Network Security Monitoring in Environments where Digital & Physical Safety are Critical

Guillaume Dupont

PhD Thesis

Network Security Monitoring in Environments where Digital and Physical Safety are Critical

Guillaume Dupont

Network Security Monitoring in Environments where Digital and Physical Safety are Critical

THESIS

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van
de rector magnificus prof. dr. ir. F.P.T. Baaijens,
voor een commissie aangewezen door het College
voor Promoties, in het openbaar te verdedigen op
dinsdag 7 juni 2022 om 11:00 uur

door

Guillaume Dupont

geboren te Lagny sur Marne, France

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter:	prof. dr. J.J. Lukkien
Promotor:	prof. dr. S. Etalle
Copromotor:	dr. J.I. den Hartog
Leden:	prof. P. Samarati (Università degli Studi di Milano)
	prof. dr. M.G.J. van den Brand
	prof. dr. M. Petkovic

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Network Security Monitoring in Environments where Digital and Physical Safety are Critical
by Guillaume Dupont

Printed by: Grefo Prepress - Eindhoven
Cover design: Grefo Prepress - Eindhoven
Cover art: Pheasant couple and plum blossom (1900-1930)
by Ohara Koson (1877-1945), Tokyo, Japan
http://hdl.handle.net/10934/RM0001.COLLECT.359274

This work in the thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics)

This work has been partially funded by the ITEA3 under the APPSTACLE project (15017)

This work has been partially funded by the ECSEL joint undertaking SECREDAS (783119-2)

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-5516-1

An electronic version of this dissertation is available at https://research.tue.nl/

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Acknowledgments

Three years ago I had the chance to go on a pilgrimage on the Camino de Santiago, where I walked from France to Spain. I cannot think of a better analogy to describe my experience as PhD candidate. Similarly to the Camino, writing this thesis has been a long and arduous journey. It takes a lot of effort, discipline and will-power to complete. Getting up early, packing, walking the ±25km to the next *albergue* (pilgrims' hostel), and repeat. Everyday, whatever the weather, one has to keep going. It can be painful at times, when blisters or a twisted ankle add an extra challenge. The pain can also be mental and emotional, with phases of fatigue, discouragement and dispiritedness. However, none of these difficulties overweight the beauty of the Camino. It is one of the most rewarding and meaningful journey one can undertake, with profound learnings and discoveries. It opens new windows and shed new light on oneself and the world. Among many things, it is the people encountered who make the Camino such a marvellous human adventure. Like the Camino, this PhD has been both a fantastic and challenging journey. Finishing it was only possible thank to a number of persons I had the chance to meet alongside the road. I am deeply grateful for all the opportunities, the help and the support they offered me throughout this journey, and I would like to thank them below.

First of all, I would like to express my profound gratitude to the committee members for taking part in my defence and for their instrumental role in my graduation. I thank prof. Pierangela Samarati, prof. Mark van den Brand and prof. Milan Petkovic for accepting to be part of the defence committee, and for all the time and effort they spent reviewing this thesis and giving their feedback. I thank prof. Johan Lukkien for chairing the committee, and Boris Skoric for his support.

Grazie mille (a thousand thanks) to prof. Sandro Etalle and Elisa Costante for giving me the chance to pursue a PhD program. Nothing would have been possible without you two. I will never forget our first meeting on my birthday in 2017, when an interview for a position at Security Matters turned into a PhD offer. Thank you for this immense gift and for seeing the potential in me. Thank you, Sandro, for enabling me to become a researcher, and for teaching me one of the most important things: to focus. (If I had received one Euro each time you told me to focus, I would be rich.) Thank you, Elisa, for all your guidance, the projects, the opportunities and the responsibilities you offered me. Thank you for all the times you championed my research and helped me bridging the gap between industry and academia. I would also like to extend my deepest gratitude to Jerry den Hartog. Thank you, Jerry, for giving me this invaluable academic training in which you taught me how to write clearly and concisely, and how to think logically and scientifically. I cannot thank you enough for all your help and the directions you gave me to conduct my research, and for the countless hours spent on my work. None of this thesis and papers would have been possible without you.

