

Anniken Wium Lie

Security Analysis of Wireless Home Monitoring Units in the Pacemaker Ecosystem

Master's thesis in Communication Technology

Supervisor: Marie Elisabeth Gaup Moe

June 2019

Anniken Wium Lie

Security Analysis of Wireless Home Monitoring Units in the Pacemaker Ecosystem

Master's thesis in Communication Technology

Supervisor: Marie Elisabeth Gaup Moe

June 2019

Norwegian University of Science and Technology

Faculty of Information Technology and Electrical Engineering

Department of Information Security and Communication Technology



Norwegian University of
Science and Technology

Title: Security Analysis of Wireless Home Monitoring Units in the Pacemaker Ecosystem
Student: Anniken Wium Lie

Problem description:

Technological evolvement has made it possible to connect about anything to the Internet, which is reflected in today's society. Everything from banking services to monitoring our activity and exercise requires an Internet connection, and we are gradually becoming more dependent on connected devices. Some people even put their very lives in the hands of this technology, as pacemakers and other medical devices are relying on wireless communication. It is therefore literary of vital importance that these devices are secured adequately, as a security breach could have fatal consequences.

Today's pacemakers are connected to other devices via wireless communication in what we refer to as the pacemaker ecosystem. In this thesis, we aim to take a closer look at a specific component of the ecosystem: the Home Monitoring Unit (HMU). The HMU is a small computer that receives medical information from a pacemaker. The information is automatically relayed from the HMU to a Data server. However, exact details of the transmission protocol and the firmware of the HMU are proprietary and only known by the vendor. Not only is this considered bad practice, but it also makes it challenging to verify if the equipment is secured and complies with current regulations.

The goal is to analyze and evaluate the implementation of the wireless protocol used between the HMU and a Data Server and look for vulnerabilities that might pose threat to the safety of pacemakers patients or disclose information about them.

Responsible professor: Marie Elisabeth Gaup Moe, NTNU/SINTEF
Supervisor: Marie Elisabeth Gaup Moe, NTNU/SINTEF
Co-Supervisor: Ravishankar Borgaonkar, SINTEF

Abstract

An increasing number of medical devices are being connected to the Internet, thereby broadening their attack surface. Some people's lives depend on these devices, hence it is of vital importance that they are secure. A pacemaker is an example of such device, communicating with other devices of what is referred to as the pacemaker ecosystem. However, the communication protocols in use are proprietary and kept secret by manufacturers. This makes it challenging to verify whether sufficient security has been implemented. Previous research has disclosed several vulnerabilities in similar systems.

In this thesis, we investigate devices from Biotronik, a leading German manufacturer. We perform a security analysis of the communication protocols between different Home Monitoring Unit (HMU) models and a Data Server. An HMU is a device that transmits patient data from a pacemaker to a Data Server where health-care personnel can access the information.

The communication is based on GSM, a wireless communication standard with several well-known security vulnerabilities. These vulnerabilities allow us to eavesdrop on the communication by setting up an illegitimate Base Transceiver Station (BTS) which the HMUs connect to.

Using Commercial off-the-shelf (COTS) equipment and decommissioned HMUs from eBay, we demonstrate how reverse engineering of the communication protocol is possible for some of the HMU models. Our analysis results in the identification of several protocol and implementation weaknesses, and demonstrates how attack vectors can be exploited. We also validate that our findings apply to older HMU models still in use by patients.

While we have discovered several vulnerabilities, our study also suggest that Biotronik have made efforts to implement security mechanisms in all their HMU models, and that their newer models are significantly more resistant to security attacks.

Sammendrag

Et økende antall medisinske enheter kobles opp mot internett, noe som gir dem en bredere angrepsflate. Enkelte menneskers liv ligger i hendene på slike enheter, og det er dermed avgjørende at de er sikre. En pacemaker er et eksempel på en slik medinsk enhet, som kommuniserer med andre enheter i det vi referer til som et pacemaker-økosystem. Kommunikasjonsprotokollene mellom enhetene i økosystemet er imidlertid proprietære og holdes hemmelige av produsentene. Dette gjør det utfordrende å verifisere om tilstrekkelig sikkerhet er implementert, Tidligere forskning har avslørt flere sårbarheter i lignende systemer.

I denne oppgaven undersøker vi enheter fra Biotronik, en ledende tysk produsent av medisinsk utstyr. Vi utfører en sikkerhetsanalyse av kommunikasjonsprotokollene mellom forskjellige hjemmeovervåkningsenheter (HMU) og en dataserver. En HMU er en enhet som overfører pasientdata fra en pacemaker til en dataserver hvor helsepersonell kan få tilgang til informasjonen.

Kommunikasjonen er basert på GSM, en trådløs kommunikasjonsstandard med flere kjente sikkerhetsproblemer. Disse sårbarhetene tillater oss å avlytte kommunikasjonen ved å sette opp en falsk basestasjon (BTS) som HMUene kobler seg til.

Ved å bruke hyllewareutstyr og brukte HMUer kjøpt på eBay, demonstrerer vi hvordan dekonstruering av kommunikasjonsprotokollen er mulig for noen av HMU-modellene. Vår analyse resulterer i identifisering av flere protokoll- og implementeringssvakheter, og demonstrerer hvordan angrepsvektorer kan utnyttes. Vi bekrefter også at funnene våre gjelder for eldre HMU-modeller som fortsatt er i bruk av pasienter.

Selv om vi oppdaget flere sårbarheter, indikerer resultatene også at Biotronik har gjort en innsats for å implementere sikkerhetsmekanismer i alle HMU-modellene, og at de nyere modellene er betydelig mer motstandsdyktige mot sikkerhetsangrep.

Preface

This Master's Thesis is the final deliverable of the Master of Science Degree in Communication Technology with specialization within Information Security at the Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU).

The thesis is also part of a larger project on the security of medical devices at SINTEF.

Much appreciation is shown to my supervisor and responsible professor, Marie Elisabeth Gaup Moe, for suggesting an exciting research topic and for providing feedback and excellent support during the work. Special thanks also go to Ravishankar Borgaonkar for valued guidance, and for helping me with acquiring the necessary equipment for the experiments.

Gratitude also goes to Guillaume Nicholas Bour, who also conducted research as part of the SINTEF project, for a great collaboration. The research results would not have been the same without your help.

Finally, I express my deepest appreciation to my family. I could not have done this without our late-night writing sessions, your proofreading, and your unconditional support.

Anniken Wium Lie

Trondheim, 18th June 2019

Contents

List of Figures	ix
List of Tables	xiii
Listings	xv
List of Acronyms	xvii
1 Introduction	1
1.1 Context and Motivation	1
1.2 Problem Description	3
1.3 Scope	3
1.3.1 Research Questions and Objectives	5
1.4 Contributions of the Thesis	5
1.5 Outline of the Thesis	6
2 Background	7
2.1 Global System for Mobile Communication (GSM)	7
2.1.1 Services	7
2.1.2 Network Architecture and Components	9
2.1.3 General Packet Radio Service (GPRS)	12
2.1.4 Radio Frequencies	13
2.1.5 Security in GSM	13
2.2 Security vulnerabilities and attacks on GSM	15
2.2.1 Illegitimate BTS	16
2.2.2 Jamming and downgrade to GSM	17
2.3 Security Model	17
3 Related Work	19
3.1 The Pacemaker Ecosystem	19
3.2 Wireless communication	20
4 Research Methodology	23

4.1	Design Science	23
4.1.1	A framework for Design Science	24
4.1.2	The Design Activity	25
4.1.3	Investigation	27
4.1.4	Correlation Between Design and Investigation	27
4.2	Threat modeling	28
4.3	Trial-and-error	28
4.3.1	Black-box Security Testing	29
4.3.2	Protocol Reverse Engineering	29
4.4	Limitations	30
4.5	Legal and Ethical Considerations	30
5	Threat Model	33
5.1	Attack Scenarios	35
5.1.1	Gather Patient Data	35
5.1.2	Harm a patient	35
5.1.3	Misuse SIM card services	36
5.2	Difficulty of attacks	36
5.3	Potential Attack Vectors	37
6	Experimental Setup	39
6.1	Home Monitoring Unit	39
6.2	Mobile Network	41
6.2.1	Setting up an Illegitimate BTS	43
6.3	Data Server	47
6.4	Additional Tools	47
7	Results	51
7.1	Investigating the HMUs	51
7.1.1	Components of the HMUs	52
7.1.2	Technical Manuals	54
7.2	Inspecting the SIM cards	56
7.2.1	Obtaining PIN codes	56
7.2.2	Inspecting the unlocked SIM cards	57
7.3	Investigating the interaction between the HMUs and a legitimate mobile network (GSM)	63
7.3.1	Original SIM Card inserted into the HMU(s)	64
7.3.2	[II-S only] Inserting a valid SIM card from another HMU	67
7.3.3	Inserting a valid HMU SIM card Into a third-party device	70
7.3.4	Inserting a SIM card with a running subscription that is not from an HMU	71

7.4	Investigating the Interaction Between the HMUs and an Illegitimate BTS	72
7.4.1	Eavesdropping on the Communication Channels	73
7.4.2	Spoofing the Data Server	77
7.4.3	Sending SMSs to the HMU	81
7.4.4	Spoofing the SIM card	81
7.5	Analyzing the Intercepted Data	82
7.5.1	Data Packets	82
7.5.2	SMS	87
7.5.3	Comparing and Analyzing the SMS User Data (UD)	88
7.6	Summary of Results	92
8	Discussion	95
8.1	Security Progression in HMU Models	95
8.1.1	SIM Cards	95
8.1.2	Vulnerabilities in GSM	96
8.1.3	Communication Channels	96
8.1.4	Data Security	97
8.1.5	Improving Security	97
8.1.6	Lack of Routines Around Revocation of Access	98
8.2	Our Results in Comparison with Related Work	98
8.3	Confirmed Attack Scenarios	99
8.3.1	An adversary can disclose data in cleartext	99
8.3.2	An adversary can access the private APN	100
8.3.3	An adversary can get free Internet Access	101
8.4	Potential Attack Scenarios	101
8.4.1	Massive data breach	101
8.4.2	Prohibit Communication on a Large Scale	101
8.4.3	Severe Patient Harm	102
8.5	Suggested Countermeasures	102
8.6	Limitations of the Results	103
8.7	Further work	103
9	Conclusion	105
	References	107
	Appendices	
A	Scripts	113
A.1	SMS Data Extraction	113
A.2	AES and DES Decryption	114

B Shared Folder	123
B.1 Creating a Shared Folder	123

List of Figures

1.1	Overview of the pacemaker ecosystem. The components are numbered: (1) Pacemaker, (2) Home Monitoring Unit, (3) Mobile Network, (4) Data Server, (5) Healthcare personnel, (6) Programmer.	2
1.2	CardioMessenger LLT and CardioMessenger II-LLT.	3
1.3	CardioMessenger 2-S and CardioMessenger 3G.	4
2.1	Overview of the TPDU for a SMS-SUBMIT message.	8
2.2	An overview of a simplified GSM architecture.	9
2.3	A simplified overview of the route of an SMS in GSM.	12
2.4	An overview of how an illegitimate BTS works.	16
4.1	Our implementation of Wieringa's [1] framework for Design Science. . .	24
4.2	Our implementation of the Design Cycle.	26
4.3	Correlation between the design and investigation activities of this thesis. The cycle illustrates how the validation model is used to investigate, and how this investigation is providing input that can further improve the model.	28
4.4	Lumax 540 VR-T, an ICD from Biotronik, whose battery has reached End of Service (EOS), as we observed when interrogating it with a programmer (right).	31
5.1	Overview of the subsystem of the pacemaker ecosystem that is the focus of this thesis.	33
6.1	Relevant components of the pacemaker ecosystem for our research: The Home Monitoring Unit (1), Mobile Network (2), and Data Server(3). . .	39
6.2	The phones used in the experiments. From left: LG-A100, Nokia 6070, Iphone 4s, Avvio 750.	41
6.3	Overview of the experimental setup for the illegitimate BTS. 1. Jammer, 2. Vert900 antennas, 3. Macbook Pro running VirtualBox. VM inside VirtualBox running OpenBTS, 4. USRP B210	42
6.4	Settings of GSM identity in OpenBTS.	45

6.5	Overview of the system architecture and how the different components are connected.	45
6.6	Screenshot from OpenBTS showing a subscriber that has not been authenticated to the illegitimate BTS.	46
6.7	The Shikra. The pinouts are placed on the right side.	48
6.8	SIM card reader, supporting regular, nano and micro SIM cards.	49
7.1	Board of the CardioMessenger LLT. 1. Micro-controller, 2. SIM card, 3. Modem.	52
7.2	Board of the CardioMessenger II-LLT. 1: Micro-controller, 2: SIM card. The modem was not visible.	52
7.3	Board of the CardioMessenger 2-S. 1: Micro-controller, 2: SIM card, 3: Modem.	53
7.4	Board of the CardioMessenger Smart 3G. 1: Micro-controller, 2: SIM card. The modem is not visible.	53
7.5	SIM cards found in the HMUs. From left: LLT, II-LLT, II-S, Smart 3G.	54
7.6	The II-S indicating normal operation.	55
7.7	Experimental setup of SIM card reader and SIMspy2.	59
7.8	Screenshot from SIMspy2 showing the FDN list retrieved from a SIM card. All the HMU SIM cards contain identical FDN lists.	60
7.9	Screenshots of the LG and the Nokia phones with HMU SIM cards inserted. The picture on the left represents the status of SIM card for the LLT and II-S, while the right presents how the II-LLT SIM card still has a valid subscription.	61
7.10	Screenshot from the Avvio phone, showing how we were able to connect to the Internet through a T-Mobile APN using the II-LLT SIM card.	62
7.11	Experimental setup used for intercepting the communication between the micro-controller and modem of the II-S. The red squares indicate where the HMU and The Shikra are connected.	64
7.12	Screenshot from Hex Fiend revealing that the credentials are sent in cleartext over the private communication channel	69
7.13	Screenshot from Hex Fiend revealing that the the remaining data sent from the HMU is not in cleartext.	69
7.14	Phone with II-LLT SIM card connected to the private APN. The assigned IP address is in the same range as the Data Server. The phone number confirms that the SIM card of the II-LLT is used.	71
7.15	Pinging the Data Server (172.16.14.1) from phone with the II-LLT SIM card inserted that is connected to the same private APN.	72
7.16	Screenshot from OpenBTS, showing an HMU being assigned an IP address from the Illegitimate BTS.	73
7.17	Screenshot from Wireshark showing SMSs after having applied the filter <i>gsm_sms</i>	74

7.18	Screenshot from Wireshark showing the UD from a single SMS from an HMU.	75
7.19	Screenshot from Wireshark showing the response received from the network upon sending an SMS to a user not registered in the network. . . .	75
7.20	Screenshot from Iphone 4s with sysmocom SIM card, showing SMSs received from the LLT 1234 is the number that the LLT SIM card was assigned in OpenBTS	76
7.21	Screenshot from Wireshark showing how the II-S HMU (192.168.99.1) is attempting to connect to an unreachable IP address (172.16.14.1)	77
7.22	Caption	78
7.23	Overview of the overall architecture, including the HMUs, the illegitimate BTS, and the spoofed Data Server	79
7.24	Screenshot from VM with IP 172.16.14.1 listening on port 2323, accepting a connection from a HMU (172.16.14.2).	79
7.25	Screenshot from Wireshark, packets are filtered on the IP address of the Data Server, 172.16.14.1.	83
7.26	Screenshots from Hex Fiend, showing the comparison of data packets from the same HMU. From left: II-LLT, II-S, Smart 3G. Parts of the credentials have been redacted.	84
7.27	Connecting to the JTAG interface on the CardioMessenger II-S.	85
7.28	The script searches through the RAM to find every every possible AES key.	86
7.29	Decrypted data packets from the II-S.	87
7.30	Structure of an SMS-SUBMIT message.	87
7.31	Screenshot from Wireshark showing the format of an SMS sent from the HMUs.	88
7.32	Structure of the SMS UD sent from the LLT.	89
7.33	Structure of the different SMS UD observed from the II-S. From the top: 0604, 0601, 0607.	89
7.34	Screenshot from Excel showing the pattern of SMSs sent by the LLT. . .	89
7.35	The SMSs from the II-S contained two identical 8-bit data blocks. . . .	90
7.36	ECB mode of operation. Taken from [2].	90
7.37	Screenshot from Excel, presenting parts of ciphertext vs. cleartext of SMS UD from the II-S.	91
B.1	Screenshot of VirtualBox, illustrating how to add a new optical drive. .	123
B.2	Screenshot of VirtualBox, illustrating what it looks like when the Guest Addition has been added.	124
B.3	Screenshot of VirtualBox, illustrating how to add a shared folder. . . .	124

List of Tables

2.1	Examples of MCC and MNC values in Norway and Germany [3][4]. . . .	10
2.2	Allocated GSM spectrum in Norway [5].	13
2.3	Allocated UMTS (3G) spectrum in Norway [5].	13
5.1	Potential attack vectors.	37
7.1	Overview of the serial numbers and IMEI numbers of the HMUs in question in a redacted format.	51
7.2	Overview of which modems support which technologies	54
7.3	PIN codes of the SIM cards in the HMUs	56
7.4	SMSs, contacts and phone number stored on the SIM cards from the three oldest HMU units.	60
7.5	Overview over the subscription status of the SIM cards and their communication capabilities.	61
7.6	Overview of what HMUs indicate normal operation while interacting with a legitimate mobile network.	65
7.7	IMSI and corresponding IMEI of the SIM cards and the HMUs. Parts of the numbers have been redacted to anonymize the HMUs.	73
7.8	Overview of which devices we were able to observe what type of data from.	82
7.9	Overview over the average entropy of data packets being sent from each HMU.	84
7.10	Overview of what the different signs mean in table 7.11 and 7.12.	92
7.11	Overview of the already implemented security measures in the HMU models.	93
7.12	Overview of the identified vulnerabilities in the HMU models.	94

Listings

7.1	Excerpt from AT-commands illustrating how the HMU II-S is repeatedly trying to establish a network connection to the available networks (24201, 24202) and obtain an IP address by sending APN credentials.	66
7.2	Excerpt from AT-commands illustrating how the HMU II-S is attempting to send an SMS without a network connection.	67
7.3	Excerpt from the AT-commands, presenting data being sent from the HMU to the Data Server. Parts of the credentials have been redacted.	68
A.1	Script for extracting relevant information from SMS TPDU's	113
A.2	Script for decrypting data that has been encrypted using either AES or DES.	114

List of Acronyms

AES	Advanced Encryption Standard.
APN	Access Point Name.
AT	ATtention.
AuC	Authentication Centre.
BSC	Base Station Controller.
BTS	Base Transceiver Station.
CLI	Command Line Interface.
COTS	Commercial off-the-shelf.
DA	Destination Address.
DCS	Data Coding Scheme.
DES	Data Encryption Standard.
DHCP	Dynamic Host Configuration Protocol.
DoS	Denial of Service.
ECB	Electronic Code Book.
EOS	End of Service.
E-SIM	Embedded SIM card.
ETSI	European Telecommunications Standards Institute.
EU	European Union.
FCC	Communications Commission.

FDA Food and Drug Administration.

FDN Fixed Dialling Number.

FIPS Federal Information Processing Standard.

GDPR General Data Protection Regulation.

GPRS General Packet Radio Service.

GSM Global System for Mobile Communication.

GUI Graphical User Interface.

HLR Home Location Register.

HMU Home Monitoring Unit.

ICD Implantable Cardiac Defibrillator.

IMD Implantable Medical Device.

IMEI International Mobile Equipment Identity.

IMSI International Mobile Subscriber Identity.

IoT Internet of Things.

IP Internet Protocol.

IS Information System.

Ki Subscriber Authentication Key.

LTE Long Term Evolution.

MCC Mobile Country Code.

ME Mobile Equipment.

MITM Man-in-the-Middle.

MNC Mobile Network code.

MS Mobile Station.

MSC Mobile Switching Centre.

MSIN Mobile Subscriber Identification Number.

NAT Network Address Translation.

NSD Norwegian Center for Research Data.

NTNU Norwegian University of Science and Technology.

OS Operating System.

PIN Personal Identification Number.

PLMN Public Land Mobile Network.

PUK PIN Unlock Code.

RAM Random Access Memory.

SCA Service Center Address.

SDR Software Defined Radio.

SIM Subscriber Identity Module.

SMS Short Message Service.

SMSC Short Message Service Center.

SSH Secure Shell.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TMSI Temporary Mobile Subscriber Identity.

TPDU Transfer Protocol Data Unit.

UD User Data.

UMTS Universal Mobile Telecommunications System.

US United States.

USRP Universal Software Radio Peripheral.

VLR Visitor Location Register.

VM Virtual Machine.

Chapter 1

Introduction

1.1 Context and Motivation

Technological development has made it possible to connect many sorts of devices to the Internet. Devices that used to be standalone, ranging from refrigerators to cars, now have the capability to automatically gather, interpret and share information. This phenomenon is referred to as the Internet of Things (IoT). One concern about this development is that security is not keeping up with the pace of innovation [6].

We are becoming increasingly dependent on connected devices in many areas of our daily lives. Some people, for example patients with Implantable Medical Devices (IMD), put their very lives in the hands of this technology. As such, it is literally of vital importance that these devices are sufficiently secured, as a security breach could have fatal consequences. Examples of IMDs are pacemakers and Implantable Cardiac Defibrillators (ICD).

Today's pacemakers and ICDs are connected to other devices via wireless communication in what we refer to as the pacemaker ecosystem. Figure 1.1 shows an overview of the ecosystem.

Pacemakers assist patients who are suffering from a heart condition to maintain a stable heart rhythm, and to send out impulses if it senses that the heart is beating slowly or unevenly. An ICD is much like a pacemaker, but has the additional ability to send out a shock to the heart if it is beating too fast [7]. For the sake of simplicity, we will refer to both pacemakers and ICDs as pacemakers throughout the thesis.

The pacemaker is implanted in the chest area of a patient, where it is connected to the heart via wires called leads. Through these, the pacemaker continuously monitors and helps maintain a stable heart rhythm. At the same time, patient data is collected and stored in the device memory and can be accessed via wireless communication channels. The patient data also includes information about the status of the device.

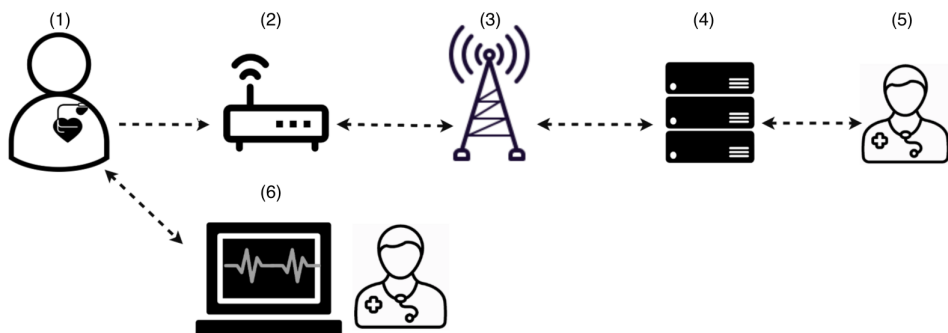


Figure 1.1: Overview of the pacemaker ecosystem. The components are numbered: (1) Pacemaker, (2) Home Monitoring Unit, (3) Mobile Network, (4) Data Server, (5) Healthcare personnel, (6) Programmer.

A *programmer* is an external device that is located in the hospital and operated by healthcare personnel. The programmer is used to configure the pacemaker initially, and later to interrogate the pacemaker for the status of the device and of the patient. Based on this data, the programmer can adjust the pacemaker configuration to better meet the patient’s need. The programmer is also used to update the firmware of the pacemaker and to pair the pacemaker with a *Home Monitoring Unit (HMU)*.

The HMU is responsible for transmitting the patient data to healthcare personnel whenever a patient is outside the hospital. This allows for continuous remote patient monitoring. However, an HMU does not have the configuration capabilities of the programmer. The data is transmitted to a *Data Server*, where healthcare personnel can access it.

The HMU either communicates over a phone line or via a wireless communication channel; only the latter is discussed in the thesis. The transmission takes place regularly, as well as when a trigger criterion is met, such as abnormal activity, either from the heart of the patient or from the device itself. The patient must be close to the HMU when the transmission takes place, somewhere between 20 centimeters and 2 metres. As such, the device is typically placed in the bedroom of a patient. Some HMUs also have a battery and can be carried around.

Introducing wireless communication into the pacemaker ecosystem has some benefits, such as enhancing patient care and hence improving the safety and the overall life quality for pacemaker patients. However, it also increases the attack surface of the system, making it more vulnerable to undesired events. Our motivation lies in the desire to assess the attack surface.

1.2 Problem Description

Manufactures of pacemakers and HMUs do not publish detailed information regarding how these devices communicate. This makes it challenging to verify if the devices are appropriately secured.

1.3 Scope

This thesis investigates the the security of different HMU models. More specifically, we analyze four different CardioMessenger HMU models from the Biotronik manufacturer. The HMU models range from year 2003 to current models, and can be seen in figures 1.2 and 1.3. Throughout this thesis, the models will be referred to with only their specific model names. For example, the CardioMessenger LLT is referred to as LLT.



Figure 1.2: CardioMessenger LLT and CardioMessenger II-LLT.



Figure 1.3: CardioMessenger 2-S and CardioMessenger 3G.

The reason for choosing Biotronik over other manufacturers was because equipment from Biotronik was available in the lab from previous experiments. Also, Biotronik HMUs that have previously been used by pacemaker patients are available for purchase online, for example on eBay. Therefore, we had access to several HMU models, both from patients in the US and the EU.

Based in Germany, Biotronik is a leading vendor of medical equipment. They have a long history of providing HMUs that support wireless communication rather than using fixed telephone lines. Biotronik was also the first vendor to get approval from the Food and Drug Administration (FDA) for a wireless HMU in 2001 [8].

In addition to Biotronik, a handful of vendors offer equipment in this domain. There have been several studies on the vulnerabilities of equipment from these vendors [9] [10] [11] [12]. Biotronik is the only vendor without any publicly disclosed vulnerabilities. However, this does not mean that vulnerabilities do not exist.

The goal of this thesis is to analyze the implementation of the wireless protocols used between the HMU and a Data Server, and to investigate their functionality and security. The main objective is to look for vulnerabilities that might disclose patient information or pose threats to the safety of patients.

1.3.1 Research Questions and Objectives

We have defined the following research questions and objectives for this thesis:

Research Questions (RQ):

RQ.1: How is patient data secured in transit between the HMU and the Data Server?

RQ.2: How are the communication channels between the HMU and the Data Server secured?

Research Objectives (RO):

RO.1: Analyze the existing security mechanisms of the artifact¹.

RO.2: Describe vulnerabilities in the artifact, if any.

RO.3: Describe possible countermeasures against vulnerabilities.

1.4 Contributions of the Thesis

To the best of our knowledge, this thesis is the first substantial analysis of the wireless communication between HMUs and Data Servers. The mobile communication standards, such as GSM, have been extensively studied. However, to our knowledge, we are the first to study their use in the context of HMUs from Biotronik.

Our analysis was performed using Commercial off-the-shelf (COTS) equipment, open source software tools, and decommissioned HMUs. Our work reveals that Biotronik has already implemented certain security mechanisms, such as encrypting the data that is transmitted over the air and using a communication channel inside a private APN.

On the contrary, we have also disclosed several security vulnerabilities that expose risk to the pacemaker ecosystem, and consequently the pacemaker patients. Exploiting these vulnerabilities, we have demonstrated attacks that affect the availability of the HMUs and the confidentiality of the data being transmitted. These attacks include spoofing the Data Server, gaining unauthorized access to a private APN in which the legitimate Data Server is located, and successfully decrypting data being transmitted from the HMU to the Data Server.

