Peer-to-Peer Networking and Applications (2021) 14:2719–2734
https://doi.org/10.1007/s12083-021-01082-w

Detecting and mitigating cyberattacks using software defined networks for integrated clinical environments

Alberto Huertas Celdrán[1,2] · Kallol Krishna Karmakar[3] · Félix Gómez Mármol[4] · Vijay Varadharajan[5]

Received: 7 May 2020 / Accepted: 21 January 2021 / Published online: 10 February 2021

Abstract

The evolution of integrated clinical environments (ICE) and the future generations of mobile networks brings to reality the hospitals of the future and their innovative clinical scenarios. The mobile edge computing paradigm together with network function virtualization techniques and the software-defined networking paradigm enable self-management, adaptability, and security of medical devices and data management processes making up clinical environments. However, the logical centralized approach of the SDN control plane and its protocols introduce new vulnerabilities which affect the security of the network infrastructure and the patients' safety. The paper at hand proposes an SDN/NFV-based architecture for the mobile edge computing infrastructure to detect and mitigate cybersecurity attacks exploiting SDN vulnerabilities of ICE in real time and on-demand. A motivating example and experiments presented in this paper demonstrate the feasibility of of the proposed architecture in a realistic clinical scenario.

Keywords Cybersecurity · Integrated clinical environments · Medical cyber-physical systems · Software defined networks · Network function virtualization

1 Introduction

The next generation of hospitals integrates technological innovations into clinical systems to enhance the patients' healthcare and quality of life while optimizing resource usage and direct/indirect healthcare costs. Among the advantages of these novel clinical environments, they enable new application scenarios such as hospital rooms sensing patients' vital signs and intelligently analyzing them to diagnose diseases, mobile personalized and non-invasive ecosystems predicting adverse events in an early phase, or multimedia assistance anywhere, anyhow, and at any time.

Medical Cyber-Physical Systems (MCPS) [1] and the fifth generation of mobile networks (5G) bring to reality the hospitals of the future and their new generation of clinical scenarios. This is possible thanks to the integration of physical devices and virtual data processing with scalable network infrastructures that meet not only the increasing bit rates and bandwidths demand but also the flexibility and dynamism required to manage the network resources in real time and on-demand. To realize the MCPS vision, a patient-centric architecture was proposed by the ASTM F2761 standard to enable integrated clinical environments (ICE) [2]. Solutions based on ICE provide open coordination, control and connectivity of medical devices belonging to medical scenarios. In this context, 5G architectures that combines network function virtualization (NFV) techniques and the software-defined networking (SDN) paradigm can help to provide self-management, adaptability, and security to medical devices and data management processes making up MCPS [3].

Despite the benefits provided by existing solutions that combine NFV/SDN and ICE, the logical centralized approach of the SDN control plane and its protocols introduce new vulnerabilities affecting the security of the network infrastructure, medical devices, and therefore, patients' safety [4]. As an example, a Denial-of-Service (DoS) attack could be performed due to the centralized approach of the SDN controller and the limitations of network devices flow-tables. Another well-known issue of

This article part of the Topical Collection: *Special Issue on Security of Mobile, Peer-to-peer and Pervasive Services in the Cloud*
Guest Editors: B. B. Gupta, Dharma P. Agrawal, Nadia Nedjah, Gregorio Martinez Perez, and Deepak Gupta

✉ Alberto Huertas Celdrán
huertas@ifi.uzh.ch

Extended author information available on the last page of the article.

SDN solutions is the open programmability of the network, both between applications and controllers, and controllers and end devices [5, 6]. Keeping in mind the previous concerns, several open challenges need more efforts to be addressed. Among the most relevant ones, we highlight the necessity of designing architectures oriented to ICE that consider the vulnerabilities of the SDN paradigm; the implementation of cybersecurity mechanisms in clinical scenarios able to detect cyberattacks affecting the SDN control plane, concretely the SDN controller and its catalogues or tables; and the use of real-time and on-demand mechanisms oriented to mitigate cyberattacks affecting heterogeneous eHealth scenarios [7, 8].

With the goal of improving the previous challenges, the main contributions of this manuscript are as follows:

- a Threat model of integrated clinical environments showing the main cybersecurity problems affecting each component of the ICE framework.
- A policy-based and SDN/NFV-based architecture, for the mobile edge computing paradigm (MEC), with the ability to detect and mitigate cybersecurity attacks exploiting SDN vulnerabilities of ICE in real time and on-demand. The proposed architecture is generic enough to detect different cyberattacks affecting ICE and other types of Cyber-Physical infrastructures.
- A motivating example of a well-known cyberthreat affecting the SDN paradigm and how our architecture detects and mitigates the cyberthreat phases by demonstrating its added value compared to existing solutions. A set of policies designed for the cyberthreat considered in the motivating example has also been proposed to demonstrate the framework adaptability. In this sense, new policies could be ingested in our architecture to detect other cyberattacks.
- Several experiments measuring the performance of our architecture in a realistic integrated clinical scenario. The obtained results demonstrate that our architecture is capable of mitigating cyberthreats affecting ICE scenarios equipped with the SDN in an acceptable time.

The remainder of this paper is structured as follows. Section 2 reviews the main related works that focus on cybersecurity applied to ICE scenarios as well as attacks affecting the SDN paradigms. Section 3 presents the ICE framework and the threat analysis of its components. Section 4 shows a realistic use case of a hospital room of the future implementing the ICE framework and relevant cyberattacks affecting the SDN plane of ICE. Section 5 outlines the components of the proposed architecture to detect and mitigate cyberattacks affecting the SDN plane of ICE, whose feasibility is validated by the conducted experiments presented in Section 6. Finally, conclusions are drawn and future work is suggested in Section 7.

2 Related work

Cybersecurity issues found in the literature regarding ICE solutions are reviewed in this section. In addition, it also considers SDN vulnerabilities and generic solutions detecting and mitigating cyberattacks affecting the SDN plane.

2.1 Cybersecurity in ICE

The cybersecurity of medical devices and hospital network infrastructures is one of the most critical issues that current clinical environments have to deal with. As a recent example, in September 2020, a ransomware attack affecting a UK hospital caused the death of a patient who had to be moved to another hospital 30 kilometers away [9]. In this dramatic scenario, the authors of [10] identified critical shortcomings of ICE in challenging scenarios such as security, quality of service (QoS), and high availability. To improve the previous aspects, the authors of [11] presented an architecture, called ICE++, for the MEC paradigm that combined SDN/NFV. As an extension of ICE++, proposed ML-based solution able to classify several well-known families of ransomware affecting the medical devices of ICE. In addition, the proposed solution was able to detect anomalies produced by unseen families of ransomware and mitigate them by using SDN/NFV techniques. Reported experiments demonstrated a good performance in terms of detection precision and recall (92.32%/99.97%) in anomaly detection and accuracy (99.99%) in classification.

In terms of cybersecurity in ICE, the authors of [12] proposed a mechanism to protect the communication of the ICE framework by using the Data Distribution Service (DDS) standard. The authors achieved this protection through a middleware enabling critical aspects such as authentication and authorization mechanisms as well as confidentiality, integrity and non-repudiation capabilities. Experiments stated that transport-level security (TLS) does not provide enough security in scenarios were resilience against insider attacks is needed. In this sense, DDS addresses or mitigates disturb, eavesdrop, and denial of service (DoS) attacks. Another work protecting the security and privacy of architectures based on ICE was proposed in [13]. The authors proposed a cloud-based secure logger receiving data from medical devices with ICE Equipment Interfaces. Cypher mechanisms were used by the logger to operate in secure communication channels, non-trusted networks, and operating systems. Despite the added-value of the proposed solution to detect replay or injection attacks (among others), it is useless to detect attacks that do not modify network packets and messages. The authors of [14] designed and deployed an authentication architecture for medical systems compliant with ICE.

The proposed architecture had three layers enabling a variety of authentication capabilities and requirements of ICE elements and networking infrastructure. Several tests demonstrated the usefulness of the authentication solution while device replacement and impersonation attacks on the OpenICE framework.

Finally, in terms of mitigation of cyberattacks using the SDN paradigm, the authors of [15] proposed the usage ovf deep packet inspection tools to analyze HTTP POST messages. The authors highlighted as relevant data the lengths of three consecutive HTTP POST messages. Later, they trained a ML classifier with samples coming from different ransomware families. The outputs of the performed experiments provided an FPR of about 4%. Another work was proposed in [16], where authors used SDN redirection capabilities and a blacklist of proxy servers to detect if infected devices communicate with proxy servers to obtain the public encryption key. They avoid this situation with the use of a flow filter to block the communication and hence the encryption of the files. The main drawbacks of this proposal are the need to keep updated the blacklist of proxy servers and the identification of those servers in advance.

2.2 Cyberattacks affecting the SDN paradigm

SDN consists of three layers and two interlayer communication interface. In this section, we describe layer-wise SDN vulnerabilities and currently available solutions. Figure 1 presents the basic SDN architecture.

2.2.1 SDN control and application plane

SDN controller possesses a high threat being a single point of failure. In this section, we will explain cyberattacks affecting the controller.

