Kalpa Publications in Computing

Volume 7, 2018, Pages 32–49

PROOFS 2018. 7th International Workshop
on Security Proofs for Embedded Systems

Attack-tree-based Threat Modeling of Medical Implants

Muhammad Ali Siddiqi[1], Robert M. Seepers[2], Mohammad Hamad[3], Vassilis Prevelakis[3], and Christos Strydis[1]

[1] Neuroscience Dept., Erasmus MC, The Netherlands – {m.siddiqi, c.strydis}@erasmusmc.nl
[2] Netherlands Organisation for Applied Scientific Research (TNO) – robert.seepers@tno.nl
[3] Inst. of Comp. & Network Eng., TU Braunschweig – {mhamad, prevelakis}@ida.ing.tu-bs.de

Abstract

Modern Implantable Medical Devices (IMDs) are low-power embedded systems with life-critical functionalities. Almost all of these devices are equipped with wireless-communication capabilities in order to aid in diagnosis, in updating the functional settings and firmware and so on, without any surgical procedure to perform these tasks manually. There is, thus, a rising trend towards increased connectivity of these devices. The down-side of this trend is, however, a proportional increase in the attack surface that can be exploited by a malicious entity. In effect, threat modeling of IMDs becomes ever more important. This is reflected by an increase in the number of vulnerabilities being found consistently in the IMDs available in market. This paper proposes a threat-modeling analysis based on attack trees to evaluate the security of these devices. As an example, three recent lightweight IMD security protocols from literature are analyzed using this approach to demonstrate its effectiveness in suggesting security improvements.

1 Introduction

Modern IMDs are autonomous, battery-powered devices with extremely high safety and reliability constraints. They are typically designed to operate for a long period of time (up to a decade or so) while implanted in the human body. Most common examples of IMDs are cardiac pacemakers and defibrillators, neurostimulators, implantable infusion pumps and cochlear implants. These devices work in a closed-loop fashion by providing some form of stimulation based on monitoring one or more physiological signals. To support and enhance the treatment capabilities of these devices, modern IMDs are equipped with a wireless transceiver. This transceiver can communicate with an external hand-held reader/programmer or a base station for e.g. local and/or remote monitoring of patient health, performing a device test, reading sensory information, updating IMD settings and/or firmware, and so on [22]. The remote monitoring aspect allows significant treatment cost reduction and helps in early detection of potential medical issues. However, these wireless capabilities – though greatly advantageous – make it possible for malicious entities to communicate with the device without the knowledge or cooperation of the victim. This vast increase in the *attack surface* leads to a number of serious issues, e.g. private-data theft, misdiagnosis, physical harm etc. The existence of vulnerabilities

has been verified by recent, successful, ethical-hacking efforts on IMDs. For example, the authors in [14] perform and document attacks against ten Implantable Cardioverter Defibrillators (ICDs) from the market. Last year alone, vulnerabilities pertaining to two implantable cardiac devices available in the market were communicated by the FDA [9].

Numerous solutions have been proposed in literature to strengthen IMD security. It is true that these solutions try to solve very narrow challenges in this young domain, e.g. emergency access, balancing safety and security etc. Even so, it has come to our attention time and again that they fall short of providing fundamental security requirements [6, 14, 13]. Thus, in trying to improve the current state of the art, we have to find a consistent way of assessing the strength of these solutions. We, therefore, introduce the use of attack-tree-based threat modeling as a non-exhaustive but structured approach for finding vulnerabilities in a very ad hoc field. Threat modeling is a systematic process for identifying and categorizing threats and security vulnerabilities of a system from the adversary's perspective. It can be used to measure and improve the security of the system against current and future threats. Furthermore, it can be used to identify the adversary profile, the valuable assets of the system, the points of potential weakness and the most applicable threats. This approach can help us gain a better insight into the mindset and goals of the attackers and where security experts should spend effort considering the type of attacks expected from the attacker profile [20]. Threat modeling in a tree structure presents a comprehensive overview of the vulnerabilities and it can be used to analyze the different attack pathways in a structured way.

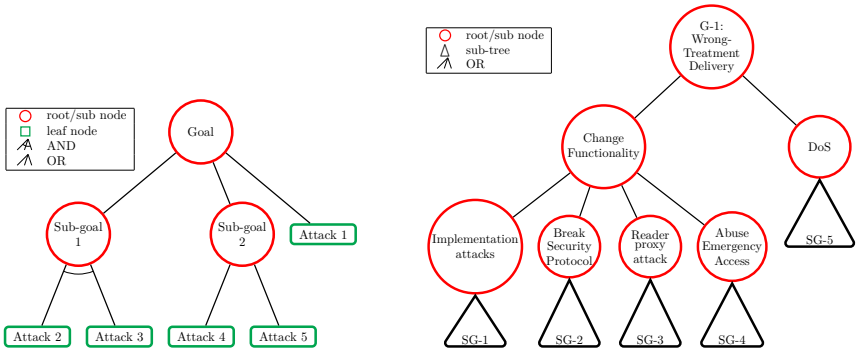
In essence, this work makes the following novel contributions:

- We introduce a systematic attack-tree-based threat-modeling approach adapted to the domain of IMDs.
- We establish attack trees for the very particular case of IMDs, which, to the best of our knowledge, is the first work formulated for these devices. The intention is to create a constantly expanded reference point by and for the whole IMD community.
- We assess these trees by evaluating the security of three recent and cutting-edge secure IMD communication protocols from literature. We subsequently give recommendations on how to improve the security of these sample protocols.

The rest of the paper is organized as follows. We provide brief background on attack trees in Section 2. In Section 3, we construct detailed, IMD-specific attack trees after defining our system and attacker models. We provide background information on the protocols chosen for our threat-analysis approach in Section 4. In Section 5, we evaluate the example protocols from Section 4 using our threat-analysis approach. We give recommendations based on our findings in Section 6 and highlight related work in Section 7. We conclude the discussion in Section 8.

2 Background on Attack Trees

Attack trees were proposed by Schneier [20] as a method to describe the security of any system. These constructs aid in improving the security or evaluating the impact of new attacks on security. Attack-tree-based analysis helps to better determine the vulnerability of a system against any specific type of attack and can rank different types of attacks based on their likelihood. It is also useful in enumerating the security assumptions of the system. In the case of a system modification e.g., generic improvements or implementation of countermeasures against a threat, attack trees help in evaluating the resulting impact on security. Since smaller



(a) Example Attack tree (b) G-1: Wrong-treatment delivery

Figure 1: Attack trees for IMDs

attack trees can fuse in larger trees, the resulting scalability helps in determining the coverage and efficiency of any countermeasure. Moreover, they can help in more efficiently allocating the security budget available and can also shed light on the resources, level of access and skills required by the attacker to perform certain attacks.

These constructs illustrate the attacks in a tree structure as shown in Figure 1a. Each tree contains a *root node*, which represents the final goal of the attack, *intermediate nodes* (sub-goals), which define different stages of the attack that lead to the root, and *leaf nodes*, which represent atomic attacks. Boolean gates are used to explain whether a node in a tree requires achieving *all* of its sub-nodes (*AND* node), or *any* of its sub-nodes (*OR* node). An attack scenario will contain a minimum set of leaves that leads to a successful traversal to a root.