¹The term artifact stems from the research methodology in chapter 3; in this theses the communication protocol is the artifact.

1.5 Outline of the Thesis

Chapter 2 presents relevant background information for this thesis. This includes an introduction to the GSM standard, its vulnerabilities, and how these can be exploited using an illegitimate BTS. Also, we present a security model that consists of fundamental aspects of security.

Chapter 3 presents related work within security research of the pacemaker ecosystem and wireless communication.

Chapter 4 introduces our methodology. We also describe limitations and discuss legal and ethical considerations that must be taken into account.

Chapter 5 presents our threat model for the relevant parts of the pacemaker ecosystem.

Chapter 6 describes the experimental setup that serves as the basis for the conducted experiments. We explain how to set up an illegitimate BTS, and also present the additional tools we have used.

Chapter 7 presents our results. The chapter also includes details of the procedures we used to obtain the results.

Chapter 8 discusses our findings in a broader context and compare the different HMU models based on our results. We also present attack vectors, possible countermeasures, and limitations of our results. Suggestions for future work are also included.

Chapter 9 concludes the thesis.

Chapter 2

Background

This chapter presents relevant background information for this thesis. We begin with presenting the Global System for Mobile Communication (GSM). GSM forms the basis for the communication between HMUs and the Data Server and is therefore fundamental for our work. Although highly successful, GSM also has vulnerabilities that can be exploited. These are discussed in this chapter. In addition, we present a security model that will be the basis for categorizing security measures and vulnerabilities throughout the thesis.

2.1 Global System for Mobile Communication (GSM)

GSM has been an international standard for mobile communication since the early 1990s. It is often referred to as the second-generation (2G) of cellular technology, and is based on digital signals as opposed to its analog predecessor (1G).

GSM has several descendants such as the Universal Mobile Telecommunications System (UMTS) and the Long Term Evolution (LTE), commonly referred to as 3G and 4G. These technologies offer better services and security than GSM. Currently, many consider GSM an outdated protocol, and some carriers have started the process of shutting it down [13]. However, there still exists many devices that depend on the GSM network, for example medical devices.

2.1.1 Services

The traditional GSM network offers three services: Phone calls, Short Message Service (SMS) and Multimedia Messaging Service (MMS). In this thesis, only SMS is relevant and will be described below.

SMS is a service for transmitting text messages between mobile subscribers. A message is commonly referred to as an SMS, which will also be the term used throughout this thesis. An SMS can consist of up to 160 characters, depending on

the encoding. The default encoding is the GSM 7-bit alphabet, but an 8-bit data alphabet and a 16-bit UCS-2 alphabet can also be used [14]. For example, binary messages are encoded using an 8-bit encoding.

SMSs are sent inside Transfer Protocol Data Unit (TPDU) using the Short Message Relay Protocol (SM-RP). There exists several types of TPDUs, but in this thesis only the SMS-SUBMIT is relevant. The SMS-SUBMIT is used for sending SMSs from an MS to a Short Message Service Center (SMSC). See section 2.3. An overview of the SMS-SUBMIT TPDU can be seen in figure 2.1.

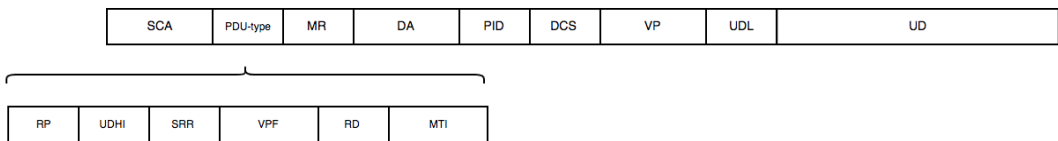


Figure 2.1: Overview of the TPDU for a SMS-SUBMIT message.

The following fields are relevant for this thesis:

- **Service Center Address (SCA):** This is the phone number of the SMSC that the SMS is forwarded to [14].
- **Destination Address (DA):** As the name implies, this is the phone number of the recipient of the SMS.
- **Data Coding Scheme (DCS):** This is an octet which defines basic information on how the recipient handset should process the received message [14].
- **User Data (UD):** This is the textual content of the SMS.

2.1.2 Network Architecture and Components

Figure 2.2 shows an overview of a simplified GSM architecture, only containing the components that are needed to understand this thesis. We will describe the different components in detail below.

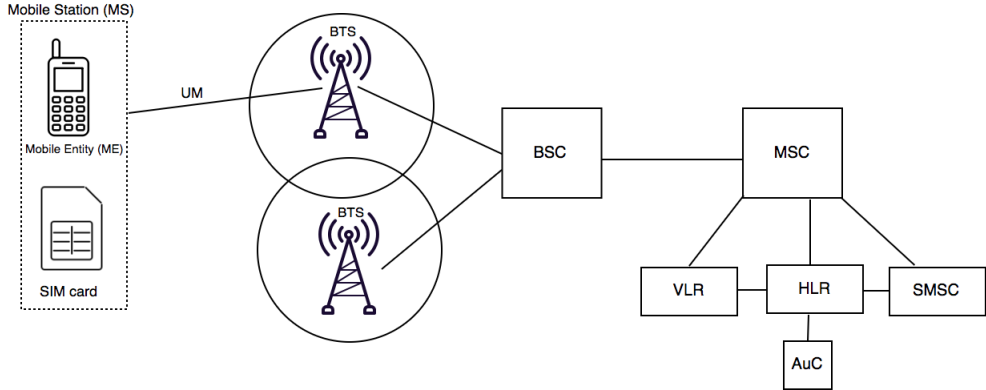


Figure 2.2: An overview of a simplified GSM architecture.

Mobile Station (MS)

A Mobile Station (MS) is the equipment used for communicating over a mobile network such as GSM. It consists of a Subscriber Identity Module (SIM) and a Mobile Equipment (ME). The ME is a physical communication device such as a cellphone or in the case of this thesis, an HMU. Each ME is uniquely identified by a 15-digit serial number called the International Mobile Equipment Identity (IMEI).

The SIM is a smart card that is placed inside the ME. The SIM is tied to a subscriber, and is needed for an ME to authenticate to a network in order to establish a connection. The SIM, however, is not tied to a specific ME.

Different types of information is stored on a SIMcard, ranging from personal data such as SMSs and contacts, configuration settings, and information needed for authentication. The first two are accessible and somewhat changeable, while the latter is securely stored.

The authentication information includes an authentication algorithm (A3), a ciphering key generation algorithm (A8) and a Subscriber Authentication Key (Ki). The Ki is a 128-bit permanent key used for authenticating a user to the network and for encrypting the communication.

The authentication information also includes an International Mobile Subscriber Identity (IMSI). The IMSI uniquely identifies a SIM, and is used for identification and authentication of a subscriber to the network. The IMSI number consists of several parts:

1. Mobile Country Code (MCC) (3 digits)
2. Mobile Network code (MNC) (3 digits)
3. Mobile Subscriber Identification Number (MSIN) (10 digits)

The MCC identifies the country of the subscription. The MNC identifies a Public Land Mobile Network (PLMN) operator within a country. A PLMN operator is an entity which offers wireless telecommunications services [15]. Finally, the MSIN identifies a subscriber within a PLMN [16]. Table 2.1 shows examples of MCCs and MNCs Norway and Germany.

MCC	MNC
242 (Norway)	01,12 (Telenor) /02,05,08 (Telia)/ 06,14 (ICE Net)
262 (Germany)	01,06 (Telekom D), 07,08,11 (O2), 02,04,09,42 (Vodafone)

Table 2.1: Examples of MCC and MNC values in Norway and Germany [3][4].

SIM cards come with built-in security mechanisms, such as Personal Identification Number (PIN) codes. A SIM card has two PIN codes, PIN1 and PIN2. The PIN1 is used to unlock the SIMcard, allowing a subscriber to communicate with other devices. Also, the PIN1 allows user to access personalized data and configuration settings such as Access Point Name (APN) configurations. An APN is a point of entry to the Internet from a mobile device, and is further described in section 2.1.3.

By default, the PIN1 code must be entered whenever the MS is restarted. The PIN2 is used to make changes to more advanced settings on the SIM, such as Fixed Dialling Number (FDN) settings. FDN is a service mode of a GSM phone's SIMcard. Numbers can be added to the FDN list, and when activated, FDN restricts outgoing calls and SMSs to only those numbers listed [17].

Both PIN codes usually consist of four digits. If any of the PIN codes are entered incorrectly three times in a row, an eight digit PIN Unlock Code (PUK) must be

provided. Both the PIN and PUK are provided by the PLMN. However the PIN codes can be changed by the user. Since the PIN1 code is being used more often than the PIN2, we will refer to the PIN1 as *PIN*, and specifically write PIN2 when I refer to PIN2.

Base Transceiver Station (BTS)

GSM has a cellular structure with a Base Transceiver Station (BTS) in each cell. A BTS is the access point for an MS to the GSM network. It consists of an antenna and radio equipment needed for radio communication with the MS over the Um interface. Several MSs can connect to the same BTS as long as they are in the coverage area of the BTS. The BTS's coverage is the size of its cell. The cells are, however, somewhat overlapping, making it possible for an MS to be in the coverage area of several BTSs at the same time. By default, an MS connects to the BTS that has the strongest signal. Upon connecting, an MS must authenticate itself to the BTS. This will be described in depth in section 2.1.5. Once connected, the BTS transmits and receives signals to and from the MS.

Base Station Controller (BSC)

The BSC manages several BTSs and has the job of allocating them with radio resources. It is also responsible for handovers between BTSs [18].

Mobile switching centre (MSC)

The Mobile Switching Centre (MSC) is responsible for routing incoming and outgoing calls and SMSs. Outgoing SMSs are routed to the correct Short Message Service Center (SMSC) while incoming SMSs are routed to the correct MS.

Short Message Service Center (SMSC)

The Short Message Service Center (SMSC) is responsible for storing and forwarding received SMSs. SMSs are stored in the SMSC until the destination MS is available. Figure 2.3 gives a simplified overview of the route of an SMS within GSM.

Home Location Register (HLR)

The Home Location Register (HLR) is a database that permanently stores information about all subscribers belonging to an area served by an MSC. The information includes the telephone number allocated to the subscriber, their current location, the Ki, allowed services and the IMSI.

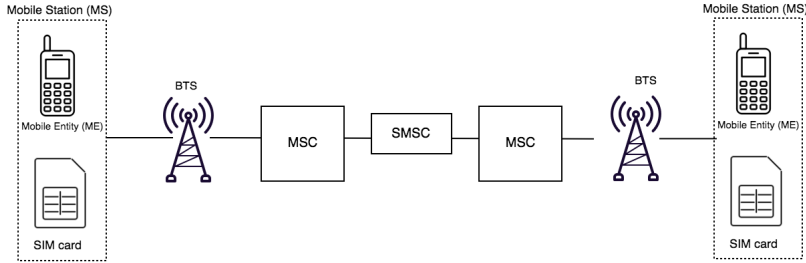


Figure 2.3: A simplified overview of the route of an SMS in GSM.

Visitor Location Register (VLR)

The Visitor Location Register (VLR) is a database that temporarily stores information about all the mobile subscribers that are currently located in the service area of an MSC. This allows for subscribers to maintain a network connection when located outside of the geographical coverage area of their home network.

Authentication Centre (AuC)

The Authentication Centre (AuC) assists in the authentication of an MS. The AuC stores a copy of the K_i s of all subscribers belonging to the HLR, as well as the algorithms used for authentication and encryption (A3, A8, A5). Upon request, the AuC generates a triplet of data (RAND, SRES and K_c). These variables are used for authentication and encryption purposes [18]. This way, the secret key K_i does not need to be transmitted during authentication.

2.1.3 General Packet Radio Service (GPRS)

The General Packet Radio Service (GPRS) is an extension of the initial GSM architecture. It introduces packet-switched functionality and is often referred to as 2.5G. The GPRS is standardized by the European Telecommunications Standards Institute (ETSI), and was the first specification to provide an Internet connection to mobile devices. It is based on the Internet Protocol (IP) and was designed to support interworking with other kinds of networks [19].

Access Point Name (APN)

In GPRS, a mechanism called an Access Point Name (APN) is used to determine how an MS is to communicate via the GSM network. For example, the APN determines what IP addresses are assigned to the MS, the security methods that should be used, and how the GSM data network should connect to the network of the customer [20].

The customer in our case is Biotronik. There exists different types of APNs, and a customer is free to customize their own APN in terms of IP addressing requirements and security requirements. For example, an APN may use public IP addresses where no security is provided, or use private IP addresses and ensure that connections are not publicly accessible.

2.1.4 Radio Frequencies

Mobile network communication takes place over specific allocated radio frequencies. Table 2.1.4 shows the allocated GSM frequency bands that are used by the different PLMNs in Norway. Table 2.1.4 shows the UMTS (3G) frequencies. I have chosen to include these as they are relevant for jamming, which will be discussed later in this chapter.

Operator	Uplink(MHz)	Downlink(MHz)
ICE	880-885	925-930
Telia	885-900	930-945
Telenor	900-915	945-960

Table 2.2: Allocated GSM spectrum in Norway [5].

Operator	Uplink(MHz)	Downlink(MHz)
Telia	1920,3 - 1940,1	2110,3 - 2130,1
Telenor	1940,1 - 1959,9	2130,1 - 2149,9
Ice	1959,9 - 1964,9	2149,9 - 2154,9

Table 2.3: Allocated UMTS (3G) spectrum in Norway [5].

2.1.5 Security in GSM

GSM provides several built-in security features [19]. The three main features are listed below and will be described in detail. In section 2.1.5 we explain why these features are insufficient and how they can be exploited.

- Subscriber identity confidentiality
- Subscriber identity authentication
- Radio-link encryption

Subscriber Identity Confidentiality

As previously mentioned, a subscriber is identified by an IMSI. This is a permanent ID, and should only be sent when it is necessary. To avoid sending the IMSI repeatedly, GSM assigns subscribers with temporary IDs upon authentication, called Temporary Mobile Subscriber Identity (TMSI). The TMSI is transmitted to the MS in an encrypted message, and is used to identify the subscriber in the subsequent communication. The TMSI is assigned for the duration that the subscriber is in the service area of the associated Mobile Switching Centre (MSC). This way, the IMSI is only transmitted when an ME first registers on a GSM network.

Subscriber Identity Authentication

Authentication is the process of verifying the identity of someone (or something). In GSM, authentication is used to verify whether the SIM of a subscriber contains the same Ki that is stored in the AuC. This is performed in a challenge-response scheme. Upon authentication, the network establishes a shared short-term encryption key and assigns a TMSI to the MS.

Authentication of the Mobile Equipment (ME), step-by-step: Challenge-response

1. The AuC generates a triplet of values (RAND, RES, Kc). The RAND is a random 128-bit challenge value. The RES is the output of the A3 algorithm with Ki and RAND as input:

$$RES = A3(RAND, Ki)$$

The Kc is a secret key used for encryption over the Um interface between the MS and the BTS, and is the output of the A8 algorithm with the RAND and Ki as input:

$$Kc = A8(RAND, Ki)$$

The RAND value is then sent to the authenticating MS.

2. The SIM card of the MS computes a RES-value using the same formula as above, using the received RAND value and the Ki and A3 algorithm stored on the SIM card. The RES value is then transmitted back to the network.
3. The AuC compares the RES received from the MS with its own previously computed RES. If equal, the ME has successfully been authenticated and joins the network.

Radio-link encryption

The key, K_c , as mentioned earlier, is used for encrypting user data and signaling over the air. The key is computed with the same formula as above:

$$K_c = A8(RAND, K_i)$$

The K_c value is used as input to the A5 ciphering algorithm, which is implemented in the hardware of the ME, to encipher and decipher the data. The encryption is symmetric; the same key is used for encryption and decryption.

$$\text{Encryption} : C = A5(P, K_c)$$

$$\text{Decryption} : P = A5(C, K_c)$$

Where P is plaintext and C is ciphertext.

There are four variants of the A5 algorithm: A5/0, A5/1, A5/2 and A5/3. A5/0 offers no encryption at all, while A5/1 and A5/2 are stream-ciphers. A5/3 is not widely deployed.

2.2 Security vulnerabilities and attacks on GSM

The security mechanisms of GSM have proven to be insufficient and contain several vulnerabilities:

- **Unilateral authentication:** As mentioned, the MSs are authenticated upon connection to the network. However, the network is not authenticated to the MS. This opens up for IMSI-catchers, as described below.
- **Limited encryption:** GSM does not offer end-to-end encryption, and only encrypts the Um interface between the MS and the BTS.
- **Security by obscurity:** The cryptographic algorithms that are used in GSM (A5/1, A5/2) were initially kept confidential, thus breaking Kerckhoffs' principle¹. They have both been reverse-engineered and have proved to offer insufficient security levels.

These weaknesses make GSM vulnerable to attacks. The following sections describe possible attacks on GSM.

¹Kerckhoffs' principle: A cryptosystem should be secure even if everything about the system, except the key, is public knowledge [21]

2.2.1 Illegitimate BTS

An illegitimate BTS is a BTS that is not part of the infrastructure of a licensed PLMN. Due to the lack of authentication in GSM, an MS will not be able to distinguish an illegitimate BTS from a legitimate one. An illegitimate BTS will typically masquerade as a real BTS by using the same identifiers (MCC, MNC) and frequency range as a real BTS. As a consequence, an MS will connect to an illegitimate BTS if it offers the strongest signal in the area without notifying the subscriber. This scenario is illustrated in figure 2.4.

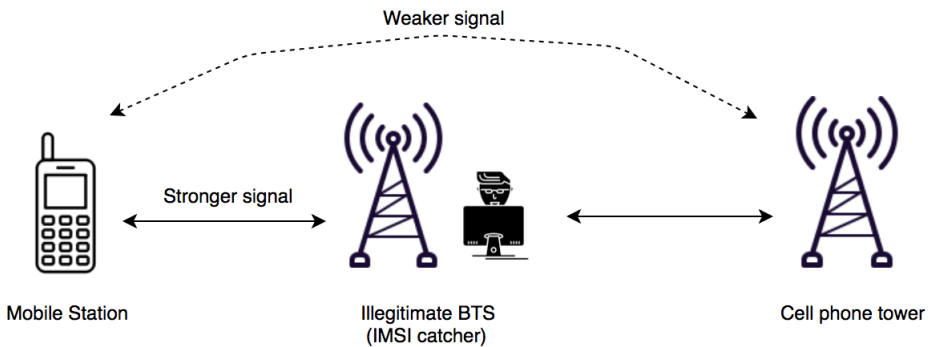


Figure 2.4: An overview of how an illegitimate BTS works.

An adversary can perform different types of attacks with an illegitimate BTS:

1. **IMSI-catching:** The illegitimate BTS is used to capture the IMSIs of MSs and hence detect the subscribers that are present at a given location and time [22]. Upon an MS requesting to access the illegitimate BTS, it requests the MS for its IMSI with a *Request IMSI* message, in which the MS responds with its IMSI. The illegitimate BTS then rejects the connection.
2. **Denial of Service (DoS) attack:** By tricking an MS into connecting to the illegitimate BTS, one is essentially performing a DoS attack. If the illegitimate BTS does not offer an Internet connection, the MS has no way of contacting any other devices outside of the network.

3. **Eavesdropping with cipher suppression:** The owner of an illegitimate BTS can eavesdrop on the communication links of the devices that connect to it. The illegitimate BTS, like a real BTS, is in charge of the type of encryption to use in the Um interface. Consequently, it can instruct the connected devices to not use encryption at all (A5/0). This is called cipher suppression. A limitation of this attack is that only outbound SMSs and data traffic can be intercepted, as the MS is disconnected from the real network. However, the outgoing traffic is of most interest in this thesis.
4. **Man-in-the-Middle (MITM) attack:** The illegitimate BTS sits between the MS and the legitimate BTS, and relays (and possibly alters) information to and from the MS. This way, communication seems to work as normal, and the MS has no way of knowing that its communication is being eavesdropped. The IMSI-catcher must have a SIM card for doing this, and acts as a BTS to the MS and as a MS to the legitimate network.

2.2.2 Jamming and downgrade to GSM

Jamming means to block certain frequencies. With a jamming device, one can block certain frequencies or all radio communication. By blocking the frequencies of UMTS and LTE, which are listed in table 2.1.4, one can force MSs in the area to downgrade to using GSM rather than a more secure protocol, like UMTS. In this scenario, the MS will communicate over a less secure network.

Jamming is not a vulnerability in GSM, but rather an attack to force MSs to communicate over GSM rather than using more secure communication protocols.

2.3 Security Model

This section presents fundamental aspects of security. At the core of information security is confidentiality, integrity and availability. This is known as the CIA triad, and forms the basis for our analysis. In addition, there are common extensions to the triad, making it a more robust model for today's constantly changing IT environment [23]. Our security model also includes safety and privacy, as the security of the pacemaker ecosystem is closely related to, and may affect these aspects with regards to a patient.

Confidentiality: Information should only be disclosed to those being authorized to access it. For example, only healthcare personnel who are logged into the Data Server should be able to access the patient data.

Integrity: Information should arrive in the same (correct and valid) state as it was sent; it should not be created, altered or destructed by unauthorized third-parties.

For example, the same patient data that was sent from a pacemaker should be received by the Data Server.

Availability: The information should be available to authorized personnel in a timely manner. For example, healthcare personnel should be informed if something is wrong with a patient or his/her pacemaker.

Authentication: Provides proof of identity of the communicating parties, as well as ensuring the origin of data. As an example, the HMU should be certain that it is in fact communicating with the correct Data Server.

Authorization: The function of defining access privileges to resources to users.

Identification: The property of recognizing and distinguishing individual users. As an example, the Data Server should be able to distinguish the different HMU devices from one another.

Accountability/Non-repudiation: Every undertaken activity should be linked to a person or an automated process.

Privacy: In the context of information security, privacy is the right of individuals to protect themselves and their information from unauthorized access, providing confidentiality [23]. Securing personal information is fundamental to achieving privacy, and privacy hence relates to all the elements described above.

Safety: The condition of being protected from danger, risk, or injury [24]. As an example, the HMU should ensure and improve patient safety, and not in any way cause harm to a patient.

Chapter 3

Related Work

In this chapter we present related work of relevance to our research, both concerning security research of the pacemaker ecosystem and means of wireless communication.

3.1 The Pacemaker Ecosystem

Marin et al. [12] demonstrated how to fully reverse-engineer the proprietary wireless communication protocol between the programmer and a widely used ICD. They confirm that their results apply to (at least) 10 types on ICDs that are currently on the market. Their experiments were performed on the long-range RF channel (2-5 meters). Using inexpensive COTS equipment, and without physical access to devices or patients, they were able to reveal several vulnerabilities.

Their analysis revealed several vulnerabilities in an unidentified proprietary protocol. The authors demonstrated how they could exploit the protocol. Due to the lack of data encryption, they were able to gain access to private sensitive data by eavesdropping on the communication. Also, they demonstrated how an adversary could take advantage of ICDs being in “standby” mode to carry out a DoS attack. Lastly, they showed how spoofing and replay attacks were possible.

Muddy Waters [10] disclosed significant vulnerabilities in the pacemaker ecosystem from St. Jude Medical (now Abbot) based on an analysis performed by MedSec. The findings include vulnerabilities with the merlin@home HMU device as well as the communication protocols between merlin@home and implantable devices. Due to the lack of proper authentication in the protocol, an attacker can impersonate a merlin@Home unit, and communicate with cardiac devices. Using a compromised device, they demonstrated different types of attacks, including “crash” attacks and battery drain attacks. With the crash attack, they were able to disable cardiac devices, while the battery drain attack could lead to premature battery depletion.

They reported that all patient data was being encrypted. However, by gaining root

access to the HMU, they were able to find cryptographic keys in plaintext, which they could potentially use to obtain user data. They also found hardcoded IP addresses of the data server and static credentials in cleartext. The authors claim that using these credentials, they could gain access to the Merlin network, potentially allowing them to perform a large scale attack. These units are readily available on eBay.

Whitescope [9] performed a security evaluation the ICD ecosystem of four unnamed vendors. Using equipment obtained on public auction sites, they performed several security tests on different parts of the ecosystem. Their results suggest similarities in the architectural framework of different vendors, and thereby also similar security risks. These risks may lead to negative consequences to patients if exploited. As a consequence of their findings, the authors present a list of questions that could aid vendors in evaluating their security controls against the identified risks.

Kristiansen et al. [25] examined parts of the pacemaker ecosystem based on equipment from Biotronik. In particular, they investigated the programmer with regards to common principles of information security. Using a disk image, they successfully emulated a programmer in a virtual machine.

When analyzing the programmer, they found a number of vulnerabilities. Using only Commercial off-the-shelf (COTS) equipment, they were able to disclose several major vulnerabilities, including the lack of authentication, lack of sensitive data encryption, lack of patching, and hard-coded pass-phrases for data extraction. As part of their research, they also performed qualitative interviews, retrieving information about the routines on how programmers were used in a Norwegian hospital.

Their findings suggest that the pacemaker ecosystem is not in accordance with best-practice in the security industry, and the authors consequently suggested several countermeasures.

3.2 Wireless communication

Mruz [26] and **Retterstøl** [22] describe IMSI-catchers (illegitimate BTS) and their inner workings in some detail. They also present security flaws in the GSM protocol and how an IMSI catcher can exploit these.

Mruz's focus was on detecting IMSI-catchers by scanning nearby BTSs for normal activity and deviations, while Retterstøl performed an analysis of the claims made by the newspaper Aftenposten with regards to IMSI-catchers in Oslo [27].

Hadžialić et al. [28] pinpoint one of the major security flaws of GSM, namely the lack of mutual authentication. Throughout the paper they explain the nature of an IMSI-catcher and a practical implementation of one. The paper also describes

possible attacks which can be performed with an IMSI-catcher, e.g., malicious SMS, and man-in-the-middle attacks.

In 2015, **Miller et al.** [29] demonstrated how they could remotely attack an unaltered vehicle, a 2014 Jeep Cherokee. The attack resulted in physical control of some aspects of the vehicle,. They used the cellular connection of the vehicle as an entry point, as the Uconnect system in the vehicle had the ability to communicate over Sprint’s cellular network, in which the device is assigned an IP address that is not accessible from the public Internet. By setting up a femtocell and forcing the vehicle to connect to their (Sprint) network, they demonstrated their ability to communicate with any other vehicle on the network. As such, their attack took advantage of missing client segregation on the Sprint network.

A similar scenario was demonstrated by **Munro** [30] in 2017, only this time the vehicles were connected to a private APN. Munro was able to confirm that the APN was accessible either by gaining access to the credentials of a Telematics Control Units (TCUs) from a vehicle, or by accessing the network through the TCU, which was considered a trusted device by the private APN. With access to the network, they observed that a large number of devices were connected to it, including vehicles from other brands, again showing the lack of device and brand segregation. Although they stopped at this point, their results suggest that they could potentially remotely compromise every vehicle connected to the network.

Chapter 4

Research Methodology

This chapter presents the research methodology, Design Science, that is the basis for our research. In addition, several methods that were used as part of the Design Science research are described in this chapter. We also discuss limitations, and legal and ethical considerations of our research.

4.1 Design Science

Design Science is a methodology that can be used when conducting research within Information System (IS). An IS can be defined as *"the study of complementary networks of hardware and software that people and organizations use to collect, filter, process, create, and distribute data"* [31]. Given this definition, one can easily argue that the pacemaker ecosystem is an IS. Design Science provides the researcher with guidelines, methods and terminology to be used when planning, structuring and performing the research.

Design Science is a problem-solving paradigm that seeks to improve a context by creating new and innovative artifacts [32]. An artifact can be almost any man-made device that has been designed for a specific context, such as a method, technique or an algorithm. The context of the artifact can be defined as *"anything that interacts with the artifact or that has influence on it"* [1].