- *Authorization Escalation.* SDN controllers run on top of Linux kernels, they are created with programming languages and are used like any other third-party software over an Operating System (OS). If any of the previous components are insecure, then the SDN controller, as well as the whole network, will be insecure, as demonstrated in [17]. An adversary can use open telnet or ssh ports to gain access to the hosting Operating system. Sometimes he can brute-force the login credentials. Once he gains access to the hosting OS, the whole network is under his control [18].
- *Buffer-overflow Attacks.* In [19, 20], authors detected that bugs in Opensource SDN controllers can lead to loopholes, which adversaries can use to launch buffer-overflow attacks. This can lead to gaining access to the Operating System (OS) console and finally to the whole network domain.
- *NorthBound API interface.* An adversary can write malicious Network applications and publish them in the SDN controller APP stores, as indicated in [21]. The target of the Network applications is to provide a back-door (hosting OS) to the adversary.
- *Web-interface.* All the SDN controllers provide a web interface for network activity monitoring and configuration. The adversary can use this interface to launch software attacks like cross-site scripting, or injection attacks, among others [21].

2.2.2 SDN data plane

Forwarding devices are dumb and have limited resources (Cam table size, Memory, CPU power). Hence, attacks like spoofing, resource exhaustion, and buffer-overflow are widespread in these cases.

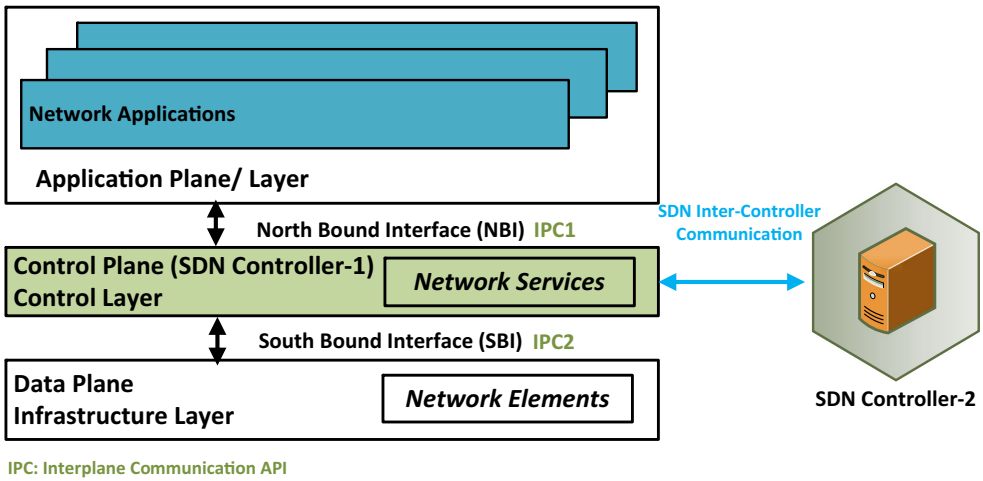

Fig. 1 SDN architecture

- *Authorization Escalation.* Every OpenFlow switch uses Linux kernel or some firmware. OpenFlow switches (HP, Dell, Pica8, etc.) use Linux kernel and adversaries can use this OpenFlow switch operation system surface to gain access to the domain hardware. Switches have open telnet ports which must be configured during the switch setup time. If they are not configured properly, an adversary can brute-force the telnet server of the OpenFlow switch to gain access to the root shell. With successful access, she/he can intercept communications through that switch, install/ change switch flow table, clog the network, launch DDoS attacks towards SDN Controller, or deploy bots for further propagation of the Malware [22].
- *Resource Exhaustion.* An adversary connected to one of the OpenFlow switches can easily know about their hardware configuration by running different fingerprinting tools. Once the adversary knows the hardware limits (Maximum CPU, Memory and CAM table size), she/he can launch specific DDoS attacks to exhaust the resource.
- *DDoS for SDN Controller.* An adversary gaining access to the OpenFlow switch shell or being a malicious host connected to the OpenFlow switches can use malware or custom scripts to generate an enormous amount of packet_in request towards the SDN controller [23]. Typically, these controllers have limited resource processing capability and hence a large amount of packet_in request causes the disruption of the service.
- *Buffer-overflow attack.* OpenFlow Switch OS are based on Linux kernel. As detected in [20], there is a huge possibility to launch Buffer-overflow attacks in this space. An adversary can make a custom script which will target a particular buffer in the OS. Overflowing the buffer will download an executable malware in the switch OS. The target of the malware is to create a shell back door with the goal of using it in the future to get the root privilege of the OS.
- *Privacy leakage of Flows.* As OpenFlow switches can listen to the network traffic, an adversary with root privileges to OpenFlow switches can intercept different types of traffic within the SDN domain such as controller messages, flows from other switches, hosts, or domains, [17].

2.2.3 SDN inter-plane communication APIs

The SDN controller maintains a secure link with the forwarding devices (OpenFlow Switches). For sending control instructions and flow rules to the OpenFlow switches SDN controller uses OpenFlow protocol. This inter-plane communication interface can be attacked in several ways:

- *Interception.* Secured links use TLS for ensuring the privacy and integrity of the communication. However, to improve the performance, most of the SDN controllers disable it. Even with the TLS, the authors of [17] detected that an adversary can launch attacks like SSL stripping, command injection, RC4 algorithm, or compression to exploit their vulnerabilities.
- *Replays or False command packet.* An adversary can replay the controller commands to the OpenFlow switches or the attacker can replay multiple legitimate switch requests to exhaust the controller. In some cases, the adversary can create false packets imitating either controller or OpenFlow switch to establish fake routes, delete legitimate flows, or request fake routes [24].

2.2.4 SDN inter-controller communication

In real world network deployments, SDN controllers need to communicate with other autonomous domain SDN controllers. The available SDN autonomous domain communication protocols mimic BGP (SDNi). However, they have many weaknesses and vulnerabilities listed in their manuals. We have also come up with some additional vulnerabilities during our experimentation.

- *Impersonation.* An adversary can capture a legitimate domain SDN controllers BGP request. Then the attacker replays it to another controller to gain access to the network domain. After gaining access, he can launch many different attacks.
- *Crossfire attacks.* Crossfire attacks are quite common in the inter-domain network space, as indicated in [25]. In a crossfire attack, the attacker cuts off a particular area of network traffic by clogging different domain network traffic. The attacker achieves it by creating a topology map of the surrounding network and then deploying malware to create zombie decoy servers.
- *Coremelt attack.* The authors of [26] proposed this attack, where legitimate communications between interdomain servers or hosts are used to clog the network. First, an adversary captures the inter-domain SDN controllers. The adversary then sends legitimate requests between the controllers. The amount of request decides the strength of the attack. With heavy requests, the communicating controllers and the networks are clogged by the adversary.

In this section we have seen the state-of-the-art in terms of vulnerabilities and attacks affecting the SDN plane of integrated clinical environments, as well as how existing solutions focused on ICE are not able to detect and mitigate cyberattacks in SDN.

3 The ICE scenario and its threat model

The main goal of this section is to highlight the cybersecurity concerns of hospital rooms of the future implementing ICE scenarios. For that, we describe the elements of the ICE framework and provide a threat model detailing the most relevant cyberattacks.

3.1 ICE framework

The ICE framework is a patient-centric architecture for ICS proposed in the ASTM F2761 standard. Among its benefits, it provides en interoperability of heterogeneous medical devices and applications belonging to a clinical scenario. The components making up the framework are the following ones:

- *ICE Equipment Interfaces*. Medical devices without connectivity capabilities are equipped with this interfaces to enable their networking capabilities.
- *ICE Supervisor*. Platform able to host medical applications that make clinical decision according to the data received from the previous interfaces and medical devices.
- *ICE Network Controller*. Component making possible the network communications of the different elements (supervisor and interfaces of medical devices, among others), as managing the discovery of new devices.
- *Data Logger*. It stores logs for forensic analysis purposes. The Data logger contains data belonging to three fields. Firstly, it contains information specific to the ICE components. Secondly, network and topology specific information is also stored. Finally, it also keeps information specific to the network applications.
- *External Interface*. Interface enabling communications with external hospital resources such as datasets, clinical systems or Electronic Health Records (EHR).

