Mitigating Network-Layer Security Attacks on Authentication-Enhanced OpenICE

Zhangtan Li
University of Chinese Academy of Sciences/TCA Lab,
Institute of Software, Chinese Academy of Sciences
lizhangtan@tca.iscas.ac.cn

Liang Cheng, Yang Zhang
TCA Lab, Institute of Software
Chinese Academy of Sciences
{chengliang,zhangyang}@tca.iscas.ac.cn

ABSTRACT

Integrated Clinical Environment (ICE) is a standardized framework for achieving medical device interoperability. It utilizes high-level supervisory and medical apps and low-level communication middleware to coordinate medical devices to accomplish a shared clinical mission. With the potential to significantly improve healthcare productivity and reduce medical errors, the interoperability of medical devices also subjects ICE systems to unprecedented security threats. In this paper, we present a set of security attacks, namely interception, tampering, and replay attack, to the network level of ICE systems, which we identify through a threat modeling analysis on OpenICE, the best-known instantiation of ICE system. For these security attacks, we devise corresponding defense mechanisms on top of OpenICE. Our experiments demonstrate that these defense mechanisms can effectively protect OpenICE from the identified attacks with acceptable computational overhead.

1 INTRODUCTION

Emerging interoperable medical systems indicate a promising future for the healthcare domain. Many case studies[1, 6–9, 14] have shown that enabling the interoperability of medical systems can reduce medical errors and improve the productivity of medical care, as compared to the traditional practices that rely upon disconnected, standalone devices. One noticeable effort in this direction is the ASTM F2761 standard [2], which defines a standardized framework, called Integrated Clinical Environment (ICE), for integrating medical devices and medical applications (or apps) into an interoperable system that intends to accomplish complex clinical procedures. Within the ICE framework, devices and apps communicate with each other clinical data and control commands through the same network, which is typically coordinated by the system's network middleware. Since its publication, the ASTM F2761 standard has been adopted by many stakeholders, such as Massachusetts General Hospital, Draeger Medical Systems and Kansas State University, to develop interoperable medical systems that follow the ICE framework.

Meanwhile, the uniqueness of ICE (i.e., the interoperability) subjects ICE systems to security risks that have never been encountered by traditional, stand-alone devices before. Security attacks to ICE systems can compromise the integrity and privacy of the data communicated among connected devices and apps, disrupt the expected clinical procedures, and even expose patients and users to severe risks.

MCPS'18, April 2018, Porto, Portugal.

The uniqueness of ICE systems also poses some challenges to make ICE more secure:

The first security challenge is to ensure only trustworthy devices and apps be connected to the system. Medical devices can be dynamically connected to an ICE system and medical apps can be dynamically installed, it is critical to ensure trustworthy devices and apps are allowed to do so. This is especially true as telemedicine/telesurgery is appearing at the horizon of healthcare, which makes it difficult for hospitals to gain full control over medical devices and apps as in the past. The dynamic integration of devices and apps into the system also constantly changes the vulnerability surface of an ICE system at different stages of clinical processes. This requires the security mechanism in the system to be adaptive, in order to align with the changing environment and rebalance security with the evolving clinical needs.

Secondly, medical devices connected to an ICE system often come from different manufacturers through a collection of device interfaces. As device interfaces endow manufacturers a great flexibility in developing medical devices in their own methods, it also creates difficulty for comprehensively analyzing the security of an ICE system down to the device level. Therefore, system developers have to design security mechanisms that are vigilant and resilient to potential security exploitations in each single device connected to the system, so that the rest of the system is protected upon such exploitations.

In view of the aforementioned challenges, the security of ICE systems should be assured at two levels: 1) the system level: security risks arising from the integration and coordination of ICE entities should be mitigated by authentication, authorization, and fine-grained access control. We have made some effort in this direction by enhancing ICE systems with authentication capabilities[3]. 2) the network level: security should be established for the network communication within ICE systems to protect the data transferred over the network. Security vulnerabilities in the network middleware adopted by ICE systems, which is referred to as the Network Controller in the ASTM F2761 standard, could enable attacks, e.g., man-in-the-middle and replay attacks, to these systems. However, the ASTM F2761 standard does not prescribe what security properties should be established in the Network Controller. In this paper, we focus on understanding the security threats at the network level of ICE systems, and propose defense solutions to mitigate such threats.

In this paper, we perform, to the best of our knowledge, the first security analysis of the Network Controller in ICE systems. The contributions of our work can be summarized as follows:

- We perform a threat modeling analysis on the Network Controller of OpenICE[1], the best-known instantiation of ICE systems. Our analysis identifies potential security attacks, namely interception, tampering and replay attacks, which can emerge at the network level of OpenICE (and any other ICE systems following the publish-subscribe communication paradigm).
- We devise three defense mechanisms to protect OpenICE from the identified attack patterns.
- We develop the prototype of these security mechanisms on top of OpenICE systems, and conduct a set of attack-defense experiments to evaluate their effectiveness and computational overhead.

The paper is organized as follows. Section 2 gives a introduction to the ICE architecture, OpenICE (an instantiation of ICE), the network middleware used by OpenICE and some security mechanisms for ICE. Section 3 presents the threat modeling analysis and the attack patterns for OpenICE. Section 4 gives the corresponding defense mechanisms. Section 5 presents results of the attack/defense experiments and performance evaluation. Section 6 is discussion of the work. Related work is discussed in Section 7. We conclude and give directions for future work in Section 8.

2 BACKGROUND

ICE: Figure 1 reiterates the ICE architecture defined in the ASTM 2761 standard, which consists of the following components:

- medical devices that are connected to the system dynamically in a plug-n-play manner;
- medical apps, which are software applications installed in the system to support the completion of the expected clinical procedures;
- the ICE Supervisor, which orchestrates medical apps and connected devices during the clinical procedures. It also provides the portal of interaction for clinicians to control and monitor the system, and exchanges clinical data with external parties such as electronic health record (EHR) systems;
- the Network Controller, which facilitates the communication among the entities in the system. The Network Controller is also known as the network middleware which is located between the applications (e.g., the devices, Data Logger and the Supervisor) and the transport layer in the Open Systems Interconnection (OSI) model.

