security.

Similarly, OWASP has published a comprehensive guide to the secure deployment of medical devices [48]. Furthermore, a CSRF vulnerability for ORTHANC was presented in [49]. Similarly, Nmap Scripting Engine (NSE) scripts have been published related to DICOM [50,51]. These scripts aim to perform brute force attacks on the AET of a DICOM server.

Table 2 outlines a set of MITRE mitigation actions along with their relevance to DICOM communication security. Each MITRE ID corresponds to a specific mitigation action designed to address a particular security concern in the context of DICOM communication.

In regard to the OWASP guide and the vulnerabilities and threats mentioned above, Table 2 lists MITRE mitigation actions that correspond to the above-mentioned security issues. According to MITRE [52], mitigations embody security principles and categories of technologies that can be employed to thwart the successful execution of a technique or subtechnique. As such, the information in Table 2 can be used to identify other potential cyberthreats by looking to the relevant MITRE ATT&CK Navigator Layer according to the particular mitigation measures.

The DICOM simulation featured in this research provides valuable insights into addressing potential cybersecurity vulnerabilities and implementing corresponding mitigation strategies. However, it is crucial to acknowledge certain inherent limitations and constraints associated with the simulation's findings. The simulation's results are contingent upon a series of assumptions and simplifications made during the modeling phase. These assumptions might not always align with the intricacies of real-world scenarios, introducing the potential for variations that could impact the accuracy of the conclusions derived from the simulation.

Table 2. MITRE Mitigation Actions and Relevance to DICOM.

MITRE ID	Mitigation Action	Realization and Relevance to DICOM
M1047	Perform audits or scans to identify weaknesses.	Regularly audit DICOM systems, configurations, and permissions to identify potential vulnerabilities that could impact DICOM communication.
M0804	Require user authentication.	Implement strong user authentication mechanisms before granting access to DICOM data or devices to prevent unauthorized access or data manipulation.
M1041	Encrypt Sensitive Information.	Apply strong encryption to sensitive DICOM data to ensure its confidentiality during transmission and storage.
M1051	Update Software.	Regularly update DICOM software components to patch vulnerabilities and reduce the risk of exploitation during communication.
M1017	User Training.	Train users involved in DICOM communication to recognize and respond to potential access or manipulation attempts, such as spearphishing or social engineering attacks.
M0930	Network Segmentation.	Segment the network to isolate critical DICOM systems from other network segments, preventing unauthorized access and protecting DICOM communication from potential threats.
M0931	Network Intrusion Prevention.	Use intrusion detection techniques that don't disrupt real-time DICOM communication protocols to identify and block unauthorized or malicious network traffic.
M1037	Limit Access to Resources Over Network.	Restrict network access to DICOM file shares and remote access points, reducing exposure to potential attackers and ensuring secure communication.
M1035	Filter Network Traffic.	Utilize network appliances and endpoint software to filter incoming and outgoing DICOM network traffic, blocking potentially harmful or unauthorized data flows.

Furthermore, the simulation did not involve the actual execution of cyberattacks or vulnerability assessments. This omission is attributed to the reliance on specific deployment parameters, making it difficult to accurately replicate authentic cyberattack scenarios. Consequently, the simulation's representations of real-world cyberattacks may lack full complexity and nuanced intricacies, potentially influencing the realism of its findings.

Moreover, the simulation's scope might not encompass the entirety of potential cyberthreats. The rapidly evolving landscape of cybersecurity introduces the possibility of emerging and sophisticated threats that were not explicitly accounted for in the simulation. This limitation is a result of the inherent challenge of predicting and accommodating the myriad attack vectors that can emerge over time.

6. Conclusions

In this research, a DICOM simulator incorporating advanced features of virtualized environments and cloud computing was developed within a virtual lab context. The focus of this study was on conducting in-depth analyses and facilitating learning related to the fundamental attributes of the DICOM protocol. The DICOM simulator provides a flexible approach that can be further developed and customized to meet the specific needs of researchers. By utilizing cloud computing capabilities, the simulator can emulate complex scenarios and facilitate interoperability between different systems and devices. This enables researchers to explore and experiment with various aspects of DICOM communication and PACS integration. Moreover, the simulation has the capability to produce network traffic through the utilization of the DICOM protocol. This traffic generation results in the creation of authentic datasets, which can be effectively employed for the advancement of machine learning, deep learning technologies, and anomaly detection techniques.

The presented topology and deployment can serve as a platform for conducting security control testing, allowing researchers to evaluate the robustness of security measures, identify vulnerabilities, and test different security configurations in a controlled and safe environment. In this way, it provides researchers with an opportunity to gain a comprehensive understanding of DICOM communication and its practical implementation.