3.2 Threat model in ICE

This section provides an overview of the different cyberthreats affecting the ICE framework. The threat model shows a diverse range of attack surface for various types of attacks. Figure 2 shows the elements making up the ICE framework as well as their cyberthreats. In the figure, ICE Components are presented using blue blocks (filled in blue), the SDN controller and physical devices are presented using

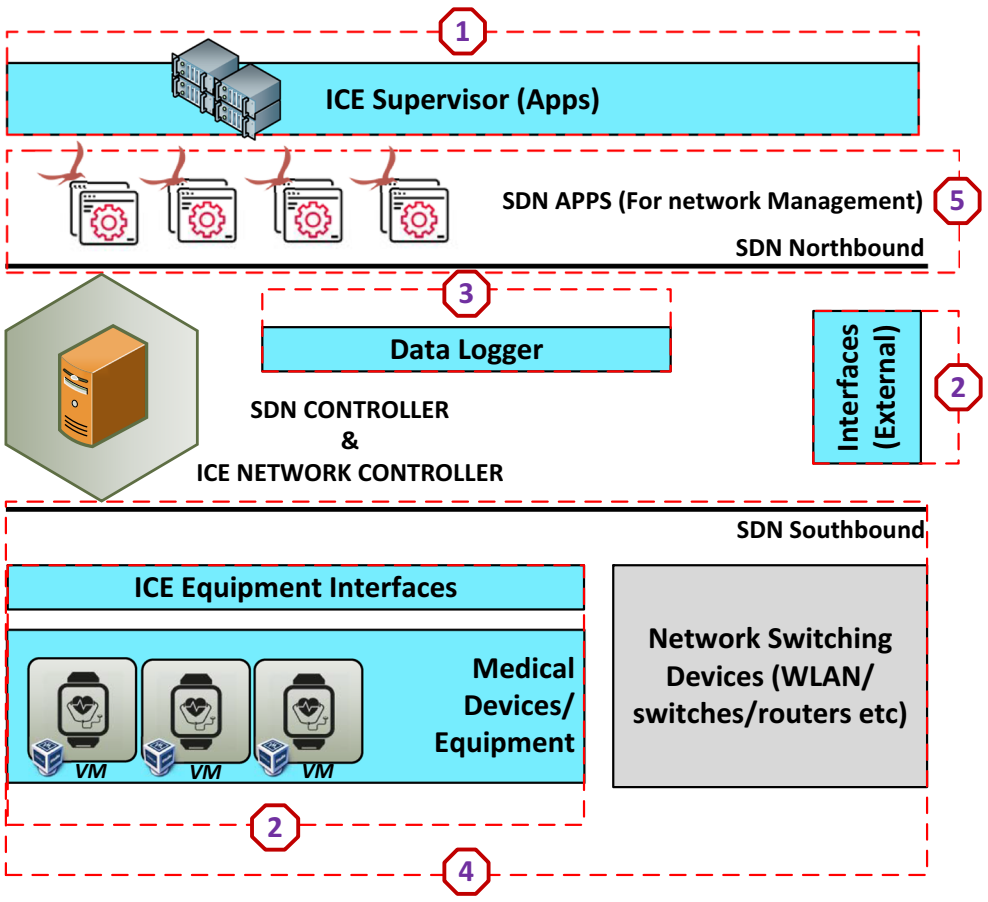

Fig. 2 Threat Model for ICE

block with grey fillings. The threats surrounding a certain component in the architecture are labelled using dashed red-line.

1. *Attacks on the ICE Supervisor.* The ICE Supervisor infrastructure runs on top of a Virtual Machine (VM) where multiple applications (apps) are hosted for supervision. The supervisor has the administrative privileges on this infrastructure. The ICE supervisor can provide clinicians minimal consoles to launch their apps. Third parties can also develop apps for this infrastructure. This infrastructure can be compromised in various ways. For instance, a supervisor can be a rogue entity who can compromise the whole ICE infrastructure. An adversary can steal the supervisors credentials and use them to take complete control over the ICE. The third-party apps can also be malicious and help the adversary to gain a backdoor in the supervisory layer. Using that backdoor, an adversary can launch a further attack to take complete control of the ICE.
2. *Malicious end-users/devices/interfaces.* ICE infrastructure consists of various, ICE compliant wired or wireless devices and interfaces. Here, the interfaces can be used for pluggable ICE compliant devices. End-users or the clinicians can use these devices. In most cases, these devices are resource-constrained and it often becomes very hard to deploy traditional security mechanisms such as encryption, etc. into them. Hence, they are vulnerable to external attacks. An adversary can easily take control of the devices and launch further attacks on the ICE. The adversary can use these devices to spread malware and create zombies. Later, he can launch DDoS attack towards the ICE network Controller.
3. *Malicious Logger.* ICE logs are valuable and one of the major targets for any adversary. There are multiple ways an adversary can manipulate or hijack the data logs. Firstly, a compromised device can send fake information to manipulate the logs. Secondly, the data logger can be a separate VM An adversary can compromise this VM to take control and manipulate the logs. Finally, the manipulated logs can be used to dupe the ICE controller.
4. *Communication Threats.* The ICE Network Controller uses this medium to control and communicate with the medical devices/interfaces. The following attacks are possible in this part.
   - An adversary using one of the medical devices can inject malicious packets in the communication medium. In addition, it can further lead to more severe attacks.
   - An Adversary can launch ARP poisoning attacks to poison the ARP cache of the medical devices. This can be used to launch Man-in-The-Middle (MiTM) attack and hijack personal medical information.
   - An adversary can send fake status report or fake packets to poison the ICE controller topological view as well as the hardware resource pool. This can lead to denial of server attacks.
   - A group of coordinated malicious devices can flood the communication medium with spurious request creating a Distributed Denial of Service (DDoS) attack.
5. *Malicious SDN Apps.* SDN Controller and ICE Network Controller use a range of network application (DHCP, forwarding, etc.) to maintain the connectivity in the ICE. However, this interface can be used to launch software specific attacks like buffer overflow, cross-site scripting etc.

4 Motivating use cases

In this section, we focus on the design of a hospital room of the future implementing the ICE framework and we explain in detail one of the most relevant cyberattacks affecting the SDN plane of our scenario. For each phase of the attack, we highlight the cybersecurity concerns that current ICE solutions are not able to manage.

The next generation of hospitals, also known as hospitals of the future, considers the elements of the ICE framework (explained in Section 3) to create innovative clinical environments. As an example we have created a scenario with a clinical room composed of:

- Two medical devices (A and B) monitoring patients' vital signs and connected each other through the network infrastructure of the hospital. Both medical devices are attached to ICE Equipment Interfaces. The medical devices are virtualized using Linux VMs which host the ICE Equipment Interfaces of the OpenICE software [27]. Each medical device runs on top of an individual VM with networking capabilities.
- One medical supervisor acquiring clinical data from the previous two medical devices, analyzing the data and suggesting doctors medical diagnosis and treatments. This component, as well as the ICE Equipment Interfaces, are connected to the ICE Network Controller to receive/send data from/to the ICE Equipment Interfaces. The medical supervisor logic is implemented by the ICE Supervisor and it runs on top of a dedicated machine.
- An SDN-enabled network with switches and routers connecting the two medical devices and the medical supervisor. The network has an SDN controller

managing the control plane of the SDN paradigm. This component is deployed together with the ICE Network Controller. In addition, there are SDN apps to manage the network connectivity, switching and routing responsibilities of the ICE scenario. The Data Logger forms part of SDN Controller core application modules for capturing network events.

In our hospital room of the future, the two medical devices A and B monitor different vital signs of a patient (blood pressure, pulse, temperature, respiration, etc.) and send the data to the medical supervisor. In a legitimate case, from the SDN domain perspective, when the medical device A sends the first packet to the nearest OpenFlow switch (SW), it does not have any flow to route the packet to the medical supervisor. Therefore, it forwards the packet to the SDN Controller. The SDN Controller with the supervision of the ICE Network Controller application instructs to forward the packet to the destination, the medical supervisor. To do it, the SDN Controller installs a flow rule in the SW and the medical device A communicates with the medical supervisor. For the next network packets, the same flow rule in the SW will allow them to communicate with each other. Figure 3 shows the main elements making up our scenario.

In the following subsections, we show the different stages of a well-known cyberattack affecting the SDN and the limitation of current ICE-based solutions to detect and mitigate each stage.

4.1 First stage: ARP cache poisoning

Address Resolution Protocol (ARP) is an integral part of Internet Protocol (IP). The ARP protocol is used to resolve the network addresses of the devices connected to the network. To find neighbour devices, each device maintains a cache memory called ARP cache. In this cache, each device stores a mapping of the MAC addresses against the IP addresses of the neighbour devices (devices in the same network). This mapped helps any device to locate the nearest router and the communicating host. However, due to the lack of proper security mechanisms in the