Given the largely unsecured or – worse yet – ad hoc manner in which security is being added to modern IMDs, attack trees offer a more structured method for designing and evaluating IMD security. This approach does not guarantee completeness but takes a methodical and scholastic approach towards IMD security, which is hoped to result in uncovering more blind spots.

3 IMD Threat Modeling

We first define our system and attacker models, after which we follow the methodology from [20] to construct IMD attack trees. The first step is to identify the attacker goals. Each goal will result in a separate attack tree. We then identify possible attacks pertaining to each goal to populate the tree. Existing trees can be reused as sub-goals to form part of a bigger tree.

3.1 IMD System Model

We consider an IMD I capable of wireless communication with an external reader R. Both entities fall within the boundary of our system model. We assume that there is only wireless (non-physical) access to I while implanted in the body, whereas physical access to R is possible. Without loss of generality, we assume that the implant application supports more than one user role in terms of lowest to highest permission levels. For example, a nurse may only be allowed to read data related to the operation of the implant, whereas a treating physician may also be allowed to suspend or resume the implant functionality or modify it for therapy updates.

The IMD communication protocol must satisfy the fundamental security services of CIANA: *Confidentiality*, *Integrity*, *Authentication*, *Non-repudiation*[1] and *Availability*. In addition, the protocol must provide the following features of particular importance to IMDs: *Access Control*, *Key Management*[2], *Key Freshness*, *Perfect Forward Secrecy (PFS)* and *Error recovery*. Also, *Emergency Access* is a crucial feature specific to IMDs, which allows paramedics access to the IMD during emergencies without compromising security. Strictly speaking, these features fall under the CIANA model, but are explicitly mentioned here as they are of special importance for IMDs, e.g., emergency access can be considered as falling under availability.

3.2 Attacker Model

In order to evaluate IMD security, we assume an attacker A whose aim is to prevent patient treatment, perform data manipulation, or steal private patient data. This is further elaborated in Section 3.3. Necessarily, A can either be an *outsider* or from various insiders with different security privileges (e.g., nurses, physicians, technicians etc.) [19]. Furthermore, we assume that A is active, i.e., has full control of the channel. Thus, A can eavesdrop, modify, block or replay messages between R and I, in addition to forging new ones. We note that these are rather conservative assumptions given the current state of the art in IMD communications. However, we consider worst-case conditions to cover any future changes in this aspect.

3.3 Attacker Goals

The attacker goals subject to the above System and Attacker models are categorized below.

3.3.1 Modification of IMD operation / Wrong-treatment delivery

Many modern IMDs are working in a closed-loop fashion, effecting some form of intervention (e.g., electrical stimulation) to a particular health issue (e.g., heart arrhythmia, epileptic seizures, chronic pain, tremor etc.). Common examples are modern-day implantable pacemakers. Modification of this functionality may result in the prevention of stimulation for treatment purposes. It may also result in over-stimulation, which could cause tissue damage. Attacker A can cause these modifications by forcing the processor to execute a different/modified binary leading to incorrect IMD functionality, or by making it run an infinite loop resulting in thermal hot-spots and battery drain. Alternatively, A can cause the sensors to read incorrect physiological data, subsequently resulting in incorrect calculations by the processor. A can also try to modify the clock frequency of the system resulting in incorrect duration of treatment and untimely triggering of stimulation.

A can also force critical IMD resources to remain unavailable when treatment action is required (e.g., during cardiac arrest) through various Denial-of-Service (DoS) attacks. A can, for instance, force the processor to run an infinite loop at full frequency resulting ultimately in energy loss and IMD shutdown (*Battery DoS*). The IMD may also be repeatedly requested to establish a secure wireless channel using incorrect credentials. This will cause repeated execution of the same authentication protocol for analyzing the request, which will – in turn – result in battery drain. Moreover, repeated communication requests may prevent the IMD from

[1] Non-repudiation is a valid concern since we assume the possibility of insider attacks, malpractices etc. For instance, a method is needed to perform computer forensics in case a patient dies or has medical issues due to a mis-configured pacemaker by a careless physician, and so on.

[2] In addition to generating keys, here key management also includes the revocation, replacement and addition of new readers.

performing its life-critical, primary task (*Function DoS*). A can also block the reader/IMD communication channel by constantly sending valid or invalid messages resulting in *Jam DoS*[3] [22].

3.3.2 Data forging

Another goal of A could be to *forge* sensitive patient data, e.g., data-logs stored in I, which can indirectly lead to patient/doctor misinformation, incorrect diagnosis and subsequent incorrect treatment. Data forging could also be the goal of an insider to cover up medical mistakes (e.g., wrong diagnosis). A can perform this by modifying the IMD memory to store incorrect data, or by manipulating the communication packets exchanged between R and I.

3.3.3 Data theft

The aim of A can also be to *steal* private patient data, which can indirectly lead to problems such as social segregation, extortion, blackmail and more [22]. This can happen if A steals data from the IMD memory, or if he/she eavesdrops on such data exchanged between R and I.

3.4 Attack Trees for IMDs

For each threat identified in Section 3.3, we now present the attack trees per attacker goal G-x.

3.4.1 G-1: Wrong-Treatment Delivery

The high-level tree to reach this goal is shown in Figure 1b. Due to complexity and paper-size limitations, we use Figure 2 to expand sizable sub-goals (SG-x). To also highlight the benefit of our approach, we coarsely assign the likelihood of attack (*L=Low, M=Moderate, H=High*) to all the leaf-nodes and propagate the resulting effect up towards the root node. The likelihood can be a function of cost, type of equipment required etc. [20]. Wrong treatment could be achieved either by changing the IMD functionality or through DoS. The IMD functionality can be modified by using implementation attacks, breaking into the secure channel (to forge communication packets), using a Reader proxy or by abusing the emergency-access mode.

SG-1: Implementation Attack[4]**.** This sub-tree consists of side-channel attacks, fault injection and attacks that exploit implementation flaws. A can employ side-channel attacks to e.g. recover the factory-installed key from a stolen R [13]. This can be done by e.g., measuring the power consumption of a processing core when running the crypto algorithm, observing the run-time behavior of an algorithm implementation etc. [15]. In the case of fault-injection attacks, A can affect the IMD operation by e.g., changing the sensor and actuator functionality through electromagnetic interference (EMI) [11]. A can also inject fault in the IMD clock source, e.g. crystal oscillator, to induce additional toggles within the clock period (clock glitch attack). This may result in timing failure of certain portions of the IMD. Barenghi et al. have listed various fault-injection attacks, which can be utilized against Reader/IMD systems [3]. A can also attempt to exploit implementation flaws in the IMD. These flaws could lie in the security protocol, e.g. in nonce generation, which create opportunities for replay attacks (see SG-2), or in the device firmware causing buffer-overflow exploits. Race conditions between different computing components of the IMD and errors in the application-code compilers also open up opportunities for exploitation. Flaws in data sanitization allow an *illegal* value to destabilize

[3]The chance of a jam DoS harming IMD operation is extremely low since the communication between R and I pertaining to critical treatment updates is very infrequent and is usually held in a controlled environment [23].