OpenICE and Data Distribution Service: Several instantiations of the ICE architecture have been proposed by the academia and industry, e.g., MDCF[2], OpenICE and OpenSDC[3], among which the most prominent example is the open-source OpenICE system developed by Goldman et al.[10]. Notably, OpenICE adopts the Data Distribution Service (DDS) as its network middleware. DDS is an Object Management Group (OMG) middleware standard that aims to enable scalable, real-time, dependable, high-performance and interoperable data exchanges using a publish-subscribe pattern. More specifically, DDS creates a communication channel, also called *topic*,

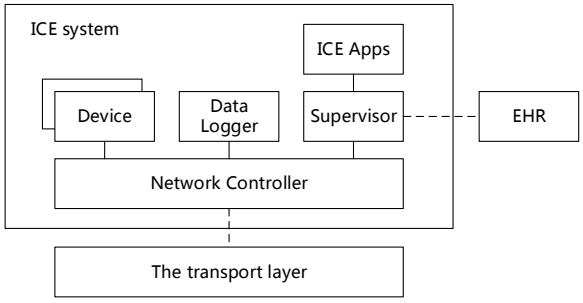

Figure 1: ICE architecture. In which Device indicates the ICE-compatible medical devices, EHR is short for Electronic Health Record system. The transport layer indicates the communication channel in the bottom level.

for each entity (i.e., the medical device or the Supervisor) that intends to send data to the rest of the OpenICE system. The sending entity keeps publishing data to that topic, while other interested entities subscribe to that topic to receive the data published to it.

DDS has its own security module named DDS security[4] that provides a set of security features, including authentication, authorization, access control, data encryption and non-repudiation, to secure the data communication in the system. It should be pointed out: since the current version of OpenICE is without DDS security, the security attacks and corresponding defense mechanisms discussed in this paper are based on the precondition that OpenICE does not integrate DDS Security. Moreover, since DDS does not understand how data communication is integrated with intricate clinical procedures, DDS security cannot effectively mitigate security threats caused by the system-level design/integration flaws of OpenICE. More details will be discussed in the section 6.

ICE Authentication: The current version of ICE standard focuses on realizing the interoperability of medical devices for the envisioned clinical scenarios, and gives little attention to potential security threats and core security features pertinent to the ICE architecture. For example, it does not provide any sort of authentication to verify the identities of devices to be connected. In view of this situation, We designed an authentication framework[3] that equips ICE with basic authentication capabilities and can be used as a basis for further security enhancement.

The proposed authentication framework consists of three layers, as illustrated in Figure 2: 1) the top level are entry points that intercept requests for connection from medical devices and initiate the subsequent authentication processes; 2) the middle level is a management component that coordinates the authentication processes by invoking APIs provided by the authentication functions at the bottom level; and 3) the bottom level is a library of implementation of various authentication protocols that fulfill different authentication needs.

Although designed for the generic ICE architecture, the authentication framework can be easily integrated to OpenICE as illustrated in Figure 3, where OpenICE components are colored in green and components of the authentication framework are in blue. As shown

[1]https://www.openice.info/
[2]https://intranet.cs.ksu.edu/taxonomy/term/106
[3]http://opensdc.sourceforge.net
[4]http://www.omg.org/spec/DDS-SECURITY/1.0/

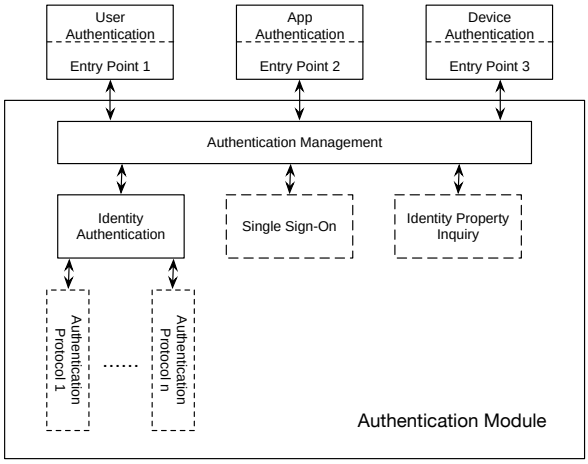

Figure 2: The authentication framework for ICE systems, where dotted boxes are optional components[3].

in Figure 3, medical devices to be connected send requests for connection and other messages to the system through their device adapters. The top level of the authentication framework resides in the ICE Supervisor, including the App Launcher and Login Component that respectively initiate the authentication processes for medical apps and human users. The second level of the authentication framework is implemented in the middle of the ICE Supervisor and DDS, while the bottom level is integrated as part of DDS.

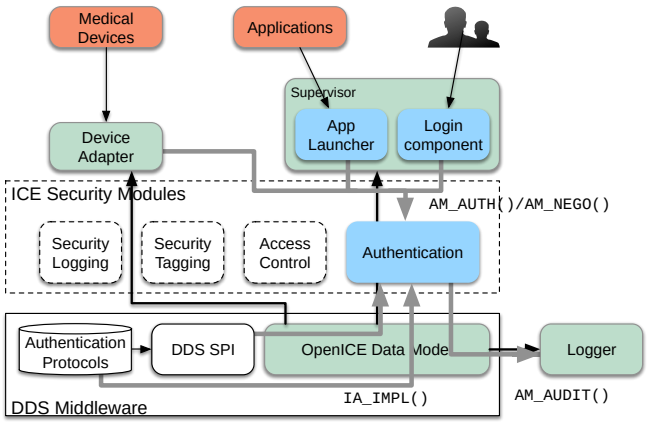

Figure 3: Integrating OpenICE with authentication

3 THREAT MODELING ANALYSIS

The authentication framework in our previous work is a basic security enhancement for OpenICE in the system level, and it can not mitigate the threats brought by the unique design characteristics and limitations of DDS when integrated into OpenICE, i.e., the threats in the network level. Therefore, we do the threat modeling for the network level of OpenICE.