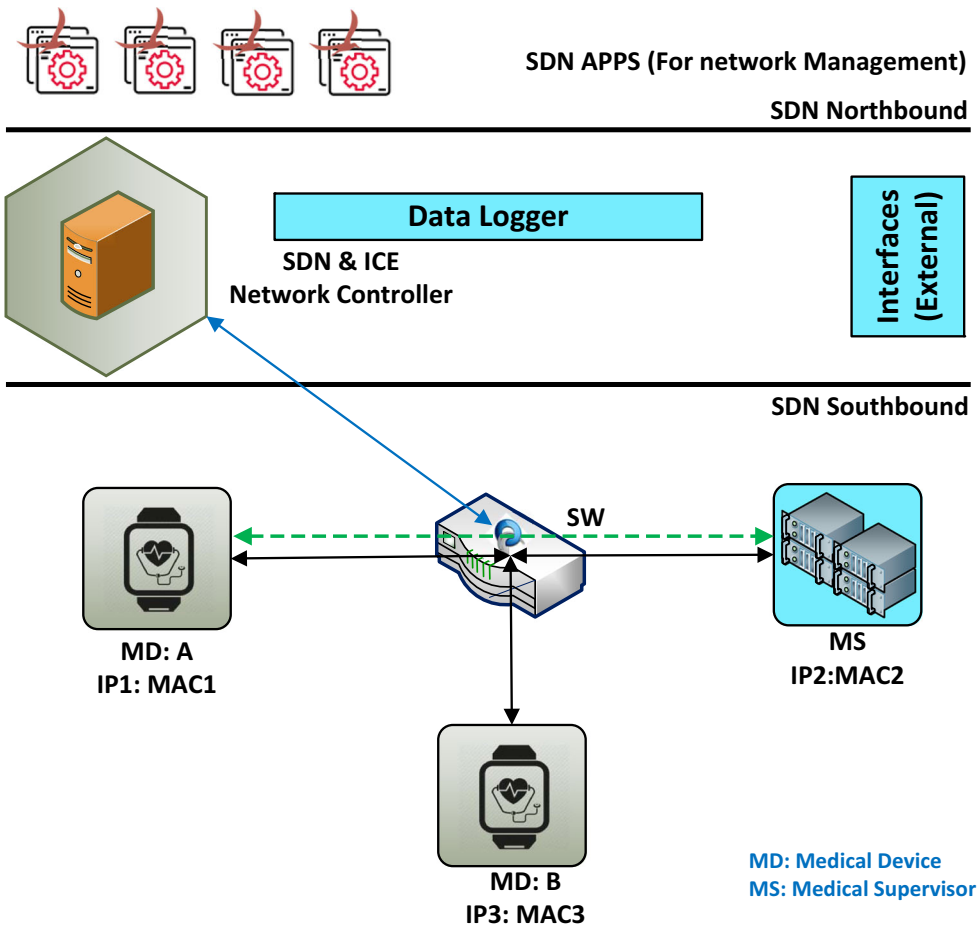

Fig. 3 Configuration of our hospital room of the future integrating the ICE framework and the SDN paradigm

networking infrastructure, this cache can be manipulated by an adversary. This attack is known as ARP poisoning or ARP spoofing attack [28].

In our hospital room of the future, we assume that the medical device B has been infected and it acts in a malicious way to poison the ARP cache of the medical device A. To poison the ARP cache of the medical device A, the medical device B sends a legitimate flow request to the SDN Controller. The main objective of the previous action is to deploy a flow rule in the network infrastructure (switch) and establish a route to the victim device. Then, the medical device B forges a gratuitous ARP request to poison the victim APR cache. This packet contains forged header information which is used to poison the ARP cache mapping. Concretely, the modified header associates in the ARP Cache of the medical device A, the IP of the medical supervisor with the MAC address of the attacker.

Concern 1 At this point of the attack, we require an ICE-based solution able to detect and mitigate automatically the poisoning of ARP Caches in real time and on-demand.

4.2 Second stage: man-in-the-middle (MiTM)

After the first stage of the cyberattack, the ARP Cache of the medical device A is poisoned. Now, the attacker (medical device B), perform the same steps to poison the ARP cache of the medical supervisor. After poisoning both ARP caches, the network packets of both devices always go through medical device B (adversary). This essentially allows the adversary to view, copy or modify any data contained in network packets sent or received by both devices. At this point, the adversary can use any packet capturing tool to intercept the flow communication, for instance, Wireshark. Hence, medical device B will be able to intercept any valuable information of medical device A and medical supervisor.

To describe the impact of this stage we present the following use case. In our scenario, the medical device A uses an embedded credential to log data to the medical supervisor. The target of the adversary is to steal this embedded credential. After the devices are compromised in stage 1, the adversary uses Wireshark to capture the communication going through it. Once the device posts the credential towards the server, the flow is captured by the adversary. Even if the Apps server uses HTTPS, it is possible to decrypt the traffic to recover the credentials. Figure 4 shows the scenario and the changes after (in red) and before (in green) the attack.

Concern 2 MiTM attacks in healthcare devices are critical, as it allows the adversary to inject/manipulate and intercept personal health-related data. Current ICE-based solutions fail to detect and mitigate such interception and tampering of data.

4.3 Third stage: phantom storm

In the last stage of the attack, the service disruption in the ICE network infrastructure is achieved. Stages 1 and 2, discussed above, are the prerequisites for exploiting this cyberattack. After performing stages 1 and 2, a phantom storm attack is executed to create a fake host (ghost) and fool the SDN and ICE controller [23].

Following our scenario, to launch the phantom storm attack, the adversary (medical device B) uses the ARP cache poisoning attack to introduce a fake medical device, the medical device C. To achieve it, the ARP caches of the medical device A and the medical supervisor will be updated to make them believe that there exists the medical device C. Then, the adversary sends forged packets to each medical device within the network domain. The purpose of the forged packets is to send a response to the fake medical device. The legitimate medical devices send the respective responses as packet_in messages. As the destination medical device is fake and SDN & ICE network controller is unaware of the network connection node of the fake medical device, it will start to flood packets looking for this device. The adversary will keep on sending such packets, and the controller will keep flooding the whole ICE network domain. Such a situation will cause a distributed denial of service and the disruption of regular operation of the ICE infrastructure. Figure 5 shows the scenario and the changes after the attack (in red).

Concern 3 Existing ICE solutions are not able to detect and mitigate DDoS attacks in real time. Hence we need ICE-based security mechanisms to prevent such DDoS attack.

5 Architecture

This section describes the components of our architecture that combines the SDN paradigm with NFV techniques to detect and mitigate cyberattacks affecting the SDN plane of ICE. The proposed architecture has been deployed at the edge of the network. That means, close to the geographical location of the patients, medical devices and interfaces comprising the ICE scenario. As an advantage, the architecture components dealing with the scenario management only suffer a minimal latency and delay in communications. Figure 6 illustrates the layers and components making up our architecture. At this point, it is important to note that the proposed architecture is generic

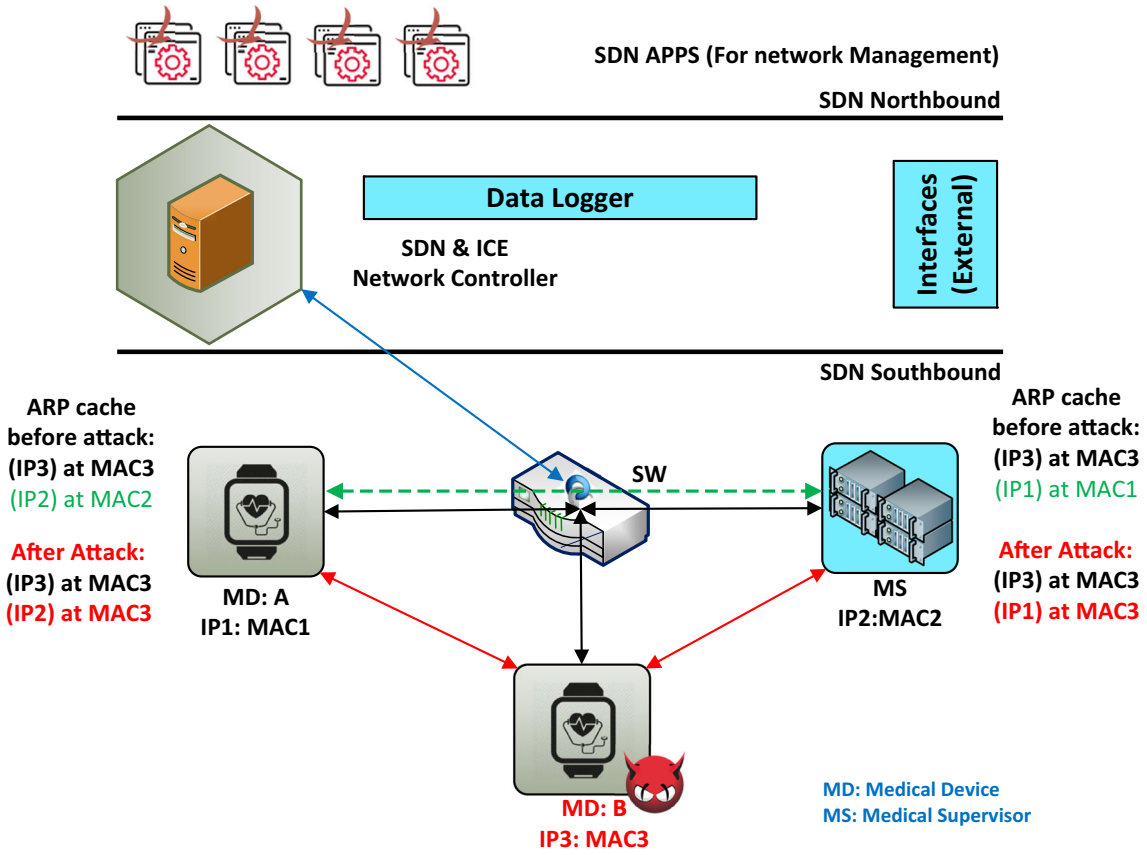

Fig. 4 MiMT attack affecting our hospital room of the future

and extensible enough to deal not only with the cyberthreat of the motivating example but also with others affecting Cyber-Physical Systems equipped with SDN.