[4]SG-1.1 and SG-1.2 are more likely to be accomplished against R than I. The likelihood annotations of these sub-goals, however, pertain to I for consistency.

SG-1 Implementation attack
OR
SG-1.1 Side-channel attack [13] (L)
OR
SG-1.1.1 Power analysis [15] (L)
OR
SG-1.1.1.1 Simple Power Analysis (L)
SG-1.1.1.2 Differential Power Analysis (L)
SG-1.1.1.3 Template Attack (L)
SG-1.1.1.4-N ...
SG-1.1.2 Timing analysis [15] (L)
SG-1.1.3-N ...
SG-1.2 Fault injection [3] (L)
OR
SG-1.2.1 Change sensor/actuator functionality through EMI [11] (L)
SG-1.2.2 Clock glitch attack (L)
SG-1.2.3 Power glitch attack (L)
SG-1.2.4-N ...
SG-1.3 Identify Implementation flaws (M)
OR
SG-1.3.1 Protocol-implementation flaw. See SG-2. (H)
SG-1.3.2 Buffer overflows (M)
SG-1.3.3 Race condition between different processing cores (L)
SG-1.3.4 Bug or flaw in application-code compiler. For deliberate errors see SG-1.4. (L)
SG-1.3.5 Flaws in data sanitization (M)
SG-1.3.6-N ...
SG-1.4 Insider Attack (L)
OR
SG-1.4.1 Attack Tool Chain
OR
SG-1.4.1.1 Compiler (L)
SG-1.4.1.2 Libraries (L)
SG-1.4.1.3 Run-time environment (L)
SG-1.4.1.4-N ...
SG-1.4.2 Attack application (L)
SG-1.4.2.1 Exploit improper access control [7] (L)
SG-1.4.2.2-N ...

SG-2 Break security protocol
OR
SG-2.1 Identify flaw in Security Protocol (H)
OR
SG-2.1.1 Identify flaw in encryption alg. (M)
OR
SG-2.1.1.1 Identify flaw in cipher (L)
SG-2.1.1.2 Identify flaw in RNG(s) to mount a replay attack (L)
SG-2.1.1.3 Identify key re-use (M)
SG-2.1.1.4 Identify lack of randomness (M)
SG-2.1.1.4.1 In biometrics [13] (M)
SG-2.1.1.4.2 Due to ECC usage [13] (M)
SG-2.1.1.5-N ...
SG-2.1.2 Identify flaw in handshake (H)
OR
SG-2.1.2.1 Man-in-the-middle attack [13] (M)
SG-2.1.2.2 Reflection attack [13] (H)
SG-2.1.2.3 Key confirmation attack [15] (L)
SG-2.1.2.4 Replay attack [15, 6, 14] (H)
SG-2.1.2.5-N ...
SG-2.2 Obtain legitimate master key for IMD (H)
OR
SG-2.2.1 Brute-force master key used in cipher (L)
SG-2.2.2 Acquire key through social engineering (H)
SG-2.2.3 Steal key (from a used IMD) using side-channel attack. See SG-1. (L)
SG-2.2.4 Insider attack. See SG-1.4 and G-2.3. (L)
SG-2.2.5-N ...

SG-3 Reader Proxy Attack
OR
SG-3.1 Hack into patient laptop/smartphone (L)
OR
SG-3.1.1 Perform remote physiological-signal measurement using camera etc. [26] (L)
SG-3.1.2 Start IMD-control application (if available) (L)
SG-3.1.3-N ...
SG-3.2 Use legitimate medical equipment to aid in remote attack [14] (M)
SG-3.3-N ...

SG-4 Abuse Emergency Access
OR
SG-4.1 Wait until emergency mode is triggered (L)
SG-4.2 Evoke emergency directly by creating a stressful incident (M)
SG-4.3 Toggle device to Emergency mode (M)
OR
SG-4.3.1 Exploit pairing device (M)
OR
SG-4.3.1.1 Physically remove the pairing device (M)
SG-4.3.1.2 Remotely attack pairing device. See SG-2. (M)
SG-4.3.2 Exploit token-based access (M)
OR
SG-4.3.2.1 Gain physical access to the token (M)
SG-4.3.2.2 Brute-force emergency password (L)
SG-4.3.3 Exploit distance-bounding protocol (M)
OR
SG-4.3.3.1 Body-coupled channel: Capture on-body (electric) signals (L)
SG-4.3.3.2 Vibration-based: Cause vibrations on the body (L)
SG-4.3.3.3 Magnetic switch (M)
OR
SG-4.3.3.3.1 Toggle magnet when proximal to I (M)
SG-4.3.3.3.2 Use strong magnet for remote attack (L)
SG-4.3.4 Biometrics (L)
OR
SG-4.3.4.1 Obtain biometrics from subject (L)
OR
SG-4.3.4.1.1 Remote measurements. See SG-3. (L)
SG-4.3.4.1.2 Physical measurements (touching patient) (L)
SG-4.3.4.2 Brute-force the biometric (L)
SG-4.3.4.3 Implementation error in (biometric) cipher/authenticator (L)
SG-4.3.5 Criticality-awareness-based access (M)

SG-5: DoS Attack
OR
SG-5.1 Disrupt channel (H)
OR
SG-5.1.1 Signal jamming on IMD-communication frequency (H)
SG-5.1.2 Keep the comm. channel busy with requests
OR
SG-5.1.2.1 Repeatedly request connection to I using normal-mode protocol (even if authentication fails) (H)
SG-5.1.2.2 Repeatedly request to activate emergency mode (even if this fails) (H)
SG-5.1.3 Send bogus packet or modify/drop packets to reset a session between R and I (H)
SG-5.2: Cause device malfunction (H)
OR
SG-5.2.1 Drain IMD battery [6] (H)
OR
SG-5.2.1.1 Overloading connection requests. (H)
SG-5.2.1.2 Execute additional code on IMD. See SG-2, SG-3 and SG-4. (L)
SG-5.2.1.3-N ...
SG-5.2.2 Overheat device (L)
OR
SG-5.2.2.1 Execute additional code on IMD. See SG-2, SG-3 and SG-4. (L)
SG-5.2.2.2 Provide continuous stimulation. See SG-2, SG-3 and SG-4. (L)
SG-5.2.2.3-N ...
SG-5.2.3 Prevent I to execute its main application. See SG-5.2.1.1. (H)
SG-5.2.4 Replay previously captured commands from R to *turn off* the therapy [6] (H)
SG-5.3: Insider attack. See SG-1.4. (L)

Figure 2: Textual representation of IMD Attack trees for sub-goals of G-1

the system, e.g., if there is an option to select an encryption algorithm among various choices, then inputting a negative crypto-algorithm identifier may cause unauthorized code execution. Implementation attacks can also be a viable option for a trusted entity that is malicious, e.g., manufacturer, developer, and so on. The insider, in this case a developer, can attack the tool chain for the software used in R and/or I by modifying the application compiler, libraries or the run-time environment, or, the application itself.