In our project, we have defined the artifact to be the communication protocol between the HMU and the Data Server, including the communication components of the HMU. The context is the pacemaker ecosystem. This means that rather than creating a new artifact, We will be looking to improve the design of an existing artifact. From a cybersecurity point of view, the problem this artifact is trying to solve is to transmit the patient data it receives from a pacemaker to the Data Server in a secure manner.

4.1.1 A framework for Design Science

Wieringa [1] introduces a framework for Design Science. The framework provides an overview of the main activities of Design Science as well as a different contexts, making us aware of other factors that must be taken into account when conducting Design Science research. Figure 4.1 presents the framework with adaptations to fit our project.

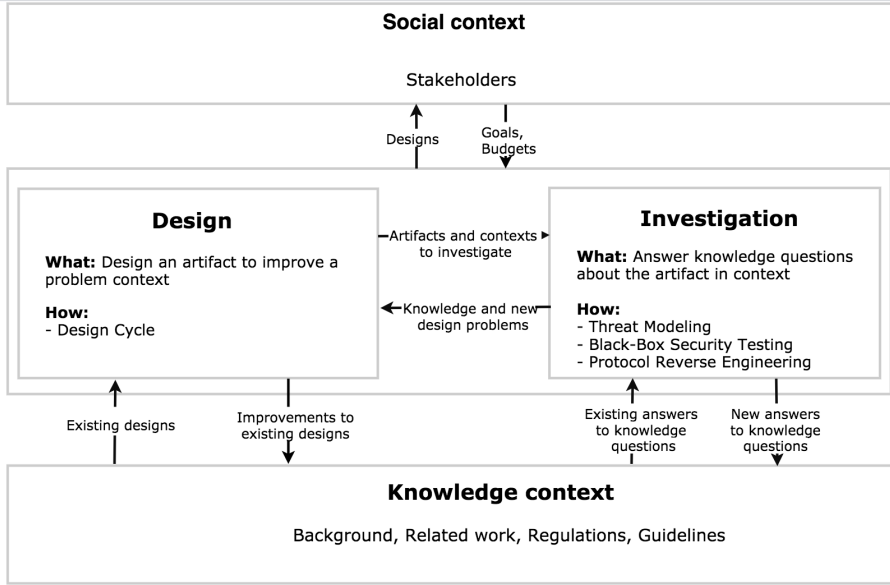


Figure 4.1: Our implementation of Wieringa’s [1] framework for Design Science.

According to Wieringa, **Design** and **investigation** are the two main activities of Design Science. These activities will be discussed in detail in section 4.1.2 and 4.1.3.

The social context consists of the human stakeholders of the project, these being parties who may affect or be affected by the project, some being aware of this, while others not. In this thesis, the stakeholders are regulators, doctors, patients and vendors. As we will not be communicating with any of the stakeholders, their goals are hypothetical and will not be further discussed in this thesis.

The knowledge context contains all a priori knowledge of relevance for the thesis, and is the place to start to get an overview of the existing knowledge. It is also used as a basis for finding relevant background information and related literature. In addition, as can be seen in figure 4.1, it is the place to go when looking for answers to knowledge questions. By knowing what knowledge already exists, one can also

determine what is missing and hence set the objective of the research. Consequently, there is a better chance of contributing to the knowledge base, which is a goal in Design Science research [32].

We are investigating an artifact in a context where most of the details are proprietary. As far as we know, no studies have been publicly disclosed on this specific artifact in context. Therefore, anything we are able to uncover can be considered new knowledge and a contribution to the knowledge base.

4.1.2 The Design Activity

The main purpose of the design activity is to design an artifact to improve a problem context, thereby solving design problems. We will not be designing any new artifact as part of this thesis. However, artifact design is not limited to creating new artifacts; already implemented artifacts may also be subject to (new) design problems. For solving design problems, Wieringa [1] introduces the Design Cycle.

Design Cycle

The Design Cycle is a tool for treating design problems, and consists of three tasks:

- Implementation evaluation/problem investigation
- Treatment design
- Treatment validation.

Figure 4.2 presents our implementation of the cycle. We have changed the order of Treatment design and treatment validation compared to the original cycle. Have chosen Implementation evaluation and not problem investigation, as we consider our work a continuation of someone else's work.

Our main focus is to review an already implemented artifact. As such, the most relevant tasks of the Design Cycle is treatment validation and implementation evaluation to study the implementation. Then, depending on the results, the treatment design task may be relevant for suggesting design improvements.

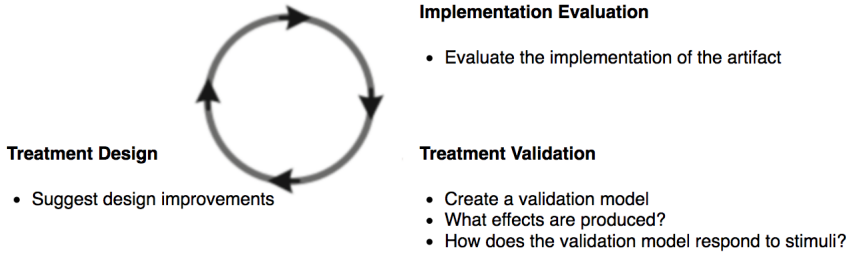


Figure 4.2: Our implementation of the Design Cycle.

When performing an *implementation evaluation*, one seeks to evaluate the interaction between an implemented artifact and its real-world problem context [book]. Determine whether the interaction is as intended. Since the artifact in questions has already been implemented, and since we have access to part of its real-world context (the mobile network and the Data Server), evaluating this implementation seems like a natural place to start. The real-world context is missing a functioning pacemaker. However, we are hoping that performing an implementation evaluation can still provide useful input regarding how the system works and its (lack of) security mechanisms.

A research method frequently used in implementation evaluation research is the observational case study. In an observational case study, the researcher observes a single artifact in a real-world context without interfering, except for the interference that may come from measurement, and measures phenomena in the artifact [book]. However, if interference is necessary, the single-case experimental method will be used, which is described shortly.

The artifacts we are looking at are no longer meant to be used in its real-world context and may therefore lack some functionality. In addition, legal and ethical considerations prevent us from experimenting too much with the real-life context. Consequently, we will also perform *treatment validation*. The goal of treatment validation is to predict how an artifact will interact with its context "*without actually observing an implemented artifact in a real-world context*" [1]. This way, we will (hopefully) be able to observe a more realistic behavior of the interaction between the artifact and the (model of the) problem context.

To perform treatment validation, a proper model must be constructed. Wieringa introduces the validation model, which is a model of the artifact and the context. Since we will be validating an already implemented artifact, we will not make a model of the artifact. Consequently, our validation model will consist of a model of the context and the artifact itself. The model of the context will initially consist of an Illegitimate BTS, which is described in section 2.2.1. The idea is to perform experiments on the model, in which the results may provide additional insight into the real-world properties of the artifact. To make generalizations by analogy from observations of the validation model, there must be a certain similarity between the model and its target.

Different research methods can be used when studying a validation model. The single-case experimental method seems to be suitable as it *"allows us to expose the model to controlled stimuli"* [1] and observe the effects. This method is usually applied in a laboratory.

Treatment design is the task of designing an artifact that improves a problem context. This is relevant for our research with regards to RO3, as we will suggest improvements to the artifact based on the vulnerabilities that we might find while performing the previously discussed tasks, if any (RO2).

4.1.3 Investigation

In the investigation activity, the goal is to answer knowledge questions about the artifact in context. The questions may or may not have existing answers. If they do, the answer can be found in the knowledge context. However, if a knowledge question does not have an answer, conducting research might be necessary. The research questions we have defined for this thesis are examples of previously unanswered knowledge questions. For answering these, as well as other questions that may arise while conducting the experiments, we will use validation model together with the methods described in sections 4.2, 4.3.1 and 4.3.2.

4.1.4 Correlation Between Design and Investigation

As can be observed in figure 4.1, the design and investigation activities interrelate in an iterative manner and provide each other with input. For example, a design activity can lead to new knowledge questions about the artifact, which again may lead to new design problems. The correlation in our research is illustrated in figure 4.3.

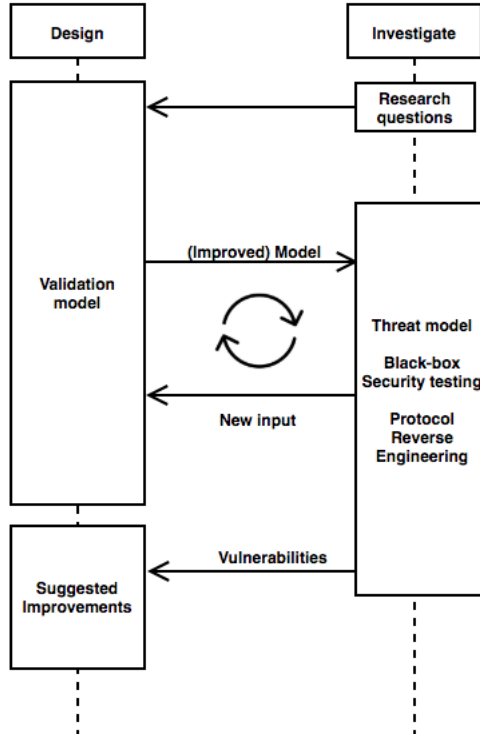


Figure 4.3: Correlation between the design and investigation activities of this thesis. The cycle illustrates how the validation model is used to investigate, and how this investigation is providing input that can further improve the model.

4.2 Threat modeling

Threat modeling is a method used when assessing the attack surface of a system. The attack surface outlines the points where an attacker could gain access to the system as well as where he could get access to data [33]. The assessment is performed by identifying the assets, threat actors, and possible threat scenarios for the system components and their interfaces.

4.3 Trial-and-error

Trial-and-error is a commonly used method within problem-solving. Hence, trial-and-error is relevant for the Design Science methodology. The method involves experimenting with and testing possible solutions to a problem and eliminating the unsuccessful attempts, until a solution which achieves the desired results is found

or until one stops trying. When using trial-and-error, one does not seek to find the best or all solutions to a problem, but simply to find one solution that solves the problem. As such, it is a non-optimal method. Also, the trial-and-error method does not answer why the solution works. In addition, the method is problem-specific in which one does not try to generalize the solution by any means, but simply solve the specific problem at hand. The advantage of trial-and-error is that it can be used without much prior knowledge.

Trial-and-error is used extensively throughout our research, and is the basis for the methods described below.

4.3.1 Black-box Security Testing

Black-box testing is a testing method where the internal structure of the system in question is not known to the tester. This means that the testing is performed from a user point of view; the tester has no additional insight to the system beside what is publicly known.

When performing black-box security testing, the tester is essentially imitating the actions of an attacker, attempting to attack the application from the outside. This is a useful approach for testing a system as the vulnerabilities discovered are realistic. A downside of this approach is that one cannot be certain to have uncovered all the vulnerabilities in the system.

Black-box security testing seems to be a suitable approach when looking for vulnerabilities in the artifacts.

4.3.2 Protocol Reverse Engineering

Reverse engineering is the process of dissecting a system to identify its components and how they interrelate and communicate. It does not involve changing the system, but rather simply examining and trying to understand its design [34].

There are many branches of reverse engineering, one of them is protocol reverse engineering. Marin [35] defines this as *"finding both the message format and how messages are exchanged between two devices without knowing its specifications"*. A proprietary protocol is often specific to a certain type of device from a specific vendor, and reverse engineering it is therefore difficult.

Marin [35] defines two ways of reverse-engineering a communication protocol: Black-box reverse engineering and firmware reverse engineering. The black-box approach is of most relevance to our research.

Black-box reverse engineering

The same principles apply here as in section 4.3.1, meaning that we are looking at a protocol without knowing its details. Marin [35] divides the process of reverse engineering a protocol into three steps:

1. Get public information from the Internet
2. Find the wireless communication parameters
3. Capture and analyze messages sent by the devices

Obtaining public information from the Internet was part of the process of writing the background and related literature chapters, but has also been a continuous process throughout the research. As for finding wireless communication parameters, knowing that the protocol is running on top of GSM facilitates this work, as GSM uses standard radio frequencies. Capturing and analyzing messages is a major part of our research, and will be described in the subsequent chapters.

4.4 Limitations

A limitation of this research is the lack of a functioning pacemaker. As described in section 1.1, the pacemaker constitutes a major part of the pacemaker ecosystem and hence the problem context of this thesis. The only pacemaker available to us had its battery depleted in 2016. This can be seen in figure 4.4. Changing the battery is impossible. Due to this, no new patient data has been received by any of the HMUs tested in our research. However, the HMUs are still transmitting data to the data server. Therefore, it is still possible to conduct the research even with an essential part of the ecosystem missing. However, a functioning pacemaker could have made the research more extensive.

4.5 Legal and Ethical Considerations

While performing our research, it is important to stay within the legal and ethical boundaries at all times. Our goal is not to attack the system, but rather to discover vulnerabilities that could pose threats and demonstrate how these vulnerabilities can be exploited.

Most of our research is performed using the validation model. However, we also perform experiments in the real-life problem context. We pay extra precautions in these experiments, making sure that we do not affect the operation of the pacemaker ecosystem and/or the safety and privacy of real patients.

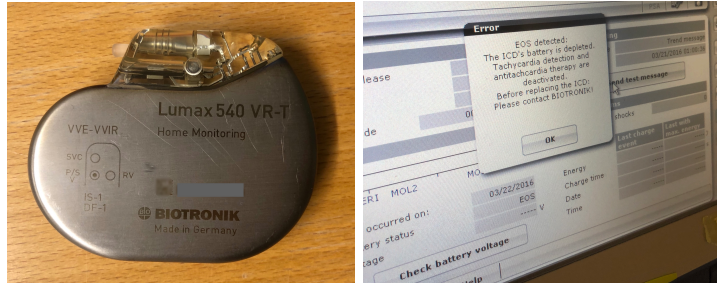


Figure 4.4: Lumax 540 VR-T, an ICD from Biotronik, whose battery has reached End of Service (EOS), as we observed when interrogating it with a programmer (right).

As we are investigating artifacts that were once owned and used by pacemaker patients, there is a possibility of finding personal information during the experiments. All personal data will be fully anonymised, and deleted after completing the thesis. As part of our research we will configure an illegitimate BTS. There is a chance that other third party MSs might try to connect to the network, and sensitive information such as IMSI numbers may be collected. If this happens, the sensitive information will be deleted immediately.

Due to the possibility of finding personal information, the project has been reported to the Norwegian Center for Research Data (NSD). In addition, our results will be disclosed according to a Coordinated Vulnerability Disclosure [36]. This way, Biotronik has the chance to mitigate any disclosed vulnerabilities before the thesis is publicly disclosed.

All experiments will be conducted in room F260, El-bygget at Norwegian University of Science and Technology (NTNU). NTNU has approval from the mobile operators in Norway regarding frequency usage on their premises. Therefore, it is legal to perform our experiments there.

Chapter 5

Threat Model

In this chapter, we assess the attack surface of a subsystem of the pacemaker ecosystem: The HMU, the Data Server and the communication protocols between them. We identify assets, potential adversaries to the system, what their incentives might be and attack vectors based on potential vulnerabilities in the system. The focus will mainly be on wireless attacks.

The blue rectangle in figure 5.1 presents the parts of the pacemaker ecosystem that will be in focus throughout this thesis. The purpose of this subsystem is to transmit data received from a pacemaker to a Data Server where health care personnel can access the data. The data is sent on a regular basis, as well as if abnormal data triggers an alert. In the figure, the black arrows indicate communication channels and the red lines indicate trust boundaries.

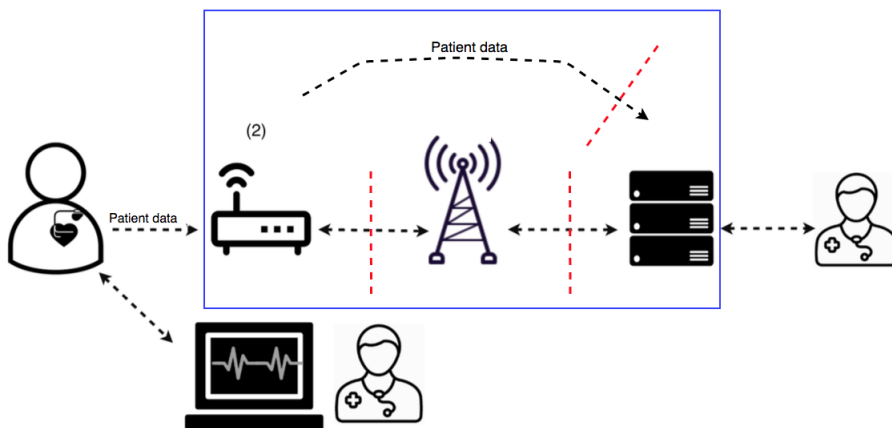


Figure 5.1: Overview of the subsystem of the pacemaker ecosystem that is the focus of this thesis.

Thus subsystem of the pacemaker ecosystem contains various *assets* (something of value) that could be of interest to an adversary:

- Physical HMU devices
- Patient data
- Patient safety
- Connection to a mobile network (SIM card)
- Reputation of the vendor

An adversary may be interested in one or more of these assets.

The adversary could be a third-party without any legitimate access to the system. For example, he/she might be someone hired by a competing manufacturer that wants to tarnish the reputation of Biotronik. It could also be someone who wants to benefit financially by selling personal information online, or using the SIM card for making free international calls. The adversary could also be someone who wants to cause harm to or blackmail a very specific person.

On the other side, the adversary could also be an insider of the pacemaker ecosystem such as a patient who may have access to a functioning device or medical personnel who has access to the Data Server.

Attacks can be divided into passive and active attacks. In a *passive attack*, the adversary does not seek to make any changes to the system but to gain information about it and from it. For example, an intruder may eavesdrop on a communication channel. Passive attacks are a threat to data confidentiality and patient privacy. Information from a passive attack can later be used as part on an active attack.

In an *active attack*, the adversary aims to break into the system to make changes to its operation. It can involve making use of information collected in a passive attack, or be performed as an independent attack. Depending on the nature of the attack, an active attack may pose a threat to all the key principles of the security model described in 2.3, as well as the safety of a patient.

Before conducting the research, our knowledge about the HMU, the DS, and the communication protocols between them is limited. However, what we do know is that all communication is taking place over a mobile network. The mobile network standard, GSM, suffers from several vulnerabilities which may be exploited. For example, one can configure an illegitimate BTS and trick devices into connecting to it. One can also force HMUs supporting newer and more secure mobile communication standards to communicate over GSM by jamming all frequencies but the GSM frequencies. Upon connection to an illegitimate BTS, one can further impose cipher suppression and eavesdrop on the unencrypted communication. Also, an adversary

can use the illegitimate BTS to deny service to the connected device altogether. These types of attack do not require physical access to the HMU; it has been suggested that an illegitimate BTS can have up to a 2 km range [37].

5.1 Attack Scenarios

This section describes examples of attack scenarios based on the the discussion above. For these examples, we assume that the adversary is a third-party.

5.1.1 Gather Patient Data

Why: An adversary wants to sell medical records online.

How: It is publicly disclosed that patient data is being sent from the HMU to the data server over a mobile network. Hence, there are (at least) two ways to possibly obtain information without requiring physical access to any patient or their device.

1. Obtain unauthorized access to the Data Server. The Data Server is connected to the Internet and might be remotely accessible.
2. Eavesdrop on the communication between the HMU and the Data Server using an illegitimate BTS. The data might be encoded or encrypted, in which the adversary manages to decode/decrypt the data.

5.1.2 Harm a patient

Why: An adversary wants to harm a specific person who has a pacemaker.

How: As the communication between the pacemaker and the HMU is outside the scope of this thesis, the adversary has to be creative when thinking of ways to harm a patient. A possible solution is to prevent (correct) data from reaching the data server. This way, either prevent continuous patient monitoring, or forge data to trick the health care personnel to believe that the status of the patient is different than what is true. This can be performed using different types of attacks:

1. A Denial of Service (DoS) attack
2. Replay attack
3. Data alteration attack

DoS: A DoS attack will prevent data transmitted by the HMU from reaching the data server. Examples of attacks could be:

- Deny the HMU to establish a network connection by tricking it into connecting to an illegitimate BTS.
- Deny the HMU to establish a network connection by removing its SIM card
- Make the server unavailable by sending it a large number of requests
- Send malicious data to the HMU, for example a malicious SMS
- Send malicious data to the Data Server

Replay attack

An adversary could act as a man-in-the-middle between the HMU and the Data Server, and deny any new packets from being transmitted, while replaying previously sent data expressing that the status of the pacemaker and the patient are normal at a time when this is not the case, thereby avoiding alerts being raised at the server side.

Data alteration

An adversary could act as a man-in-the-middle between the HMU and the Data Server, and make changes to the data that is being sent before relaying the information to the Data Server. The adversary could either make the healthcare personnel believe that everything is fine, or trigger alerts stating the the patient is at great risk when in fact there is nothing wrong.

5.1.3 Misuse SIM card services

Why: An adversary wants to make make calls, send SMSs and access the Internet without having to pay for it.

How: As every HMU contains a SIM card, an adversary could either order an HMU online and hope that the SIM card still has a valid subscription, or he/she could steal an HMU from a patient. The latter requires physical access to a patient's home.

5.2 Difficulty of attacks

None of the potential attacks discussed require exceptional skills, experience, or expensive equipment. Most of the attacks can also be performed without access to a patient and/or their HMU.

The most advanced equipment needed is the illegitimate BTS, which can be configured using Commercial off-the-shelf (COTS) hardware and software components. The

cost for the hardware is around 1500 USD. The software is open source and freely available online. In addition, old/used HMUs can be purchased online for about 20 USD.

5.3 Potential Attack Vectors

Table 5.1 summarizes the potential attack vectors from the attack scenarios in section 5.1.

Attack Vector	Threat to
Remote server access	Authenticity
Interception on the communication channel(s)	Confidentiality
Decoding/Decryption of data	Confidentiality
DoS of HMU	Availability
DoS of Data Server	Availability
Replay of data	Integrity, Availability, Safety
Data Alteration	Integrity, Safety
SIM card fraud	Accountability

Table 5.1: Potential attack vectors.

Chapter 6

Experimental Setup

This chapter introduces the software and hardware components that were used for setting up an Illegitimate BTS that is part of the validation model discussed in section 4.1.2. In addition, the steps for configuring the Illegitimate BTS. In addition, an overview of the different tools we used as part of the experiments are included at the end of the chapter.

Figure 6.1 presents the components of the pacemaker ecosystem that were included in the experiments of this thesis.

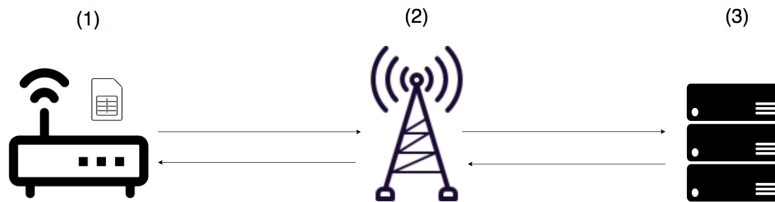


Figure 6.1: Relevant components of the pacemaker ecosystem for our research: The Home Monitoring Unit (1), Mobile Network (2), and Data Server(3).

6.1 Home Monitoring Unit

In this thesis we are investigating four different models of the wireless HMUs from the German vendor Biotronik.

- CardioMessenger LLT
- CardioMessenger II-LLT
- CardioMessenger II-S
- CardioMessenger Smart 3G

Pictures of the different models are found in section 1.3. All the HMUs were bought on eBay, and have previously been owned (and used) by pacemaker patients in the and the US. Consequently, we have only been investigating devices that were no longer in use by patients, and the research has therefore not affected any real patients.

We did not attempt to create a prototype or model of the HMUs as part of the research. However, we did use mobile phones and additional SIM cards to understand the behavior of the HMUs better as well as to test certain functionalities in ways not possible with the HMUs.

Mobile Phones

A phone, unlike an HMU, has a Graphical User Interface (GUI). Through the GUI, one can interact with the inserted SIM card and observe how the SIM card behaves and responds to different stimuli. One can also access the data stored on the SIM and make changes to certain settings. Below is a list of all the phones we used as part of the experiments, as well as an explanation of what they were used for. The phones can be seen in figure 6.2.

- **LG-A100:** Used for accessing personal data on the SIM cards
- **Nokia 6070:** This replaced the LG-A100 when it stopped working.
- **Iphone 4s:** This phone fits a Micro SIM card. We used it for testing the Internet access of the SIM cards, as it supports Web browsing. The phone also has the option of only using GPRS for accessing the Internet, and one can manually configure the Access Point Name (APN) settings.
- **Avvio 750:** This phone fits full-size SIM card and is able to everything the other phones can do. In addition, allows us to access the Fixed Dialling Number (FDN) settings of the SIM card, as well as accessing a terminal.



Figure 6.2: The phones used in the experiments. From left: LG-A100, Nokia 6070, Iphone 4s, Avvio 750.

SIM cards

Using additional SIM cards allowed us to both replace the SIM cards of the HMUs as well as interacting with HMUs. For the experiments where we needed empty SIM cards to play around with, we used SIM cards from Sysmocom. These are dummy SIM cards that are not tied to any subscriber, and is used only for research purposes. We also used a SIM card with a valid subscription.

6.2 Mobile Network

A substantial part of the experiments that were performed as part of this thesis used a model of a mobile network; an illegitimate BTS. An illegitimate BTS is not part of the a licensed network, but serves as a simulated environment for devices that are connected to it. The illegitimate BTS that was configured for this thesis was only accessible for the devices that were part of the experiments. Consequently, we avoid having uninformed third-party equipment who happens to be inside the range of the illegitimate BTS connect to our network.

Setting up an illegitimate BTS requires a computer and COTS equipment, such as standard hardware equipment and open-source software. The following list presents the equipment we used in this thesis, which are also shown in figure 6.3. Following, we present each component in detail.

- MacBook Pro (Retina, 13-inch, Mid 2014), OSX Sierra 10.12 with VirtualBox v.6.0.2 installed
- Virtual Machine (VM) running Ubuntu 14.04 Server with OpenBTS, Asterisk, SmQueue and SipAuthServe installed.
- USRP B210 with VERT 900 antennas connected to the Ubuntu 14.04 via USB



Figure 6.3: Overview of the experimental setup for the illegitimate BTS. 1. Jammer, 2. Vert900 antennas, 3. Macbook Pro running VirtualBox. VM inside VirtualBox running OpenBTS, 4. USRP B210

OpenBTS

OpenBTS is a Linux-based open source software developed by Range Networks, implementing most of the GSM stack above the radio modem. The code is freely accessible on GitHub¹, and a complete guide for installing it is described in the book *"Getting Started with OpenBTS"*.² Together with a USRP, OpenBTS has the capabilities to simulate a GSM mobile network. Hence, it can be used to set up an illegitimate BTS.

To fully support calls and SMSs, OpenBTS requires some additional applications:

- **SIP Message Queue (SMQueue):** Stores and forwards text messages, and is needed for delivering SMSs between the mobile stations that are connected to the illegitimate BTS. SMQueue replaces the the Short Message Service Center (SMSC) in the real GSM network.

¹<https://github.com/RangeNetworks/dev>

²http://openbts.org/site/wp-content/uploads/ebook/Getting_Started_with_OpenBTS_Range_Networks.pdf

- **SIP Authorization Server (SipAuthServe):** Manages the subscriber database, and replaces the GSM HLR.
- **Asterisk:** Is responsible for establishing calls between mobile stations that are connected to the illegitimate BTS.