The upper layer is called *System Management* and provides an overview of the status of the ICE elements and SDN-based network infrastructure. This layer makes automatic and on-demand decisions to guarantee the security of the ICE elements. Decisions to mitigate particular cyberattacks against the SDN plane are enforced by the lowers of layers. The *Operation Support System* (OSS) deals with the logic of the scenario and exposes an interface to establish policies able to detect and mitigate cyberattacks. This layer also allocates the *Monitor*, *Analyzer*, *Decision* and *Orchestrator* components. In a nutshell, the Monitor is in charge of gathering network packets sent and received by ICE medical supervisor and medical devices in real time. The Analyser allows the administrator to process network packets in different ways to extract relevant metrics suitable for detecting cyberattacks affecting the SDN plane. These metrics are critical for the Decision component, which detects cyberattacks and decides mitigation mechanisms according to the predefined policies and metrics. Finally, the Orchestrator component orchestrates and interacts with the lower layers to enforce the mitigation actions.

In the middle layer, the *VF Manager* (VFM) and the *Virtualized Functions* (VF) are responsible for managing and allocating virtual functions. On the one hand, the VFM is composed of different managers belonging to the control plane. Among them we highlight the *Security Manager* and the *ICE Manager*, focused on deploying, controlling, and monitoring, *Virtualized Functions* (VF) focused on ensuring the security of the ICE elements and the clinical scenario. In the same layer, the VF represents the data plane and create, manage, and dismantle VFs running on the VMs exposed by the lower layer. It provides flexibility and cost-efficiency during detection and mitigation of cyberattacks. As an example of VFs related to ICE, we could have the ICE Supervisor, the Data Logger, or the ICE Equipment Interfaces (which has not been included in the figure due to room limitations). From the security perspective, different VFs such as Intrusion Detection Systems (IDS), Deep Packer Inspectors (DPI), Honeynets, and many others could be considered.

The *SDN controller* manages the network communications and connectivity of the ICE elements in real time and

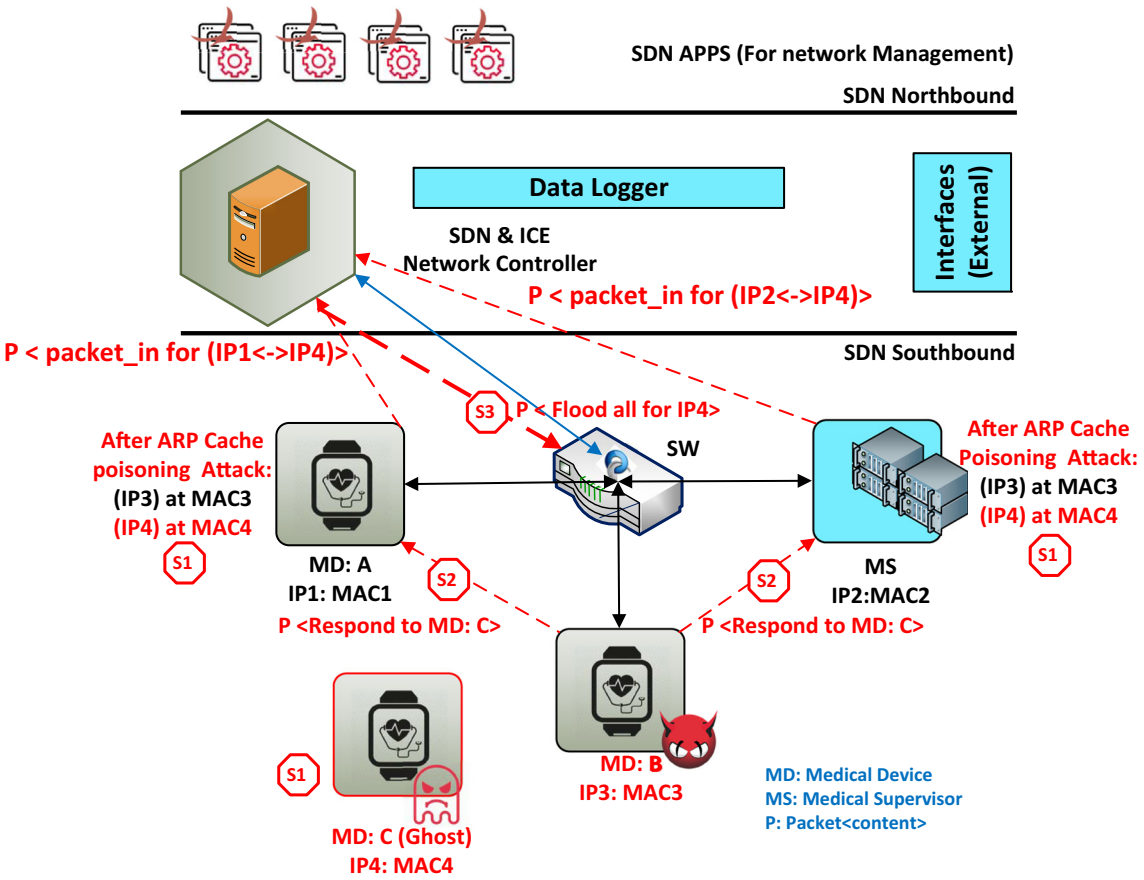

Fig. 5 Phantom storm attack affecting our hospital room of the future. S1: Adversary poisons the ARP cache of MS and MD:A with a fake host (MD:C). S2: Adversary sends forged packet to all hosts to respond to MD:C. S3: Upon packet_in request from all the hosts the Controller floods the network

on-demand. This component interacts with the hardware and VMs of the clinical environment.

Finally, the lower layer contains the data and control plane NFV technologies. In the control plane, the *Virtualized Infrastructure Manager* (VIM) creates, manages and monitors the life cycle of VMs running on top of *Virtualized Infrastructure* instantiated over generic *Physical Resources*.

5.1 Detection and mitigation policies

The schema of the policies managed by our architecture to detect and mitigate cyberattacks affecting the SDN plane in ICE is defined as follows:

$$Device \wedge Metric \wedge Location \wedge Date \rightarrow Result$$

Device is the medical device managed by the policy; *Metric* is the parameter considered by the policy to detect the attack; *Location* is the geographical position in which the policy is applied; *Date* is optional and establishes the moment when the policy will be implemented; *Result* determines the countermeasures to mitigate the detected attacks; $\wedge$ means "and"; and $\rightarrow$ means "then". It is worthy to note that the consequent of the policy is the Result, while the rest of fields are the antecedent. In terms of results (countermeasures), for the hospital of the future hospital we point out the management of the SDN plane and ICE elements such as medical supervisors, equipment interfaces, or loggers.

6 Experimental use case deployment, detection and mitigation

In this section, we implement the attack explained in the motivating example and demonstrate how our architecture is able to manage the three concerns highlighted in Section 3. For each stage of the cyberattack, we show its implementation and the detection and Orchestrator mechanisms orchestrated by our solution in real time and on-demand.