3.1 Modeling Clinical Scenarios

We perform a threat modeling analysis over OpenICE to understand the potential security threats that arise due to the limitations of DDS and the design flaws of OpenICE. To make our analysis relevant, we first define a generic clinical scenario for OpenICE, which is generalized from the set of clinical scenarios listed in the ASTM F2761 standard. It should be pointed out that our threat modeling is based on the precondition that OpenICE does not integrate DDS Security and the assumption that the adversary has physical access to the network of OpenICE.

The generic scenario, as illustrated in Figure 4, consists of a patient; a clinician who operates the system; medical devices connected to the patient for delivering treatment or monitoring patient conditions; the ICE Supervisor coordinating connected devices and other medical apps; and the network middleware facilitating network communication within the system.

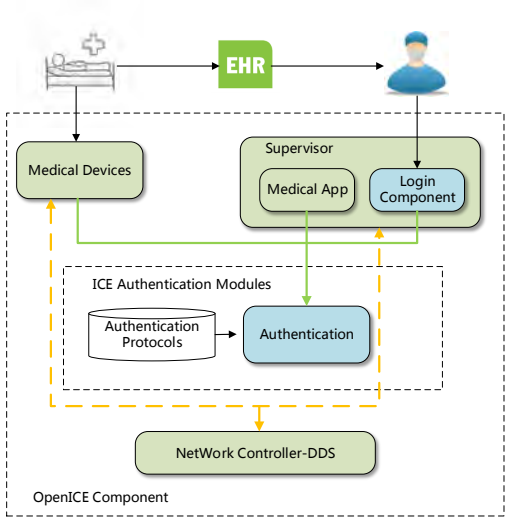

Figure 4: A generic clinical scenario, where solid and dotted lines represent local and DDS communications respectively.

In this generic scenario, clinical data and control signals are communicated among connected devices, medical apps, and the ICE Supervisor. Such data and control can also be sent out to the external EHR system for logging. Most of the network communication in the system, except for the communication with the external EHR system, has to go through DDS using a collection of topics that DDS establishes for publishers and subscribers. In order to maintain their connection to the systems, every connected device is required to continuously send messages to the *HeartBeat* topics. As the name implies, the *HeartBeat* topic is used for transmitting signals which indicate that the sender is alive in the network. Among all entities in the system, the ICE Supervisor and the authentication module are considered as the only trusted computing bases (TCB).

3.2 Identification of Attack Classes

Venkatasubramanian et al. enumerated five classes of security attacks to an interoperable medical devices (IMD) environment, namely destroy, disturb, reprogram, denial of service, and eavesdroping [17]. There is nothing fundamentally new with these attack patterns, but they cover all components of the ICE architecture, as well as the communication between the ICE Supervisor and the connected devices. Therefore, we choose to apply these attack classes to OpenICE in our analysis to explore possible attacks.

Table 1 reiterates the definition of these security attacks [13]. Among these attack classes, destroy attacks which physically destroy some of the components in ICE are obviously not in our discussion. Since we focus on the newtwork level, reprogramming attacks which modify data or code in a medical device are also excluded from our analysis. In addition, our analysis does not include denial of service attacks either which we consider should be analyzed in realistic clinical settings.

3.3 Attack Patterns for OpenICE

Given the communication patterns in OpenICE (as illustrated in Figure 4), we identify three types of attacks as instantiations of the disturb and eavesdrop attack classes. In particular, we identify interception attacks for the eavesdrop attack class, and tampering and replay attacks in the disturb class. Each of these types of attacks assumes a simple scenario, as illustrated in Figure 5, which includes the ICE Supervisor, a genuine device *A*, and an adversary device *B*. The primary target of these attacks is the communication between device *A* and the ICE Supervisor, while the communication between genuine devices is not in our discussion.

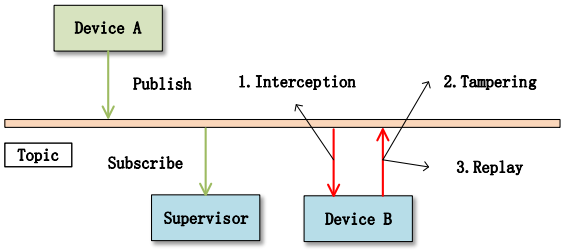

Figure 5: The attack scenario, where orange line represents the communication channels (i.e., topics in DDS), green lines represent normal publication/subscription communications, and red lines indicate communications to launch the attacks.

3.3.1 Interception. The ICE Supervisor and A exchange data through a topic, say *t*. Assume that B (unauthorized) is connected to the same network and somehow acquires the identifier of *t*. Then B can subscribe to *t* and intercept any communication between the ICE Supervisor and A using topic *t*, because DDS poses no restrictions on to which topic a connected device can subscribe.

3.3.2 Tampering. Due to the lack of authentication, the adversary device B can publish tampered data to a topic to which the ICE Supervisor subscribes. The tampering can also tamper data used by other devices or applications and the tampered data has the potential to trigger unexpected operation in the subscribers and in turn induce medical errors and incidents.

3.3.3 Replay Attack. Replay attack on a security protocol typically manifests as an adversary resending messages that are legitimate in other contexts with the goal of deceiving the receivers to think that they have completed the protocol successfully. As for the attack scenario in Figure 5, an adversary device *B* can launch a replay attack to disguise itself as *A*. More specifically, any device connected to OpenICE needs to continuously publish messages, which include the identifier of that device, into the *HeartBeat* topic, in order to maintain its connection with the system. This requirement makes it possible for *B* to intercept the identifier of *A* (using interception attacks), and forge heartbeat messages and continuously publish them to the *HeartBeat* topic after *A* is disconnected. In this way, *B* can deceive the system to think that *A* is connected back.