Software Defined Radio (SDR) and Universal Software Radio Peripheral(USRP)

A Software Defined Radio (SDR) is a communication system where software is used to perform core radio functions that have traditionally been implemented in hardware, such as tuning, synchronization and signal processing [38]. Implementing these functions in software makes the components programmable and hence easy to adjust. An SDR has many fields of application, and can for example be used for listening to and transmitting radio signals on different frequencies, depending on its frequency range.

A Universal Software Radio Peripheral (USRP) serves as a hardware platform for an SDR, and can be bought online.³

VirtualBox

VirtualBox⁴ is an open-source program used for creating and managing Virtual Machine (VM), hence making it possible for multiple Operating System (OS) to run on a single physical computer in parallel.

Jammer

Jamming means to disrupt the communication on a specific frequency and is accomplished by transmitting noise on the same frequency. In our experiment, we use a jammer to block 3G frequencies and hence force all HMUs to communicate over GSM. The range of the jammer we used is very limited and did not affect any third-party devices.

6.2.1 Setting up an Illegitimate BTS

Configuring and Running OpenBTS

Out-of-the-box, OpenBTS comes with default settings. For example, OpenBTS is by default configured as a test network with values MCC=001 and MNC=01, and GPRS disabled. Some of the settings must be changed to customize the network for the intended purpose.

³<http://www.ettus.com>

⁴<https://www.virtualbox.org>

OpenBTS is equipped with a Command Line Interface (CLI) that can be used to interact with the program in real-time. Through the CLI, one can inspect and modify the OpenBTS configuration parameters, as well as monitoring the system status. To access the CLI, OpenBTS and the additional applications must be running:

```
$ sudo start openbts
$ sudo start sipauthserve
$ sudo start smqueue
$ sudo start asterisk
```

To access the CLI:

```
$ OpenBTSCLI
```

As mentioned, the HMUs previously belonged to patients in the UK and the US. However, as will be discovered in the results, the SIM cards belonging to the HMUs are German and from T-mobile/Telekom D. Based on this, we decided to use MCC and MNC value to match the SIM card operator. The values do not really make a difference in our experiments, as we are not deliberately trying to spoof the operator, and do not change the fact that the IllegitimateBTS must still offer the strongest signal in the area for the HMUs to connect to it.

To change the MCC and MNC values, type the following commands into the CLI:

```
OpenBTS> config GSM.Identity.MCC 262
OpenBTS> config GSM.Identity.MNC 01
```

In addition, we set the name of the network to be *Telekom*:

```
OpenBTS> config GSM.Identity.Shortname Telekom
```

The current settings of OpenBTS can be inspected with the `config` command. Figure 6.4 presents the settings upon setting the MCC, MNC and shortname values.

As most of the HMUs are capable of communicating over the internet, we enabled GPRS on the illegitimate BTS. This is achieved with a simple `config` command in the CLI:

```
OpenBTS> config GPRS.enable 1
```

It is important to ensure that Dynamic Host Configuration Protocol (DHCP) is enabled on the environment that OpenBTS is running in. DHCP is used to assign

```

OpenBTS> config GSM.Identity
GSM.Identity.BSIC.BCC 2      [default]
GSM.Identity.BSIC.NCC 0      [default]
GSM.Identity.CI 10           [default]
GSM.Identity.LAC 1000        [default]
GSM.Identity.MCC 262
GSM.Identity.MNC 01          [default]
GSM.Identity.ShortName Telekom

```

Figure 6.4: Settings of GSM identity in OpenBTS.

an IP-address to connected device that is requesting it. The devices will be assigned an IP address in the range 192.168.99.0/24.

Figure 6.5 presents an overview of how the components are linked together using the setup described above. *sgsntun* is a virtual network interface with private IP addresses. OpenBTS is by default configured to forward all incoming and outgoing traffic from the interface to the public Internet (eth0). This is accomplished using Network Address Translation (NAT), which is a tool for mapping between IP address spaces.

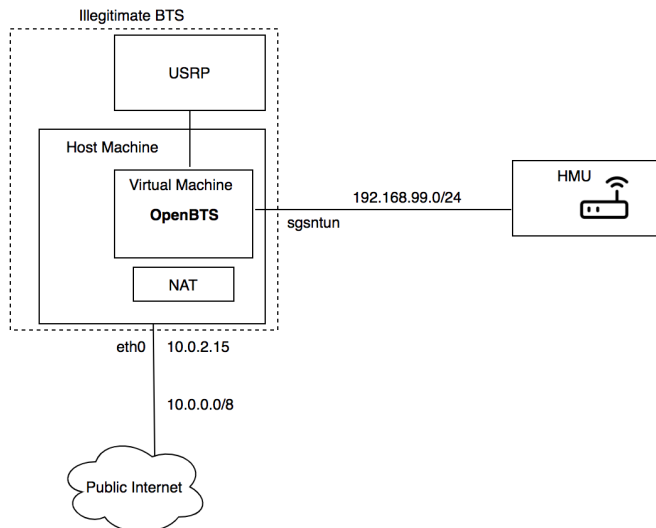


Figure 6.5: Overview of the system architecture and how the different components are connected.

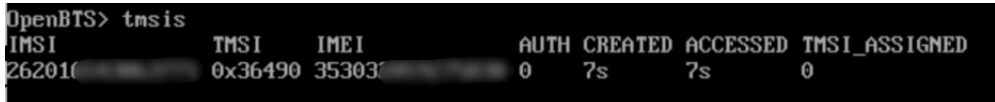
Connecting the HMUs to the Illegitimate BTS

To connect the HMUs to the Illegitimate BTS, we placed the devices in close proximity. For the HMU that supported communication over 3G, we used the jammer to block out 3G signals in the area. If the device was not able to connect, we tried changing the distance between HMU and USRP and/or using the antennas.

When a device attempts to connect to our Illegitimate BTS for the time, it is denied access. However, its IMSI and IMEI numbers are collected, and can be accessed with this CLI command:

```
OpenBTS> tmsis
```

The device should appear in the list with Auth set to 0, which means that the device has not yet been authenticated. This can be seen in figure 6.6. The IMEI and IMSI numbers are used to authenticate the device to the illegitimate BTS.



IMSI	TMSI	IMEI	AUTH	CREATED	ACCESSED	TMSI_ASSIGNED
26201	0x36490	35303	0	7s	7s	0

Figure 6.6: Screenshot from OpenBTS showing a subscriber that has not been authenticated to the illegitimate BTS.

Registering subscribers to the Illegitimate BTS

There are two ways of adding a subscriber to OpenBTS. For devices where the Ki of the SIM card is known, OpenBTS can perform the same A3 RAND-RES authentication as in GSM (ref 6.2). Since we do not have access to this information for the SIM cards from the HMUs, we will be using the other method, called cached authentication. Cached authentication means that an MSs will be authenticated with its IMEI and IMSI, and upon being authenticated once and added to the network, it is automatically authenticated until it is removed. Cached authentication is performed with the following steps:

1. Enter the directory dev/NodeManager
2. Run the file nmcli.py with the following parameters:

```
./nmcli.py sipauthserve subscribers create "[device name]"
IMSI[IMSI] "[phone number]"
```

This command will assign the selected phone number to the MS. The phone number can be any number, even a number belonging to a subscriber in the real world.

Upon attempting to reconnect the to the illegitimate BTS, registered devices will be authenticated and is thus able to connect to the network (Auth=1).

6.3 Data Server

As part of the experiments, we both observed the HMU(s) communicating with the real Data Server of Biotronik, as well as a simulated server we set up using an Ubuntu 16.04 Server VM.

6.4 Additional Tools

This section provides an overview of the tools we have used in as part of the research.

tcpdump

tcpdump is an open-source command-line packet analyzer tool that is freely available online.⁵ The tool can be configured to intercept different types of network packets on an interface, and displays the results to the user, either in real-time in the command-line, or save it to a file.

Wireshark (Tshark)

Wireshark is also an open-source network packet analyzer.⁶ Just as TCPDump, wireshark can capture packets on an interface, or read from file. The program contains a user interface where additional information about the packets are displayed to the end-user in a neat format. The command-line equivalent of Wireshark is called Tshark.

The Shikra

The Shikra is a hardware tool that provides an interface to different low-level interfaces of a device. The Shikra can be purchased online.⁷ Using the Shikra allows a user to investigate and reverse engineering embedded systems. It is connected to a computer via USB and to embedded systems using the pinouts. The Shikra can be seen in figure 6.7.

⁵<https://www.tcpdump.org>

⁶<https://www.wireshark.org>

⁷<https://int3.cc/products/the-shikra>

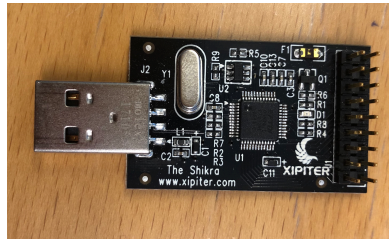


Figure 6.7: The Shikra. The pinouts are placed on the right side.

ATtention (AT) Commands

ATtention (AT) commands is a set of commands that is used for instructing modems to perform different actions. The commands were originally developed for the Hayes Smartmodem 300 baud modem, but now has a broader area of application. For example, ETSI has specified an AT commands set for GSM Mobile Equipment in GSM 07.07 [39]. Using these commands, one can instruct a modem to make calls, send SMSs, and transmit data over the internet among other things. A modem usually comes with instructions regarding what AT-commands it supports.

CyberChef

Cyberchef is an online tool for analyzing data. It provides almost 300 different data operations, such as exploring data formats and decoding data. The tool can be used by anyone who is intereseted in data analysis, and does not require any advances skills. It is freely available online.⁸

Qute

Qute is a Terminal Emulator application for Android devices. It provides a Unix command line shell for runing Unix commands. The application can be downloaded from Google Play store, but is also accessible online.⁹

iptables

Iptables is a command-line tool that is intalled on Ubuntu machines by default. It can be used to set up, maintain, and inspect the rules for IP packet. It provides different tables, for example for packet filtering and Network Address Translation (NAT). Die¹⁰ provides an overview over the different commands that can be used.

⁸<https://gchq.github.io/CyberChef/>

⁹<https://apkpure.com/qute-command-console-terminal-emulator/com.ddm.qute>

¹⁰<https://linux.die.net/man/8/iptables>

netcat

Netcat is a command-line tool for reading from and writing to network connections. It comes pre-installed on Ubuntu machines by default. SANS provides a cheat sheet for netcat commands.¹¹

Hex Fiend

Hex Fiend is a hex editing tool for Mac. It can for example be used for comparing hex files, and is freely available online.¹²

SIMspy2 and SIM card reader

SIMspy2 is a Windows tool for extracting information about a SIM card.¹³ A SIM card reader is a device that allows you to access the content of a SIM card on a computer. Combining the two allows a user to read the data stored on a SIM card, such as its settings, services, contacts as well as both saved and deleted SMSs, call log and contacts. The SIM card reader can be seen in figure 6.8.



Figure 6.8: SIM card reader, supporting regular, nano and micro SIM cards.

OpenOCD

OpenOCD is a free open-source software tool for performing on-chip debugging.¹⁴ For installation details, refer to [40]

¹¹https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf

¹²<https://ridiculousfish.com/hexfiend/>

¹³<http://www.nobbi.com/download.html>

¹⁴<https://repo.or.cz/w/openocd.git>

Chapter 7

Results

In this chapter we present our results with reference to the research questions and research objectives (RO.1 and RO.2) in section 1.3.1.

7.1 Investigating the HMUs

Our research commenced with investigating different aspects of the HMU devices. We studied the external and internal components of the HMUs, as well as any publicly disclosed information. The information uncovered in this section could be useful for subsequent experiments by explaining the behavior of the HMUs.

Every HMU has a unique serial number, and IMEI. An overview of these values are found in table 7.1 below, although parts of the values have been redacted. The table also presents the release year of each HMU as well as the year we believe the exact device to be from. The years are based on their Federal Communications Commission (FCC) IDs, which is a unique identifier assigned to any device registered with the United States Federal Communications Commission [41].

HMU	Serial Number	IMEI	Release year
LLT	4508XXXX	3528XXXXXX66495	2003
II-LLT	4712XXXX	3548XXXXXX38406	2006/2010
II-S	4816XXXX	3530XXXXXX75830	2008
Smart 3G	6476XXXX	3568XXXXXX24640	2013/2017

Table 7.1: Overview of the serial numbers and IMEI numbers of the HMUs in question in a redacted format.

7.1.1 Components of the HMUs

Figures 7.1, 7.2, 7.3 and 7.4 presents the internal components of the HMUs. The following sections will describe the relevant components in detail.

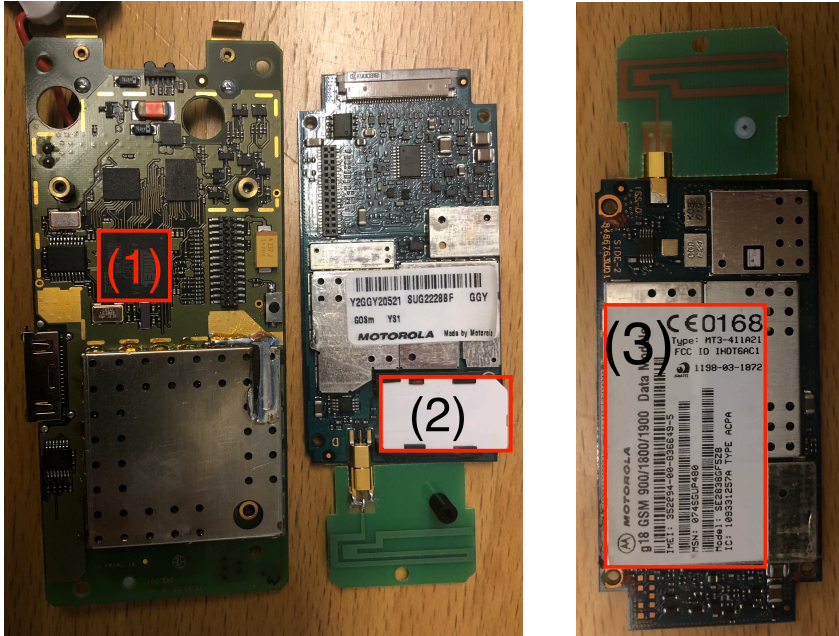


Figure 7.1: Board of the CardioMessenger LLT. 1. Micro-controller, 2. SIM card, 3. Modem.

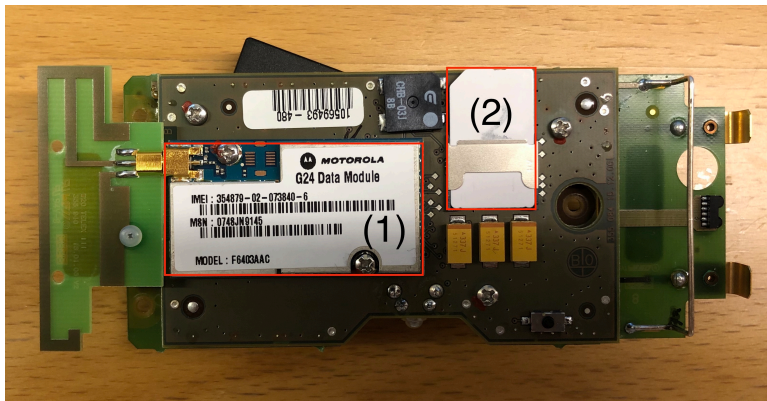


Figure 7.2: Board of the CardioMessenger II-LLT. 1: Micro-controller, 2: SIM card. The modem was not visible.

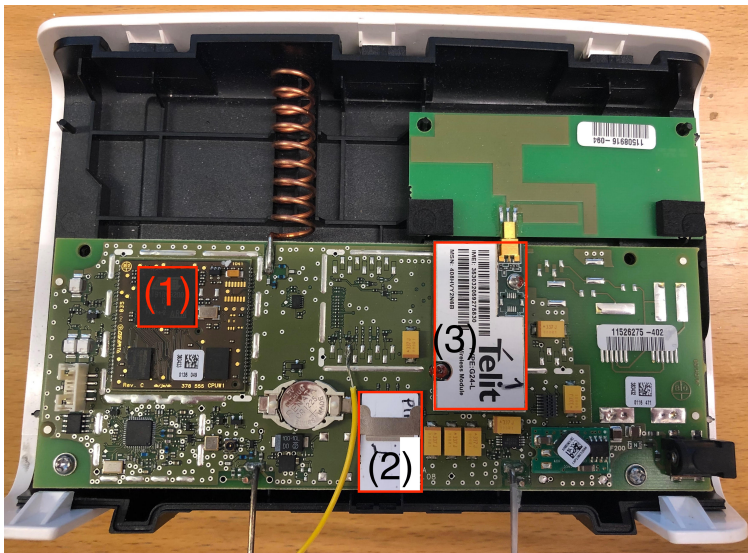


Figure 7.3: Board of the CardioMessenger 2-S. 1: Micro-controller, 2: SIM card, 3: Modem.

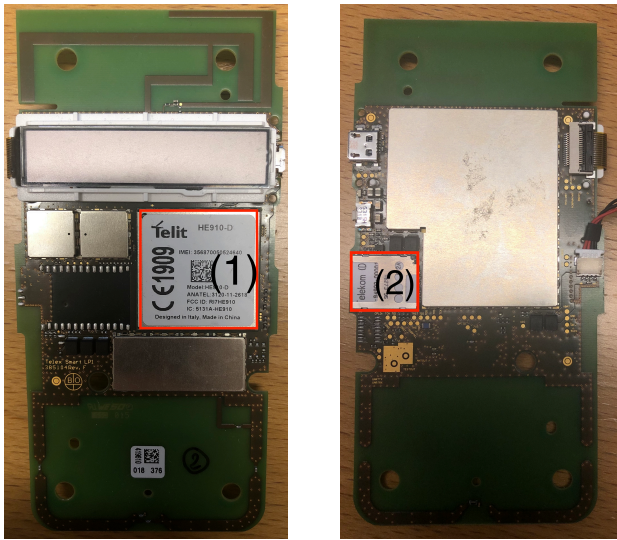


Figure 7.4: Board of the CardioMessenger Smart 3G. 1: Micro-controller, 2: SIM card. The modem is not visible.

Modems

Table 7.2 provides an overview of the modems that are used in the different HMU models.

HMU	Modem	Year	Supports
LLT	Motorola g18	2000 ¹	GSM, GPRS
II-LLT	Motorola G24	2005 ²	GSM, GPRS
II-S	Telit G24-L	2011 ³	GSM, GPRS
Smart 3G	Telit HE910-D	2012 ⁴	GSM, GPRS, 3G

Table 7.2: Overview of which modems support which technologies

SIM Cards

A SIM card, as described in section (background), is used to authenticate and connect a device to a mobile network, allowing it to communicate with other devices. Also, A SIM can store personal data and configuration settings.

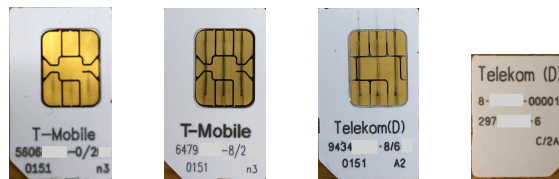


Figure 7.5: SIM cards found in the HMUs. From left: LLT, II-LLT, II-S, Smart 3G.

The SIM cards inside the HMUs were fairly easy to spot and detach from the boards. As can be seen in figure 7.5, all SIM cards have been issued by T-Mobile D or Telekom D, which today are the same company and a leading German telecommunication provider [42]. The three oldest SIM cards are full size while the SIM belonging to the Smart 3G is a Micro SIM. They were all locked, and access to their PIN codes were needed to unlock them.

Finding 1

The SIM card can be detached from all the HMUs.

7.1.2 Technical Manuals

In addition to the HMU themselves, we also had access to the "technical manuals" that are given to the patients. The manuals mostly contain instructions for using the HMU and do not contain much technical information about how the HMUs communicate. In terms of transmission of patient data, the manuals state the following:

- **LLT:** *"The CardioMessenger works like a cellphone and transmits the information received from the implant as a short message (SMS) via a cellular telephone network to the Biotronik Service Center" [43]*
- **II-LLT:** *"The CardioMessenger works like a cell phone and automatically transmits the information received from your device as encoded messages to the BIOTRONIK Service Center" [44].*
- **II-S, Smart 3G:** *"The CardioMessenger collects the information and transmits it to the BIOTRONIK Service Center as encoded messages via a mobile connection" [45][46].*

In addition, the manuals offer a thorough description of what is referred to as the "traffic light" system for the LLT, II-LLT and II-S, and "icons" for the Smart 3G. These are indications of the current status of the HMU. The most relevant light/indication is "OK", which indicates normal operation. For the different models, normal operation is indicated by:

- **LLT, II-LLT:** Green flashes of the "OK" light.
- **II-S:** "OK" light illuminating green.
- **Smart 3G:** The operation and battery icons remain permanently activated.

An example of this behavior can be seen in figure 7.6.



Figure 7.6: The II-S indicating normal operation.

7.2 Inspecting the SIM cards

7.2.1 Obtaining PIN codes

To access the information stored on the SIM cards, we needed their PIN codes. Inserting the SIM cards into a phone, we quickly discovered that the LLT SIM card used the default T-mobile PIN code that is easily accessible online [47]. This implies that all HMUs of this model has the same PIN code. This was not the case for the other SIM cards. However, we observed that the counter for number of attempts left to provide the correct PIN code was reset every time the SIM was restarted in an HMU. Based on this, we determined that the PIN code is hard coded in the memory of the HMU. As Bour [40] was already investigating the firmware of the HMUs, we kindly asked them to look for the PIN codes. With their help, we were able to find the PIN codes of the II-LLT and the II-S, but not the 3G. The uncovered PIN codes can be seen in table 7.3.

HMU	PIN1 code
LLT	1234 (Default)
II-LLT(1)	1519
II-LLT(2)	3463
II-S	5638
Smart 3G	Unknown

Table 7.3: PIN codes of the SIM cards in the HMUs

Finding 2

The PIN codes of most of the SIM cards were obtained.

We initially had two II-LLT HMU devices. However, one of them (II-LLT(1)) seemed to be defect and we decided not to include it for any further investigation. The II-LLT(2) is referred to as II-LLT for the rest of the thesis. The fact that two HMUs of the same model had different PIN codes indicates an improvement with regards to security.

We were not able to get hold of the PIN2 codes for any of the SIM cards belonging to the HMUs. It was not observed in the firmware/memory of any of the HMUs, and the default PIN2 from T-mobile did not work for any of the SIM cards.

7.2.2 Inspecting the unlocked SIM cards

Having unlocked most of the SIM cards, we set out to inspect their content to uncover information that might serve as a basis for subsequent experiments. We were mainly looking for personal data, configuration settings and subscription status.

Two different methods were used to perform the investigation: Inserting the SIM cards into a Phone and using a dedicated SIM card reader and SIMspy2. The methods are somewhat overlapping, however they both provide information that the other does not. Both of the methods were performed for each of the SIM cards with PIN code access with the goal of finding answers to the following questions:

- Does the SIM card have a valid subscription, i.e can it authenticate and connect to a network?
- If yes, is it possible to send/receive SMS, make/receive calls and/or connect to the Internet?
- What personal data (SMSs, contacts, phone number associated with the SIM card) can be found on the SIM card?
- What services are enabled on the SIM card?

METHOD 1: Inserting HMU SIM Card Into a Mobile Phone

Inserting a SIM card into a phone makes it feasible to observe the subscription status of the SIM card as well as to test its communication capabilities.

Prerequisites

- HMU SIM cards
- Phones
- Additional SIM card with a valid subscription
- Skype

Procedure

1. Insert the HMU SIM card into a phone and enter the PIN code.
2. Explore settings of the phone (APN settings etc.), the phone book, and the SMS inbox/outbox for information of interest.
3. Attempt to connect to the available networks. If no connection is established, skip the following steps.

4. Attempt to send SMS and make calls to the additional SIM card, and other phone numbers stored on the SIM card if any.
5. Open a Web Browser and attempt to surf the Internet.
6. Attempt to call the HMU SIM card using Skype, if any phone number is found on the SIM card. If any contacts are found, attempt to call these numbers as well.

METHOD 2: Using a SIM card reader and SIMspy2

Using a SIM card reader and SIMspy2 allows us to access certain information stored on the SIM cards, such as deleted contacts and SMSs.

Prerequisites

- HMU SIM cards
- Computer running Windows (10)
- SIM card reader
- SIMspy2

Procedure

1. Insert the HMU SIM card into the card reader.
2. Plug the USB-cable into a USB port of a computer running the SIMspy2 software.
3. Enter the PIN code of the SIM card upon request.
4. Explore the available information from the menu in the user interface.

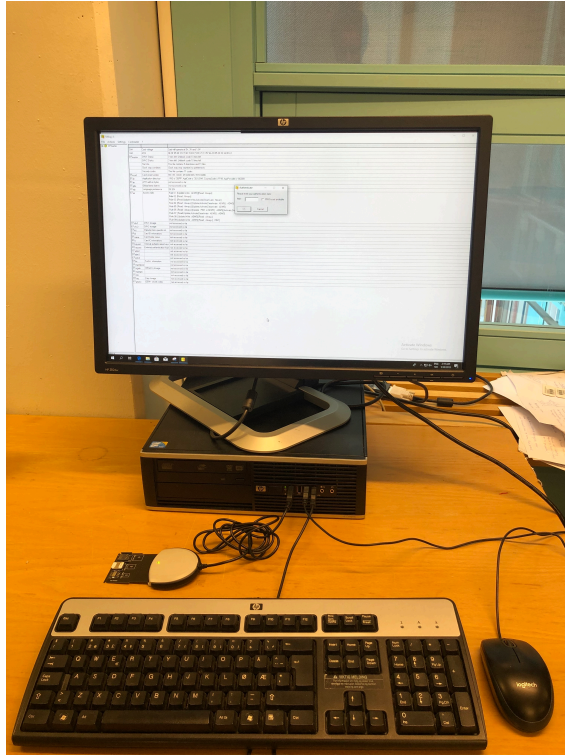


Figure 7.7: Experimental setup of SIM card reader and SIMspy2.

Personal Data

The SIM card from the LLT had four contacts stored on it. The numbers starting with +49 seem to be legitimate, German, phone numbers; this is also true for the phone numbers belonging to the LLT and II-LLT SIM cards. 151 and 171 are non-geographic area codes indicating that the numbers derive from T-mobile. The last number (*99#) appears to be a service number. The SMSC numbers are the standard numbers in German SIM cards. None of the SIM cards contained any (deleted) SMSs.

The SIM cards of the newer HMUs (II-LLT, II-S) had no contacts stored on them, and one can speculate whether the removal is due to security improvements. This is further investigated in section 7.4.4.

HMU	Contacts	Phone Number
LLT	<ul style="list-style-type: none"> • +49 1712XXXX56 • +49 1712XXXX57 • +49 1712XXXX58 • *99# 	+49 15120XXXX30
II-LLT	-	+49 15129XXXX05
II-S	-	

Table 7.4: SMSs, contacts and phone number stored on the SIM cards from the three oldest HMU units.

It was not possible to call any of the phone numbers that were stored as contacts, neither using the HMU SIM card(s) nor Skype. The numbers appear to have been blocked from receiving calls.

Enabled Services

All the SIM cards had Fixed Dialling Number (FDN) enabled, as described in section 2.1.2. The FDN list for all the SIM cards was identical and can be observed in figure 7.8.