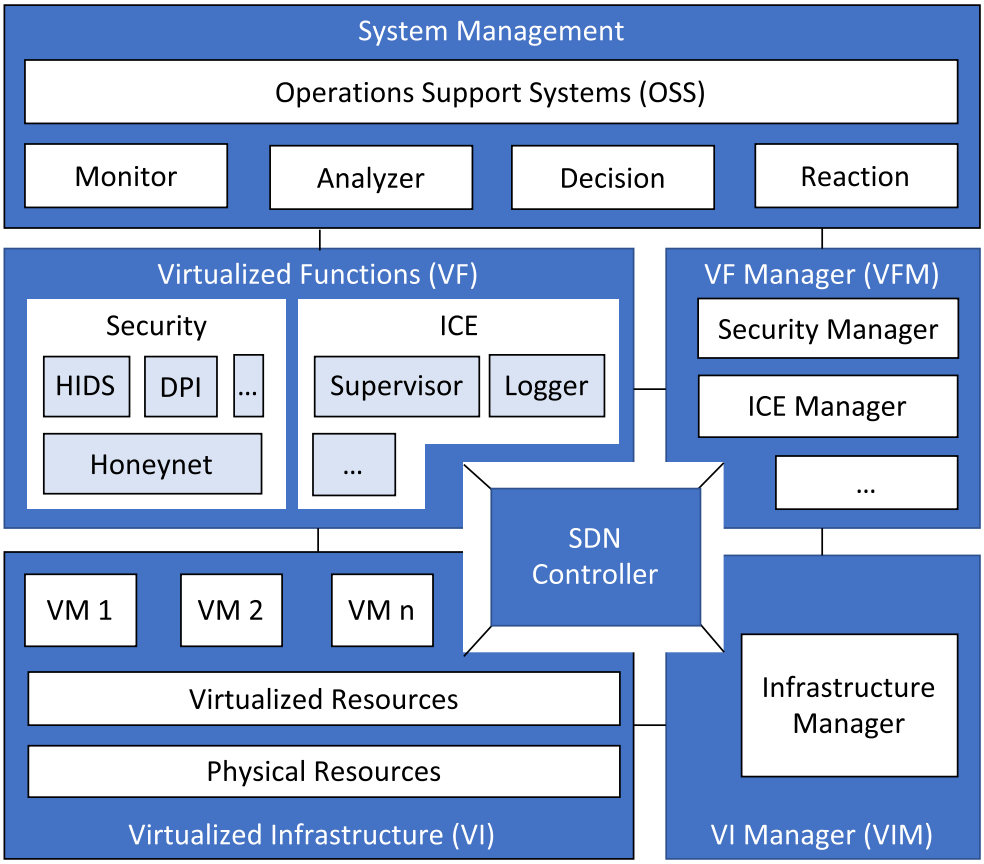

Fig. 6 SDN/NFV Architecture providing cybersecurity in ICE

6.1 Use case set-up

The hospital room of the future (detailed in Section 4) has been emulated over VMware NSX cluster at the Advanced Cyber Security Engineering Research Centre (ACSRC), University of Newcastle. The cluster uses three Dell Power Edge M640 Server, each server consisting of two Intel Xeon Gold 6126 @ 2.6 GHz processors and 384 GB(6x64) of RAM. We have used ONOS as the SDN Controller which runs over an Ubuntu 18.04 VM and mininet to create and manage the data plane infrastructure. The ICE supervisor and ICE Equipment Interfaces devices are created using OpenICE and are mapped into the mininet virtual hosts. In this experimental set-up, the medical supervisor and two medical devices indicated in Section 4 have the following names, IP and MAC addresses:

- *Medical Supervisor (MS) - Name: Node h101, IP: 192.168.56.101, MAC: 46:70:e1:d2:8f:fc*
- *Medical Device A (MDA) - Name: Node h102, IP: 192.168.56.102, MAC: 22:f0:6e:00:09:fd*
- *Medical Device B (MDB) - Name: Node h103, IP: 192.168.56.103, MAC: 56:eb:f4:de:2d:52*

6.2 First stage: ARP cache poisoning

Once having the scenario set-up, the MDB (attacker) executes the attack by using the SDNPWN script. On top of Fig. 7 it is shown the console of MDB when the ARP cache poisoning attack is executed against the MDA. On the bottom, it can be seen how before the attack the content of MDA ARP cache is right (highlighted in green) and after the attack, the target IP 192.168.56.101 in MDA was mapped to the MDB MAC address (highlighted in red).

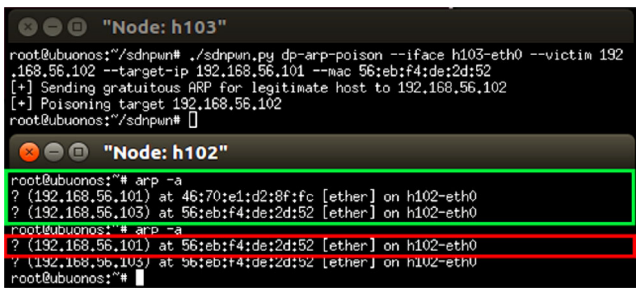

Fig. 7 ARP poisoning attack on the ICE scenario

To detect this ARP cache poisoning attack and the first concern of Section 4, the network administrator defines the next network management policy. This policy gets the catalogue of the SDN controller to obtain the IP address of the medical supervisor (MS), which was registered during the scenario set-up. After that, the policy inspects the ARP Cache of the medical device A (MDA) and checks if the IP address of the medical supervisor stored in the catalogue is the same as the one stored in the ARP cache. If not, an ARP cache poisoning attack is on-going and the policy replaces the ARP cache of the infected medical device A (MDA).

SDNController(?controller) ∧ hasCatalog(?controller,?catalog) ∧ hasDevice(?catalog,#MS) ∧ hasIP(?catalog,?medSuperIP) ∧ hasDevice(?catalog,#MDA) ∧ hasARPCache(catalog,?arpCacheMDA) ∧ hasIP(?arpMD,#MDA) ∧ notEqual(?arpCacheMDA,?medSuperIP)
→ replaceARPCache(#MDA,?arpCacheMDA)

The next mitigation policy specifies the steps required by our architecture to replace the ARP Cache of MDA and mitigate the ARP cache poisoning attack.

Orchestrator(?orch) ∧ hasInstance(?orch,?instance) ∧ hasVmMDA(?instance,?vmMDA) ∧ hasArpCache(?vmMDA,?insVnfArpCacheMDA) ∧ hasCatalog(?orch,?catalog) ∧ hasArpCache(?catalog,?catVnfArpCacheMDA) → VFM(?vfm) ∧ deploy(?vfm,?catVnfArpCacheMDA) ∧ dismantle(?vfm, ?insVnfArpCacheMDA)

To deploy the mitigation action, the Orchestrator module receives the notification and looks at the deployed instances to find the VM where the MDA has been deployed and the VNF implementing the ARP Cache of MDA. After finding them, the orchestrator looks at its catalog to find the descriptor of a generic VNF ARP Cache. Once detected, it checks if the VM has enough computing, storage and networking resources to allocate the new VNF (not included in the policy for simplicity sake). If so, it deploys the VNF implementing the new ARP cache and dismantle the old ARP cache (the infected one). The action is communicated to the Orchestrator component, which updates the catalogues with the information of the new VNF.

6.3 Second stage: MiTM

In this stage of the attack, the adversary (MDB) launches a MiTM attack towards MS and MDA. To implement it, MDB executes the ARP Cache Poisoning attack and modifies the ARP Cache of MDA and MS. After the successful attack, MDB uses Wireshark to intercept the network packets exchanged by both devices and obtain sensitive data (as shown in Fig. 8).

Regarding the second concern, which focuses on the detection and mitigation of a MiTM attack, the administrator defines the following policy to detect medical devices receiving and sending the same network packets in a short period of time. For that reason, the SDN controller gathers the packages sent and received by the medical device B (MDB) and compares the sizes and timestamps of the received and sent network packets. If their values are in a given range (predefined by the administrator), the MiMT attack is detected, and the policy replaces the MDB

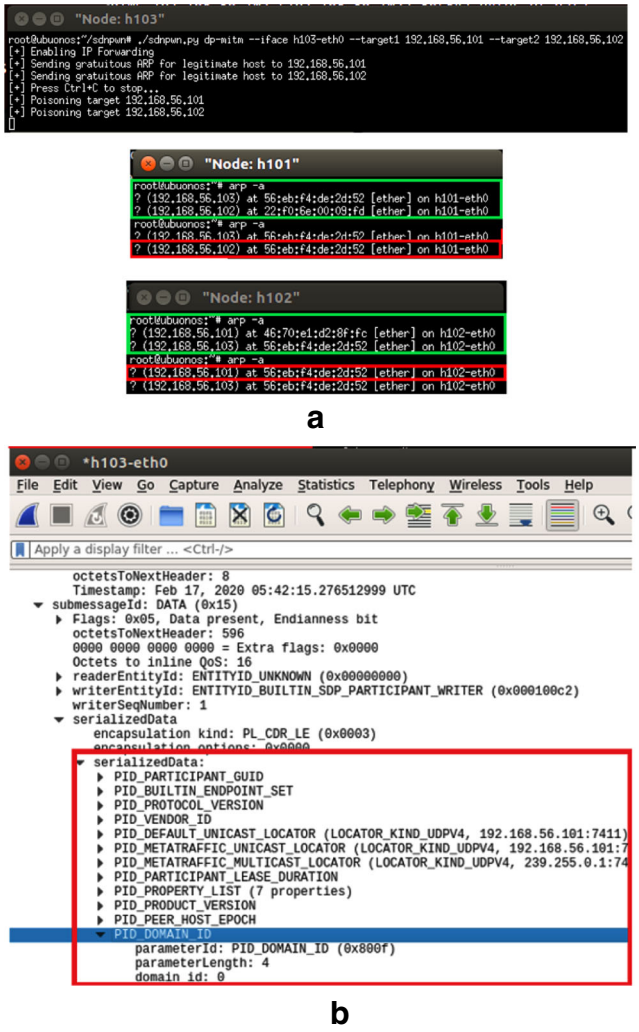

Fig. 8 MiTM attack affecting our ICE scenario. **a** Terminal view of Node h103: 192.168.56.103 (a malicious adversary) running *sdnpwn.py* script towards target 1 (Node h101:192.168.56.101) and target 2 (Node h102:192.168.56.102). The script launches ARP poisoning attack and the terminal view shows the ARP cache of node h102 and h102. **b** Wireshark network packet captured by the malicious adversary for ICE specific information

SDNController(?controller) ∧ hasTopology(?controller,?topology) ∧ hasDevice(?topology,#MDB) ∧ hasReveivedPk(?topology,?in_pkt) ∧ hasSentPk(?topology,?out_pkt) ∧ hasTimestamp(?in_pkt,?in_timestamp) ∧ hasTimestamp(?out_pkt,?out_timestamp) ∧ inRange(?in_timestamp,?out_timestamp) ∧ hasSize(?in_pkt,?in_size) ∧ hasSize(?out_pkt,?out_size) ∧ inRange(?in_size,?out_size) → replace(#MD_B,?iceInterface)

The next mitigation policy specifies the steps required by our architecture to mitigate the MiMT, and replace and configure the VM and VNF implementing MDB.