SG-2: Break Security Protocol. Another option for A could be to break the security protocol between R and I. A can, for instance, identify flaws in the used cipher for data confidentiality or the used random-number generator (RNG) for nonce generation (for replay attacks). A can also look for cases of key re-use, lack of randomness in biometrics or lack of randomness due to the use of Error-Correcting Codes (ECCs) if the protocol employs fuzzy cryptographic primitives [13]. Alternatively, A can find flaws in the protocol handshake process. Marin et al. [13] proposed attacks specific to physiological-signal-based security protocols. They showed that a well-cited Dynamic-Cardiac-Biometrics (DCB)-based protocol (H2H) [18] had weaknesses against *Man-in-the-middle (MITM)* and *reflection attacks*. A can also opt for other common attacks such as *replay attacks* and/or *key-confirmation attacks* [15]. Halperin et al. [6] demonstrated successful replay attacks on a commercial ICD over a short-range communication channel by replaying previous messages sent by the reader. Marin et al. [14] showed that adversaries can launch successful replay attacks on multiple commercial IMDs over both short- and long-range communication channels.

Besides, A can try to obtain the legitimate IMD cipher master key in order to break the security protocol. A can, for instance, attempt a brute-force attack or use social engineering, e.g. by employing blackmail or phishing on a trusted entity or the IMD manufacturer.

SG-3: Reader Proxy Attack. This sub-goal pertains to the scenario where A uses legitimate equipment in place of the reader. For protocols based on physiological signals (e.g. heartbeats), A can hack into the patient smartphone/laptop camera and measure subtle color variations of the patient skin to detect heartbeats using remote photo-plethysmography (rPPG) [26, 22]. A can also hack the smartphone to run an IMD control application, if available (see the control application in [1], for example). Another approach could be to buy an inexpensive, compatible base station that only gathers telemetry data from the IMD and sends it to the hospital to facilitate remote monitoring. A can use it to activate the IMD and then use his/her own equipment to send malicious messages, as shown in [14] for a commercial IMD.

SG-4: Abuse Emergency Access. In the case of emergencies, the IMD should permit access to a paramedic's reader for immediate treatment despite the fact that they are likely unknown to each other and therefore do not share a secret key [21]. Following SG-4, A can abuse this emergency mode. A can wait for the emergency to happen, evoke a fake emergency situation directly by creating a stressful incident, or toggle the device to this mode based on the type of access scheme employed by the IMD [21]: A can attack a pairing (wearable) device used in a *device-pairing* scheme, which handles the authentication of readers during normal mode of operation and triggers fail-open access to the IMD if it is physically distanced from the IMD. Exploitation of *token-based* schemes would require an attacker to get access to a token (wearable, tattoo etc.) that has the emergency password for the paramedics. *Distance-bounding* methods enforce touch-to-access[5] by making sure that the communication distance between R and I is short. This can be done by employing short-distance communication channels such as the human body (using electric conductivity ofr vibrations), a magnetic switch (to

[5]Touch-to-access ensures that I can only allow access to the entities who can physically touch the patient for a prolonged period of time. This policy assumes that only R is permitted to touch the patient and that it is infeasible for an attacker A to get in close proximity to the patient since the patient would reject physical contact with untrusted entities [22].

G-2: Data Forging
OR
G-2.1 Modify I memory data after initiating communication with it. See SG-2, SG-3 and SG-4. (L)
G-2.2 Inject/modify communication packets between R and I [6]. See SG-2, SG-3 and SG-4. (L)
G-2.3 Insider attack (by attacking operations/logistics) (L)
OR
G-2.3.1 Forging/mishandling of treatment logs (L)
G-2.3.2-N ...
G-2.4-N ...

G-3: Data theft
OR
G-3.1 Steal private data (data logs etc.) (H)
OR
G-3.1.1 Compromise the reader where logs are downloaded (M)
G-3.1.2 Steal data by tampering with a used implant (L)
G-3.1.3 Retrieve private data from the IMD memory after initiating communication with the implant. See SG-2, SG-3 and SG-4. (L)
G-3.1.4 Eavesdrop on communication between R and I [6]. See SG-2, SG-3 and SG-4. (H)
G-3.1.5-N ...
G-3.2 Investigate which IMD type is implanted inside the patient by finding the IMD identifier (H)
OR
G-3.2.1 Retrieve identifier by initiating communication with the implant. See SG-2, SG-3 and SG-4. (H)
G-3.2.2 Eavesdrop on communication between R and I. See SG-2, SG-3 and SG-4. (H)
G-3.2.3 Retrieve identifier/model number etc., by looking at the reader (H)
G-3.2.4-N ...
G-3.3 Insider attack. See G-2.3. (L)
G-3.4-N ...

Figure 3: Textual representation of IMD Attack trees for G-2 and G-3

disable security) etc. For the devices relying on the human-body channel, depending upon the implementation, A can try to remotely capture on-body electric signals, cause vibrations on the body by calling the patient cell-phone etc. For the devices employing a magnetic switch, A can pass a magnet over I to disable security when in close proximity. A relatively expensive alternative is to use a strong magnet in case of a remote attack. IMDs can also employ *biometrics* for emergency access. A can opt for performing measurements remotely (as discussed in SG-3) or physically while touching the patient e.g., by impersonating a nurse. A can also opt to brute-force the biometric if it lacks perfect entropy or try to find implementation flaws in the security primitives employed in biometric-based encryption schemes, e.g., fuzzy vault etc. *Criticality-awareness*-based schemes, unlike previous methods, do not follow a touch-to-access policy. In these schemes, I monitors patient vitals and triggers fail-open access in case of emergency. Here, A can try to fool I to believe that it is in a medical emergency.

SG-5: Denial of Service. To achieve DoS, A can disrupt the wireless channel to block communication between R and I in order to prevent medical intervention. This can be done by jamming the IMD frequency band, by repeatedly requesting connection to I even if the authentication fails, or by sending/modifying/dropping packets to reset a communication session between R and I. A can also cause malfunction or shorten lifetime of I by draining battery [6] by overloading I with connection requests or by running additional code via code injection (using SG-2, SG-3 and SG-4). Marin et al. [14] carried out battery DoS for certain pacemakers by switching the devices from standby to (energy-consuming) interrogation mode with relative ease. Through continuous execution, the additional malicious code can also overheat the device, e.g., by causing continuous stimulation. The overloading of connection requests from A can also result in the inability of I to execute its main application.

3.4.2 G-2: Data Forging

The attack tree for data forging (G-2) is shown in Figure 3. Note that the listed attacks are common to those discussed in Section 3.4.1. Moreover, an insider e.g., a doctor or nurse can try to attack the logistics, e.g., by malicious handling of the treatment logs, in order to cover up medical mistakes.