This PhD program would not have been successful without the help and generosity of Damanio Bolzoni and Emmanuele Zambon. Thank you very much for welcoming me in your company and providing me with such an amazing work environment and research opportunities. I had the chance there to work with Daniel dos Santos and Alexios Lekidis on exciting projects. Many thanks Daniel for being such a role model and mentor. I learnt a great deal by looking up to you, from performing research to managing myself and others. Thank you for being a kind, warm-hearted and caring person, and for having been by my side when the path was steep and rocky. Thanks a lot Alexios for the good times spent working together on automotive security, at project meetings across Europe, and on stage presenting our work.

Additionally, I am thankful for all the persons who contributed to making these five years a joyful and fulfilling work experience, both at the university and at Forescout/Security Matters. I would like to thank particularly the research team at Forescout for having me. Thank you so much Alessandro, Quasim, Stanislav, and all the interns I have had the chance to meet. Special thanks to Sylvio and Mario, who have been with me (almost) from the start. Thank you for all the unforgettable memories together, for celebrating our successes and accomplishments, and for supporting each other throughout our respective journeys. Many thanks as well to all the colleagues at Forescout for making the office such an enjoyable place. It was a delight to come to work every day and being greeted by friendly faces. Thank you all for the great discussions and the fun after-works and celebrations. I also want to thank Luca for choosing me as a teaching assistant. Thank you for your trust and all the opportunities you gave me, it was a privilege to be part of this adventure. I thank all the members of the security group of the TU/e, and all the other PhD students for sharing their research and interests. Particularly, I would like to thank Cristoffer who took the load off my shoulders when I could no longer bare the weight. Thank you, Cristoffer, for your tremendous support and kindness. I gratefully acknowledge the assistance and help of Mirjam, Marieke and Anjolein with all administrative matters and for the organisation of memorable events with all the colleagues. I am also thankful for the great work of Gino and his colleagues from Grefo Prepress who designed and printed this thesis.

I thank from the bottom of my heart all my family members for their immense support and nurturing, and for helping me in so many ways to pursue my goals. A big thank you to my siblings for always believing in my endeavours and motivating me in the face of adversity. I cannot thank my parents enough for everything they did for me. Thank you so much Mom and Dad for raising me the way you did, for the loving and caring environment you provided for my growth, for encouraging me to pursue my interests and passions, and for investing in me and my education without hesitation. Many thanks to my family-in-law for all their tremendous help, support and kindness, and particularly to my parents-in-law who, in their infinite generosity, provided me with everything I needed to start writing this thesis. I also thank my grandparents, for nurturing my curiosity, opening my mind and instilling in me much of their hard-earned wisdom. Finally, I would like to thank my wife, without whom the completion of this PhD would not have happened. No words or actions would ever be enough to express how grateful and thankful I am for your dedication and committent. Thank you so much for all the love and kindness you poured at me every single day, and for the relentless support and untiring care you gave me. Thank you for "being

in this together" and for always staying by my side, no matter how early the morning or late the night. Thank you for being my light and shelter; with you there are no tunnels too dark, and no storms too strong. Thank you for giving me the meaning and purpose I need to keep on walking. Thank you for sharing and celebrating all the successes, big and small, from the first paper published to this very thesis. This is not *my* success, but *ours*. You are truly the best, my everything.

I would like to seize the opportunity to also thank some important people who shaped my life and enabled me to become who and where I am today. First, I thank my first grade teacher, who taught me how to read and write, as well as my teachers from the subsequent years up to university, who equipped me with the skills required to access higher education. I thank all these teachers who planted seeds in me, broadened my horizons, and fostered a passion for learning and a thirst for knowledge. I was fortunate to have access to a solid education, and I am privileged to have received foundational reading, numeracy and ICT skills, among others. I do not take those skills for granted, as there are around the world still too many children, young men and women, especially, who are not given these skills upon which their future success in our digitalised and globalised world depends. Additionally, I would like to thank the inspirational teachers who saw potential in me and motivated me to seek out and pursue opportunities. In particular, I am forever thankful for Thierry who encouraged me to pursue my university education when I wanted to stop. I thank as well my internship supervisors who gave me the experience required to secure a good position after my Master. Thank you Julien and Twan for being inspiring professionals and rich human beings. I thank my first manager Pepijn who gave me the golden (or orange) ticket to the Netherlands. Thank you for this great career head start and for this unforgettable adventure in the Low Lands. I thank my new manager Rahul for his trust and for giving me a fantastic opportunity after the PhD to apply my skills and knowledge to make the world a safer place. Finally, I would like to express my sincere and profound gratitude to the European Union tax payers who contributed to my salary and enabled me to live from my research. Many thanks as well to the coffee farmers from Africa and South America who fuelled my work and helped me stay alert and engaged every day (it is estimated that 3,650 cups of coffee were *required* to complete this PhD).