Orchestrator(?orch) ∧ hasInstance(?orch,?instance) ∧ hasVmMDB(?instance,?insVmMDB) ∧ hasCatalog(?orch,?catalog) ∧ hasVmMDB(?catalog,?catVmMDB) ∧ hasVnfMDB(?catalog,?catVnfMDB) → VIM(?vim) ∧ dismantle(?vim,?insVmMDB) ∧ deploy(?vfm,?catVmMDB) ∧ VFM(?vfm) ∧ deploy&configure(?vfm,?catVnfMDB)

To enforce the rule to replace the MDB, once notified the Orchestrator, it interacts with the VIM to deploy the new ICE Equipment Interface in the available hardware (steps 1 and 2 of Fig. 9). When the VIM gets the demand, it dismantles the VM hosting the MDB. Once the Orchestrator component receives the confirmation (step 4), it interacts with the VIM to create a new VM to host the new MDB (step 5). When the VIM receives the notification, it creates a new VM with the MDB (ICE Equipment Interface) on top of the existing infrastructure (step 6). The action is confirmed (step 7) and the Orchestrator component updates the instances and catalogues. Once the new MDB is deployed, the Orchestrator sends the configuration of the virtual medical device to the VFM (step 8). THe IP address of the medical supervisor or the frequency of communication are some of the parameters exchanged during the previous communication. After that, VFM configures the MDB with those parameters and when the configuration is finished, the FVM and the Orchestrator receives the confirmation (steps 10 and 11).

6.4 Third stage: phantom storm

In this attack, the adversary (MDB) uses the SDNPWN toolkit to launch a phantom-storm attack. As described in Section 4.3, the malicious script creates a fake or phantom host, which IP address is 192.168.56.104. This fake host sends a fake communication request to the medical supervisor and the MDA device. After that, both the medical supervisor and MDA requests the SDN controller

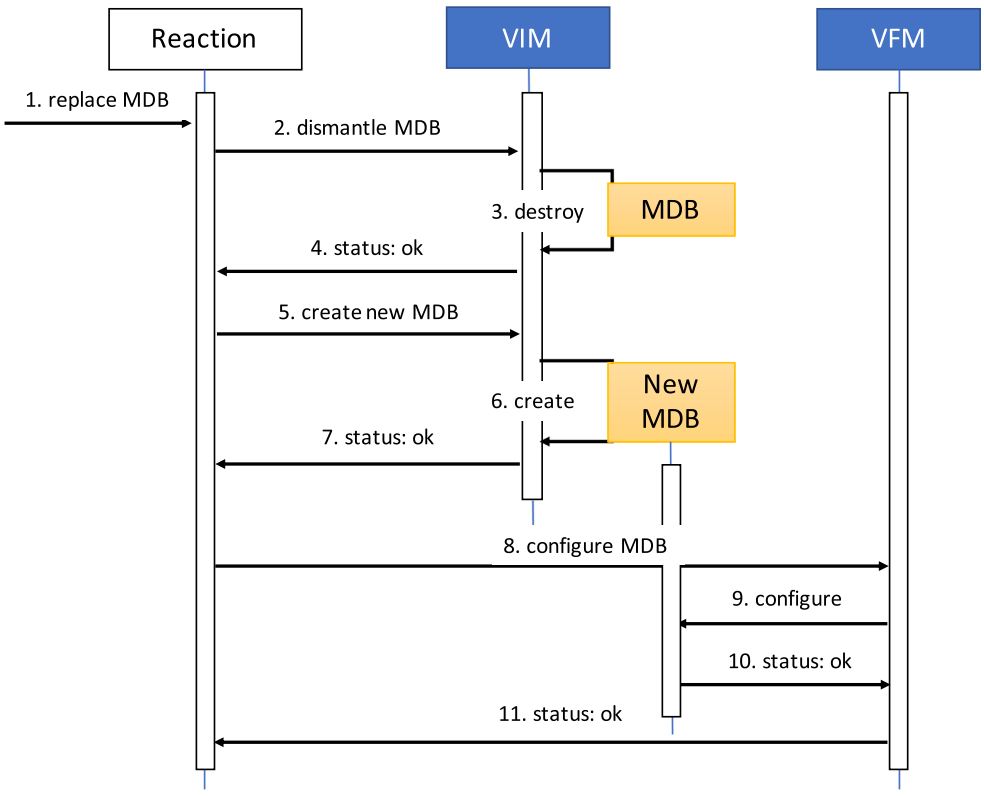

Fig. 9 Medical device B replacement

to establish a communication with the phantom host. However, in this scenario, there does not exist any legitimate device with the IP 192.168.56.104. Hence, the controller is unaware of the physical connection point of this particular device. Finally, the SDN controller floods the network with packets searching for 192.168.56.104. Figure 10 presents a successful Phantom-storm attack. In the GUI interface, it is visible that a single host has two IPs assigned in the same interface. This behaviour is due to ONOS GUI classifying the hosts, based on the packet source.

To demonstrate the impact of this attack, we have measured the average delay between the MS and MDA before and after the attack. The average delay before the attack was $0.042ms$ and after the attack became $0.296ms$, more than 500%. It is important to note that we have considered a small network with four hosts and having a larger network with much more devices, the delay will increase.

To address the third and last concern of Section 4, the administrator of our architecture defines the following policy able to detect and mitigate the phantom storm cyberattack.

SDNController(?controller) ∧ hasTopology(?controller,?topology) ∧ hasNetworkPacket(?controller,?pkt) ∧ hasDestination(?pkt,?medDev) ∧ doesNotBelong(?medDev,?topology) → discard(?controller,?pkt)

This policy obtains the network topology through the SDN controller, after that it monitors the network packets traveling across the network to check if destination device belongs to the network, or in contrast it is a ghost. If so, the policy detects the phantom storm attack and discard the network packets with the IP of the ghost medical device. To discard those network packets a OpenFlow rule drooping the network packets to MDB.

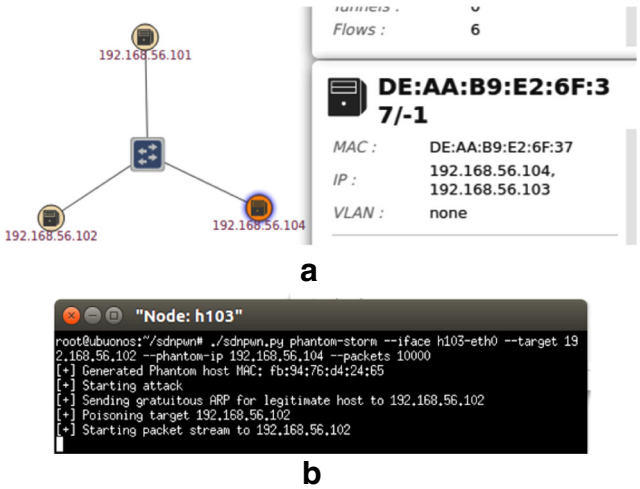

Fig. 10 Phantom-storm Attack on the ICE scenario **a** ONOS GUI showing two IP addresses assigned to the same physical interface. **b** Terminal view of the adversary while launching a Phantom storm attack

6.5 Experimental results

The aim of this section is to measure the performance of the proposed solution when replacing an infected medical device due to the cyberattack detailed in Section 4. With that goal in mind, we performed several experiments measuring the time required by our architecture to deploy and launch a VM. Previously, the VM has been configured with different operating systems and an instance of OpenICE that simulates the behaviour of a medical device (steps 5 and 6 of Fig. 9).

In order to measure the architectural performance, we have considered the following two different hardware configurations:

- Personal computer. Equipped with 16GB of RAM and Quad Core Intel(R) CPU i7-3770 at 3.40 GHz. The operating system is Ubuntu 16.04 LTS Desktop
- Server. Equipped with 64GB or RAM, a 36 Core Intel(R) Xeon(R) CPU E5-2697 v4 at 2.30GHz, and Ubuntu 16.04 LTS Server as operating system.

Considering the two previous hardware configurations, we have followed the next steps to measure the performance of our solution:

1. Create three VMs with 1GB of RAM, a single-core CPU, and 10 GB of hard disk.
2. Install three diverse operating systems (one per VM) with different resource requirements (openSUSE Leap 42.3, Ubuntu 16.04 LTS Server, and Windows 7).
3. Install and configure the OpenICE software and instantiate an ICE Equipment Interface simulating a medical device.
4. Measure the time required to instantiate the VMs in each of the two hardware configurations. For each hardware and operating system configuration, we have executed the previous steps 10 times and calculated the mean value.

Table 1 shows the times obtained for the deployment and initiation of the ICE Equipment Interface running on top of different software and hardware configurations.