Another condition is that *A* is still connected with the ICE Supervisor, and *B* also intercepts the identifier of *A* to use the same ID as its own device identifier during its initialization. In this case, the adversary even does not need to forge the heartbeat messages for replay, and the ICE Supervisor cannot distinguish the genuine device and the adversary. Thus, the adversary can bypass authentication, and acquire unauthorized access to the system.

With replay attacks, an adversary can bypass our authentication framework for OpenICE, and imitates a legitimate device to communicate with the rest of the system. As a result, the adversary can further compromise the security of OpenICE, such as stealing critical clinical data or broadcasting fake clinical data to interrupt system operation. Replay attack can also be launched in some other topics and deceive the subscribers to receive the old messages.

It should be pointed out that the attacks described in this section are three kinds of classical attacks and they provide the basis for more complicated attacks such as Man-in-the-Middle Attack. More importantly, these attacks are also applicable to other instantiations of ICE systems that adopt the Publish-Subscribe paradigm in their communication mechanisms.

4 MITIGATING THE SECURITY ATTACKS

The three types of attacks identified in our analysis emerge primarily due to two factors: 1) DDS does not offer means to limit the subscribers to one topic; and 2) messages are communicated in the system in plaintext, which allows the malicious devices to acquire sensitive information, such as device identifiers, contained in these messages. Mitigation measures can thus be designed to prevent the security attacks by eliminating these two enabling factors.

However, it is challenging to design such mitigation measures without the ability of directly modifying DDS. Eclectically, we propose defense mechanisms against these types of attacks on top of DDS, so that there is no need to change the implementation of DDS.

4.1 Mitigating Interception Attacks

Interception attacks can be prevented by mandating the messages be encrypted before publishing to the corresponding topics. In this way, no sensitive information is disclosed to the intercepters. We

Table 1: General attack model for ICE[13]

Attack Pattern	Description	Susceptible Components
Destroy	Physically destroy ICE components; e.g., cut an infusion pump tube.	All Architectural Components of ICE
Disturb	Modify exchanged data to prevent correct operation of components; e.g., man-in-the-middle or replay attacks.	All Architectural Components of ICE
Reprogram	Modify data or code in an ICE component to prevent its correct operation; e.g., modify infusion pump software to deliver extra medication.	All Architectural Components Except the Communication Network
Denial of Service	Exploit bugs or interfaces that were not designed with security in mind.	All Architectural Components of ICE
Eavesdrop	Listen in on the deployed ICE environment to learn sensitive information.	Communication Network

implement a symmetrical encryption mechanism, illustrated in Figure 6, to secure the content of messages published in OpenICE topics. Given a topic, the mechanism requires its legitimate publisher(s) and subscriber(s) to establish key agreement following the X509 standard[5]: a device and the ICE Supervisor follow the TLS[6] handshake protocol to authenticate each other, during which the key agreement is reached (including the establishment of session key and the key to compute Message Authentication Codes) if the authentication succeeds. For simplicity, we use self-signed certificate to accomplish the authentication, while in practice, the certificate authority should be added. With the session key the device encrypts the data before publishing it to the topic. The ICE Supervisor maintains a database of the session keys established for each active device, from which it retrieves the session key to decrypt the data published from the corresponding device before displaying the data on the user interface.

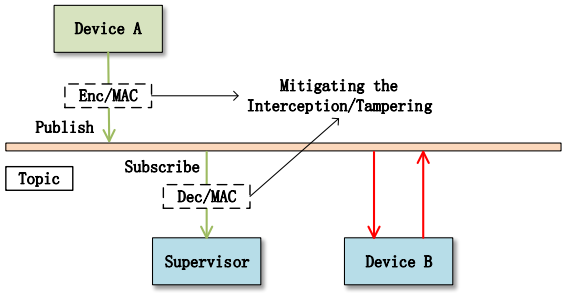

Figure 6: The Defense scenario. The dotted boxes indicate the Encryption/Decryption and Message Authentication Code modules.

4.2 Mitigating Tampering Attacks

The primary enabler of tampering attacks is that DDS does not authenticate an entity before it publishes messages to a topic. Hence, subscribers of the topic have no means to confirm that the messages come from the genuine publisher. Defending against tampering attacks can thus be achieved by authenticating the true identity of message publishers. We choose to utilize the Message Authentication Code (MAC) mechanism to ensure the authenticity and integrity of the published messages, because it not only is effective but also requires the only minimum extension to DDS.

In particular, our solution enforces a device to negotiate one encryption key with the ICE Supervisor during its authentication process. Later on, when that device publishes a message to one of its topics, it is required to calculate the MAC value for the message using the negotiated key and the HmacSHA1 algorithm[7]. This MAC value is appended to the message when it is published[8]. When receiving this message, the ICE Supervisor re-calculates its MAC value in the same way, and compares it with the one appended to the message. If the two MAC values do not match with each other, the message is considered as being compromised or from a fake publisher, and thus discarded. Moreover, upon receiving consecutive compromised messages from the same publisher, the ICE Supervisor is revised to notify the user the occurrence of a potential tampering attack.

4.3 Mitigating Replay Attacks

Replay attacks can happen because the ICE Supervisor only uses the device identifier, which is in plaintext, included in the heartbeat messages to identify their publishers. Thus, an adversary can easily disguise itself as a genuine device by re-publishing the same heartbeat messages from the genuine device.

Typical protection against replay attacks is to augment messages with timestamps, so that they are only valid for a limited period of time. But this type of protection may not be ideal for medical systems like OpenICE. Especially, for heartbeat messages that the connected devices have to constantly send to the ICE Supervisor to maintain their connection to the system, adding timestamps may augment more overhead. Moreover, as a distributed system, it is not easy for OpenICE to maintain synchronized time.

We propose a simple solution to prevent replay attacks that avoids the heavy computational overhead of timestamps. In our

[5]https://en.wikipedia.org/wiki/X.509

[6]We just refer to the classical TLS handshake protocol to implement the authentication module, so it does not mean we integrate OpenICE with TLS. Moreover, we add defense mechanisms on top of the network middleware, i.e. DDS, so we do not focus on the communication in the bottom level, e.g., the transport layer.