Future Work

A significant future milestone involves fostering enhanced interoperability between PACS while encompassing other medical imaging devices or formats. This expanded interoperability can facilitate seamless communication and data exchange across various SOP classes, extending the simulation support to encompass a wider range of scenarios and medical imaging modalities. Moreover, an area of promising exploration is the extension of research in the development and utilization of digital twins for medical imaging modalities. The development and utilization of digital twins for replicating healthcare or medical modalities can unlock a multitude of possibilities.

Finally, an important focus of future research involves addressing cyberthreats, conducting security assessments, creating a readily deployable topology (cyber range), and developing adversary emulation strategies aligned with the threats mentioned in this research.

Author Contributions: Conceptualisation, S.K. and R.C.; methodology, S.K.; software, S.K.; validation, S.K. and R.C.; formal analysis, E.M. and T.F.; writing—original draft preparation, S.K.; writing—review and editing E.M., C.N. and R.C.; supervision, E.M. and T.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements No. 883273 (AI4HEALTHSEC) and No. 101070450 (AI4CYBER).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Sample logs can be obtained by contacting the authors.

Acknowledgments: The authors would like to acknowledge support and funding provided by the projects AI4HEALTHSEC and AI4CYBER.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bercovich, E.; Javitt, M.C. Medical imaging: From roentgen to the digital revolution, and beyond. *Rambam Maimonides Med. J.* **2018**, *9*, e0034. [CrossRef] [PubMed]
2. Dash, S.; Shakyawar, S.K.; Sharma, M.; Kaushik, S. Big data in healthcare: Management, analysis and future prospects. *J. Big Data* **2019**, *6*, 54. [CrossRef]
3. Osteaux, M.; Van den Broeck, R.; Verhelle, F.; de Mey, J. Picture archiving and communication system (PACS): Medical perspectives. *J. Belg. Radiol.* **1997**, *80*, 128–132. [PubMed]