The main conclusions obtained after performing the previous experiment is that the time required to deploy the ICE Equipment Interface is inversely proportional to

Table 1 Deployment time of a new medical device instantiated with OpenICE

Hardware	VM deployment and OpenICE instantiation		
Details	openSUSE Leap	Ubuntu 16.04	Windows 7
Personal Computer	7.536 s	8.694 s	15.946 s
Server	6.864 s	7.972 s	12.623 s

the amount of computational resources needed. As can be seen in Table 1, the selection of the operating systems is also an important decision. Specially in integrated clinical environments where the time is critical to ensure the patients' safety.

In conclusion, this section has demonstrated that the proposed architecture and its associated policies are able to detect the different phases of a relevant cyberthreat affecting the SDN paradigm. Furthermore, the mitigation mechanism adopted by our architecture is able to replace an infected medical device in a time ranging from 6.864 s to 15.946 s depending on the underlying hardware and operating system. Also, it is important to keep a trade-off between the purpose of the clinical scenario and the selection of hardware and software to virtualize medical equipment.

7 Conclusion and future work

In this work, we have proposed a novel architecture oriented to the mobile edge computing combining ICE and SDN/NFV. The proposed architecture is able to detect in real time cyberattacks affecting the SDN plane of ICE scenarios. Different policies have demonstrated, as proof of concept, the feasibility of our solution to detect different phases of a relevant cyberattacks affecting the SDN plane. Additionally, two use cases have demonstrated the added value of the proposed solution compared to the state-of-the-art. Finally, several experiments have demonstrated that the proposed architecture is able to detect and mitigate the cyberattacks in an acceptable response time.

As future work, we plan to extend our architecture with intelligent components able to detect new cyberattacks affecting not only the SDN plane but also the medical devices making up the ICE scenario. In addition, we also plan to evaluate the architecture with those attacks and validate its feasibility.

Acknowledgements This work has been funded by the Government of Ireland, through the IRC post-doc fellowship (grant code GOIPD/2018/466), by the Spanish Government grant with code RYC-2015-18210 (co-funded by the European Social Fund), by Armasuisse S+T with project code CYD-C-2020003, and by the Univerity of Zürich UZH.

Funding Open Access funding provided by Universität Zürich.

References

1. Stankovic JA (2016) Research directions for cyber physical systems in wirelessand mobile healthcare. ACM Trans Cyber-Phys Syst 1(1):1:1–1:12. https://doi.org/10.1145/2899006
2. 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. Standard ASTM, International, West Conshohocken, PA
3. Huertas Celdrán A, Gil Pérez M, García Clemente FJ, Martínez Pérez G (2018b) Sustainable securing of medical cyber-physical systems for the healthcare of the future. Sustainable Computing: Informatics and Systems 19:138–146. https://doi.org/10.1016/j.suscom.2018.02.010
4. Gómez Mármol F, Gil Pérez M, Martínez Pérez G (2016) I don't trust ICT: research challenges in cyber security. In: 10th IFIP WG 11.11 international conference on trust management (IFIPTM 2016), Darmstadt, Germany, IFIPAICT, vol 473, pp 129–136. https://doi.org/10.1007/978-3-319-41354-99
5. Scott-Hayward S, O'Callaghan G, Sezer S (2013) Sdn security: a survey. In: 2013 IEEE SDN for future networks and services (SDN4FNS), pp 1–7. https://doi.org/10.1109/SDN4FNS.2013.6702553
6. Benton K, Camp LJ, Small C (2013) Openflow vulnerability assessment. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in softwaredefined networking, pp 151–152
7. Nespoli P, Papamartzivanos D, Gómez Mármol F, Kambourakis G (2018) Optimal countermeasures selection against cyber attacks: a comprehensive survey on reaction frameworks. IEEE Communications Surveys and Tutorials 20(2):1361–1396. https://doi.org/10.1109/COMST.2017.2781126
8. Díaz López DO, Dólera Tormo G, Gómez Mármol F, Martínez Pérez G (2016) Dynamic counter-measures for risk-based access control systems: an evolutive approach. Future Generation Computer Systems 55:321–335. https://doi.org/10.1016/j.future.2014.10.012

9. Dunhill J (2020) Critical patient dies after cyber attack disables hospital computers. https://www.iflscience.com/technology/critical-patient-dies-after-cyber-attack-disables-hospital-computers/. Online; Accessed 28 Oct 2020
10. Huertas Celdrán A, García Clemente FJ, Weimer J, Lee I (2018) Ice++: im-proving security, QoS, and high availability of medical cyber-physical systems through mobile edge computing. In: IEEE 20th international conference one-health networking, applications and services (Healthcom), pp 1–8. https://doi.org/10.1109/HealthCom.2018.8531185
11. Fernández Maimó L, Huertas Celdrán A, Perales Gómez AL, García Clemente FJ, Weimer J, Lee I (2019) Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. Sensors 19(5). https://doi.org/10.3390/s19051114
12. S Hamed DA, Goldman J (2016) Toward a safe and secure medical internet of things. IIC Journal of Innovation
13. Nguyen H, Acharya B, Ivanov R, Haeberlen A, Phan LTX, Sokolsky O, Walker J, Weimer J, Hanson W, Lee I (2016) Cloud-based secure logger for medical devices. In: Proceedings of the IEEE first international conference on connected health: applications, systems and engineering technologies (CHASE), pp 89–94. https://doi.org/10.1109/CHASE.2016.48
14. Cheng L, Li Z, Zhang Y, Zhang Y, Lee I (2017) Protecting interoperable clinical environment with authentication. SIGBED Rev 14(2):34–43. https://doi.org/10.1145/3076125.3076129
15. Cabaj K, Gregorczyk M, Mazurczy W (2018) Software-defined networking-based crypto ransomware detection using http traffic characteristics. Comput Electr Eng 66:353–368. https://doi.org/10.1016/j.compeleceng.2017.10.012
16. Cabaj K, Mazurczyk W (2016) Using software-defined networking for ransomwaremitigation: the case of cryptowall. IEEE Netw 30(6):14–20. https://doi.org/10.1109/MNET.2016.1600110NM
17. Benton K, Camp LJ, Small C (2013) Openflow vulnerability assessment. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, pp 151–152. https://doi.org/10.1145/2491185.2491222
18. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Du-rumeric Z, Halderman JA, Invernizzi L, Kallitsis M, et al. (2017) Understanding the Mirai Botnet. In: 26th USENIX security symposium (USENIX security 17), pp 1093–1110
19. Xiao F, Zhang J, Huang J, Gu G, Wu D, Liu P (2020) Unexpected data dependency creation and chaining: a new attack to sdn. In: 2020 IEEE symposium on security and privacy (SP), pp 264–278
20. Luo P, Zou D, Du Y, Jin H, Liu C, Shen J (2020) Static detection of real-world buffer overflow induced by loop. Computers & Security 89:101,616. https://doi.org/10.1016/j.cose.2019.101616
21. Lee S, Kim J, Woo S, Yoon C, Scott-Hayward S, Yegneswaran V, Porras P, Shin S (2020) A comprehensive security assessment framework for software-defined networks. Computers & Security: 91: 101720
22. Shin S, Lee S, Kim J, Porras P, Yegneswaran V (2017) Athena: a framework for scalable anomaly detection in software-defined networks. In: The 47th IEEE/IFIP international conference on dependable systems and networks (2017). IEEE Communications Society
23. Smyth D, Cionca V, McSweeney S, O'Shea D (2016) Exploiting pitfalls in software-defined networking implementation. In: 2016 International conference on cyber security and protection of digital services (Cyber Security). IEEE, pp 1–8
24. Cao J, Li Q, Xie R, Sun K, Gu G, Xu M, Yang Y (2019) The crosspath attack: disrupting the SDN control channel via shared links. In: 28th USENIXSecurity symposium (USENIXSecurity, vol 19, pp 19–36
25. Kang MS, Lee SB, Gligor VD (2013) The crossfire attack. In: 2013 IEEE sympo-sium on security and privacy. IEEE, pp 127–141
26. Studer A, Perrig A (2009) The coremelt attack. In: European symposium on research in computer security. Springer, pp 37–52
27. David AJP, Goldman JM (2017) Openice medical device interoperability plat-form overview and requirement analysis. In: Biomedical engineering / Biomedizin ische Technik, pp 39–47. https://doi.org/10.1515/bmt-2017-0040
28. Prabadevi B, Jeyanthi N, Abraham A (2020) An analysis of security solutions for ARP poisoning attacks and its effects on medical computing. International Journal of System Assurance Engineering and Management 11(1):1–14

Affiliations

Alberto Huertas Celdrán[1,2] · Kallol Krishna Karmakar[3] · Félix Gómez Mármol[4] · Vijay Varadharajan[5]

Kallol Krishna Karmakar
KallolKrishna.Karmakar@newcastle.edu.au

Félix Gómez Mármol
felixgm@um.es

Vijay Varadharajan
Vijay.Varadharajan@newcastle.edu.au

1 Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, Zürich, CH 8050, Switzerland

2 Telecommunications Software, Systems Group, Waterford Institute of Technology, Waterford, Ireland

3 University of Newcastle, New South Wales, Australia

4 Departamento de Ingeniería de la Información y las Comunicaciones, University of Murcia, Murcia, 30071, Spain

5 University of Newcastle, New South Wales, Australia