3.4.3 G-3: Data Theft

When it comes to data theft (see Figure 3), A could either be interested in stealing treatment-related data/logs etc., or just the nature of the medical condition itself. Stealing of private data can be done by compromising the reader, by stealing a used implant or by using attacks

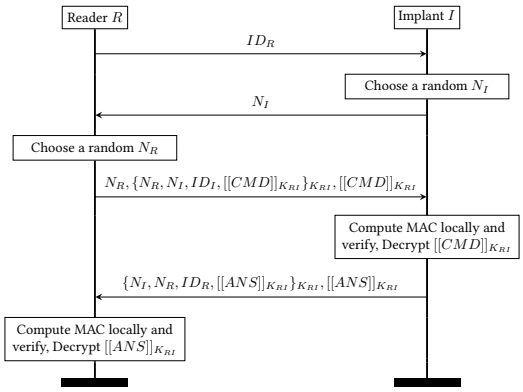


Figure 4: P-Sec [23, 22]

from Section 3.4.1 to hack into I and modify the IMD memory or, finally, to eavesdrop and decrypt communications between R and I [6]. In order to discern the nature of the medical condition, A can try to investigate the type of IMD implanted in the patient by finding the IMD identifier. This can be done by looking at the model/type of R. Alternatively, A can opt for remote attacks using SG-2, SG-3 and SG-4.

4 Example IMD Protocols

In order to demonstrate our proposed threat-modeling approach, we use the protocols proposed in [23, 21, 16], which were designed to enable secure communication between R and I. These protocols were selected for our analysis because they are custom-made for low-power IMD systems and entail state-of-the-art research concepts such as zero-power defense [6], dynamic-biometrics-based security, touch-to-access policy, emergency access, and so on. Moreover, it is well known that the IMD manufacturers rely on "security through obscurity" by concealing the protocol specifications [14], which is another reason to evaluate the above protocols from academia. The three protocols are summarized below and are denoted by P-Sec, P-KeyEx and P-Auth, respectively, for brevity.

4.1 P-Sec: Lightweight Secure Communication Protocol

The main purpose of this protocol is to ensure confidentiality, integrity and mutual authentication of the messages exchanged between R and I. It uses a lightweight symmetric block cipher for data confidentiality. Moreover, cipher-based Message-Authentication-Code (MAC) is used for integrity and authentication. The protocol uses freshly generated unique random numbers *number-once (nonce)* in the MAC calculation to prevent replay attacks.

The identifiers of R and I are denoted by ID_R and ID_I respectively. The encryption of message M with key K is represented by $[[M]]_K$. Similarly, $\{M\}_K$ refers to the MAC of message M with key K. N_X denotes a nonce generated by entity X. The protocol steps are shown in Figure 4. R initiates the protocol by sending ID_R, which is used by I to choose the correct key K_{RI}. I responds by sending N_I to R. R then generates N_R and encrypts the command (CMD) using K_{RI}. It also computes a MAC that includes $[[CMD]]_{K_{RI}}$ and the nonces. R then sends N_R together with the calculated MAC and the encrypted command to I. I checks if the

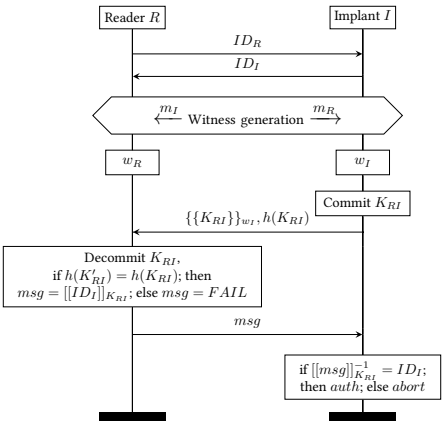


Figure 5: P-KeyEx [21]

message has been received correctly and was indeed sent by R by locally calculating the MAC and checking its equality to the received value. I aborts the protocol in case the validation fails. Otherwise, it confirms R as a legitimate entity. I then decrypts and executes the CMD and sends the subsequent answer (ANS) in a similar fashion to how it was done for CMD in the previous step. R receives the message and calculates the local version of the MAC. R and I are considered mutually authenticated if both the MAC values are equal and as a result, the ANS is decrypted and processed by R. Otherwise, R drops the reply.

4.2 P-KeyEx: Lightweight Authenticated Key-Exchange Protocol

The purpose of P-KeyEx is to establish trust between R and I, and to perform key exchange for any symmetric-key-based data confidentiality protocol. It achieves this by using the cardiac Inter-Pulse Interval (IPI) [18], which is the time difference between two consecutive heartbeats, as an RNG. An IPI value is obtained by both R and I by simultaneously measuring a cardiac signal from the same person. Each of these values are used to derive time- and person-specific random numbers, which are referred to as *witnesses* w_R and w_I, respectively. This makes them useful in entity authentication and key exchange [21]. However, if I transports K_{RI} by encrypting it using w_I as symmetric key, R cannot decrypt it, since in practice w_R is not exactly equal to w_I. The protocol addresses this by employing a *fuzzy-commitment* scheme [10], which applies ECCs on K_{RI} before its encryption to counter the difference between w_R and w_I. The commitment operation $\{\{x\}\}_w$ of x using witness w is defined as $\{\{x\}\}_w = ECC(x) \oplus w$.

The protocol is depicted in Figure 5. Both R and I exchange ID_R and ID_I in order to bind K_{RI} to these identifiers upon successful exchange. To generate witnesses, R and I simultaneously obtain a block of IPIs and communicate the occurrence of any *heartbeat misdetection* using misdetection flags (m_R and m_I). In the case of a misdetection they replace the block of IPIs with fresh IPIs. This process is repeated until the gathered IPIs are enough to generate w_R and w_I. I generates a random K_{RI} (through its internal RNG) and fuzzy commits it using w_I, calculates a cryptographic hash of K_{RI} ($h(K_{RI})$) for *data integrity*, and sends the entire message to R. R after receiving this message applies the inverse process of commitment and obtains K'_{RI} (where $K'_{RI} = K_{RI}$ iff $w_R \approx w_I$). R validates the correct transfer of K_{RI} by locally computing the hash $h(K'_{RI})$ and comparing it to $h(K_{RI})$ received from I. In case of a match, R encrypts ID_I with K_{RI} using its regular cipher and sends it to I. I decrypts the

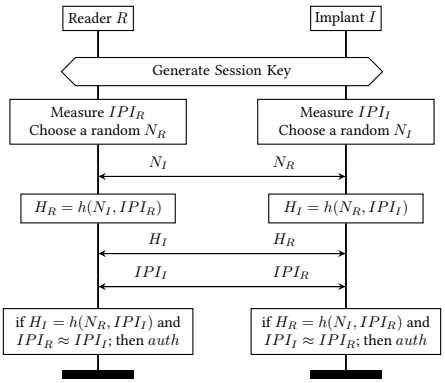


Figure 6: P-Auth [16]

ID_I with K_{RI} using the same cipher. If the decrypted ID_I is correct then the key exchange is a success. At this point, both R and I are mutually authenticated since both have implicitly verified that $w_R \approx w_I$. If either of $R's$ hash comparison or $I's$ identifier comparison fails, then the protocol fails.