[7]https://en.wikipedia.org/wiki/Hash-based_message_authentication_code

[8]DDS maintains a data structure for each topic established, which defines the format of messages published to that topic. We extend such data structure with an additional field to store the MAC value.

solution, the ICE Supervisor is extended to maintain a list of authenticated devices that are currently connected to the system. Whenever a device is disconnected (i.e., the device stops sending the heartbeat messages), the ICE Supervisor removes it from the list, so that it has to be re-authenticated when connecting back to the system. This enables the ICE Supervisor to detect any replayed heartbeat messages when the corresponding devices have been disconnected. Accordingly, for the replay attack on other topics, the replayed messages can also be detected, since the corresponding device has been removed from the list.

However, the solution talked above can not mitigate the replay attack on the other condition(i.e., when the genuine device is still connected with the ICE Supervisor). In this case, the adversary even does not need to forge the heartbeat messages for replay attack. The adversary just need to utilize the genuine device's ID as its own device identifier during its initialization and then disguises itself as the genuine device to communicate with the ICE Supervisor. Therefore, the keypoint of the mitigation is to improve the authentication for the topic such as the *HeartBeat* topic and this is beyond the discussion of this paper.

It should be pointed out that all these defenses are applicable to any ICE systems that follow the Publish-Subscribe paradigm of communication, except for the defense against replay attack that is based on the implementation of OpenICE.

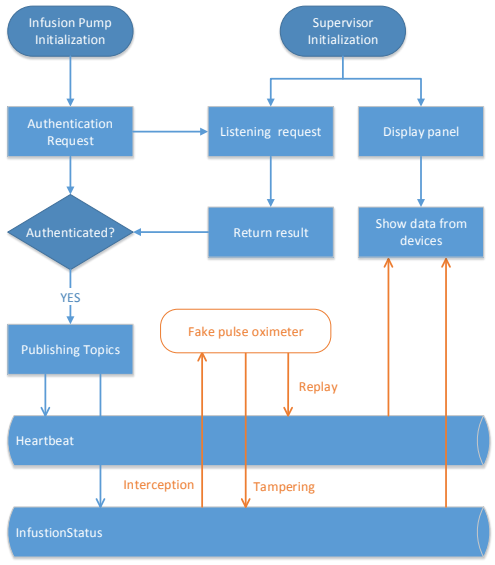

Figure 7: The attack diagram mapped into the workflow of OpenICE with authentication module. The solid blue lines indicate the workflow of OpenICE, the orange rectangle denotes the fake pulse oximeter and the orange solid lines indicate the attack from it.

5 EVALUATION

We have performed three sets of experiments to evaluate the security attacks to OpenICE and our defense mechanisms against them. The first set of experiments intended to confirm the practicality of security attacks described in section 3; the second set evaluated the effectiveness of the defense mechanisms; and the last set evaluated the performance of these mechanisms.

All these experiments were performed using the same hardware settings: a computer (with a 4-core Intel 3.3GHz i5 CPU, 6MB cache and 12GB RAM) was used to run the OpenICE Supervisor (version 1.8) and other simulated (genuine) medical devices, and a laptop (with a dual-core Intel 2.1GHz i7 CPU, 4MB cache and 8GB RAM) ran simulated adversary medical devices. These two computers were connected through a local area network, so that the ICE Supervisor and simulated devices could communicate with others.

5.1 Attack Experiments

In the attack experiments, we created two simulated devices: the genuine device simulates an infusion pump that delivers fluids, medication or nutrients into patients; and the adversary device simulates a pulse oximeter that monitors the blood oxygen saturation level of patients. For simplicity, we refer to the infusion pump as *the pump* and the pulse oximeter as *the fake device* in the rest of the paper. Specifically, we reused the code of the genuine infusion pump and pulse oximeter in OpenICE to create these two simulated devices. We also added some new modules into the fake device to launch the attacks.

Therefore, the OpenICE system under experiment was organized as illustrated in Figure 7, where solid blue lines represent the normal workflow of the system and orange solid lines represent the attacks initiated by the fake device.

To enable the fake device to launch attacks, we extended its initialization function with an attack module for each type of attacks described in section 3. Each attack module was implemented as a thread in which the fake device utilized the DDS API to read and write the target topic. These attack modules were executed asynchronously, because the interception attack is the foundation of the other two types of attacks.

In the experiments, we assume that the fake device possesses the knowledge of the topic that it intends to attack. This is reasonable because OpenICE stores the information of all topics in a text file in plaintext, which can be easily accessed to and compromised by an adversary if not protected appropriately.

5.1.1 Interception Attack. With this attack, the fake device aimed at intercepting the messages that the pump published to topic *InfusionStatus* reporting its operational status. In order to do so, the fake device created a corresponding *Subscriber* in the interception thread to declare its intent to receive messages published in *InfusionStatus*. Once the pump published messages to this topic, the interception thread can obtain the messages, i.e., launching the interception attack.

Figure 8 demonstrates a message intercepted by the fake device, which included the information on the device identifier of the pump, and the name of the drug currently being delivered, and other infusion parameters used by the pump.

5.1.2 Tampering Attack. With the device identifier of the pump acquired through interception attacks, the Tampering Attack module in the fake device then forged a message that only changed

Figure 8: Data published in the InfusionStatus topic and intercepted by the fake device.

the Drug Name field of the intercepted message from *Morphine* to *Morphine Tampering*, and published it to *InfusionStatus*. The ICE Supervisor received this message from the topic and displayed it on the front panel, as illustrated in Figure 9. This confirms that the current version of OpenICE is vulnerable to tempering attacks.

Figure 9: The tampered message displayed on the ICE Supervisor front panel.

5.1.3 Replay Attack. In this experiment, the pump was first disconnected from the OpenICE system. The fake device then forged heartbeat messages using the device identifier of the pump, and published them to the *HeartBeat* topic. The result of this experiment confirms that the ICE Supervisor continued to display the pump (actually the fake device) on its front panel, since it is unable to verify the true identity of the publisher of the received heartbeat messages.