Reflecting on these past years, many people gave me a generous and helping hand when I could not climb alone; I owe them where I stand today. I will never forget, and will commit myself to giving back to the best of my abilities.

I remember on the Camino a fellow Pilgrim who, as we bid each other farewell, said: "When you arrive in Santiago de Compostela, this is not the end. The Camino continues, all your life. In fact, your life *is* the Camino. So keep walking, and *¡buen Camino!*" Today, this PhD program may be over, this is the end of an adventure, and the begin of a new one. The road goes ever on. Let's keep walking.

¡Buen Camino!

The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say.

J.R.R. Tolkien

Summary

The digitalisation of our society has profoundly transformed our economy and the technological landscape, leading to an unprecedented dependency on networked systems. Modern environments are comprised of a broad diversity of devices connected together as ecosystems, spreading across space (decentralised and distributed architectures) and time (varying device lifecycle). In this context, some of these environments stand out due to the types of devices they have and the kind of data they process. Cyber-physical systems and IoT devices are deployed, integrating the digital world with the physical world, and large volumes of users' (personal) data are collected, processed and stored. While these technologies and operations yield many benefits, they also introduce new threats to these environments and, to a greater extend, to our society. Cyber attacks on these specific environments can have critical impact on the *safeness* of their users, i.e., their physical safety and digital privacy, through the exploitation of vulnerabilities in cyber-physical systems and the abuse of (personal) data. In this thesis, we focus on such environments we call *safeness-critical*.

Protecting safeness-critical environments is both challenging and essential. It requires to be *in control* of the environment and its assets: one must have a thorough understanding of the infrastructure, as well as the ability to detect and respond to threats and intrusions. Network security monitoring, a strategy focused on visibility, can help to obtain this control. It is a well-established concept in the IT world where a number of solutions such as intrusion detection systems have been developed based on security principles fitting the specificities of that domain. However, it is unclear how the current network security monitoring solutions perform in the context of safeness-critical environments and their unique specificities. This leads us to consider the question: *To what extent can we perform network security monitoring in safeness-critical environments?*

To answer this question we select and investigate two environments that can be regarded as safeness-critical: healthcare delivery organisations (e.g., hospitals) and modern cars. Recent security research and real-life incidents have demonstrated the risks to their users' safeness, calling for effective security measures to protect their networks and devices (and ultimately their users). In this context, the objective of this thesis is to identify the requirements for network security monitoring and intrusion detection, and to provide the means to create and evaluate intrusion detection systems for these environments.

Our investigations, organised into two distinct studies of the aforementioned environments, comprise the following methodology. In both studies we begin with the analysis of the environment's technological landscape and related threats. This first phase sheds light on the technologies and security challenges, and it helps us to identify gaps with regards to current network security monitoring capabilities. We then investigate the limitations of the monitoring solutions applied in the environment. Finally, we develop novel security mon-

itoring approaches suited for the specificities of the environment, as well as frameworks to evaluate them. Our work yields a number of contributions, including a device classification method addressing some of the limitations of traditional approaches, the identification of vulnerabilities in healthcare network protocols, a taxonomy of intrusion detection systems for automotive networks, as well as a unified framework for the evaluation of these systems.

Our results enable us to identify three main requirements for the application of network security monitoring in safeness-critical environments in a broader context. First, one has to obtain a clear picture of the environment's infrastructure and assets. Second, one needs the ability to "understand" the protocols and devices, including their related threats. Third, one requires the ability to evaluate network security monitoring tools such as intrusion detection systems in a unified manner. Having these requirements fulfilled represents a starting point from which effective network monitoring capabilities can be designed to better protect the environment and its users from cyber threats. How well these requirements are fulfilled largely determines the extend to which network security monitoring is possible and thus answers our research question.