4.3 P-Auth: Biometric Authentication Protocol

Similar to P-KeyEx, the purpose of P-Auth is to authenticate the two entities using IPIs. However, as opposed to P-KeyEx, P-Auth does not perform symmetric-key exchange. The protocol steps are shown in Figure 6. P-Auth starts with a session key establishment using any suitable key exchange protocol. Both nodes measure the IPIs and generate random nonces N_R and N_I. These nonces are then exchanged after which both nodes calculate a hash of the locally measured IPIs and the received nonce. They exchange these hashes (H_R and H_I) *before* sharing their IPIs with each other. This prevents the peer node to replay the received IPI value. Both nodes then locally calculate the hash of the received IPIs and the locally generated nonce. If the resulting value is not equal to the received hash value, or the local and received IPIs are not similar enough, the authentication fails.

5 Evaluation Using Attack Trees

In this section, we evaluate the three protocols described in Section 4 by reflecting them against our attack trees introduced in Section 3.4. We will only use the portion of the attack trees relevant to the evaluation of security protocols. However, keep in mind that the trees are designed to have much broader applicability as these are generic trees for Reader/IMD systems.

5.1 P-Sec analysis

The goals/sub-goals relevant to the evaluation of P-Sec are SG-1, SG-2 and SG-5 (see Figure 2).

SG-1: Implementation Attack. These attacks are by definition implementation-specific, yet, we discuss them here to highlight prominent ways in which the protocol implementations can be exploited, since some of the implementations are dependent on protocol architecture.

Looking at SG-1, we can immediately see that P-Sec's dependence on pre-installed keys[6]

[6]It was not the objective of the authors of P-Sec to address key management, as explicitly mentioned in [23].

opens the door for A to perform side-channel attacks to steal the factory-installed keys from the reader **(SG-1.1)**. Obviously, any protocol implementation involving the storage of secret data is in principle susceptible to these attacks. Therefore, such pre-installed keys need to be properly protected. Protocol-implementation flaws **(SG-1.3.1)** e.g., incorrect nonce-generation implementation, also invite attacks from A. By construction P-Sec allows distinguishing individual users (or user groups) because of pre-shared unique keys, thus facilitating access control. Hence an insider with valid authentication credentials would not be able escalate privileges if access control is implemented correctly **(SG-1.4.2.1)**.

SG-2: Break Security Protocol. We start the analysis of the protocol itself by first looking at the encryption algorithm using **SG-2.1.1**. The MISTY1 cipher employed by P-Sec was recently broken using *Integral Cryptanalysis* [25] **(SG-2.1.1.1)**, which makes the cipher-based RNG used for nonce generation vulnerable too **(SG-2.1.1.2)**, hence making the protocol susceptible to replay attacks. Moreover, for all block ciphers with a block size of 64 bits, including MISTY1, the likelihood of cipher-text block collisions increases if large amounts of data are encrypted with the same session key (in most modes of operation e.g., CBC, CFB, CTR etc.), thus making them vulnerable to *birthday attacks* [4]. Although these attacks are currently unrealistic for low data-traffic IMDs, they still warrant a fix considering that these devices are intended to remain implanted for many years. Therefore, the cipher implementation should make sure that it encrypts less data per session key and that the session keys are changed more frequently. Since, in P-Sec, MISTY1 is used in CTR mode for nonce generation using the same key for all the sessions, this makes it vulnerable to these attacks. One of the most important issues with the protocol is that the authentication key is the same as the session key (K_{RI}), which is re-used in every session **(SG-2.1.1.3)**. This means that we lose *PFS*, i.e., if this shared key is compromised in the future, then all the previous communication (which is recorded by the adversary) will also be compromised. In a device that is expected to operate for a long time, satisfying PFS has high significance. Moreover, the use of a single factory-installed K_{RI} makes it a *single point of failure*, which is undesirable [13].

We analyze the handshake using **SG-2.1.2**. Since the protocol relies on symmetric encryption, it is by design robust against MITM attacks **(SG-2.1.2.1)**. It should be noted, however, that because of symmetric encryption, the protocol does not guarantee *non-repudiation*. Moreover, the protocol was originally designed for a few readers with pre-distributed symmetric keys. There is no mechanism to introduce new readers or keys, or remove existing ones. Unique/shared device keys are not less secure but are problematic in terms of *(key) management*. This also means that P-Sec does not facilitate swift diagnosis and treatment in a secure manner during *emergencies*. This is because the paramedic R and I are likely unknown to each other and therefore do not share a K_{RI}. Since the type of messages between R and I are not symmetric (similar) in both directions, the protocol is not vulnerable to reflection attacks [13] **(SG-2.1.2.2)**. The use of the same key, or in other words, the lack of *key freshness* throughout all the communication sessions between R and I rules out key-confirmation attacks **(SG-2.1.2.3)** but creates replay-attack opportunities for A **(SG-2.1.2.4)**. In order to protect against these attacks, P-Sec employs nonces. This, however, depends on an error-free implementation of nonce generation. As a best practice, it is recommended to use different and unique keys for each session.

Moreover, when it comes to obtaining K_{RI}, it would be infeasible for A to find the 128-bit MISTY1 key using brute force **(SG-2.2.1)**. However, if A does manage to acquire the key, e.g., through social engineering **(SG-2.2.2)**, the protocol does not facilitate its replacement, and thus the only option would be to replace I through surgery.

SG-5: Denial of Service. Looking at SG-5, A can try DoS attacks by requesting a new

session by sending a valid ID_R **(SG-5.1.2.1, SG-5.2.1.1)** (see Figure 4). After receiving N_I from I, A can send any message, which will force I to perform MAC calculations, resulting in battery drain. A can also disrupt the protocol and hence manage to cause function DoS by sending a bogus packet to I when it is waiting for a response from R, which will result in failed authentication and subsequently protocol reset **(SG-5.1.3)**. This will prevent the device from responding to legitimate requests.

5.2 P-KeyEx analysis

The sub-goals relevant to the evaluation of P-KeyEx are SG-1, SG-2, SG-3, SG-4 and SG-5.

SG-1: Implementation Attack. In SG-1, we see that the protocol is independent of the need to install keys at manufacturing time and, hence, is free from resulting issues **(SG-1.1)**, as was the case with P-Sec. When it comes to flaws in protocol implementation **(SG-1.3.1)** we recognize that one possible candidate could be the improper design of the RNG that generates K_{RI}, however, it is not clear from [21] what algorithm is used to generate this random key. P-KeyEx does not allow distinguishing individual users (or user groups) for access control since it does not employ any pre-shared secret between R and I. Hence it is vulnerable to exploits targeting improper access control **(SG-1.4.2.1)**.