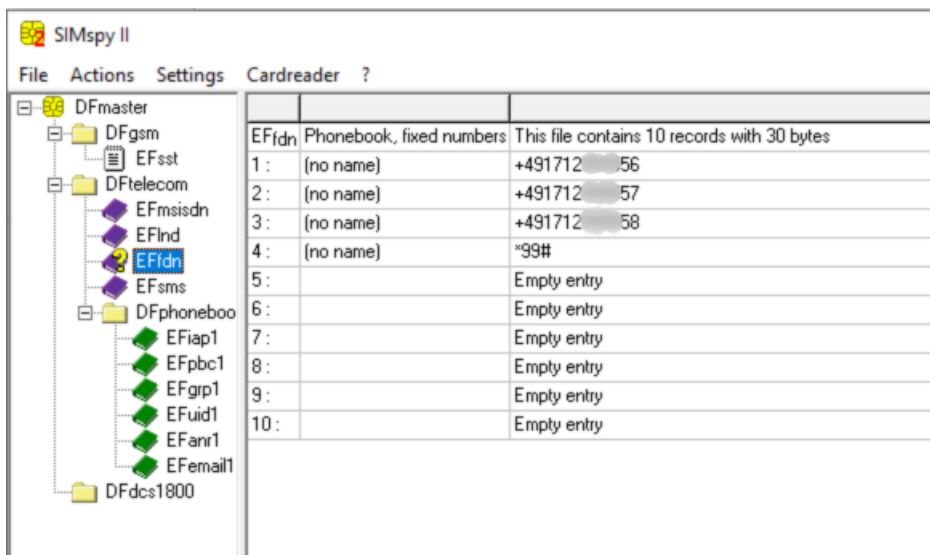


Figure 7.8: Screenshot from SIMspy2 showing the FDN list retrieved from a SIM card. All the HMU SIM cards contain identical FDN lists.

These numbers are identical to the numbers that are stored on the SIM card of the LLT. Modification or disabling of FDN requires access to PIN2, which we did not have for the HMU SIM cards.

Subscription Status and Communication Capabilities

Table 7.5 provides an overview of the subscription status of the SIM cards and their communication capabilities. Two of the SIM cards (LLT, II-S) were not registered to an operator, and could thus not be used for any form of communication.

HMU	Subscription	Call to/from	Send SMS to/from	Internet Access
LLT	No	No	No	No
II-LLT	Yes	Yes/No	Yes/No	Yes
II-S	No	No	No	No

Table 7.5: Overview over the subscription status of the SIM cards and their communication capabilities.

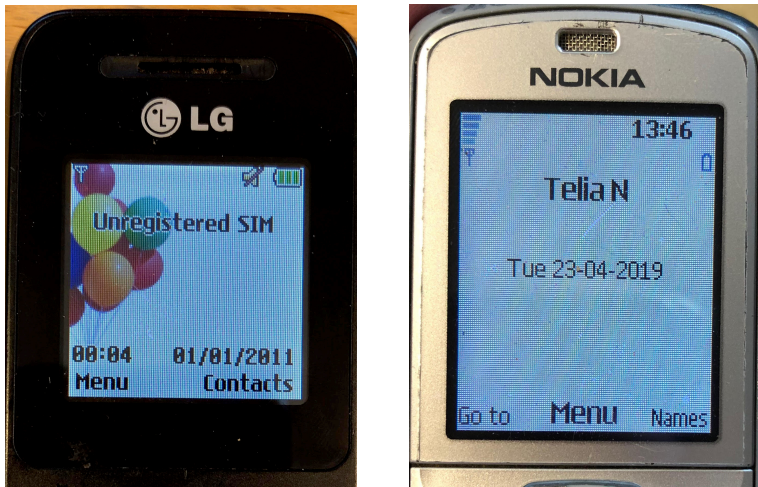


Figure 7.9: Screenshots of the LG and the Nokia phones with HMU SIM cards inserted. The picture on the left represents the status of SIM card for the LLT and II-S, while the right presents how the II-LLT SIM card still has a valid subscription.

The II-LLT SIM did have a running subscription with T-Mobile D and was able to connect to all the available networks in the area (Telia and Telenor). It was possible to call the MS containing the SIM card from Skype, but the calls were diverted to an unknown number. It was not possible to call any number from the SIM card. However, different error messages were observed when attempting to call a number

in the FDN list and when trying to call other numbers, so this seems to be related to the fact that it is not possible to call these numbers. If anything, it confirmed FDN was in fact restricting the outgoing calls from the SIM card, and hence limiting the usability of the SIM card (for an adversary).

It was not possible to send SMSs from the SIM. We observed an error message saying: *"You can only send messages to your fixed dialling numbers"*, thus confirming that the FDN restrictions of the SIM card affect outgoing SMSs.

It was possible to send SMSs to the valid SIM card. The behavior of the HMUs with regards to SMSs is further analyzed in section 7.4.3.

We were able to connect to the Internet with the II-LLT SIM card over GPRS. The SIM card was connected to a T-mobile APN (Internet.t-mobile) using default credentials. This can be seen in figure 7.10 . To stay within the ethical research boundaries, we did not test whether there was a data limit. Our goal was not to attack the system through this vulnerability, but rather to determine whether the vulnerability existed, which it did. Consequently, being able to connect to the Internet is a potential attack vector.

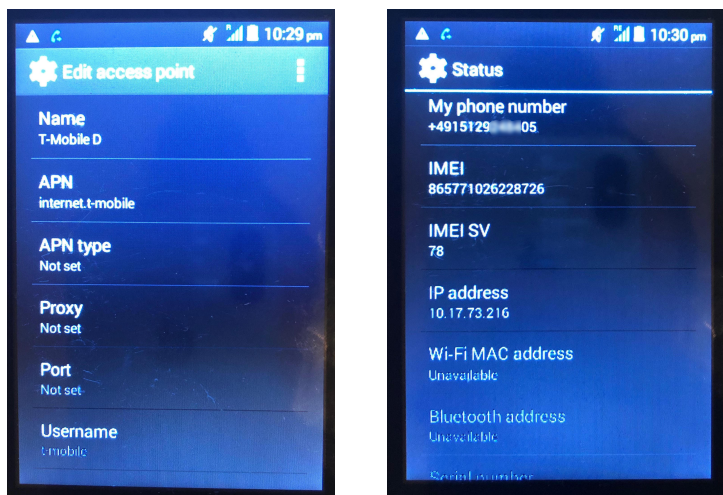


Figure 7.10: Screenshot from the Avvio phone, showing how we were able to connect to the Internet through a T-Mobile APN using the II-LLT SIM card.

Finding 3

With a valid SIM card from an HMU (II-LLT), it is possible to access the public Internet. However, calls and SMSs are restricted by Fixed Dialling Number (FDN).

7.3 Investigating the interaction between the HMUs and a legitimate mobile network (GSM)

To evaluate the HMUs as part of the implementation evaluation of the HMUs we discussed in section 4.1.2, we observed the behavior of HMUs interacting with a legitimate mobile network and investigated how their behavior was affected by various stimuli. Several experiments were performed, with different types of stimuli, where every subsequent type of stimuli was a consequence of the previous findings.

As mentioned in section 4.4, this context is not entirely realistic due to the lack of a pacemaker. However, the experiment might provide some useful information with regards to the behavior of the artifacts.

We used the setup from [40] to eavesdrop on the internal communication from the micro-controller to the modem of the II-S (the setup did not work for the other models). The following procedure presents how we used this setup, in which was used for all of the experiments in this section. The setup can be seen in figure 7.11. The same methods were used by Bour [40] to obtain the PIN codes.

Prerequisites

- Macbook
- The Shikra with wires connected to the HMU II-S board
- HMUs with SIM card

Procedure

1. (Only for the II-S) Connect the solder-on wire and the wire connected to antenna to the Shikra and insert into USB port of a computer. This is illustrated in figure 7.11.
2. (Only for the II-S) Run *screen*, a terminal emulator included in Mac OS, by running the following python script:

```
python3 cm2-serial.py -L /dev/tty.usbserial-142 115200
```

Screen is run within the script. The script is used to include time stamps to the intercepted commands. The output of the script can be found in a file with the name [initial timestamp].txt.

3. Turn on the HMU and observe the AT-commands.

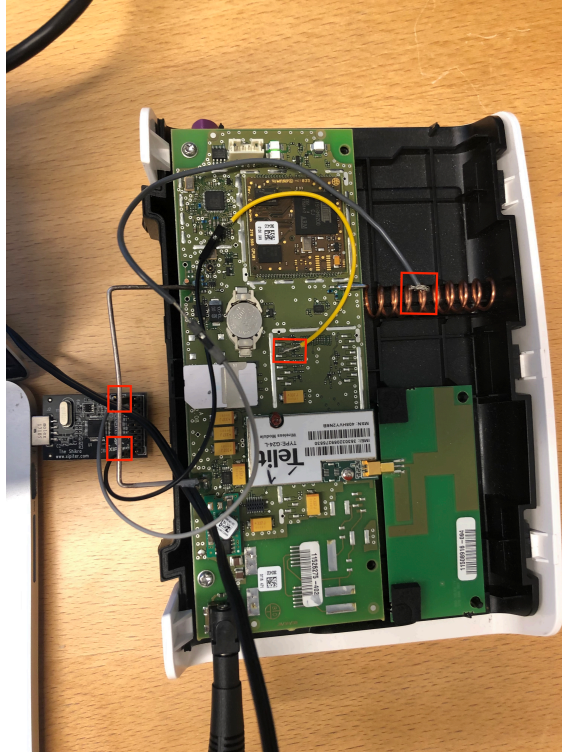


Figure 7.11: Experimental setup used for intercepting the communication between the micro-controller and modem of the II-S. The red squares indicate where the HMU and The Shikra are connected.

7.3.1 Original SIM Card inserted into the HMU(s)

In this experiment, we applied the observational case study method described in Chapter 4 and observed the interaction between the HMUs, the private APN, and the Data Server without any interference except for monitoring of the internal communication. We turned on the HMU and observed its behavior. In addition to the procedure above, we observed the external behavior of the HMUs with regards to their lights/icon.

The external behavior of some of the HMUs is inconsistent with the descriptions from the technical manuals

Table 7.6 presents whether the HMUs indicated normal operation while interacting with a legitimate mobile network. As can be observed all the HMUs, even the ones we previously observed to not have a valid SIM card, did in fact indicate normal operation. This is clearly not consistent with the information given to the patients in the technical manuals. As for the Smart 3G, we do not have enough information to know whether its status and indication of operation is consistent.

HMU	Indicating normal operation
LLT	Yes
II-LLT	Yes
II-S	Yes
Smart 3G	Yes

Table 7.6: Overview of what HMUs indicate normal operation while interacting with a legitimate mobile network.

Finding 4

The external behavior of the HMUs is inconsistent with the technical manuals.

[II-S] The HMU is not able to establish a network connection

By eavesdropping on the communication channel between the micro-controller and the modem of the II-S, we could observe the AT-commands being sent from the micro-controller and the modem. As described in section 6.4, AT-commands are commands that instruct the modem on what to do. Looking at listing 7.2, we can see that the micro controller was repeatedly asking the modem to establish a connection to a mobile network (**AT+COPS**).

The HMU was also trying to connect to Access Point Name (APN) (**AT+MIPCALL=1**), which is a point of entry to the Internet from a mobile device (described in section 2.1.3), without success. The parameters of the MIPCALL are authentication details (APN name, username, password). As can be observed in listing 7.2, the first part of the username corresponds to the serial number of the HMU that is easily found on the sticker on the HMU casings.

Eventually, the modem was reset after about 10 ten minutes (**AT+MRST**), in which the process is restarted. This pattern was observed repeatedly.

```

1 [2019-03-10 10:00:11] AT+CPIN="5638"
2 [2019-03-10 10:00:12] AT+CPIN?
3 [2019-03-10 10:00:13] ATS24=0
4 [2019-03-10 10:00:13] ATS100=0
5 [2019-03-10 10:00:13] ATS102=0
6 [2019-03-10 10:00:13] AT+MSCTS=0
7 [2019-03-10 10:00:18] AT+CREG?
8 [2019-03-10 10:01:25] AT+COPS=?
9 [2019-03-10 10:01:46] AT+COPS=1,2,"24202"
10 [2019-03-10 10:01:47] AT+COPS=3,0
11 [2019-03-10 10:01:51] AT+CREG?
12 [2019-03-10 10:01:52] AT+COPS=1,2,"24201"
13 [2019-03-10 10:01:52] AT+COPS=3,0
14 [2019-03-10 10:01:57] AT+CREG?
15 [2019-03-10 10:01:59] AT+CREG?
16 [2019-03-10 10:01:59] AT+COPS=0,2
17 [2019-03-10 10:01:59] AT+CGMI
18 [2019-03-10 10:01:59] AT+COPS=3,2
19 [2019-03-10 10:01:59] AT+COPS?
20 [2019-03-10 10:01:59] AT+COPS=3,0
21 [2019-03-10 10:01:59] AT+CSQ
22 [2019-03-10 10:02:02] AT+CSQ
23 [2019-03-10 10:02:02] AT+CREG=2
24 [2019-03-10 10:02:02] AT+CREG?
25 [2019-03-10 10:02:02] AT+CREG=0
26 [2019-03-10 10:02:08] AT
27 [2019-03-10 10:02:08] AT+CPMS="SM"
28 [2019-03-10 10:02:08] AT+CMGL=4
29 [2019-03-10 10:02:08] AT+MIPCALL=1,"biotroni.ic.t-mobile","4816
    XXXX@cm3-homemonitoring.de","TP0pSXXXXX"
30 [2019-03-10 10:05:51] AT+MIPCALL=1,"biotroni.ic.t-mobile","4816
    XXXX@cm3-homemonitoring.de","TP0pSXXXXX"
31 [2019-03-10 10:08:23] AT+COPS=?
32 [2019-03-10 10:08:45] AT
33 [2019-03-10 10:08:45] AT+COPS=1,2,"24202"
34 [2019-03-10 10:08:45] AT+COPS=3,0
35 [2019-03-10 10:08:50] AT+CREG?
36 [2019-03-10 10:08:50] AT
37 [2019-03-10 10:08:50] AT+COPS=1,2,"24201"
38 [2019-03-10 10:08:50] AT+COPS=3,0
39 [2019-03-10 10:08:55] AT+CREG?
40 [2019-03-10 10:10:51] AT+MRST

```

Listing 7.1: Excerpt from AT-commands illustrating how the HMU II-S is repeatedly trying to establish a network connection to the available networks (24201, 24202)

and obtain an IP address by sending APN credentials.

Finding 5

The serial number of the HMU (II-S) is part of the credentials used for authenticating to the Access Point Name (APN).

The HMU is trying to send SMSs

After attempting to establish an APN connection for a while without success, the HMU attempted to send SMSs. The phone number in which the HMU was sending the SMS to was one of the numbers previously observed in section 7.2.2.

```
1 [2019-03-10 20:32:06] AT+CMGF=0
2 [2019-03-10 20:32:06] AT+CMGS=120
3 [2019-03-10 20:32:06] 079194710167000011560C91947121XXXX6500F6C
4 86A06046680D7C8BDB19684EF24287FA3954CA200AFBBD44C170DB1A8C452D
5 C03891BD43302DAC8DA33CE225FF99E976F9B066CA00AA5A25AB8A47218D21F
6 232ED41AED4E0F22F42E6189968D7CF6B965B73A768D7CF6B965B73A768D7CF
7 6B965B73A7205C7FDBEA102B22
```

Listing 7.2: Excerpt from AT-commands illustrating how the HMU II-S is attempting to send an SMS without a network connection.

It is unclear why the HMU would try to send SMSs or establish connection to an APN without a network connection.

Finding 6

SMS is used as a mean of communication (for the II-S).

These results correspond to what we observed in section 7.2.2: The II-S does not have a valid SIM card. This can explain the lack of an established network connection. As for the credentials provided to connect to the network, further investigation was required to determine whether they were still valid.

7.3.2 [II-S only] Inserting a valid SIM card from another HMU

Based on the findings from the previous experiment, we wanted to determine whether the APN credentials were still valid. As this would require stimuli, we were no longer just observing the interaction without interfering. As such, the single-case experimental method was applied, as described in section (4.1.2). The stimuli in this experiment was a valid SIM card from the II-LLT HMU, so that the II-S would hopefully be able to establish a network connection. Also, this would reveal if the

HMU's accept non-original SIM cards. The only thing we did was to change the PIN code of the II-LLT SIM card to match the PIN code of the II-S.

As it turned out, the HMU accepted the SIM card and was able to establish a connection to the network. Also, the credentials proved to be valid, and upon authentication, the HMU proceeded with establishing a TCP socket/connection to an IP address and port number (172.16.14.1, 2323). Consequently, the HMU started to transmit data (**MIPOPEN=1 and MIPSEND=1**). The data transmitted will be analyzed in section.

The Data Server is located on a private APN

The IP-address (172.16.14.1) the HMU was establishing a TCP connection with is a private IP address. It is reasonable to believe that this is the IP address of the Data Server, and that it has been configured on a private subnet, where it cannot be reached from the public Internet. The fact that the HMU was communicating directly to a private IP address is also indicating that the HMU and Data Server are connected to the same private network.

Finding 7

The Data Server is not accessible from the public Internet.

Finding 8

Upon authentication of the HMU, the HMU and the Data Server are connected to the same private APN.

Credentials are sent in the clear, but the subsequent data is not

As can be observed in figure 7.12 first data packet transmitted via the TCP socket contained credentials, and indicates the HMU authenticating itself to the Data Server. Also, the credentials were identical to the APN credentials used to access the private network. The remaining data that was sent over the connection was not in clear text, as can be seen in figure 7.13.

```

1 AT+MIPCALL=1,"biotroni.ic.t-mobile",
2 "48XXXX65@cm3-homemonitoring.de","TP0XXXXXqI"
3 AT+MIPOPEN=1,10671,"172.16.14.1",2323,0
4 AT+MIPSETS=1,1372
5 AT+MIPSEND=1,"3438XXX36353440636D332D686F6D656D6F6E69746F7269
6 6672E64650D54504FXXXXX71490D"
7 AT+MIPSEND=1,"0500BE00000002DF02B6080BC5FC998B4A0631AC8D78C39ECA
8 3C5D309F7A1BBE7F5C8EF3A2F814AD9D156F0E5C183F48AB298B6C9CC20223F6

```

9 93EE78BA10913123614CCD8F3F3E269BD6C7 "

Listing 7.3: Excerpt from the AT-commands, presenting data being sent from the HMU to the Data Server. Parts of the credentials have been redacted.

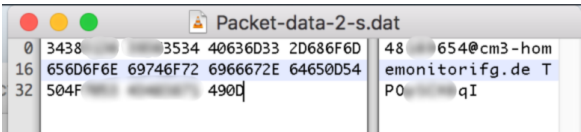


Figure 7.12: Screenshot from Hex Fiend revealing that the credentials are sent in cleartext over the private communication channel

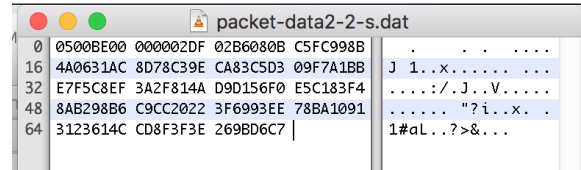


Figure 7.13: Screenshot from Hex Fiend revealing that the the remaining data sent from the HMU is not in cleartext.

No SMSs are sent

With a TCP connection established, we did not observe any SMSs being sent. This may indicate that SMS is used as a backup for when a TCP connection is not possible, and is further discussed in section 7.4.2.

Finding 9

(II-S) The APN credentials are still valid. They are not tied to a SIM nor the device and is used multiple places: for authenticating the HMU to both the APN and the Data Server.

Finding 10

(II-S) A TCP socket is used for transmitting data packets.

Finding 11

(II-S) The Data Server credentials are sent in cleartext over the APN network, however the rest of the data is not.

7.3.3 Inserting a valid HMU SIM card Into a third-party device

Having determined that the only certain SIM cards could access the APN, and the credentials are not tied to a specific SIM card, we wanted to determine whether we could access the APN from a third-party device. We inserted the SIM card from the II-LLT with a valid subscription into the Avvio phone (see section 6.2, and used the credentials from the II-S. Concerning ethical boundaries, we had to take precautions when investigating the private network. HMUs in current use by pacemaker patients could be connected to the network and the server, and we did not want to affect the operation of these components. Also, we did not have authorization from Biotronik to access the network/Data Server. Consequently, we only determined whether it was possible to reach the server and whether one could access the public Internet.

Prerequisites

- SIM card from the II-LLT
- Phone with the possibility to change APN settings (Avvio)
- Application for accessing a Terminal running on the phone (Qute)

Procedure

1. Insert the SIM card into the phone
2. Change the APN settings of the phone to match the APN settings of the II-S HMU
3. Allocate the IP address of the device (Usually found in network settings). If no IP address is allocated, skip the remaining steps.
4. Enter a web browser on the phone and try to browse the Internet
5. Try to ping the Data Server (ping 172.16.14.1)

Result: As can be seen in figure 7.14, the device was successfully authenticated and was assigned an IP address from the same private range as the server. We were also able to ping the server, as can be seen in figure 7.15. However, it was not possible to access the public Internet.

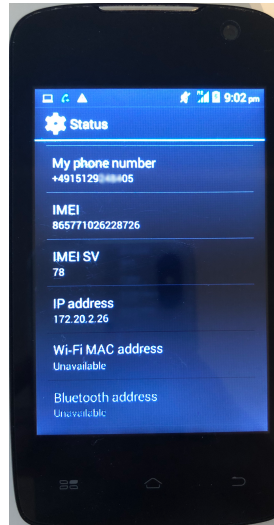


Figure 7.14: Phone with II-LLT SIM card connected to the private APN. The assigned IP address is in the same range as the Data Server. The phone number confirms that the SIM card of the II-LLT is used.

Finding 12

With a valid SIM card and credentials, it is possible to access the private APN where the Data Server is located, without having access to an HMU that is still in use by a patient.

7.3.4 Inserting a SIM card with a running subscription that is not from an HMU

Having determined that the HMU accepts non-original SIM cards and that the HMU and the data server are communicating over a private network, we wanted to further investigate the security of this network. Therefore, we inserted a SIM card that had no relation with the pacemaker ecosystem, and followed the same procedure.

Interestingly, we did not observe the HMU attempting to authenticate to the APN at all during this experiment. To further investigate whether this was due to restrictions on the network or the SIM card, we also tried to access the network with the same SIM card inserted into a phone, following the procedure steps 1-3 from the previous experiment. As it turned out, the device was not assigned an IP address. This indicates that there have been set some restrictions on the network with regards to access control, so that only certain SIM cards can access it.

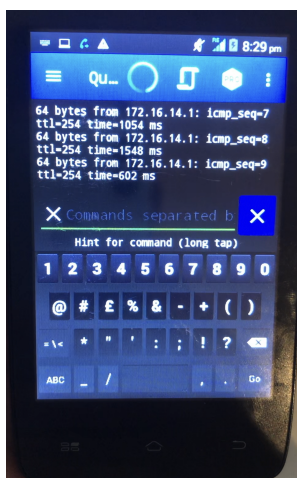


Figure 7.15: Pinging the Data Server (172.16.14.1) from phone with the II-LLT SIM card inserted that is connected to the same private APN.

Finding 13

The private APN performs access control, only accepting certain SIM cards.

7.4 Investigating the Interaction Between the HMUs and an Illegitimate BTS

In this part of the research, we made use of an illegitimate BTS. The details around the setup is found in section 6.2.1. Using an illegitimate BTS would allow us to intercept traffic from all the HMUs. Due to the nature of an illegitimate BTS, all SIM cards are considered valid, so we were expecting to observe traffic even from the SIM cards that were not valid in the real mobile network.

As discussed in the threat model (see section 4.2, setting up an illegitimate BTS and tricking devices to connect to it, is considered an active attack. Forcing unencrypted communication is also considered an attack, which is what an Illegitimate BTS configured with OpenBTS does by default. While the HMUs are connected one can further perform additional attacks on the HMUs to uncover the attack surface of the system.

It was necessary to use a jamming device to force the Smart 3G device to communicate over 2G instead of 3G. The jamming signals had a limited range and did not affect any other devices in the area.

Results: All the HMUs were successfully connected to the illegitimate BTS, and their IMEI and IMSI numbers were collected. These numbers are found in table 7.7. The IMSIs further confirm that the SIM cards are German, as 262 is the MCC of Germany.

HMU	IMEI	IMSI
LLT	3528XXXXXX66495	26201XXXXXX6889
II-LLT	3548XXXXXX38406	26201XXXXXX5647
II-S	3530XXXXXX75830	26201XXXXXX3771
Smart 3G	3568XXXXXX24640	26201XXXXXX7863

Table 7.7: IMSI and corresponding IMEI of the SIM cards and the HMUs. Parts of the numbers have been redacted to anonymize the HMUs.

Finding 14

All the HMUs are connecting to the illegitimate BTS.

All the HMUs but the LLT requested an IP address from the illegitimate BTS, which they got. As the HMUs were never connected to the illegitimate BTS at the same time, they were all assigned the IP address 192.168.99.1. This is illustrated in figure 7.16. The fact that the LLT did not request an IP address is not surprising, as we have already uncovered (in section 7.1.2) that this HMU is only sending SMSs to the data server even though the modem supports GPRS.

```
OpenBTS> sgsn list
GMM Context: imsi=2620[REDACTED]5647 ptmsi=0x54001 tlli=0xc0054001 state=GmmRegisteredNormal age=43 id
le=7 MS#1,TLLI=c0054001,80402b3c IPs=192.168.99.1
```

Figure 7.16: Screenshot from OpenBTS, showing an HMU being assigned an IP address from the Illegitimate BTS.

7.4.1 Eavesdropping on the Communication Channels

Having confirmed that all the HMUs were able to connect to the illegitimate BTS, the next step was to intercept in the potential communication channels. From previous experiments, we had observed the II-S transmitting both SMS and data packets, and were hoping to observe similar behavior for all the HMUs.

To be able to observing any traffic in real-time, we created a shared folder between the host machine and the virtual machine running tcpdump (Ubuntu 14.04). The details around this can be found in Appendix B.

Prerequisites

- tcpdump (to capture traffic)
- Wireshark (to display the captured traffic)

Procedure

1. Run tcpdump on the Ubuntu 14.04 command line with these options:

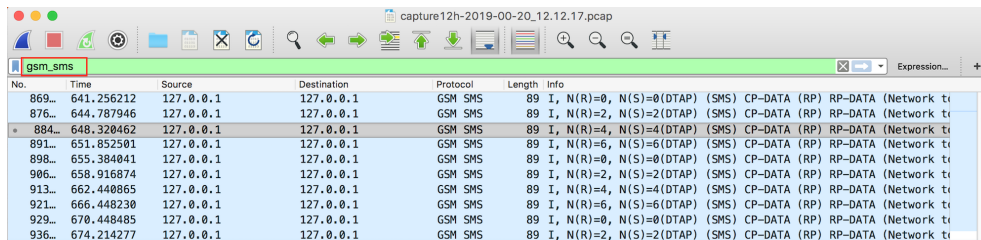
```
tcpdump -i any -n s0 (-G 3600) -w [filename](.pcap)
```

This command will record all (-n any) traffic and save it to a .pcap file

2. Turn on the HMU and make sure it connects to the illegitimate BTS
3. Open the file in Wireshark on the host machine
4. Apply a suitable filter in GSM. For SMSs, use *gsm_sms* to filter out any traffic but SMSs. For filtering on an IP address: *ip.addr == [IP address]* and on TCP port: *tcp.port == [port number]*

Results

We were able to intercept SMSs from all HMUs but the most recent unit (Smart 3G). Figure 7.17 and 7.18 shows intercepted SMSs in Wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
869...	641.256212	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=0, N(S)=0(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
876...	644.787946	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=2, N(S)=2(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
* 884...	648.320462	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=4, N(S)=4(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
891...	651.852501	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=6, N(S)=6(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
898...	655.384041	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=0, N(S)=0(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
906...	658.916874	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=2, N(S)=2(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
913...	662.440865	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=4, N(S)=4(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
921...	666.448238	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=6, N(S)=6(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
929...	670.448485	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=0, N(S)=0(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...
936...	674.214277	127.0.0.1	127.0.0.1	GSM SMS	89	I, N(R)=2, N(S)=2(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network t...