5.2 Defense Experiments

We performed three experiments to examine whether the defense mechanisms presented in section 4 are effective in preventing the attacks to OpenICE.

Defense against Tampering Attacks: As aforementioned, we utilized the MAC mechanism as illustrated in Figure 10 as the green boxes to defend against tempering attacks to OpenICE systems. In the experiment, the pump was forced to calculate and append a MAC value to every message it published to topic *InfusionStatus*. Upon receiving messages from *InfusionStatus*, the ICE Supervisor verified its authenticity and integrity by checking their MAC values. The result of this experiment showed that the tampered message as crafted in the tempering attack experiment was no longer displayed on the front panel of the ICE Supervisor.

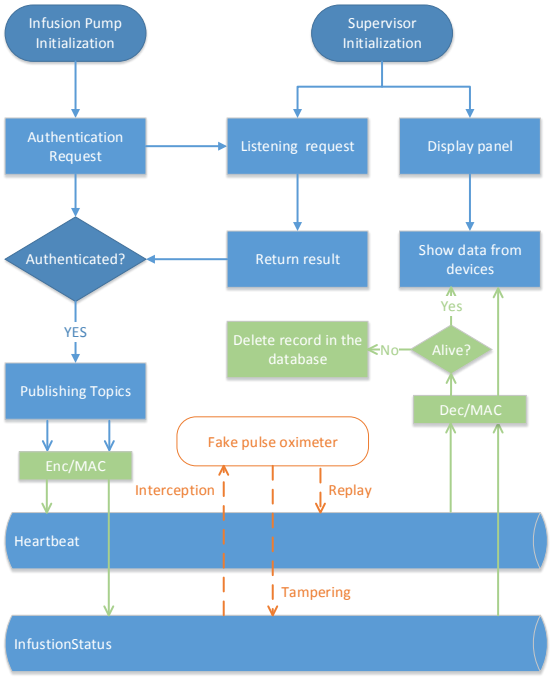

Figure 10: The Defense Diagram mapped into the workflow of OpenICE. The green rectangles indicate the security modules and the solid green lines indicate the data flow through security modules.

Defense against Interception Attacks: In this experiment, the defending mechanism required the pump to negotiate a key with the ICE Supervisor during its authentication, and use this key to encrypt every message it published to the system. For example, the pump used the key to encrypt the messages published to topic *Enc_InfusionStatus*, except for the device identifier information. The device identifier was not encrypted because the ICE Supervisor needs it in plaintext to retrieve the decryption key. The encryption and decryption modules are illustrated in Figure 10 as the green boxes.

Our experiment showed that the encryption-based defense can effectively mitigate tampering attacks. Figure 11 shows a message that was published in topic *Enc_InfusionStatus*, where all fields of the message but the device identifier were encrypted. Thus, the fake device cannot understand the content of this message without knowing the encryption key for *Enc_InfusionStatus*.

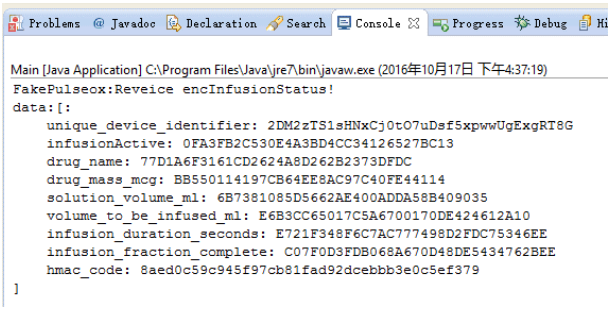

Figure 11: Content of an intercepted message

Defense against replay attacks: Our defense mechanism forced the ICE Supervisor to remove the record of a connected device when it is disconnected. This defense was integrated to OpenICE during the experiment as illustrated in Figure 10. The fake device was first made to launch the interception attack. Then, the pump was disconnected from the system. Using the pump's device identifier acquired from the interception attack, the fake device forged heartbeat signals and sent them to the ICE Supervisor. The defense made the ICE Supervisor aware of the disconnection of the pump, and hence rejected fake heartbeat signals from the fake device. This is confirmed by the fact that the pump was not listed on the front panel of the ICE Supervisor when the fake device launched the replay attack.

5.3 Performance

The last set of experiments we performed were to evaluate the computational overhead of the presented defense mechanisms, which compared the system performance of OpenICE, in terms of the time latency, CPU and memory usage, with and without the defense mechanisms. What needs illustration is that the computational overhead is based on our experiment environment which is different from the clinical environment.

5.3.1 Latency Test. Table 2 summarizes the time latency of the OpenICE system, under the circumstances where the defense mechanism was installed or otherwise. The time latency of device booting is defined as the duration from the time when a device starts its initialization to the time when it is recognized and displayed on the ICE Supervisor front panel. The time latency of displaying a device is the time duration from when the ICE Supervisor first receives its heartbeat messages (without defense) or its authentication request (with defense), to when the ICE Supervisor displays it on the front panel. Notably, the time latency values listed in Table 2 are calculated as the average of five rounds of experiments.

Table 2: The latency test

Latency	Without Defense (ms)	With Defense (ms)
Device booting	592	2927
Topic encryption	/	79
MAC calculation	/	3
Device displaying	close to 0	849

Table 2 suggests a significant increase in the latency during device booting (from 592ms to 2927ms) when the system was equipped with the defense mechanisms. Most of this increase was caused by the computation involved in device authentication and key negotiation. However, we do not consider that such increase of latency from device booting (in seconds) would pose significant risks to clinical tasks.

5.3.2 CPU&Memory Usage. To estimate the CPU and memory usage imposed by the defense mechanisms, we simulated the OpenICE system in two settings: one with only one infusion pump connected and the other with 15 infusion pumps connected simultaneously, given the available resources (i.e., the memory size of the experiment computers restricts the number of infusion pumps connected simultaneously up to 15). The experiment results are illustrated in Figure 12.