SG-2: Break Security Protocol. When evaluating against SG-2, we first analyze the symmetric cipher chosen for P-KeyEx, PRESENT-80 **(SG-2.1.1.1)**. The protocol uses a block size of 64 bits, however, birthday attacks are not a concern since the encryption process ($[[ID_I]]_{K_{RI}}$) uses a randomly generated key for every session. If we assume a strong internal RNG implementation, P-KeyEx protects against replay attacks **(SG-2.1.1.2, SG-2.1.2.4)**. In terms of key re-use **(SG-2.1.1.3)**, the protocol provides fresh keys for every session. The authors have done a comprehensive analysis of the randomness of biometrics (DCBs) employed in the protocol **(SG-2.1.1.4.1)**, hence there are no exploitable opportunities for the attacker in this regard. Usually when using ECCs **(SG-2.1.1.4.2)**, the effective key length is reduced since a portion of the key is used to provide redundancy for error correction, which sacrifices entropy [13]. However in P-KeyEx the 80-bit K_{RI} has the same effective key length since it is encoded with 204 bits using BCH codes, which creates a Hamming distance of 37 bits between code words [21]. Similar to the reasoning for P-Sec, P-KeyEx is not vulnerable to MITM and reflection attacks **(SG-2.1.2.1-2)**. Due to the key-confirmation steps at the end of the protocol, P-KeyEx is robust against key-confirmation attacks **(SG-2.1.2.3)**. The generation of fresh K_{RI} for every session protects P-KeyEx against brute-force attacks and makes side-channel attacks and social engineering inapplicable **(SG-2.2.1-3)**.

SG-3: Reader Proxy Attack & SG-4: Abusing Emergency Access. The only reader proxy attack applicable to P-KeyEx is the use of rPPG for heartbeat measurement **(SG-3.1.1)**. However, the high frame-rate requirement for the cameras, the need for the subject to be stable etc., make these attacks highly unlikely in practice. This *remote attack* is also the only relevant method for P-KeyEx in SG-4 **(SG-4.3.4.1.1)**.

SG-5: Denial of Service. Looking at SG-5, we see that P-KeyEx is susceptible to DoS attacks[7]. A can send ID_R to I to initiate a session **(SG-5.1.2.1, SG-5.2.1.1)**, and can also exchange valid misdetection flags to keep the session alive, even though he/she is not performing the witness generation his/herself (see Figure 5). This will force I to perform fuzzy commitment and hash calculation. A subsequent *msg* packet from A will result in an unnecessary decryption operation from I. Thus, A can cause serious battery drain using this method. A can also continuously modify or drop the misdetection flags during witness generation, which would

[7]It was not the objective of the authors of [21] to address availability.

Table 1: Identified threats per protocol

Sub-goal	P-Sec	P-KeyEx	P-Auth
1	SG-1.1, SG-1.3.1	SG-1.4.2.1	SG-1.4.2.1
2	SG-2.1.1.1, SG-2.1.1.2, SG-2.1.1.3		SG-2.1.2.2
3		SG-3.1.1	
4		SG-4.3.4.1.1	
5	SG-5.1.2.1, SG-5.1.3, SG-5.2.1.1	Same as P-Sec	Same as P-Sec

Table 2: Evaluated-protocol services and additional features (N/E: non-eligible)

Type of service/feature	P-Sec	P-KeyEx	P-Auth
Confidentiality	✓	N/E	N/E
Integrity	✓	N/E	N/E
Authentication	✓	✓	
Non-repudiation			
Availability			
Access control	✓		
Emergency access		✓	✓
Key management		✓	N/E
Key freshness		✓	N/E
Perfect forward secrecy		✓	N/E
Error recovery			

result in R and I not being able to agree on IPIs needed for generating witnesses **(SG-5.1.3)**. Thus, A would be able to block any legitimate access (jam DoS).

5.3 P-Auth analysis

The evaluation of P-Auth has an attack-tree traversal similar to P-Sec and P-KeyEx. P-Auth shares issues pertaining to availability **(SG-5.1.2.1, SG-5.1.3, SG-5.2.1.1)** and access control **(SG-1.4.2.1)**. However, the major difference is the vulnerability to reflection attacks **(SG-2.1.2.2)** since the handshake is quite notably symmetric. A can exploit this by initiating connection with either R or I and then replaying the same messages that are received from either of these nodes after the session-key establishment. For instance, if A is trying to communicate with I, it can send its nonce, hash and IPI value equal to N_I, H_I and IPI_I, respectively. This would satisfy the checks for hash equality and IPI similarity at the final stage, resulting in incorrect authentication of A.

6 Recommendations

The results of the protocol evaluations in Section 5 are summarized in Table 1. Recall that here we have only shown the portion of the attack trees relevant to the evaluation. The impact of identified threats on the CIANA services is shown in Table 2, which also lists the coverage of the desired features specific to IMDs. Note that these protocols cannot be compared directly because of their different security aims. N/E stands for *non-eligible*, which denotes a service that was not intended to be supported by the protocol in question. Looking at the weaknesses, P-Auth, in particular, fails to address *authentication* because of its vulnerability to reflection attacks. As an example, this can be resolved if both the nodes verify that the two nonces or IPI values are not exactly the same. The protocols do not guarantee *non-repudiation* because they

do not employ digital signature and public key infrastructure[8]. This is a valid concern since we assume the possibility of attacks from trusted entities (see SG-1.4 and G-2.3). As evident from the likelihood of attacks (see Figure 2), the protocols are most susceptible to DoS attacks, which hurt *availability*. We recognize that any similar protocol in isolation is susceptible to some form of DoS attack, however, in the case of IMDs, it is highly recommended to at least incorporate protection within the protocols against battery and function DoS, as evident from Section 3.3.1. One practical way to solve this issue is to use RF-energy-harvesting-based authentication [6, 23]. Also, although P-KeyEx and P-Auth try to provide authentication in terms of establishing trust, they lack role identification/distinction, hence failing to address *access control*. P-Sec can, however, support this feature since it facilitates multiple pre-shared passwords, allowing for multiple users (or user groups) to be distinguished. It can for instance appropriately utilize certain bits of ID_R for privilege information. It is not possible for A to modify these bits since ID_R is already pre-installed in I. The protocols do not support *error recovery*, i.e., any form of error during the protocol messages results in session termination. While this is typically done at a different layer and is out of scope for these protocols, it might be useful to incorporate this feature within the protocols for *lightweight* implementations.

It can also be observed from Table 2 that the protocols have a largely non-overlapping coverage of the targeted features. If P-KeyEx is combined carefully with P-Sec to provide authenticated key exchange, the resulting scheme resolves P-Sec's issues related to symmetric-key usage and supplements it with emergency access. Moreover, in order to provide long-term *security*, some simple modifications are recommended: The block size employed for the PRESENT-80 cipher can be changed from 64 to 128 bits (assuming MISTY1 is not used because of the reasons highlighted in Section 5) to protect against birthday attacks. If this is not possible due to energy constraints, at least the session key should be changed frequently.