Figure 7.17: Screenshot from Wireshark showing SMSs after having applied the filter *gsm_sms*.

▼ TP-User-Data
SMS body: 0604d364f2d8e8fac0bbef24287fa3954ca200afbdbc44c1...

0000	09	01	84	00	b9	00	07	91	94	71	01	67	00	00	78	11q·g·x·
0010	b9	0c	91	94	71	21	55	55	65	00	f6	c8	6a	06	04	d3	...q!UU e...j...
0020	64	f2	d8	e8	fa	c0	bb	ef	24	28	7f	a3	95	4c	a2	00	d.....\$(...L...
0030	af	bb	dc	44	c1	70	db	b9	0c	d0	ce	8d	36	54	bd	fd	...D·p·...6T...
0040	ae	b8	d5	65	1a	4f	05	25	ff	99	e9	76	f9	b0	66	ca	...e·0·% ...v...f·
0050	00	aa	5a	25	ab	8a	47	21	8d	21	f2	32	ed	41	ae	d4	..Z%·G! ·!·2·A·

Figure 7.18: Screenshot from Wireshark showing the UD from a single SMS from an HMU.

Finding 15

The HMUs are sending SMSs (except for the Smart 3G). The User Data of the SMSs is not in cleartext.

All the HMUs were sending SMSs to the same phone number. With the current setup, as this number had not been registered with the illegitimate BTS, error messages were also observed in Wireshark. An example of this can be seen in figure 7.19. Due to the flexible number policy in OpenBTS as mentioned in section 6.2.1, it was easy to register a SIM card with this number to OpenBTS. This way, by inserting this SIM into a phone we could also observe the SMSs on the receiver side.

▼ TP-User-Data
SMS text: Can't send your SMS to +49171 : Phone not registered here.:

Figure 7.19: Screenshot from Wireshark showing the response received from the network upon sending an SMS to a user not registered in the network.

For the II-S and II-LLT, no SMSs were observed being received on the phone. For the LLT, for every message sent and observed in Wireshark, an SMS with with either no content (Avvio) or a single "x" was received (Iphone 4S, LG), depending on the phone used. Figure 7.20 shows this. These are standard ways of displaying a binary SMSs on equipment that does not know how to decode the content.



Figure 7.20: Screenshot from Iphone 4s with sysmocom SIM card, showing SMSs received from the LLT 1234 is the number that the LLT SIM card was assigned in OpenBTS

The HMUs are not able to establish a TCP connection

Looking the AT-commands, can observe that the HMU II-S is receiving an IP address, however is not able to connect to the "server" and consequently no data is being sent. This is not surprising, as the IP address is not coming from the APN but from the illegitimate BTS.

By inspecting the data packets received from the HMU II-S in Wireshark, the same behavior can be observed. Filtering on the IP subnet that the HMU's IP is in, we can see that the HMU (192.168.99.1) is trying to reach a private IP address (172.16.14.1) on port 2323, in which it is not able to. We already knew that this IP address belongs to the server, but this demonstrates how we could have learned about the Data Server without listening on the AT-commands. As such, this could potentially have been a fully wireless attack.

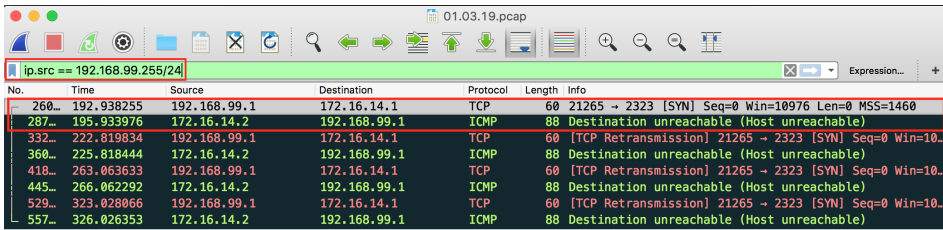


Figure 7.21: Screenshot from Wireshark showing how the II-S HMU (192.168.99.1) is attempting to connect to an unreachable IP address (172.16.14.1)

7.4.2 Spoofing the Data Server

To be able to eavesdrop on the data being sent over the TCP socket, we had to trick the HMU into believing that it was connected to the DS. We already knew the IP address and port number of the Data Server that the II-S was connecting to. Since these parameters seems to be static, we decided to set up a machine with the same IP address to listen on the same port, and see if this would trigger communication. This would only work if the HMU did not authenticate the server.

Since the HMU was assigned an IP-address outside of the private range that the IP address of the server was from, we also had to set up Network Address Translation (NAT) to make it seem like they were on the same network.

Prerequisites:

- A machine that can act as the data data server (We used an Ubuntu 16.04 Server VM)
- netcat
- iptables
- ifconfig

Procedure

1. Set up a private subnet in VirtualBox (Settings -> preferences -> Networks) with the same details as the server you want to simulate (172.16.0.0/12 in this case)
2. Add this network to both VMs, the one running OpenBTS and the one running the data data server (Settings -> network -> add network [select the network that was created in the previous step])
3. Both VMs should now each get a new network interface, which can be seen by running the ifconfig command.

4. Configure the machine acting as the server with the same private IP-address as the original data data server:

```
ifconfig [interface] 172.16.14.1/12
```

5. On machine running OpenBTS:

```
ifconfig [interface] 172.16.14.2/12
```

6. Also on the machine running OpenBTS, run the following command:

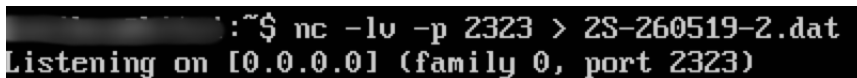
```
$ cd /etc/OpenBTS/iptables.rules
```

With the latter command, NAT is used to mask the IP address of the HMU (192.168.99.1) with the IP address on the same network as the spoofed server (172.16.14.2), thus making it look like as if the HMU and data data server are on the same subnetwork. In other words, all traffic being sent from an HMU (192.168.99.1) to the server (172.16.14.1) will look like it was sent from 172.16.14.2.

7. Ensure that it is possible to establish a connection between the VMs by using the Ping command line program.
8. On the data data server VM, run netcat with the following command

```
nc -l (-v) -p [port number] > [outfile]
```

This command instructs netcat to listen (-l) on the port number [port] for a connection, and store the result (output) in the file [outfile]. For these experiments, the port number is 2323. The result can be seen in figure 7.22.



```
~$ nc -lv -p 2323 > 2S-260519-2.dat
Listening on [0.0.0.0] (family 0, port 2323)
```

Figure 7.22: Caption

Figure 7.23 shows an updated version of figure 6.5, now including the spoofed server and NAT between the OpenBTS network (192.168.99.0/24) and the server network (172.16.0.0/12).

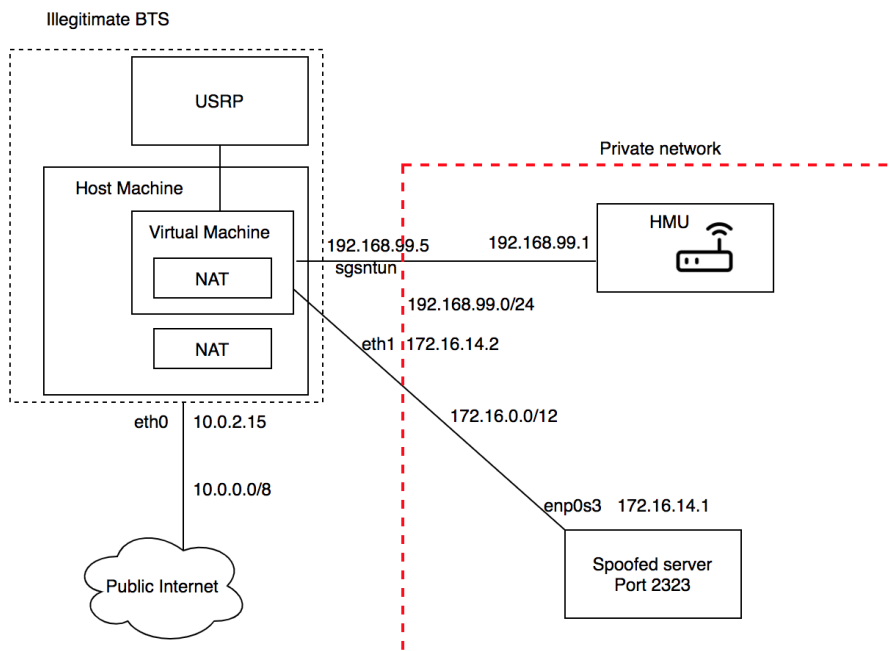


Figure 7.23: Overview of the overall architecture, including the HMUs, the illegitimate BTS, and the spoofed Data Server

Results: This setup worked not only for the II-S, but also for the II-LLT, II-S and Smart 3G HMUs, and we could observe the data being sent to what they believed to be the real Data Server. Figure 7.24 shows an HMU successfully connecting to the spoofed server. This indicates that none of them authenticate the server.

```

~$ nc -lv -p 2323 > 2S-260519-2.dat
Listening on [0.0.0.0] (family 0, port 2323)
Connection from [172.16.14.2] port 2323 [tcp/*] accepted (family 2, sport 55890)

```

Figure 7.24: Screenshot from VM with IP 172.16.14.1 listening on port 2323, accepting a connection from a HMU (172.16.14.2).

Finding 16

The HMUs do not authenticate the Data Server. This allows for a Man-in-the-Middle (MITM) attack where the adversary can eavesdrop on the communication.

Finding 17

All the HMUs with Internet access are connecting to what seems to be the same Data Server, and are transmitting data to it.

Finding 18

The Data Server has a static IP address and port number.

Just as we observed with the II-S, the II-LLT also has its serial number included in the credentials being sent in clear-text. This further confirms that all HMUs have individual credentials. In theory, this is a good secure measure. However, as we have observed, the credentials are not tied to the HMU or its SIM card. Hence, only one set of credentials is needed to access the private network and the Data Server with any valid SIM card. We were not able to find any credentials for the Smart 3G.

Finding 19

The II-LLT and II-S have unique credentials that include the serial number of a particular device for connecting to the private APN and the Data Server.

No SMSs were observed upon connection with the spoofed Data Server

Just as we observed with the II-S in section 7.3.2, we could not observe any SMSs for the rest of an ongoing session once the HMU had established a connection to the spoofed Data Server. Hence, this further implies that SMS are used as a backup for when a GPRS connection is not possible.

Finding 20

For some of the HMUs (II-LLT, II-S), SMS seems to be a backup for data packets over the TCP connection.

7.4.3 Sending SMSs to the HMU

Malicious SMSs is a known attack vector in GSM [28], and it was hence of interest to investigate whether the HMUs are prone to such an attack. As we recall from section 7.2.2, it was possible to send an SMS to the II-LLT SIM card.

OpenBTS has a built-in feature that let us send an "echo" SMS from the CLI 6.1 using the following command:

```
OpenBTS> sendsms [IMSI of HMU] [sender] ["message"]
```

Using the same setup and procedures from section 7.3, we eavesdropped on the internal communication of the II-S. For the II-S, the AT-commands revealed a repeating pattern of commands:

```
1 AT+CPMS="SM"  
2 AT+CMGL=4 #List all SMS  
3 AT+CMGD=4 #Delete all SMS
```

From the AT-commands we can understand that the HMU is listing (**AT+CMGL=4**) and deleting (**AT+CMGD=4**) all the SMSs it can find on the SIM card (**AT+CPMS="SM"**). Looking at Bour's results [40], they found a function inside the firmware of the II-S HMU that seems to be parsing SMSs.

For the other HMUs, we inserted their SIM cards into the smart card reader and explored their content using SIMspy2, using the same procedure as in section 7.2.2. In all the SIM cards, the SMS sent from OpenBTs was found in deleted SMSs, hence indicating the same pattern as for the II-S.

Based on these findings, we can assume that it is possible for the HMUs to receive SMSs also when connected to the real mobile network. Although our experiment stopped here, we showed that the HMUs are vulnerable to SMS attacks.

Finding 21

SMSs are processed by the HMU. This opens up for using SMS as an attack vector.

7.4.4 Spoofing the SIM card

From 7.3.2, we knew that the II-S HMU accepts a non-original SIM card. We wanted to exploit this by trying to insert a SIM card with different "settings". We used information from previous experiments as a basis for these experiments.

[LLT] Inserting SIM card With different contacts stored

From experiment 7.2.2, we saw that the SIM of the LLT had contacts stored on it, and that the contacts included the phone number that the HMUs are sending SMSs to. we wanted to learn whether the HMU reads directly from SIM regarding where to send SMSs. To do this, we inserted a SIM card with different contacts stored to see how this affected the behaviour of the LLT. This experiment was conducted without the spoofed Data Server available, as we wanted to observe SMSs.

As it turned out, the behavior of the HMU was not affected and it continued sending SMSs to the same number as before, hence indicating that the HMU is not reading from the SIM memory. This is good, as it excludes the possibility for an adversary to insert a SIM card with other contacts stored on it, thereby redirecting all SMSs to themselves.

Finding 22

The HMUs have the recipient number for SMSs stored in memory. It is not read from the SIM card.

7.5 Analyzing the Intercepted Data

This section further analyzes the gathered data from the different experiments. Table 7.9 presents the types of data that was observed being transmitted by the different HMUs.

HMU	SMS	Packet data
LLT	Yes	No
II-LLT	Yes	Yes
II-S	Yes	Yes
Smart 3G	No	Yes

Table 7.8: Overview of which devices we were able to observe what type of data from.

7.5.1 Data Packets

All the data packets we observed were sent to the same IP address and port number (172.16.14.1, 2323). Wireshark was not able to identify the application layer protocol that was used, indicating that a proprietary communication method is used on top of TCP. This is indicated by *:data in protocols in frame*, that can be seen in figure 7.25.

Hence, we could exclude the use of standard protocols such as HTTP, FTP and Telnet. Considering the port number, Port 2323 is not a standard port. It is occasionally used for Telnet, however Telnet would have been recognized by Wireshark. This applies to the packets from all of the HMUs.

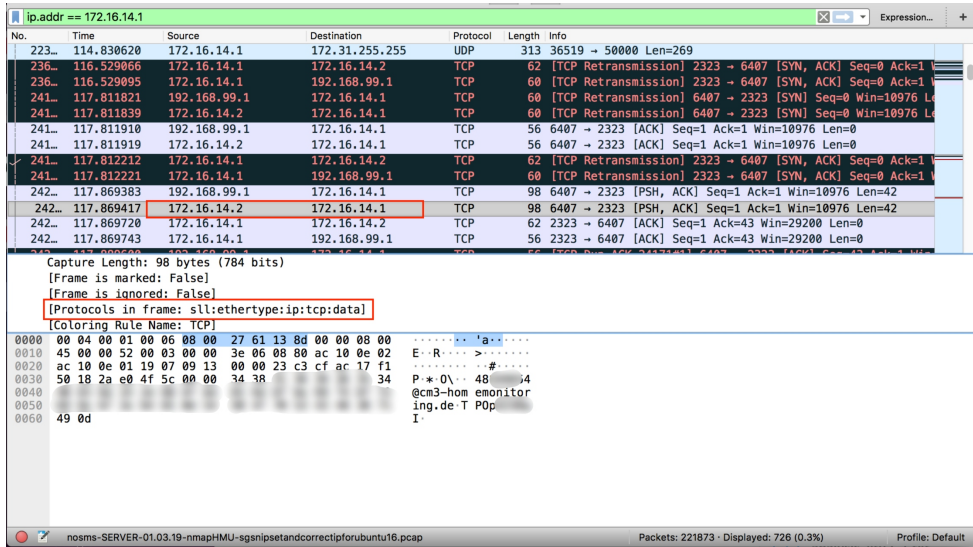


Figure 7.25: Screenshot from Wireshark, packets are filtered on the IP address of the Data Server, 172.16.14.1.

For every connection to the (spoofed) Data Server, the II-LLT and the II-S HMUs always transmitted their credentials in clear text as the first data packet. An example of this from the II-S can be seen in figure 7.25. The credentials for the two devices were not identical, and can be linked to the device as the username contained the serial number of the HMU with following format: *serial number*@cm3-homemonitoring.de, *password*.

As mentioned in section 7.3.2, for the II-S, the same credentials were observed for authenticating the HMU to the APN. This means that they are reusing credentials, and we can assume that this is also the case for the II-LLT. However, we cannot confirm this.

Regarding the Smart 3G, no clear-text was observed and only one packet (168 bytes) was sent for every connection before it was terminated. This suggests a different data pattern than for the other HMUs. The beginning of every packet had a few identical bytes, but we could not determine their meaning or content.

Finding 23

The Smart 3G is not sending credentials in cleartext, and seems to be sending different data than the II-LLT and the II-S.

Figure 7.26 presents the data being sent over the TCP sockets for the different HMUs. As can be seen, most of the data appears to be random.

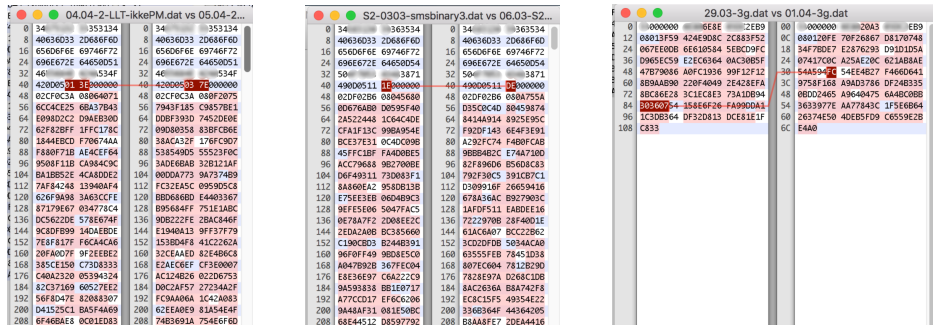


Figure 7.26: Screenshots from Hex Fiend, showing the comparison of data packets from the same HMU. From left: II-LLT, II-S, Smart 3G. Parts of the credentials have been redacted.

The average entropy of the data from the HMUs can be found in table 7.9. An entropy above 5 indicates some sort of encrypted or compressed data, while properly encrypted or compressed data of a reasonable length should have an entropy of over 7.5 [48]. As such, the data from II-LLT and II-S seem to be compressed and/or encrypted. For the Smart 3G the entropy is lower, however this may be because the data packets are too short.

HMU	Average Entropy
II-LLT	7.76
II-S	7.90
Smart 3G	6.35

Table 7.9: Overview over the average entropy of data packets being sent from each HMU.

Decrypting Packet Data

As part of their research, Bour [40] succeeded to decrypt data that was transmitted by a similar device as the II-S: The CardioMessenger II-S T-line. The difference between the devices is that the T-line transmits data over the telephone line instead

of using wireless communication. This means that the devices have identical boards. The only difference is the components on the boards. For example, the II-S contains a wireless modem while the II-S T-line does not. Due to the similarities, we suspected that they might also use the same encryption scheme, namely Advanced Encryption Standard (AES). Therefore, we collaborated with Bour [40] to test the same method on the II-S.

Tools

- Raspberry Pi Zero with OpenOCD installed, used as a JTAG adapter
- Wires for connecting the Raspberry Pi to the HMU

Procedure

1. Connect the wires to the identified pins on the JTAG interface on the HMU board. This is presented in figure 7.27.
2. Run the `dump_image` command in OpenOCD to read from the memory of the HMU. This will take a couple of minutes.

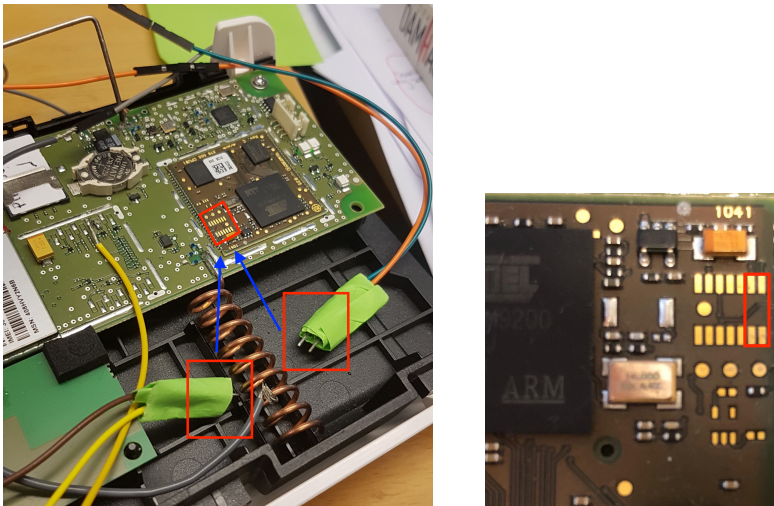


Figure 7.27: Connecting to the JTAG interface on the CardioMessenger II-S.

We did not have access to soldering equipment, so we had to hold the wires. While connected to the JTAG pins, we had full access to the microcontroller, were able to dump the Random Access Memory (RAM).

From Bour's [40] results, we knew that the key would be in the RAM memory during any encryption process. We also knew that the key size was either 168, 192, or 256 bits to match the different key sizes of AES encryption; we started with 186 bits (16 bytes). To find the correct key, we used Bour's [40] script, which is listed in Appendix A.2. The script essentially finds all potential keys by going through the RAM dump 16 bytes at a time, and stores every possible key. This is demonstrated in figure 7.28.

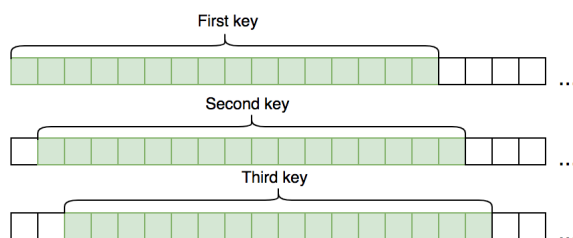


Figure 7.28: The script searches through the RAM to find every every possible AES key.

With the possible keys, we had a reduced key space, making a brute force attack easier. In addition, based on the previous analysis performed by Bour [40], we knew what the beginning of the plaintext was. Consequently, the script tries every key in the key space until the expected output is found. For additional details, refer to Bour [40].

The data turned out to be encrypted with AES encryption using a 168 bit key. The data packets only contained log data of what network operator the HMU has been connected to. This can be observed in figure 7.29.

It is reasonable to believe that actual patient data is encrypted using the same encryption scheme and key, however this is yet to be confirmed.

Finding 24

AES encryption is used for encrypting information being transmitted from the II-S. The key can be found in memory of the HMU (II-S).

We did not attempt to do the same for the other HMUs. This could be considered in further work.

```

T-Mobile UKo002,003{23430ñ{#a.*
T4APP 2.20üà
K\=$#J#].*?<zz>
T-Mobile UKo002,003{23430i£#a)r
T4APP 2.20ü@
P\=$πÜ#})r?<vv>
T-Mobile UKo002,003{23430_#a)
T4APP 2.20üü
P\=$.E#])?<r!!r>
T-Mobile UKo002,003{23430D[ #a)Ä
T4APP 2.20ü2P\=$!i#})Ä?<n##n>
T-Mobile UKo002,003{23430ü #a)J
T4APP 2.20ühP\=$"
#)J?<j%<j>
T-Mobile UKo002,003{23430Fë# a)~
T4APP 2.20ü4P\=$vô#})~?<f'f>
T-Mobile UKo002,003{23430'Ø# a)J
T4APP 2.20ü~P\=$" #)J?<sb)<b>
T-Mobile UKo002,003{23430Yó# a)
T4APP 2.20üñP\=$Bπ#})?<$^++^>
T-Mobile UKo002,003{23430âô# a)'jü
T4APP 2.20üø°Q\=$I"#)'jü?<SZ--Z>
T-Mobile UKo002,003{23430n#aèü
T4APP 2.20üò)\=$G1#]èü?<SV//V>
T-Mobile UKo002,003{23430ñb#aä™
T4APP 2.20üö)\=$] #]ä™?<R11R>
T-Mobile UKo002,003{23430è$#aäÄ
T4APP 2.20ü2ü)\=$'i#]äÄ?<SN33N>
T-Mobile UKo002,003{234305«#aÜ£
T4APP 2.20ü&t)\=$+i#]Ü£?<J55J>
T-Mobile UKo002,003{23430~£#aÑL
T4APP 2.20üf£)\=$ô#\ÑL?<SF77F>
Telekom.deo002,003{23430Rf#a~¿
T4APP 2.20üÜ.a\=$...±#\~¿?<A99A>
Telekom.deo002,003{23430ö`#a;π
T4APP 2.20ü`Ic\=$VÜ#;\;π?<$<;<>

```

Figure 7.29: Decrypted data packets from the II-S.

7.5.2 SMS

All the HMUs sent SMSs to the same phone number. Merely by looking at the UD part of the SMSs, we determined that it was not cleartext. To look for possible hints, we studied the meta data of the SMSs in Wireshark. This can be seen in figure 7.30. Recalling from section 2.1, a SUBMIT SMS has the structure as shown in figure 7.30.

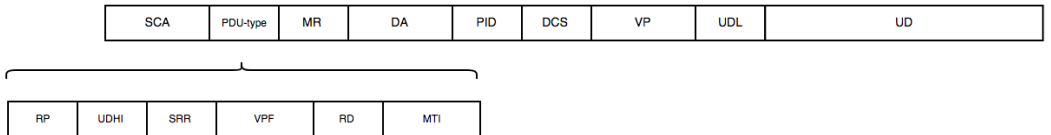


Figure 7.30: Structure of an SMS-SUBMIT message.

Based on the relevant values, we can say the following about the SMSs:

DA: +491712XXXX56. All the HMUs are sending SMSs to the same recipient number.

DCS:246

- **8 bit data:** This indicates that the UD is encoded with a user-defined coding scheme, and that the message can consist of up to 140 octets.
- **Message Class 2 (U)SIM specific message:** This specifies that the SMS should be sent directly to and stored on the SIM card of the receiver. No message reaches the phone's SMS inbox [49][50].

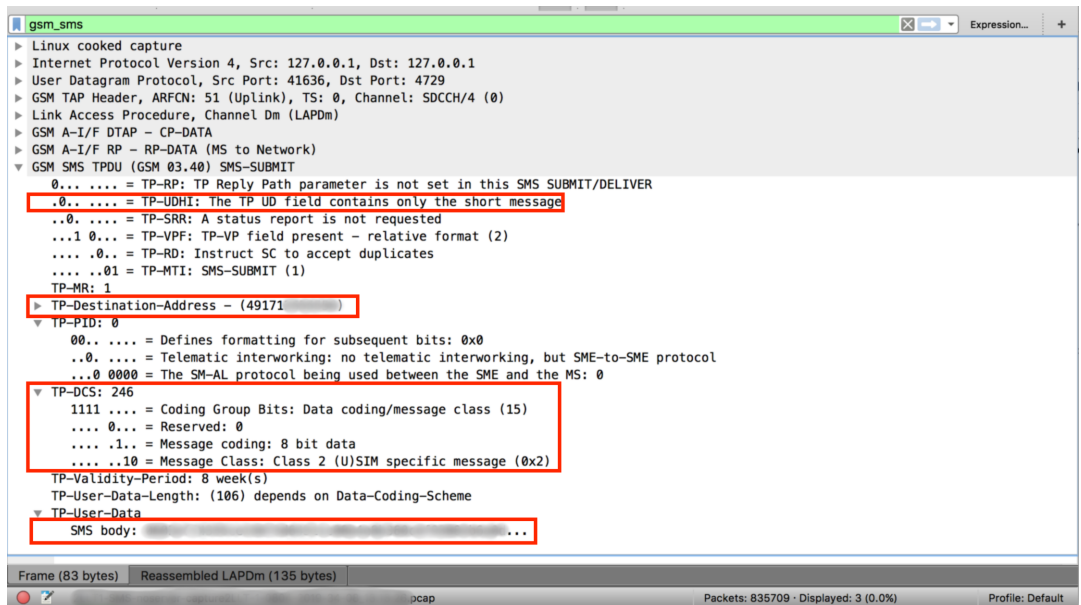


Figure 7.31: Screenshot from Wireshark showing the format of an SMS sent from the HMUs.