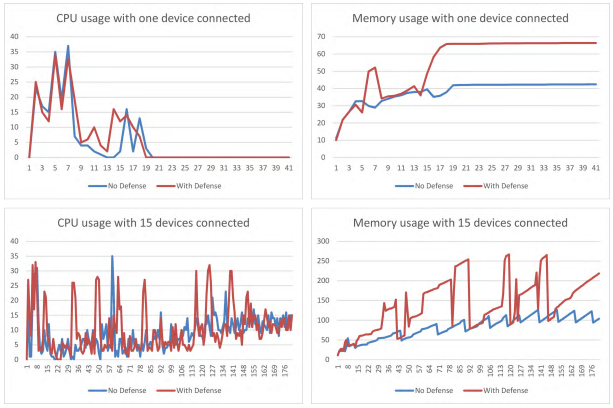

Figure 12: CPU and Memory usage, where blue and red curves denote the resource usage with the defense mechanisms being turned off and on, respectively.

As shown in Figure 12, the distribution of computational resource usage over system operation demonstrates a similar pattern, with the defense mechanisms were turned either on or off. However, turning on the security modules did cause a certain degree of increase in CPU and memory usage or, in other words, certain performance degradation in the system. In particular, with the defense mechanisms, the system's CPU usages increased by 19.9% (or 15.8%) with one device (or 15 devices) connected, even though the peak CPU usage remained the same no matter the defense mechanisms were turned on or off. The peak memory usage, on the other hand, increased from 42 MB to 66 MB with one device connected, and from 126 MB to 267 MB with 15 devices connected, when the defense mechanisms were turned on. The average memory usage increased by 43.7% and 71.4% in these two settings when the defense mechanisms were enabled.

5.3.3 Cost of Time Stamp Calculation. We did the third experiment to evaluate the overhead of appending timestamps to the communicated messages, compared with our defense against the replay attack. In this experiment, the authentication module was enabled, and ten devices were connected to OpenICE simultaneously. Encrypted timestamps were appended to heartbeat messages

(using the keys established between the devices and the ICE Supervisor) published by the connected devices, and the ICE Supervisor decrypted the timestamps to evaluate the validity of the received heartbeat messages.

The results of this experiment are illustrated in Figure 13, which indicates that calculating timestamps and appending them to heartbeat messages caused the average CPU usage to increase by 19.98%, and the peak CPU usage raised to 60% (from the original 39%). In addition, Figure 13 also sees an increase in the average memory usage by 45.9% due to calculating timestamps. These results confirm that using timestamps to ensure time validity of communicated messages can cause significant computational overhead and in turn system performance degradation in OpenICE. In contrast, as blue curves show, our defense mechanism to prevent replay attacks only imposed a slight burden on system performance.

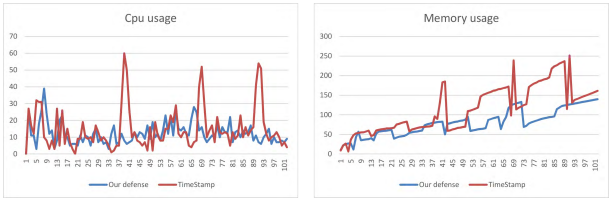

Figure 13: CPU and Memory usage, where blue curves represent the usage with our defense presented in 4.3 against replay attack and red curves represent that with timestamp calculation.

From the results of the attack-defense experiment, we can conclude that: Firstly, the identified attacks are practical, due to the limitation of OpenICE without DDS security. Secondly, the presented defense mechanisms are effective in preventing these attacks. Thirdly, the computational cost of the defense mechanisms can cause certain degradation in system performance. However, whether such degradation can cause clinical risks, such as delay or disruption to clinical procedures should be evaluated in real clinical practices or by consulting with healthcare providers.

6 DISCUSSION

We have to emphasize that the security attacks and corresponding defense mechanisms discussed in this paper are based on the precondition that OpenICE does not integrate DDS Security. In fact, in its latest release, DDS Security has provided certain security features such as authentication, access control and cryptography. Thus, DDS Security has the ability to mitigate interception and tampering attacks, by providing access control over topics per device [13]. However, DDS Security cannot provide fine-grained access control over the *HeartBeat* topic for connected devices. That is, in OpenICE every device has the privileges to read from and write to the *HeartBeat* topic. If a genuine device is compromised, an adversary can access to the *HeartBeat* topic and acquire its device identifier to initiate replay attacks. In comparison, our defenses are implemented on top of DDS. They may not be as efficient as DDS security, but they can also mitigate the interception and tampering attacks. In addition, we present a simple solution which is much more effective than the timestamp to mitigate the replay attack discussed above which can not be mitigated by DDS security.

All the attacks and defenses discussed above are not only applicable to OpenICE. Actually, these attacks and defenses are based on the publish-subscribe paradigm used by DDS, thus they are applicable to any ICE systems using this communication paradigm, such as the Medical Device Coordination Framework (MDCF).

7 RELATED WORK

The security of interoperable medical systems in general, and ICE systems in particular, has been attracting increasing attentions from both industry and academia. Most previous research in this direction focused on establishing reasonable high-level security requirements for ICE systems, such as[4, 5]. Vasserman et al.[15] defined a set of fundamental security requirements for safe and secure next-generation medical systems consisting of dynamically composable units, tied together through a real-time safety-critical middleware. They suggested to eliminate these security requirements from the system level. Venkatasubramanian et al.[16, 17] enumerated five typical categories of possible security attacks to the ICE platform, based on how the ICE platform may be attacked and the consequence caused by such attacks.

Few security mechanisms have been proposed and implemented on realistic ICE systems. In fact, the only security mechanism for ICE systems aware by the authors was proposed by Salazar and Vasserman [11, 12], which implemented a modularized prototype for device authentication and communication encryption within the MDCF context. It provides preliminary access control to ICE systems with a break-the-glass feature. The defense mechanisms presented in this paper distinguishes itself from that by Salazar and Veasserman in the following aspects: 1) our mechanisms target specifically at OpenICE, a realistic instantiation of the ICE framework. The MDCF, on the other hand, intends to provide a model-based design and simulation framework for ICE systems; 2) our defense mechanisms have been shown effective to prevent security attacks that target at vulnerabilities at the network layer in OpenICE, as compared to their mechanism focuses primarily on system-level security design of ICE systems.