Samenvatting

De digitalisering van onze maatschappij heeft onze economie en het technologische landschap grondig veranderd, wat heeft geleid tot een ongekende afhankelijkheid van verbonden systemen. Moderne omgevingen bestaan uit een grote verscheidenheid aan apparaten die als ecosystemen met elkaar verbonden zijn en verspreid zijn in plaats (gedecentraliseerde en gedistribueerde architecturen) en tijd (variërende levenscyclus van apparaten). In deze context onderscheiden sommige van deze omgevingen zich door de soorten apparaten die ze hebben en het soort gegevens dat ze verwerken: De inzet van cyber-fysieke systemen en IoT apparaten integreert de digitale wereld met de fysieke wereld, en grote hoeveelheden (persoonlijke) gegevens van gebruikers worden verzameld, verwerkt en opgeslagen. Hoewel deze technologieën en operaties veel voordelen opleveren, introduceren ze ook nieuwe bedreigingen voor deze omgevingen en, in bredere zin, voor onze maatschappij. Cyberaanvallen op deze specifieke omgevingen kunnen een kritieke impact hebben op de '*safeness*' van hun gebruikers, d.w.z. hun fysieke veiligheid en digitale privacy, door misbruik te maken van kwetsbaarheden in cyber-fysieke systemen en het misbruik van (persoons) gegevens. In dit proefschrift richten we ons op dergelijke omgevingen die we '*safeness-critical*' noemen.

Het beschermen van safeness-critical omgevingen is zowel uitdagend als essentieel. Het vereist dat men de controle heeft over de omgeving en de systemen er in: men moet een grondig begrip hebben van de infrastructuur, evenals het vermogen om bedreigingen en indringers te detecteren en erop te reageren. Netwerkbeveiligingsmonitoring ('network security monitoring'), een strategie gericht op zichtbaarheid, kan helpen om deze controle te verkrijgen. Het is een beproefd concept in de IT-wereld waar een aantal oplossingen, zoals inbraakdetectiesystemen, zijn ontwikkeld op basis van beveiligingsprincipes die passen bij de specifieke kenmerken van dat domein. Het is echter onduidelijk hoe de huidige oplossingen voor netwerkbeveiligingsmonitoring presteren in de context van safeness-critical en hun specifieke kenmerken. Dit brengt ons bij de vraag: *in hoeverre kunnen we netwerkbeveiligingsmonitoring uitvoeren in safeness-critical omgevingen?*

Om deze vraag te beantwoorden selecteren en onderzoeken we twee omgevingen die als safeness-critical kunnen worden beschouwd: zorgorganisaties (bijvoorbeeld ziekenhuizen) en moderne auto's. Recent beveiligingsonderzoek en real-life incidenten hebben de risico's voor de safeness van hun gebruikers aangetoond en vragen om effectieve beveiligingsmaatregelen om hun netwerken en apparaten (en uiteindelijk hun gebruikers) te beschermen. In deze context is het doel van dit proefschrift om de vereisten voor netwerkbeveiligingsmonitoring en inbraakdetectie te identificeren en de middelen te bieden om inbraakdetectiesystemen voor deze omgevingen te creëren en te evalueren.

Ons onderzoek is verdeeld in twee studies van de bovengenoemde respectievelijke omgevingen met de volgende methodologie. In beide studies beginnen we met de analyse van het

technologische landschap van de omgeving en gerelateerde bedreigingen. Deze eerste fase werpt licht op de technologieën en beveiligingsuitdagingen en helpt ons om hiaten te identificeren met betrekking tot de huidige mogelijkheden voor netwerkbeveiligingsmonitoring. Vervolgens onderzoeken we de beperkingen van de monitoringoplossingen die in de omgeving worden toegepast. Ten slotte ontwikkelen we nieuwe beveiligingsmonitoringbenaderingen die geschikt zijn voor de specifieke kenmerken van de omgeving, evenals raamwerken om deze te evalueren. Ons werk levert een aantal bijdragen op, waaronder een apparaat-classificatie methode die enkele van de beperkingen van traditionele benaderingen aanpakt, de identificatie van kwetsbaarheden in netwerkprotocollen voor de gezondheidszorg, een taxonomie van inbraakdetectiesystemen voor autonetwerken, evenals een uniform raamwerk voor de evaluatie van deze systemen.