4. Nichols, J.A.; Herbert Chan, H.W.; Baker, M.A. Machine learning: Applications of artificial intelligence to imaging and diagnosis. *Biophys. Rev.* **2019**, *11*, 111–118. [CrossRef] [PubMed]
5. Latreche, A.; Kelaiaia, R.; Chemori, A.; Kerboua, A. Reliability and validity analysis of MediaPipe-based measurement system for some human rehabilitation motions. *Measurement* **2023**, *214*, 112826. [CrossRef]
6. Latreche, A.; Kelaiaia, R.; Chemori, A.; Kerboua, A. A New Home-Based Upper-and Lower-Limb Telerehabilitation Platform with Experimental Validation. *Arab. J. Sci. Eng.* **2023**, *48*, 10825–10840. [CrossRef] [PubMed]
7. Hussain, T.; Nguyen, Q.T. Molecular imaging for cancer diagnosis and surgery. *Adv. Drug Deliv. Rev.* **2014**, *66*, 90–100. [CrossRef] [PubMed]
8. Bidgood, W.D., Jr.; Horii, S.C.; Prior, F.W.; Van Syckle, D.E. Understanding and using DICOM, the data interchange standard for biomedical imaging. *J. Am. Med. Inform. Assoc.* **1997**, *4*, 199–212. [CrossRef] [PubMed]
9. Lebre, R.; Costa, C. An Efficient and Reliable Architecture for Distributing Medical Imaging Data. In Proceedings of the 2021 International Conference on e-Health and Bioengineering (EHB), Iasi, Romania, 18–19 November 2021; pp. 1–4. [CrossRef]
10. Supriya, S.; Padaki, S. Data security and privacy challenges in adopting solutions for IOT. In Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 15–18 December 2016; pp. 410–415. [CrossRef]
11. Jeyakumar, V.; Abirami, K.R.; Saraswathi, S.; Kumaran, R.S.; Marthi, G. Secure medical image storage and retrieval for Internet of Medical Imaging Things using blockchain-enabled edge computing. In *Intelligent Edge Computing for Cyber Physical Applications*; Elsevier: Amsterdam, The Netherlands, 2023; pp. 85–110. [CrossRef]
12. Mustra, M.; Delac, K.; Grgic, M. Overview of the DICOM standard. In Proceedings of the 2008 50th International Symposium ELMAR, Zadar, Croatia, 10–12 September 2008; Volume 1, pp. 39–44.
13. Fennell, N.; Ralston, M.D.; Coleman, R.M. PACS and Other Image Management Systems. In *Practical Imaging Informatics: Foundations and Applications for Medical Imaging*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 131–142. [CrossRef]
14. Cooke, R.E., Jr.; Gaeta, M.G.; Kaufman, D.M.; Henrici, J.G. Picture Archiving and Communication System. U.S. Patent 6,574,629, 3 June 2003.
15. Desjardins, B.; Mirsky, Y.; Ortiz, M.P.; Glozman, Z.; Tarbox, L.; Horn, R.; Horii, S.C. DICOM images have been hacked! Now what? *Am. J. Roentgenol.* **2020**, *214*, 727–735. [CrossRef] [PubMed]
16. Stites, M.; Pianykh, O.S. How secure is your radiology department? Mapping digital radiology adoption and security worldwide. *AJR Am. J. Roentgenol.* **2016**, *206*, 797–804. [CrossRef] [PubMed]
17. Beek, C. Mcafee Researchers Find Poor Security Exposes Medical Data to Cybercriminals. McAfee Blogs. 2018. Available online: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-researchers-find-poor-security-exposes-medical-data-to-cybercriminals/ (accessed on 5 September 2023).
18. Oliveira, P.A.M.; Junior, E.C.; Andrade, R.M.; Santos, I.S.; Neto, P.A.S. Ten Years of eHealth Discussions on Stack Overflow. 2022.
19. Cronin, S.; Kane, B.; Doherty, G. A qualitative analysis of the needs and experiences of hospital-based clinicians when accessing medical imaging. *J. Digit. Imaging* **2021**, *34*, 385–396. [CrossRef] [PubMed]
20. Eichelberg, M.; Kleber, K.; Kämmerer, M. Cybersecurity in PACS and medical imaging: An overview. *J. Digit. Imaging* **2020**, *33*, 1527–1542. [CrossRef] [PubMed]
21. Coutinho, B.; Ferreira, J.; Yevseyeva, I.; Basto-Fernandes, V. Integrated cybersecurity methodology and supporting tools for healthcare operational information systems. *Comput. Secur.* **2023**, *129*, 103189. [CrossRef]
22. Kumar, P.; Singh, A.; Sengupta, A. Securing Cyber-Resilience in Healthcare Sector. In *Cyber Security in Intelligent Computing and Communications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 211–226. [CrossRef]
23. Wang, Z.; Li, Q.; Wang, Y.; Liu, B.; Zhang, J.; Liu, Q. Medical protocol security: DICOM vulnerability mining based on fuzzing technology. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 2549–2551. [CrossRef]
24. Wang, Z.; Li, Q.; Liu, Q.; Liu, B.; Zhang, J.; Yang, T.; Liu, Q. DICOM-Fuzzer: Research on DICOM vulnerability mining based on Fuzzing technology. In Proceedings of the Communications and Networking: 14th EAI International Conference, ChinaCom 2019, Shanghai, China, 29 November–1 December 2019; Springer: Berlin/Heidelberg, Germany, 2020; pp. 509–524. [CrossRef]
25. Paul, M.; Maglaras, L.; Ferrag, M.A.; AlMomani, I. Digitization of healthcare sector: A study on privacy and security concerns. *ICT Express* **2023**, *9*, 571–588. [CrossRef]
26. Abouelmehdi, K.; Beni-Hssane, A.; Khaloufi, H.; Saadi, M. Big data security and privacy in healthcare: A Review. *Procedia Comput. Sci.* **2017**, *113*, 73–80. [CrossRef]
27. GitHub-ionianCTF/dicom_simulation: A DICOM Simulator in Python Sending ".dcm" Files from a Local Folder to a PACS server Using Pynetdicom and Pydicom.—Github.com. Available online: https://github.com/ionianCTF/dicom_simulation (accessed on 8 July 2023).
28. Zhou, Z.; Law, M.Y.; Huang, H.; Cao, F.; Liu, B.J.; Zhang, J.; Mogel, G.T.; Zhuang, J. Educational RIS/PACS simulator. In Proceedings of the Medical Imaging 2003: PACS and Integrated Medical Information Systems: Design and Evaluation. *SPIE* **2003**, *5033*, 139–147. [CrossRef]
29. Orthanc—DICOM Server—Orthanc-Server.com. Available online: https://www.orthanc-server.com/ (accessed on 8 July 2023).
30. Onis Viewer—Dicom Viewer and PACS—Onis-Viewer.com. Available online: https://onis-viewer.com (accessed on 8 July 2023).