7.5.3 Comparing and Analyzing the SMS User Data (UD)

To facilitate the task of comparing the UD from different SMSs, we created a script that takes in pcap files and extracts the UD along with its timestamp and what HMU the SMS is from. Upon running the script we sorted the data and imported it to excel, where the comparison was performed. The script along with this procedure can be found in Appendix A.

LLT

All the UD fields had the same length, 123 bytes, and began with 0178. For every SMS sent from the LLT, with the exception of a few, there was a consistent pattern in

the UD. The pattern is presented in figure 7.32, where green represents the bytes that were identical for every SMS, orange represents bytes that were similar for several subsequent SMSs, and red represents bytes that were different for every SMS. This pattern indicates that there is no randomization in the encoding/decoding scheme. We did not observe any repeating data within the UD of the SMSs.



Figure 7.32: Structure of the SMS UD sent from the LLT.

II-S

Three different patterns were found in the SMSs from the II-S. The same patterns were found from the interaction with the real mobile network and the illegitimate BTS. The SMSs can be divided into three groups, beginning with 0601, 0604 and 0607. All SMSs within the same group had the same length and pattern, as illustrated in figure 7.33. Also, SMSs were transmitted in certain patterns, where every other message started with 0604 and then the message between either started with 0601 or 0607. This can be seen in figure 7.34.

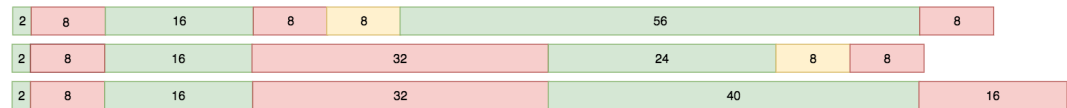


Figure 7.33: Structure of the different SMS UD observed from the II-S. From the top: 0604, 0601, 0607.



Figure 7.34: Screenshot from Excel showing the pattern of SMSs sent by the LLT.

We also observed that the UD of every SMS contained two 8-byte blocks of identical data. This suggests a block structure of 8 bytes (64 bits). The blocks were different for every SMS, but always on the same place in the UD.

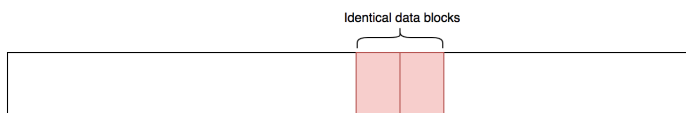


Figure 7.35: The SMSs from the II-S contained two identical 8-bit data blocks.

II-LLT

We were only able to capture three SMSs from this device. This could be due to the fact that the Illegitimate BTS was not able to provide the strongest signal in the area, and that the HMU connected to Telenor or Telia instead most of the time. The messages were very similar to the SMSs from the II-S device: They and also began with 06 and contained two consecutive identical 8-byte blocks.

II-LLT and II-S

The SMSs from both the II-LLT and the II-S begin with 06. Bour [40] discovered that this could be an indication of Data Encryption Standard (DES) encryption. The SMSs further contain two identical blocks of data. This suggests that an 8 byte (64 bits) block size is being used, which further suggests that the SMSs could have been encrypted using DES. Further, having two identical blocks narrows it down to DES encryption with Electronic Code Book (ECB) mode, if DES is in fact used. This is because ECB, as can be seen in figure 7.36 encrypts every block separately, without any randomization. Consequently, two identical plaintext blocks will give identical ciphertext blocks.

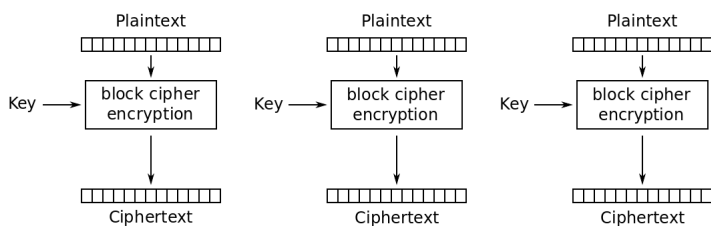


Figure 7.36: ECB mode of operation. Taken from [2].

The fact that every SMS of the same group (II-S) were so similar further indicates that no initialization vector have been used. With an initialization vector, two runs of the encryption algorithm does not produce the same ciphertext. This further strengthened the hypotehsis that it could be DES encryption ECB mode of operation.

Decrypting SMSs

Based on these assumptions, and as we already had a memory dump of the II-S RAM, we worked together with Bour [40] to make some changes to the script we described in section 7.5.1 so that it could be used to decrypt data having been encrypted with DES encryption with ECB mode. The script is listed in Appendix A.2. Using this, we were able confirm that DES ECB is in fact used to encrypt the SMSs from II-S. Due to the similarity of the SMSs from the II-LLT, there is a great chance that the same encryption scheme is used, allhtough this has not been confirmed.

The decrypted content of several SMSs can be seen in figure 7.37. Comparing this with the decrypted data packets in figure 7.29, one can observe that the exact same data is sent over both communication channels. The content of the data packet have been partitioned into several SMSs, due to the size restrictions on SMSs, hence explaining the different groups of SMSs. This confirms our hypothesis of SMS being used as a backup for when the HMU cannot establish an Internet connection.

06:01:2fa2:2e:be:dc:bd:4b:f3:57:fb:fb:1c:ef:58:a3:44:68:d7:cf:6b:96:5b:73:a7:e9:89:96:fa...	...Telekom.de@00000002,0030{023430#>~\$...
06:04:c7:2f:70:97:06:6c:1d:b0:ef:24:28:7fa3:95:4ca2:00:af:bb:dc:44:c1:70:db:92:c9:36:8d...	...T4APP 2.20•00chv\00000000=\$000_R«0-60...
06:01:74:86:d4:bc:42:23:98:e1:57:fb:fb:1c:ef:58:a3:44:68:d7:cf:6b:96:5b:73:a7:6a:e1:24:2c...	...Telekom.de@00000002,0030{023430#>~\$...
06:04:a9:40:6f:c1:06:b8:b7:38:ef:24:28:7fa3:95:4ca2:00:af:bb:dc:44:c1:70:db:24:7b:7c:7d...	...T4APP 2.20•00A@v\00000000=\$00A_R0ú03...

Figure 7.37: Screenshot from Excel, presenting parts of ciphertext vs. cleartext of SMS UD from the II-S.

Finding 25

DES encryption with ECB mode is used for securing the SMSs transmitted by the II-S and possibly the II-LLT. The key can be found in the memory of the HMU (II-S).

Finding 26

The same data is sent as payload of data packets and as User Data in SMSs (II-S).

As for SMSs from the LLT, Bour [40] did not find a match for 01 in the firmware of the II-S T-line. Seeing that the LLT is an older device, and taking the information from the technical manuals into consideration, one could argue that the security of the LLT would likely be equal or worse than for the II-LLT and the II-S. The manuals of II-LLT and II-S clearly states that the messages have been encoded, while the manual of LLT does not. One would also expect the level of security for data to match the rest of the device security, for example the use of a default PIN-code. However, these are just assumptions that should be further looked into.

7.6 Summary of Results

Table 7.11 presents the already implemented security measures, and table 7.12 presents vulnerabilities that are present in the HMUs. Reference to the security model in section 2.3. The different signs signifies the following:

Sign	Meaning
X	Yes
(X)	Likely, but not confirmed
?	We are not certain
-	Not applicable
	No

Table 7.10: Overview of what the different signs mean in table 7.11 and 7.12.

Security Measure	LLT	II-LLT	II-S	Smart 3G	Improves	In reference to finding
No data is sent in cleartext	X	X	X	X	Confidentiality	11, 15
Non-default PIN code		X	X	(X)	Confidentiality, Accountability	2
DES encryption for securing data sent in SMS		(X)	X	?	Confidentiality	25
AES encryption for securing data sent in data packets		(X)	X	?	Confidentiality	24
Private APN for communication with Data server	-	X	X	X	Confidentiality	7,8
Individual credentials for every HMU for authenticating to the APN and the Data Server	-	X	X	(X)	Identification	19
Access Control on the APN	-	X	X	X	Authenticity	13
Infrastructure data is not read from SIM card	X	X	X	(X)	Integrity, Authentication	22
Restrictions on outgoing calls and SMSs with Fixed Dialling Number (FDN)	X	X	X	(X)	Accountability	3

Table 7.11: Overview of the already implemented security measures in the HMU models.

Vulnerability	LIT	II-LIT	II-S	Smart 3G	Threat to	In reference to finding
Removable SIM card	X	X	X	X	Availability, Accountability	1
Available PIN code	X	X	X		Confidentiality	2
Valid SIM card		X		-	Accountability	3
Lack of mutual authentication (BTS)	X	X	X	X	Availability	14
Lack of mutual authentication (Data Server)	-	X	X	X	Availability	10,16,17
Hardcoded infrastructure data	X	X	X	X	Authenticity	18
Eavesdropping on the communication channel(s)	X	X	X	X	Confidentiality	15,16, 17, 18, 19,20
Accessible cryptographic key		(X)	X	?	Confidentiality	20,24,25
Receiving SMS	X	X	X	?	Availability	6,15,21
Hardcoded credentials	-	X	X		Accountability	9,19
Possible to eavesdrop on internal communication		X	X		Confidentiality	5,21
Valid APN credentials	-	X	X	?	Authenticity	9
Valid Data Server credentials	-	X	(X)	(X)	Authenticity	9
Credentials sent in clear-text	-	X	X		Confidentiality	5,11,23
Inconsistency between external behaviour and what is stated in the technical manuals	X		X	?	Safety	4
Lack of dynamic access control in the APN	-	X	X	?	Authenticity	12
The same data is sent over several communication channels with varying security measures	-	(X)	X	?	Confidentiality	26

Table 7.12: Overview of the identified vulnerabilities in the HMTU models.

Chapter 8

Discussion

In this chapter, we discuss the development of security levels in different HMU models, and consider our results in a larger context. We also demonstrate how the discovered vulnerabilities can pose threats to both patient safety and privacy if exploited by an adversary. Towards the end of the chapter we suggest potential countermeasures, and discuss the limitations of our results. Finally, we propose future research topics.

8.1 Security Progression in HMU Models

In our experiments, we have analyzed four different wireless HMU models from Biotronik, a German manufacturer. The models are of varying age, ranging from year 2003 to 2017. As described in 7.6, the HMU models implement different security mechanisms and expose various vulnerabilities. In this section we discuss how the security of the HMUs has progressed as newer models have been introduced.

8.1.1 SIM Cards

We have observed a clear improvement in terms of the PIN code and the ease of obtaining it. The LLT, which is the oldest model, uses a default PIN code that one can easily find by searching online, while newer models use non-default codes that are different for each device. While we were able to obtain the PIN codes for the II-LLT and the II-S using the methods described by Bour [40], this did not work for the Smart 3G.

Outgoing calls and SMSs are restricted by Fixed Dialing Numbers (FDN) on all the HMU SIM cards we were able to investigate, and we assume that this is also true for the Smart 3G SIM card. However, we observed a lack of restrictions on Internet access for the II-LLT. We do not know if this has been improved in newer models.

8.1.2 Vulnerabilities in GSM

The three oldest HMU models only support GSM, while the newest model (Smart 3G) also supports 3G. As we recall from Chapter 2, GSM has several known vulnerabilities that have been mitigated in the 3G standard. However, due to the backwards compatibility between 3G and GSM, we demonstrated how we could force the Smart 3G to communicate over GSM by jamming the 3G signal in a small area. Consequently, all the HMU models are affected by the vulnerabilities of GSM, and we confirmed that it was possible to trick all HMU models into connecting to an illegitimate BTS. When connected to an illegitimate BTS, the HMU is not able to transmit data to the Data Server, which could affect availability and patient safety.

These results show that even if security problems have been addressed in newer standards, and these newer standards are supported in newer models, vulnerabilities from the past are still possible to exploit.

8.1.3 Communication Channels

The three oldest HMU models (LLT, II-LLT, II-S) support communication using SMS. For these models, we performed a cipher suppression attack using the illegitimate BTS, and were able to force the HMUs to communicate over an unencrypted communication channel.

We also discovered that the II-LLT, the II-S and the smart 3G communicate with the Data Server over the Internet using TCP. Even in the oldest model (II-LLT), several security mechanisms have been implemented. This includes individual credentials per device, access control, and a private communication channel to the Data Server using a private APN. These observations indicate that Biotronik has a long-standing concern for security.

An issue that is present for all the HMU models communicating over the private APN is the lack of mutual authentication between the HMU and the Data Server. Exploiting this, we were able to impersonate the server using a private IP address and a port number that were found in the memory of the II-S. Although not confirmed, this could imply that all the HMU models connect to the same private APN. If so, an adversary that successfully intrudes the APN network by way of a vulnerability found in an older HMU, could potentially also gain access to a newer HMU.

For the II-LLT and the II-S, no security was observed beyond the use of the private APN. Consequently, when an illegitimate BTS and the spoofed server were introduced (see section 7.4.2), we were able to eavesdrop on the unencrypted channel. We observed their credentials in cleartext. This vulnerability is mitigated in the Smart 3G, and

no credentials were observed. However, we were not able to learn what security mechanisms have been implemented in the Smart 3G model.

8.1.4 Data Security

As discussed in the previous section, HMUs transmit data to the Data Server through SMSs, and through a private APN.

In the SMSs intercepted by us, no data was observed in cleartext. We were not able to determine the SMS encoding scheme used by the LLT, but it seems likely that it is using a proprietary binary format. However, we cannot exclude that some form of encryption is used.

For the II-S, we confirmed that all SMS data was encrypted before transmission. This is considered good practice as it provides confidentiality to the data. However, the encryption scheme in use, single DES, was considered broken in 1999 [51], and was withdrawn as a Federal Information Processing Standard (FIPS) in 2005 [52].

All observed data on the private APN was encrypted. The II-S uses the AES (128 bit) encryption scheme, which is considered sufficient. However, collaborating with Bour [40], we were able to locate the symmetric key in the firmware of the device. Once this key is found, it is easy to decrypt all data transmitted from the HMU to the Data Server. The II-LLT seems to use the same encryption scheme as II-S.

We were not able to identify the encryption scheme used in the Smart 3G. However, its entropy suggests something of similar security level as its predecessors. No attempt was made to dump the firmware of the device. It may be possible, given sufficient time and persistence, to find the keys in the Smart 3G unit. However, this is uncertain and more research is required.

An interesting discovery we made is that identical data was sent both as the UD in SMSs and through the APN connection. Thus, the same data was transmitted using quite different levels of security. The results of our experiments suggest that the II-S and the II-LLT use SMS as a backup transmission channel when a APN connection is not available. However, an adversary can easily deny an HMU access to the Internet, thereby making the HMUs transmit the data through SMSs a less secure manner. As the 3G does not seem to be sending SMSs at all, this vulnerability is probably eliminated.

8.1.5 Improving Security

Biotronik deserves credit for implementing basic security mechanisms in all of its wireless HMU models. We have observed that the newer models consistently include

better security mechanisms in certain areas. However, vulnerabilities remain, even in the newest models. This shows that security is an ongoing activity which probably never will be finished.

8.1.6 Lack of Routines Around Revocation of Access

As we discovered, Biotronik has essentially done a good job of distributing individual credentials to each device. They have also implemented access control to the APN, requiring both valid credentials the use of a SIM card bundled with an HMU. We confirmed this in section 7.3.4 by attempting to access the APN using valid credentials and a SIM card not bundled with an HMU. In this case, we were denied access to the APN.

This implies that Biotronik indeed has a system in place that allows them to revoke access on a per-device basis, both for the APN, the Data Server, and the mobile network (SIM cards). However, it seems that Biotronik does not have a proper decommissioning system in place to determine which HMUs are currently in use by patients and which devices that are no longer in use.

If such administrative system had been in place, Biotronik could revoke access from HMUs that are no longer in use by patients. However, one can argue that it is potentially more dangerous to mistakenly revoke the access of a legitimate user than retaining access for old devices.

8.2 Our Results in Comparison with Related Work

This section presents a comparison of our results with the related work presented in Chapter 3.

As we have demonstrated, a problem for Biotronik is the availability of used HMU units on online auction sites such as eBay. In their paper, Whitescope [9] confirmed that this is also the case for other vendors; Whitescope investigated equipment from four unnamed vendors.

Another report from Muddy Waters [10] states that this also is an issue for the merlin@home HMU from Abbott. The report also analyzes the security levels of merlin@home devices, which gives us the opportunity to compare HMUs from different vendors.

In comparison with the merlin@home device, we have confirmed that Biotronik avoids certain vulnerabilities that were discovered in the merlin@home. For example, Biotronik provides unique credentials for every HMU device where the merlin@home devices had static, shared credentials.

However, we also discovered that HMUs from Biotronik and merlin@home share some of the same vulnerabilities. These include hardcoded server information and hardcoded credentials in the firmware. These vulnerabilities were also identified in the devices analyzed by Whitescope [9].

The other attacks performed on the merlin@home, as discussed in section 3.1, are outside the scope of this thesis.

Miller et al. [29] and Munro [30] report how they were able to gain unauthorized access to wireless communication protocols used in vehicles. Our work achieved the same, but in the pacemaker ecosystem. Miller et al. [29] and Munro [30] also attempted to scan for other devices connected to the same networks. We did not try this, but their work suggest that device segregation might also be an issue in Biotronik’s private APN.

In our experiments, we have successfully constructed and configured an illegitimate BTS using COTS equipment for investigating the pacemaker ecosystem. To the best of our knowledge, no previously disclosed research has done this. Retterstøl [22] and Mruz [26] configured an illegitimate BTS using COTS equipment as part of their work, but for different purposes.

8.3 Confirmed Attack Scenarios

With reference to the threat model in Chapter 4, we were able to demonstrate how several of the suggested attack vectors could be utilized due to exposed vulnerabilities which can be seen in table 7.12. This section presents different confirmed attack scenarios on some of the HMU models.

8.3.1 An adversary can disclose data in cleartext

The first step towards disclosure of information is to eavesdrop on the communication channels. An adversary could set up an illegitimate BTS and trick the HMU into connecting to it. This is possible due to the lack of mutual authentication between the HMU and the mobile network. Using cipher suppression, the adversary could further eavesdrop on the unsecured communication channel where SMSs are being sent by the HMU to the Data Server.

We performed this attack in a laboratory, with only centimeters between the HMUs and the illegitimate BTS. However, research has shown that equivalent attacks can be performed within 2 km range with powerful antennas [37].

The adversary can also eavesdrop on the data packets that the HMU is transmitting. As the private APN restricts access to communication, the adversary would have

to work around this. Having eavesdropped on the internal communication of an old HMU device bought online, the adversary knows the hardcoded and static IP address and port number of the Data Server. Combining this with the lack of mutual authentication between the HMU and the Data Server, the adversary could configure a spoofed Data Server with the same IP address and port number as the authentic server. As such, the HMU would transmit data to the spoofed server, allowing the adversary to eavesdrop on the data packets. This attack would not require physical access to a patient or his/her HMU.

However, all transmitted data is encoded or encrypted. To fully disclose the data, the adversary would have to learn the cryptographic keys that are used. Bour [40] confirmed that the keys are stored in the firmware of the device, but that each unit uses a unique key. As such, the adversary would need physical access to the particular device belonging to a patient. This significantly complicates the attack.

In our experiments, the HMUs were not connected to pacemakers and therefore did not transmit any real patient data. Instead, the HMUs transferred simple log data which we were able to fully decrypt. It seems reasonable to assume that real patient data would be protected by the same security methods as the log data. However, this cannot be confirmed without a functional pacemaker, which we have not had access to.

8.3.2 An adversary can access the private APN

Potentially, an adversary could order a used HMUs online, and thereby get hold of a device where the SIM card still has a valid subscription. We have confirmed that this is possible.

To obtain valid credentials for the private APN, the adversary would need to eavesdrop on the communication between the micro-controller and the modem of a used HMU, where the APN credentials are found. These credentials are still valid even though the HMU is no longer used by a patient. As we uncovered in section 7.4.2, even though every HMU has unique credentials, the credentials are neither tied to the HMU nor its bundled SIM card. As such, the adversary could combine the credentials and SIM card from different HMUs. This demonstrates how the manufacturer is leaving attack vectors open in old HMUs.

The combination of the SIM card, the credentials, and the APN information, allows an adversary to access the private APN from a third-party device in which it may have root access. This opens up for a range of attack vectors.

Due to legal and ethical considerations, we did not explore the private APN except for confirming that the Data Server was present. We did so by pinging it. As such,

we do not know what an adversary could do with this type of access. We also do not know what other devices are connected to the same private APN.

8.3.3 An adversary can get free Internet Access

As some of the used HMUs contain valid SIM cards, an adversary could order used equipment online and use the SIM card for accessing the Internet for free. All the HMU SIM cards we have tested, are blocked from making calls and sending SMSs. However, once the PIN code of the SIM card had been uncovered, there were no restrictions on Internet access. The PIN code can be obtained by following the steps in [40].

This type of attack has been seen previously in other contexts. For example, the SIM cards bundled with traffic lights in South Africa were stolen and used for making calls for thousands of dollars [53].

8.4 Potential Attack Scenarios

Adversaries may also use attack scenarios which we have not explored in this thesis, but that are based on the vulnerabilities discovered in our research. In this section we briefly discuss some potential attack scenarios.

8.4.1 Massive data breach

We have shown that access to the private APN is possible where the Data Server is located. We do not know for sure what data is stored on the Data Server. In a worst-case scenario, patient records from all the HMUs that connect to the network are disclosed to the adversary.

8.4.2 Prohibit Communication on a Large Scale

Another potential attack scenario is to prohibit communication on a large scale. This attack also requires access to the private APN. The attack can be done in two ways, either by attacking the Data Server or by attacking other HMUs that are connected to the same APN:

1. Perform a DoS attack on the server, in which the Data Server is saturated with communication requests, so that it cannot respond to the traffic from the other legitimate HMUs. As a results, no data can be accessed by healthcare personnel.

2. Attack the other HMUs that are connected to the same network, for example by sending malicious SMSs or deploying malicious software to HMUs on a large scale, thereby making them inoperable.

8.4.3 Severe Patient Harm

In April 2018, the FDA issued an alert concerning premature battery depletion of certain pacemakers from the manufacturer St. Jude Medical (now Abbott) [54]. This was triggered by the death of two patients that could be linked to this failure [55]. After these incidents, patients with faulty pacemaker batteries were instructed to always have their HMU switched on to give them an early warning in case the battery started depleting while waiting for their pacemaker replacement surgery [56].

Battery depletion or other technical failures could potentially also occur in a pacemaker from Biotronik. If a technical failure of the pacemaker occurs and the HMU experiences a DoS attack, healthcare personnel will not be informed about the malfunctioning pacemaker. A patient has few ways of discovering a DoS attack. As discussed in section 7.3.1, the HMU will not indicate a DoS attack; the lights on the HMU still indicates normal operation.

The likelihood of a technical failure and a DoS attack happening at the same time may be small, but the combination is, in a worst-case scenario, fatal for the patient.

Another scenario where a patient may suffer is when an adversary compromises an HMU and takes over reporting to the Data Server. The adversary may report normal operation of the pacemaker, while in fact the pacemaker tries to alert healthcare personnel about a problem. This attack scenario requires sophisticated knowledge of the inner workings of the HMU, as well as physical access to the unit. Therefore, it seems unlikely.

8.5 Suggested Countermeasures

Based on our findings, and with reference to our RO3, we present possible countermeasures that could improve security of the artifacts, and consequently the pacemaker ecosystem as a whole.

- **Embedded SIM card (E-SIM):** An E-SIM is a built-in SIM card that provides the same services as a physical SIM card. An E-SIM would mitigate the attacks that require a SIM card from an HMU, as E-SIMs cannot be removed from a device. However, implementing E-SIM cards is not possible with existing models. Also, E-SIM is relatively new technology which may its own security vulnerabilities.

- **Decommissioning by dynamic access control:** Whenever an HMU is no longer used by a patient, a process is needed to invalidate their SIM cards and credentials.
- **Add strong encryption to the transport layer:** Use Transport Layer Security (TLS), or Secure Shell (SSH). This will prevent an attacker from eavesdropping on the communication over the TCP socket. Using TLS or SSH will require the HMUs and Data Server to use public key encryption rather than private key encryption, which is used in today's solution.
- **Mutual authentication:** Implementing TLS or SSH will also solve the issue of mutual authentication.
- **Stronger SIM card restrictions:** In addition to using Fixed Dialling Numbers (FDN) for restricting outgoing calls and SMSs, additional restrictions should be imposed to restrict Internet connections, incoming calls, and SMSs.
- **Consistent signals:** Biotronik must ensure that the lights/indication of operation are consistent with the actual status of the HMU at any given time. This way, a patient can be certain of the status of the HMU.

8.6 Limitations of the Results

While we believe that this thesis presents important findings about the security status in the wireless communication of HMUs, our work has several limitations. First, we do not know if the vendor has already addressed the vulnerabilities we have discovered. There may exist firmware updates that have addressed some of the identified security issues. However, we do not know of any such updates.

Second, we did not have access to any patient data, only log data. Consequently, we cannot prove that the same security mechanisms apply to patient data. Third, some of the HMU models we tested are old, and may not be in use by patients any longer.

8.7 Further work

The work described in this thesis has uncovered security issues in HMUs. Within the scope of our research objectives, there are still questions which should be investigated.

- **Repeat the experiments with patient data:** In our experiments, we only had access to log data, which we were able to decrypt. It would be interesting to perform the same experiments, only with actual patient data, to observe whether the same security schemes are used. This would require access to a functioning pacemaker.

- **Further investigate the Smart 3G:** Due to time restrictions, we did not investigate the Smart 3G as thoroughly as we wanted. This is the newest HMU model from Biotronik, and a full analysis of its security mechanisms would be interesting to perform. The Smart 3G is also the first HMU model with a USB interface, which could potentially open up for new attack vectors.
- **Regulations and guidelines:** To fully determine whether the security of the HMU models are sufficient, they should be evaluated against current regulations and guidelines. For example, the EU enforced the General Data Protection Regulation (GDPR) in May of 2018 [57]. The EU has also adopted two new regulation regarding medical equipment, the 2017/745 [58] and the 2017/746 [59], that will be put into effect in 2020 and 2022 respectively.

Also, the FDA provides pre- and postmarket management of Cybersecurity in Medical Devices [60] [61]. In Norway, the *Code of conduct for information security and data protection in the healthcare and care services sector* [62] is a recognized guideline.

- **Firmware updates:** As mentioned, we have not been able to determine if and how the HMUs are updated. In their study, Whitescope [9] identified the capability of remote firmware updates of HMUs from several (unknown) vendors. If this is also the case for Biotronik, this could potentially be a new attack vector.

Chapter 9

Conclusion

In this thesis, we have performed a security analysis of the proprietary communication protocol between HMUs and a Data Server in the Biotronik pacemaker ecosystem. Our initial research questions were:

***RQ1.** How is patient data secured in transit between the HMU and the data server?*

***RQ2.** How are the communication channels between the HMU and the Data Server secured?*

We used the Design Science methodology to structure our work, and included methods such as threat modelling, black-box security analysis and reverse engineering in a trial-and-error manner. We configured an illegitimate BTS using COTS equipment, and used it to investigate and attack different HMU models.