Thus, attack trees can provide a very handy tool for quantifying weaknesses of the IMD systems. What is more, with an increasing volume of literature proposing security protocols for IMDs and related fields, employing attack trees can help to compile such contributions into new powerful protocols with a larger coverage of attacks, as demonstrated in the above evaluation of example protocols. The attack trees can be made more focused if the information about the commercial Reader/IMD-system implementations is made available. In essence, this work highlights the top-down approach instead of evaluating actual implementations.

7 Related Work

Attack trees have been used as a tool to illustrate the attack scenarios within various domains and systems but their usage in the medical domain has been very limited. To the best of our knowledge there is no such work that specifically targets IMDs. Taylor et al. [24] have formulated an attack tree specifically applicable to the patient-controlled analgesia application, and subsequently suggested mitigating solutions. This work was extended by Xu et al. [27] who discuss a methodology to generate these attack trees. Luckett et al. [12] discuss the use of attack graphs for vulnerability identification, risk assessment and subsequent derivation of mitigation strategies to protect ambulatory medical devices (AMBs).

Populating the attack trees in this paper has been based on in-house endeavor. Additionally, these were carefully expanded by consulting the following recent work in literature. Humayed et al. [8] discuss threats, vulnerabilities, attacks and security challenges of cyber-physical systems

[8]IMDs normally do not have energy and computing resources to support asymmetric cryptography. Moreover, supporting non-repudiation also requires a robust logging infrastructure, which cannot be supported by a limited on-device memory.

including medical devices. ALTawy et al. [2] study the trade-offs between security, safety and availability in cyber physical systems using IMDs as a case study. Camara et al. [5] survey the security goals for future IMDs and analyze the protection mechanisms discussed so far in literature. Rathore et al. [17] provide an overview of the attacks pertaining to IMDs. Rushanan et al. [19] have done a rigorous survey of security schemes and attacks pertaining to IMDs and health-related BANs, and highlight emerging threats, but there is a need to perform a similar type of analysis to cover new attacks (e.g., [14, 9]) and protocols (e.g., [21, 16]), since the work was done more than three years ago. Marin et al. [13] evaluate security of two physiological signal based security protocols for IMDs and have proposed solutions to improve security of such systems. Based on our work, we envision an open-access attack-tree resource where current research efforts can reflect upon and also contribute to.

8 Conclusions

In this paper, we have proposed a systematic threat-modeling approach to analyze IMD security. This attack-tree-based approach offers a comprehensive and highly structured picture of the strengths and weaknesses of the IMD systems. As a case study, we applied our threat analysis on three state-of-the-art IMD secure-communication protocols found in literature. We have showed that this evaluation makes the task of coming up with security improvements significantly easier. By using our approach, we have not only confirmed the capabilities/limitations of the protocols (as identified by their authors) but also discovered certain limitations (e.g. susceptibility to DoS and reflection attacks etc.). Moreover, it has enabled us to easily visualize and propose a combined use of these protocols, for better coverage of the identified security services and features. This work, thus, paves the way for building more robust and secure protocols for IMDs and mobile-health systems. What is more, it provides a structured approach towards performing system-level security evaluation to include many possible attack surfaces. As future work, we intend to develop an open-access attack-tree resource with the aim to consolidate the research efforts in this domain. We also intend to extend the attack-tree with operational security aspects based on past experiences from medical practice. We hope that this effort is a step in the right direction, towards the much needed standardization of IMD security.

9 Acknowledgments

This work has been supported by the EU-funded project SDK4ED (Grant Agreement No. 780572).

References

[1] Abbott. Proclaim^{TM} Elite Recharge-Free SCS System, 2018.

[2] Riham AlTawy and Amr M Youssef. Security tradeoffs in cyber physical systems: A case study survey on implantable medical devices. *IEEE Access*, 4:959–979, 2016.

[3] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[4] Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-) security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 456–467. ACM, 2016.

[5] Carmen Camara, Pedro Peris-Lopez, and Juan E. Tapiador. Security and privacy issues in implantable medical devices: A comprehensive survey. *Journal of biomedical informatics*, 55:272–289, 2015.

[6] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. In *2008 IEEE Symposium on Security and Privacy*, pages 129–142. IEEE, 2008.

[7] High-Tech Bridge. Improper Access Control [CWE-284], 2015.

[8] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-physical systems security–a survey. *IEEE Internet of Things Journal*, 2017.

[9] ICS-CERT. Advisories ICSMA-17-009-01A and ICSMA-17-241-01, 2017.

[10] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM CCS*, pages 28–36, 1999.

[11] D. F. Kune, J. Backes, S. S. Clark, D. Kramer, M. Reynolds, K. Fu, Y. Kim, and W. Xu. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *2013 IEEE Symposium on Security and Privacy*, pages 145–159. IEEE, 2013.

[12] Patrick Luckett, Jeffrey McDonald, and William Glisson. Attack-Graph Threat Modeling Assessment of Ambulatory Medical Devices. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[13] Eduard Marin, Enrique Argones-Rúa, Dave Singelée, and Bart Preneel. A survey on physiological-signal-based security for medical devices. *IACR Cryptology ePrint Archive*, 2016:867, 2016.

[14] Eduard Marin, Dave Singelée, Flavio D Garcia, Tom Chothia, Rik Willems, and Bart Preneel. On the (in) security of the latest generation implantable cardiac defibrillators and how to secure them. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 226–236. ACM, 2016.

[15] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[16] Steffen Peter, Bhanu Pratap Reddy, Farshad Momtaz, and Tony Givargis. Design of secure ecg-based biometric authentication in body area sensor networks. *Sensors*, 16(4):570, 2016.

[17] H. Rathore, A. Mohamed, A. Al-Ali, X. Du, and M. Guizani. A review of security challenges, attacks and resolutions for wireless medical devices. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1495–1501. IEEE, 2017.

[18] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. Heart-to-heart (H2H): authentication for implanted medical devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1099–1112. ACM, 2013.

[19] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson. SoK: Security and privacy in implantable medical devices and body area networks. In *2014 IEEE Symposium on Security and Privacy*, pages 524–539. IEEE, 2014.

[20] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[21] Robert M Seepers, Jos H Weber, Zekeriya Erkin, Ioannis Sourdis, and Christos Strydis. Secure key-exchange protocol for implants using heartbeats. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 119–126. ACM, 2016.

[22] Robert Mark Seepers. *Implantable Medical Devices: Device security and emergency access*. PhD thesis, Erasmus University Medical Center, Rotterdam, Netherlands, December 2016.

[23] Christos Strydis, Robert M Seepers, Pedro Peris-Lopez, Dimitrios Siskos, and Ioannis Sourdis. A system architecture, processor, and communication protocol for secure implants. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(4):57, 2013.

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

[25] Yosuke Todo. Integral cryptanalysis on full MISTY1. *Journal of Cryptology*, 30(3):920–959, 2017.

[26] Wim Verkruysse, Lars O Svaasand, and J Stuart Nelson. Remote plethysmographic imaging using ambient light. *Optics express*, 16(26):21434–21445, 2008.

[27] J. Xu, K. K. Venkatasubramanian, and V. Sfyrla. A methodology for systematic attack trees generation for interoperable medical devices. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–7. IEEE, 2016.