31. Hussain, M.A.; Langer, S.G.; Kohli, M. Learning HL7 FHIR using the HAPI FHIR server and its use in medical imaging with the SIIM dataset. *J. Digit. Imaging* **2018**, *31*, 334–340. [CrossRef] [PubMed]
32. Potter, G.; Busbridge, R.; Toland, M.; Nagy, P. Mastering DICOM with DVTk. *J. Digit. Imaging* **2007**, *20*, 47–62. [CrossRef] [PubMed]
33. DVTK/RIS_Emulator at Master·151706061/DVTK—Github.com. Available online: https://github.com/151706061/DVTK/tree/master/RIS_Emulator (accessed on 8 July 2023).
34. Piraino, D. Radiology information system and picture archiving and communication system: Interfacing and integration. In *Digital(r) Evolution in Radiology*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 47–55. [CrossRef]
35. Oemig, F.; Snelick, R.; Oemig, F.; Snelick, R. Testing Tools. *Healthcare Interoperability Standards Compliance Handbook: Conformance and Testing of Healthcare Data Exchange Standards*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 525–558. [CrossRef]
36. GitHub—Hapifhir/hapi-hl7v2—Github.com. Available online: https://github.com/hapifhir/hapi-hl7v2 (accessed on 8 July 2023).
37. Mantri, M.; Taran, S.; Sunder, G. DICOM integration libraries for medical image interoperability: A technical review. *IEEE Rev. Biomed. Eng.* **2020**, *15*, 247–259. [CrossRef] [PubMed]
38. Cawthra, J.; Hodges, B.; Kuruvilla, J.; Littlefield, K.; Niemeyer, B.; Peloquin, C.; Wang, S.; Williams, R.; Zheng, K. *Securing Picture Archiving and Communication System (PACS) Cybersecurity for the Healthcare Sector*; Nist Special Publication: Gaithersburg, MD, USA, 2019.
39. Mileva, A.; Velinov, A.; Dimitrova, V.; Caviglione, L.; Wendzel, S. Information hiding in the dicom message service and upper layer service with entropy-based detection. *Entropy* **2022**, *24*, 176. [CrossRef] [PubMed]
40. Fritz, S.L.; Roys, S.R.; Munjal, S. Design requirements for DICOM patient, study, and results management. In Proceedings of the Medical Imaging 1996: PACS Design and Evaluation: Engineering and Clinical Issues, Newport Beach, CA, USA, 13–15 February 1996; Volume 2711, pp. 98–108.
41. Welcome to the Orthanc Book 2014; Orthanc Book Documentation—Book.Orthanc-Server.com. Available online: https://book.orthanc-server.com/index.html (accessed on 8 July 2023).
42. Association, N.E.M. Digital Imaging and Communication in Medicine (DICOM). NEMA PS 3 Supplement 23 Structured Reporting. 1997. Available online: https://www.dicomstandard.org/current (accessed on 5 September 2023).
43. Hamamoto, K. Standardization of JPEG quantization table for medical ultrasonic echo images. In Proceedings of the ICECS'99. Proceedings of ICECS'99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No. 99EX357), Paphos, Cyprus, 5–8 September 1999; Volume 2, pp. 683–686. [CrossRef]
44. Security—Dicomstandard.org. Available online: https://www.dicomstandard.org/using/security (accessed on 24 August 2023).
45. GitHub—Pydicom/Pynetdicom: A Python Implementation of the DICOM Networking Protocol—Github.com. Available online: https://github.com/pydicom/pynetdicom (accessed on 8 July 2023).
46. GitHub—Pydicom/Pydicom: Read, Modify and Write DICOM files with Python Code—Github.com. Available online: https://github.com/pydicom/pydicom (accessed on 8 July 2023).
47. Docker—Github.com. Available online: https://github.com/docker (accessed on 22 August 2023).
48. Frenz, C. OWASP Secure Medical Device Deployment Standard Version 2.0. Available online: https://owasp.org/www-pdf-archive/OWASP_Secure_Medical_Devices_Deployment_Standard_7.18.18.pdf (accessed on 8 July 2023).
49. DICOM Security—Sdnewhop/Dicom:—Github.com. Available online: https://github.com/sdnewhop/dicom (accessed on 22 August 2023).
50. Dicom-Brute NSE Script 2014; Nmap Scripting Engine Documentation—Nmap.org. Available online: https://nmap.org/nsedoc/scripts/dicom-brute.html (accessed on 22 August 2023).
51. Dicom-Ping NSE Script 2014; Nmap Scripting Engine Documentation—Nmap.org. Available online: https://nmap.org/nsedoc/scripts/dicom-ping.html (accessed on 22 August 2023).
52. Mitigations—Enterprise | MITRE ATT&CK—Attack.mitre.org. Available online: https://attack.mitre.org/mitigations/enterprise/ (accessed on 22 August 2023).