Regarding RQ.1, our research has shown that all data is encrypted or encoded. For the HMU models where we have confirmed that encryption is used, the identified encryption schemes are based on symmetric key cryptography of varying security levels. We found that AES was used for data packets and single DES was used for SMS User Data. As we have not tried to decrypt patient data, we cannot confirm that the same schemes are used for patient data, but it seems plausible.

We are not certain which encryption/encoding schemes are used by the oldest and newest HMUs, if any. For the oldest unit, it seems that the User Data is not encrypted but still coded in a binary format, which we have not been able to decode. For the most recent unit, we believe encryption is used, but have not been able to establish which scheme is being used.

As for RQ.2, two different communication channels were observed. In the communication channel where SMSs are transmitted, Fixed Dialling Number (FDN) restricts the outgoing calls and SMSs from the HMUs. For the communication channel using the private APN, access control is used, only allowing certain SIM cards to access

the network. In addition, valid credentials must be provided. The credentials are unique for every HMU device.

Our results suggest that Biotronik has made efforts to implement basic security mechanisms in all their HMU models and that the overall security is improving in newer models.

However, our results also imply that these security mechanisms are insufficient. Based on disclosed vulnerabilities, we have demonstrated attacks that could potentially compromise the safety and privacy of patients. Also, even if the newer devices are better secured, the older models are still being used by patients. In addition, decommissioned HMUs are available to purchase online. These are still given access to the private APN and the Data Server, and could potentially affect the security of all HMUs that try to connect to the Data Server.

We believe that the number and scope of Internet-connected health devices will continue to grow in the foreseeable future. Close attention should be paid to the security design of these devices to ensure the safety and privacy of their many users. As such, this area will remain an important research topic in the years to come.

References

- [1] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.
- [2] Block cipher mode of operation. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.
- [3] Mobile Country Codes (MCC) and Mobile Network Codes (MNC). <https://www.mcc-mnc.com>. [Online; Accessed 13-June-2019].
- [4] Nummerplan: E.212, Navn: Mobil nettverkskode (MNC). <https://www.nkom.no/npt/numsys/E.212.pdf>, 06 2019. [Online; Accessed 13-June-2019].
- [5] Frekvensar til offentleg mobilkommunikasjon.
- [6] U.S. Department of Homeland Security. STRATEGIC PRINCIPLES FOR SECURING THE INTERNET OF THINGS (IoT, Version 1.0). 11 2016. [Online; Accessed 12-May-2019].
- [7] Pacemaker or ICD: Which Do I Need? <https://www.webmd.com/heart/pacemaker-or-icd-which-do-i-need#3-8>. [Online; Accessed 25-September-2018].
- [8] Laurent Castellucci. FDA approves Biotronik Home Monitoring System pacemaker. <https://www.medscape.com/viewarticle/786432>, 10 2001. [Online; Accessed 11-May-2019].
- [9] Security Evaluation of the Implantable Cardiac Device Ecosystem Architecture and Implementation Interdependencies. <https://www.ledecodur.ch/wp-content/uploads/2017/05/Pacemaker-Ecosystem-Evaluation.pdf>. [Online; Accessed 04-September-2018].
- [10] Carson C. Block. MW is Short St. Jude Medical (STJ:US). 2016.
- [11] Carl D. Livitt. St. judes witness report. https://medsec.com/stj_expert_witness_report.pdf. [Online; Accessed 23-August-2018].
- [12] Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, Bart Preneel. On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them. *Proceedings of the 32nd Annual Conference on Computer Security Applications*, page 226–236, 2016.

- [13] Heimar Lecht. 2G and 3G networks are shutting down globally?! <https://1ot.mobi/blog/2g-and-3g-networks-are-shutting-down-globally>, 03 2018. [Online; Accessed 19-January-2019].
- [14] Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40). Specification, ETSI, July 1996. [Online; Accessed 14-April-2019].
- [15] Vocabulary for 3GPP Specifications V.15. Specification, 3GPP, July 2018. [Online; Accessed 16-June-2019].
- [16] <http://www.mcc-mnc.com>. [Online; Accessed 21-January-2019].
- [17] ETSI TS 122 101. Specification, ETSI, March 2009. [Online; Accessed 03-June-2019].
- [18] Suraj Srinivas. The GSM Standard (An overview of its security), 2001.
- [19] ETSI. Digital cellular telecommunications system (Phase 2+ Security-related network functions (3GPP TS 03.20 version 8.6.0 Release 1999). 2008.
- [20] Digi. Digi Connect Application Guide Cellular IP Connections (Uncovered) . http://ftp1.digi.com/support/documentation/appguide_connectcellar_ipconsiderations.pdf, 07 2005. [Online; Accessed 12-May-2019].
- [21] Kerckhoffs's principle. <http://www.crypto-it.net/eng/theory/kerckhoffs.html>. [Online; Accessed 19-march-2019].
- [22] Torjus B. Retterstøl. Base Station Security Experiments Using USRP. Master's thesis, NTNU Trondheim, 2015.
- [23] Herberg J. Mattord Michael E. Whitman. *Management of Information Security*. CENGAGE Learning, Boston, Massachusetts, 2016.
- [24] Safety. <https://www.dictionary.com/browse/safety>. [Online; accessed 28-march-2019].
- [25] Eivind S. Kristiansen, Anders B. Wilhelmsen. Security Testing of The Pacemaker Ecosystem. Master's thesis, NTNU Trondheim, 2018.
- [26] Andre Mruz. Mobile Network Security Experiments With USRP. Master's thesis, NTNU Trondheim, 2016.
- [27] Per Anders Johansen, Andreas Bakke Foss. Stortinget og statsministeren OVERVÅKES. [Online; Accessed 17-June-2019].
- [28] Mesud Hadžialić, Mirko Škrbić, Kemal Huseinović, Irvin Kočan, Jasmin Mušović, Alisa Hebibović and Lamija Kasumagić. An Approach to Analyze Security of GSM Network. Telfor 2014, 2014. [Online; Accessed 01-November-2018].
- [29] Dr. Charlie Miller, Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. October 2015.

- [30] Ken Munro. Hacking IoT vendors smart cars via private APNs. <https://www.pentestpartners.com/security-blog/hacking-iot-vendors-smart-cars-via-private-apns/>, 10 2017. [Online; Accessed 9-May-2019].
- [31] Dave Bourgeois, David T. Bourgeois. What Is an Information System. <https://bus206.pressbooks.com/chapter/chapter-1/#footnote-5-1>. [Online; Accessed 24-April-2019].
- [32] Alan R. Hevner, Salvatore T. March, Jinsoo Park, Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly Vol. 28 No. 1, pp. 75-105/March 2004*, 2004.
- [33] Shane Argo. Attack Surface Analysis Cheat Sheet. https://www.owasp.org/index.php?title=Attack_Surface_Analysis_Cheat_Sheet&oldid=156006, 2013.
- [34] Elliot Chikofsky, James H. Cross. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software 0740-7459/90/0100/0013, 1990.*, pages 13–17, 1990.
- [35] Eduard Marin. *Security and Privacy of Implantable Medical Devices*. PhD thesis, Arenberg Doctoral School, March 2018. [Online; Accessed 14-April-2019].
- [36] National Cybersecurity Centre. Coordinated vulnerability disclosure: The guideline. 2018.
- [37] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, Jean-Pierre. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. [Online; Accessed 01-November-2018].
- [38] SDR-Radio. <https://www.sdr-radio.com>.
- [39] ETSI. Digital cellular telecommunications system (Phase 2+ (GSM 07.07)); AT command set for GSM Mobile Equipment (ME) (GSM 07.07). July 1996. [Online; Accessed 4-April-2019].
- [40] Guillaume Nicholas Bour. Security Analysis of the Pacemaker Home Monitoring Unit: A BlackBox Approach. Master's thesis, NTNU Trondheim, May 2019.
- [41] Searchable FCC ID Database. <https://fccid.io>.
- [42] Telekom. <https://www.telekom.com/en>. [Online; Accessed 14-February-2019].
- [43] BIOTRONIK, Berlin, Germany. *Technical Manual CardioMessenger-LLT*, 2008.
- [44] BIOTRONIK, Berlin, Germany. *CardioMessenger II-LLT*, revision J edition, 3 2013.
- [45] BIOTRONIK, Berlin, Germany. *Technical Manual CardioMessenger-®II-S, revision F*, 2 2013.
- [46] BIOTRONIK, Berlin, Germany. *Technical Manual CardioMessenger® Smart, revision H*, 3 2016.

- [47] Michael Curtis. How to Retrieve a T-Mobile Pin. <https://itstillworks.com/retrieve-tmobile-pin-7671093.html>. [Online; Accessed 07-January-2019].
- [48] [https://gchq.github.io/CyberChef/#recipe=Entropy\(\)](https://gchq.github.io/CyberChef/#recipe=Entropy()).
- [49] Data Coding Scheme. <https://www.openmarket.com/docs/Content/apis/v3smpp/smpp-data-coding.htm>. [Online; Accessed 12-April-2019].
- [50] 3GPP TS 23.038 V15.0.0 (Technical Specification Group Core Network and Terminals; Alphabets and language-specific information (Release 15)). Specification, 3GPP, June 2018.
- [51] Paul Van De Zande. The day DES died. <http://target0.be/madchat/crypto/papers/paper722.pdf>, 07 2001. [Online; Accessed 10-June-2019].
- [52] NIST. Federal Register / Vol. 70, No. 96 / Thursday, May 19, 2005 / Notices. <https://web.archive.org/web/20080625202735/http://csrc.nist.gov/publications/fips/05-9945-DES-Withdrawl.pdf>, 05 2005. [Online; Accessed 1-June-2019].
- [53] South Africa thieves hit traffic lights for Sim cards. <https://www.bbc.com/news/world-africa-12135841>, 2011.
- [54] FDA. Battery Performance Alert and Cybersecurity Firmware Updates for Certain Abbott (formerly St. Jude Medical) Implantable Cardiac Devices: FDA Safety Communication. <https://www.fda.gov/medical-devices/safety-communications/battery-performance-alert-and-cybersecurity-firmware-updates-certain-abbott-formerly-st-jude-medical>, April 2018. [Online; Accessed 30-May-2019].
- [55] Jim Finkle, Ransdell Pierson. Two deaths spark recall of St. Jude heart devices. <https://www.reuters.com/article/us-st-jude-medical-batteries/two-deaths-spark-recall-of-st-jude-heart-devices-idUSKCN12B199>. [Online; Accessed 30-May-2019].
- [56] Jeff Fecho. Premature Battery Depletion with Implantable Cardioverter Defibrillator. <https://www.cardiovascular.abbott/content/dam/bss/divisionalsites/cv/pdf/reports/BatteryDepletion-Doctor-Letter-11Oct2016-US.pdf>, 10 2016. [Online; Accessed 1-June-2019].
- [57] General Data Protection Regulation. <https://gdpr-info.eu>. [Online; Accessed 17-February-2019].
- [58] REGULATION (EU) 2017/745 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0745&from=EN>. [Online; Accessed 03-March-2019].
- [59] REGULATION (EU) 2017/746 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0746&from=EN>. [Online; Accessed 03-March-2019].

- [60] Content of Premarket Submissions for Management of Cybersecurity in Medical Devices. <https://www.fda.gov/media/119933/download>. [Online; Accessed 19-February-2019].
- [61] Postmarket Management of Cybersecurity in Medical Devices. <https://www.fda.gov/media/95862/download>. [Online; Accessed 19-February-2019].
- [62] Direktoratet for e-helse. Code of conduct for information security and data protection in the healthcare and care services sector, v.5.3. [Online; Accessed 03-March-2019].

Appendix

Scripts

In this appendix, we list the source code used to extract User Data from SMSs (A.1) and to decrypt data using AES and DES (A.2).

A.1 SMS Data Extraction

```
use POSIX qw(strftime);

foreach $file ('ls *pcap') { #Replace pcap with the name of the
    correct folder containing the desired pcap files, or replace
    *pcap with the name of a single file
    chop $file;
    foreach $line ('tshark -V -R gsm_sms -2 -r $file -T pdml |
        egrep \"gsm_sms.sms_body|frame.time_epoch\"') {
        $_ = $line;
        if (/field name=\"frame.time_epoch\"/) {
            if (/show=\"([\\d\\.]+)\"/) {
                $timestamp = $1;
                $datetime = strftime \"%Y-%m-%d,%H:%M:%S\",
                    localtime($timestamp);
            }
        } elsif (/field name=\"gsm_sms.sms_body\"/) {
            if (/show=\"([0-9a-f:]+)\"/) {
                $data = $1;
                $start = substr $data, 0, 2;
                if ($start eq \"06\") { $box = \"2-S\" }
                elsif ($start eq \"01\") { $box = \"LLT\" }
                else { $box = \"unknown\" }
                print \"$box,$datetime,$data,$file\\n\";
                $datetime=\"\"; $timestamp=\"\"; $data=\"\";
            }
        }
    }
}
```

}

Listing A.1: Script for extracting relevant information from SMS TPDU's

A.2 AES and DES Decryption

```
#!/usr/bin/env python3

"""
    File name: cm-decrypt.py
    Version: 1.1
    Author: Guillaume Bour
    Last modified: 2018/06/05
    License: GNU General Public License v3.0
    Description: A script to decrypt the data sent by the HMU to
                 the backend server, given an AES key. Can also be used
                 to
                 brute force the AES key given a binary dump of either
                 the
                 RAM or the flash memory.
    Changelog:
        1.1: Add support for SMS decryption (basic and
             brute force).
"""

import os
import sys
import time
import argparse
import struct
import math
import zlib

from stat import *
from Crypto.Cipher import AES
from Crypto.Cipher import DES

NAME = "CM_DECRYPT"
VERSION = "1.1"

GZIP_MAGIC_HEADER = "1f8b"
ESC_ELT = 0x10

ENC_DES = 6
ENC_3DES_CBC = 7
```

```

ENC_AES_CBC = 8

def print_file_info(filename):
    print("**_{v}_{**}\n".format(NAME, VERSION))
    print("[*]Opening_{...}".format(filename))

    with open(filename, "rb") as bin_file:
        print("\n--_FILE_INFORMATION_--\n")
        try:
            st = os.stat(filename)
        except IOError:
            print("Failed_to_get_information_about_{...}".format(
                filename))
            sys.exit(-1)
        else:
            print("File_size:_{bytes}_{hex}_{:02X}".format(st[
                ST_SIZE], st[ST_SIZE]))
            print("File_created:_{...}".format(time.asctime(time.
                localtime(st[ST_MTIME]))))
            print("File_modified:_{...}".format(time.asctime(time.
                localtime(st[ST_MTIME]))))
            print("File_entropy:_{bits_per_byte}\n".format(H(
                bin_file.read(), True)))

def sanitize(bin_data):
    binary = b""
    skip = False
    for b in bin_data:
        if not skip and b == ESC_ELT:
            skip = True
        else:
            binary += bytes([b])
            skip = False

    # Fix weird bug
    binary = bytes.fromhex(binary.hex().replace('aa01', 'aa'))
    return binary

def getKeys(file_name, keySize = 16):
    key = b"\x00" * keySize
    with open(file_name, "rb") as bin_file:
        next_byte = bin_file.read(1)

```

```

        while next_byte != "":
            key = key[1:] + next_byte
            next_byte = bin_file.read(1)
            yield key

# Adapted from http://blog.dkbza.org/2007/05/scanning-data-for-entropy-anomalies.html
def H(data, round_r = False):
    if not data:
        return 0
    entropy = 0
    for x in range(256):
        p_x = float(data.count(x))/len(data)
        if p_x > 0:
            entropy += - p_x*math.log(p_x, 2)
    if round_r:
        return round(entropy, 2)
    return entropy

def aes_decrypt(data, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return cipher.decrypt(data)

def des_decrypt(data, key):
    cipher = DES.new(key, DES.MODE_ECB)
    return cipher.decrypt(data)

def parse_message_layer(data):
    if not QUIET:
        print("\n**_Message_Layer**\n")

    msgs = zlib.decompress(data, zlib.MAX_WBITS|32)
    h = H(msgs, True)
    msgs = msgs.hex()

    print("Size_of_the_recovered_data:_{}}_bytes".format(len(msgs)))
    print("Number_of_packets:_{}}".format(len(msgs.split("0a"))))
    print("Entropy:_{}}".format(h))

    print(msgs)

```

```

def parse_compression_layer(data):
    if not QUIET:
        print("\n**_Compression_Layer_**\n")

    type_p = struct.unpack(">B", data[0:1])[0]
    magic_header = struct.unpack(">H", data[1:3])[0]
    payload = data[1:]

    if not type_p == 9:
        print("[ERROR]_Not_a_compression_layer_packet!")
        sys.exit(-1)

    if not QUIET:
        is_mh = "_=>_gzip_compressed_data,_from_Unix" if "{}".format(
            GZIP_MAGIC_HEADER) in payload.hex() else ""
        print("Compression_packet:_Yes")
        print("Magic_header:_0x{:02x}_{}".format(magic_header, is_mh))
        print("Entropy:_{}".format(H(payload, True)))

    parse_message_layer(payload)

def parse_encryption_layer(data, key):
    if not QUIET:
        print("\n**_Encryption_Layer_**\n")

    type_p = struct.unpack(">B", data[0:1])[0]
    padding = struct.unpack(">B", data[1:2])[0]
    iv = data[2:18]
    r_data = data[18:]

    type_txt = ""
    if type_p == ENC_AES_CBC:
        type_txt = "AES_CBC"
    elif type_p == ENC_3DES_CBC:
        type_txt = "3DES_CBC"
    elif type_p == ENC_DES:
        type_txt = "DES"
    else:
        print("[ERROR]_Encryption_type_not_known:_{}".format(
            type_p))
        sys.exit(-1)

```

```

if not QUIET:
    div = "Yes" if len(data) % 16 == 0 else "No"
    print("Length of the packet: {} (hex: {:02x}) (div by 16? {})".format(len(data), len(data), div))
    print("Type of packet: {} (hex: {:02x}) => {}".format(type_p, type_p, type_txt))
    print("Padding: {} (hex: {:02x})".format(padding, padding))
    print("IV: {}".format(iv.hex()))
    print("Key: {}\n".format(key.hex()))

if type_p == 8:
    blocks = ( len(r_data) // 16 ) * 16
    data = aes_decrypt(r_data[:blocks], key, iv)
    parse_compression_layer(data)
else:
    print("[ERROR] {}: encryption type not implemented".format(type_txt))
    sys.exit(-1)

def parse_transport_layer(bin_content, key, get_payload = False):
    :
    if not QUIET:
        print("** Transport Layer **\n")

    type_p = struct.unpack(">B", bin_content[0:1])[0]
    length = struct.unpack(">H", bin_content[1:3])[0]
    unknown = struct.unpack(">B", bin_content[3:4])[0]
    p_id = struct.unpack(">H", bin_content[4:6])[0]
    cm_id = struct.unpack(">I", bin_content[6:10])[0]
    checksum = struct.unpack(">H", bin_content[len(bin_content)-2:])[0]

    if not QUIET:
        print("Type of packets: {} (hex: {:02x})".format(type_p, type_p))
        print("Length: {} bytes (hex: {:04x})".format(length, length))
        print("Unknown: {} (hex: {:02x})".format(unknown, unknown))
        print("Packet ID: {} (hex: {:04x})".format(p_id, p_id))
        print("CM ID: {} (hex: {:08x})".format(cm_id, cm_id))

```

```

        print("Checksum: {} (hex: {})".format(checksum,
        checksum))

payload = bin_content[10:len(bin_content)-2]

if get_payload:
    return payload
else:
    parse_encryption_layer(payload, key)

def bruteforce_aes(data, filename_bin):
    if not QUIET:
        print("\n[*] Brute forcing the AES key...")

    type_p = struct.unpack(">B", data[0:1])[0]
    iv = data[2:18]
    r_data = data[18:]
    blocks = ( len(r_data) // 16 ) * 16
    c = 0

    if type_p != ENC_AES_CBC:
        print("[ERROR] Encryption type not recognized or not implemented {}".format(type_p))
        sys.exit(-1)

    if not QUIET:
        with open(filename_bin, "rb") as file_b:
            print("[*] {} keys to try".format(len(file_b.read())
            ))

    for key in getKeys(filename_bin):
        if len(key) != 16:
            return None
        d = aes_decrypt(r_data[:blocks], key, iv)

        if "{}{}{}".format("09", GZIP_MAGIC_HEADER, "0800") in d
        .hex():
            return (key, c - 15)

    if not QUIET:
        c += 1
        if c % 10000 == 0:
            print(".", end="")
            sys.stdout.flush()

```

```

    return None

def bruteforce_des_sms(data, filename_bin):
    if not QUIET:
        print("\n[*] Brute forcing the DES key (SMS)...")

    type_p = struct.unpack(">B", data[0:1])[0]
    r_data = data[2:]
    blocks = ( len(r_data) // 8 ) * 8
    c = 0

    min_ent = 8
    min_ent_key = None
    addr = 0

    if type_p != ENC_DES:
        print("[ERROR] Encryption type not recognized or not implemented({})".format(type_p))
        sys.exit(-1)

    if not QUIET:
        with open(filename_bin, "rb") as file_b:
            print("[*]{} keys to try".format(len(file_b.read())
            ))

    for key in getKeys(filename_bin, 8):
        if len(key) != 8:
            return (min_ent_key, addr - 7)

        d = des_decrypt(r_data[:blocks], key)
        h = H(d)

        if h < min_ent:
            min_ent = h
            min_ent_key = key
            addr = c

        if not QUIET:
            c += 1
            if c % 10000 == 0:
                print(".", end="")
                sys.stdout.flush()
    return (min_ent_key, addr - 7)

```



```

def bruteforce_key(data, filename_bin, sms=False):
    if not QUIET:
        sms_txt = "(SMS)" if sms else ""
        print("[*] Brute_force_mode{}".format(sms_txt))
        print("[*] Binary: {} \n".format(filename_bin))

    res = None

    if not sms:
        data = parse_transport_layer(data, None, True)

    start = time.time()
    res = bruteforce_des_sms(data, filename_bin) if sms else
        bruteforce_aes(data, filename_bin)
    end = time.time()
    t = int((end - start) * 100) / 100

    if res:
        (key, addr) = res
        print("\n[*] Key_Found_in {}s! \n".format(t))
        print("Key: {}".format(key.hex()))
        print("Addr: 0x{:08x}".format(addr))
    else:
        print("\nKey_not_found! Time_elapsed: {}s".format(t))

def decrypt_data(data, key):
    if not QUIET:
        print("[*] Decrypt_mode")
    parse_transport_layer(data, key)

def decrypt_data_sms(data, key):
    if not QUIET:
        print("[*] Decrypt_mode (SMS)")

    r_data = data[2:]
    blocks = ( len(r_data) // 8 ) * 8
    d = des_decrypt(r_data[:blocks], key)

    if not QUIET:
        print("[*] Decrypted_data:")
    print(d.hex())

```

```

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Decrypt a file
        sent by the Biotronik HMU or break the key used to
        encrypt the data.")
    parser.add_argument("filename", type=str, help="Filename of
        the binary data to be decrypted.")
    parser.add_argument("-q", "--quiet", action="store_true",
        help="Only prints the result.")
    parser.add_argument("-n", "--no-sanitize", action="
        store_true", help="Do not sanitize the data before
        attempting decryption (useful for the GSM version.")
    parser.add_argument("-s", "--sms", action="store_true", help
        ="Data is SMS.")
    group = parser.add_mutually_exclusive_group(required=True)
    group.add_argument("-k", "--key", type=str, help="The key to
        decrypt the data (in hex).")
    group.add_argument("-b", "--binary", type=str, help="The
        binary file used to brute force the key.")
    args = parser.parse_args()

    QUIET = args.quiet

    if not QUIET:
        print_file_info(args.filename)

    with open(args.filename, "rb") as bin_file:
        data = bin_file.read()

        if not args.no_sanitize:
            data = sanitize(data)
            if not QUIET:
                print("[*] Data sanitized!")

        if args.key:
            if args.sms:
                decrypt_data_sms(data, bytes.fromhex(args.key))
            else:
                decrypt_data(data, bytes.fromhex(args.key))
        else:
            bruteforce_key(data, args.binary, args.sms)

```

Listing A.2: Script for decrypting data that has been encrypted using either AES or DES.

Appendix B

Shared Folder

B.1 Creating a Shared Folder

This appendix presents the steps for creating a shared folder between the Host machine and the Virtual Machine (VM) (Ubuntu 14.04).

Procedure:

1. Go to <http://download.virtualbox.org/virtualbox/> and download guest addition corresponding to the running version of Virtualbox.
2. Open VirtualBox and select the VM you want to create a shared folder with
3. Open the settings of the VM. Click on "Storage" and click the "add optical drive" symbol that is indicates in figure B.1. Press "Choose disk".

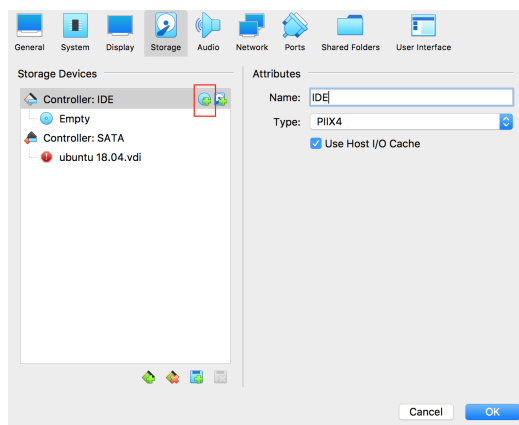


Figure B.1: Screenshot of VirtualBox, illustrating how to add a new optical drive.

4. Select the file that was downloaded in step 1. The disk will now appear under "Controller: IDE". This can be seen in figure B.2.

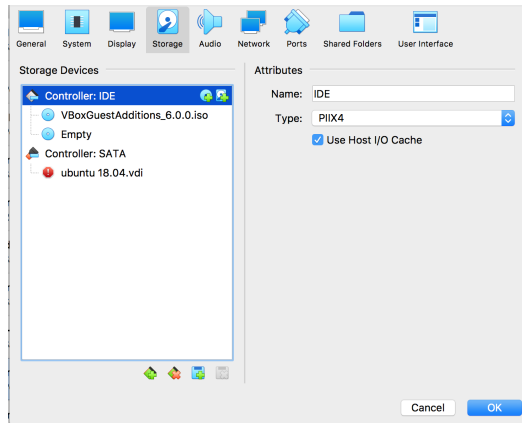


Figure B.2: Screenshot of VirtualBox, illustrating what it looks like when the Guest Addition has been added.

5. In the settings, click on "Shared Folders" -> click on the "Add new shared folder" symbol, as indicated in figure B.3.

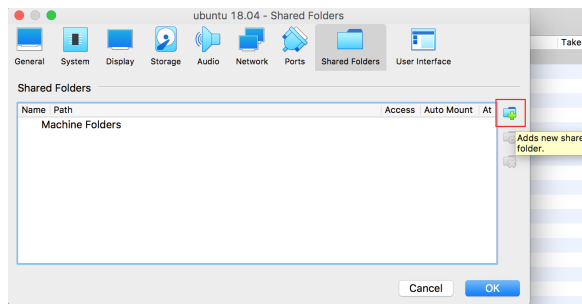


Figure B.3: Screenshot of VirtualBox, illustrating how to add a shared folder.

6. Select the folder that is to be shared with the VM.
7. Start the VM.
8. Run the following commands

```
$ sudo mount /dev/cdrom /media/cdrom
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo /media/cdrom/./VBoxLinuxAdditions.run
```

9. Restart the VM.

10. Create a shared directory

```
$ mkdir ~/[name of directory]
```

11. Mount the shared folder from the Host machine

```
$ sudo mount -t vboxsf [name of shared folder] ~/[name of directory]
```