Based on OpenICE, Soroush et al.[13] developed two prototypes of security mechanisms for OpenICE based on TLS and DDS Security, respectively. They pointed out that DDS Security could provide more resilience against insider attacks utilizing authenticated but compromised medical devices. However, these mechanisms cannot address security challenges caused by the unique design characteristics of OpenICE. Our work discovers that DDS Security cannot provide fine-grained access control over the *HeartBeat* topic, so that an adversary can leverage compromised devices inside the system to get some key information, and in turn launch some attacks such as the replay attack.

8 CONCLUSIONS

Emerging interoperable medical systems indicates a promising future for healthcare, and ICE is a prominent effort towards this vision. However, security threats can jeopardize the safety of interoperable medical systems and expose patients and users to unwanted risks. In this paper, we have analyzed the security threats that can leverage

the vulnerabilities at the network layer in ICE systems (OpenICE in particular) to compromise their security. A collection of security mechanism have been proposed and evaluated to effectively mitigate these security threats. All the attacks based on these security threats and the corresponding security mechanisms are practicable in some other ICE systems which use the publish-subscribe paradigm in their network layer.

It is also worthy of noting that DDS security can mitigate some traditional security threats at the network level of OpenICE. But it cannot address security threats due to the system-level design of OpenICE. Therefore, acceptable security can only be established in interoperable medical devices in a systematic manner, by bringing appropriate security design and comprehensive consideration to system, communication, and device levels.

We next plan to improve the performance of our security mechanisms and evaluate the scalability of the security mechanisms in real clinical scenarios.

ACKNOWLEDGMENTS

We are grateful for the insightful comments proposed by the anonymous reviewers. This research was supported in part by National Natural Science Foundations of China (Grant No. 61471344) and National Key R&D Program of China (Grant No. 2017YFB0802902).

REFERENCES

[1] David Arney, Sebastian Fischmeister, Julian M Goldman, Insup Lee, and Robert Trausmuth. 2009. Plug-and-play for medical devices: Experiences from a case study. *Biomedical Instrumentation & Technology* 43, 4 (2009), 313–317.

[2] ASTM International 2009. *ASTM F2761-09(2013), Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) - Part 1: General requirements and conceptual model.* ASTM International.

[3] Liang Cheng, Zhangtan Li, Yi Zhang, Yang Zhang, and Insup Lee. 2017. Protecting interoperable clinical environment with authentication. *ACM SIGBED Review* 14, 2 (2017), 34–43.

[4] Denis Foo Kune, Krishna Venkatasubramanian, Eugene Vasserman, Insup Lee, and Yongdae Kim. 2012. Toward a safe integrated clinical environment: a communication security perspective. In *Proceedings of the 2012 ACM workshop on Medical communication systems.* ACM, 7–12.

[5] John Hatcliff, Andrew King, Insup Lee, Alasdair Macdonald, Anura Fernando, Michael Robkin, Eugene Vasserman, Sandy Weininger, and Julian M Goldman. 2012. Rationale and architecture principles for medical application platforms. In *Cyber-Physical Systems (ICCPS), 2012 IEEE/ACM Third International Conference on.* IEEE, 3–12.

[6] Andrew King, Dave Arney, Insup Lee, Oleg Sokolsky, John Hatcliff, and Sam Procter. 2010. Prototyping closed loop physiologic control with the medical device coordination framework. In *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care.* ACM, 1–11.

[7] Andrew King, Sam Procter, Dan Andresen, John Hatcliff, Steve Warren, William Spees, Raoul Jetley, Paul Jones, and Sandy Weininger. 2009. An open test bed for medical device integration and coordination. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on.* IEEE, 141–151.

[8] Andrew L King, Sanjian Chen, and Insup Lee. 2014. The middleware assurance substrate: Enabling strong real-time guarantees in open systems with openflow. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2014 IEEE 17th International Symposium on.* IEEE, 133–140.

[9] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, et al. 2012. Challenges and research directions in medical cyber–physical systems. *Proc. IEEE* 100, 1 (2012), 75–90.

[10] Jeffrey Plourde, David Arney, and Julian M Goldman. 2014. Openice: An open, interoperable platform for medical cyber-physical systems. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on.* IEEE, 221–221.

[11] Carlos Salazar. 2014. *A security architecture for medical application platforms.* Ph.D. Dissertation. Kansas State University.

[12] Carlos Salazar and Eugene Y Vasserman. 2014. Retrofitting communication security into a publish/subscribe middleware platform. In *Software Engineering in Health Care.* Springer, 10–25.

[13] Hamed Soroush, David Arney, and Julian Goldman. 2016. Toward a Safe and Secure Medical Internet of Things. *IIC Journal of Innovation* 2, 1 (2016), 4–18.

[14] Curtis R Taylor, Krishna Venkatasubramanian, and Craig A Shue. 2014. Understanding the security of interoperable medical devices using attack graphs. In *Proceedings of the 3rd international conference on High confidence networked systems.* ACM, 31–40.

[15] Eugene Y Vasserman and John Hatcliff. 2013. Foundational Security Principles for Medical Application Platforms. In *International Workshop on Information Security Applications.* Springer, 213–217.

[16] Eugene Y Vasserman, Krishna K Venkatasubramanian, Oleg Sokolsky, and Insup Lee. 2012. Security and interoperable-medical-device systems, part 2: Failures, consequences, and classification. *IEEE security & privacy* 10, 6 (2012), 70–73.

[17] Krishna K Venkatasubramanian, Eugene Y Vasserman, Oleg Sokolsky, and Insup Lee. 2012. Security and interoperable-medical-device systems, part 1. *IEEE security & privacy* 10, 5 (2012), 61–63.