Onze resultaten stellen ons in staat om drie belangrijke vereisten te identificeren voor de toepassing van netwerkbeveiligingsmonitoring binnen safeness-critical omgevingen in een bredere zin. Ten eerste moet men een duidelijk beeld verkrijgen van de infrastructuur en systemen van de omgeving. Ten tweede moet men in staat zijn om de protocollen en apparaten te "begrijpen", inclusief de bijbehorende bedreigingen. Ten derde is een uniforme manier voor het evalueren netwerkbeveiligingsmonitoringtools zoals inbraakdetectiesystemen vereist. Het voldoen aan deze vereisten vormt een startpunt van waaruit effectieve netwerkbewakingsmogelijkheden kunnen worden ontworpen om de omgeving en zijn gebruikers beter te beschermen tegen cyberdreigingen. Hoe goed aan deze eisen wordt voldaan, bepaalt voor een groot deel in hoeverre netwerkbeveiligingsmonitoring mogelijk is en beantwoordt daarmee onze onderzoeksvraag.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ARP	Address Resolution Protocol
BAS	Building Automation System
BYOD	Bring Your Own Device
CAN	Controller Area Network
CIA	Confidentiality, Integrity and Availability
CMS	Central Monitoring System
CPS	Cyber-Physical System
CT	Computed Tomography (scan)
CVE	Common Vulnerabilities and Exposure
CVI	Clustering Validation Index
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DICOM	Digital Imaging and Communications in Medicine
DMS	Data Management System
DoS	Denial of Service
ECU	Electronic Controller Unit
EHR	Electronic Health Record
FTP	File Transfer Protocol
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
HDO	Healthcare Delivery Organisation
HL7	Health Life Seven
HTTPS	Hypertext Transfer Protocol Secure
HVAC	Heating, Ventilation and Air Conditioning
ICS	Industrial Control System
ICT	Information and Communication Technologies
IDS	Intrusion Detection System
IoMT	Internet of Medical Things
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LIN	Local Interconnect Network
LIS	Laboratory Information System
MAC	Media Access Control
MITM	Man-In-The-Middle

ML	Machine Learning
MOST	Media-Oriented System Transport
MRQ	Main Research Question
NIDS	Network Intrusion Detection System
NSM	Network Security Monitoring
NTP	Network Time Protocol
OBD-II	On-Board Diagnostics II
OS	Operating System
OT	Operational Technology
PACS	Picture Archiving and Communication System
PHI	Protected Health Information
PII	Personally Identifiable Information
POCT	Point-Of-Care Testing
RDP	Remote Desktop Protocol
RQ	Research Question
SCE	Safeness-Critical Environment
SIEM	Security Information and Event Management
SMB	Server Message Block
SNMP	Simple Network Management Protocol
SOC	Security Operation Center
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
V2X	Vehicle-to-Everything
VLAN	Virtual Local Area Network
VoIP	Voice over IP

À ma mère. Comme promis.

1

Introduction

The rapid spread of new technologies is transforming our lives and societies at unprecedented speed and scale [239]. The digitalisation of our economy profoundly altered many economic and social activities, leading to a dependency on these technologies [258]. In this brave new world powered by advanced robotics, Artificial Intelligence (AI), the Internet of Things (IoT) and cloud computing, the rhythms and operations of modern life are regulated by Internet-connected devices [239]. These systems integrate the cyber world with the physical world [176], while providing seamless connectivity between heterogeneous networks [210]. They are reshaping the digital landscape in numerous ways, making it drastically different than before when it consisted of Information and Communication Technologies (ICT) devices (e.g., computers, servers and IP phones). Computers and smartphones are nowadays being outnumbered by other connected devices, and the gap is forecasted to keep on widening over time [137, 188]. Our ever-increasing reliance on connected systems makes their presence around us inevitable: they monitor our buildings, regulate traffic in cities, deliver energy to our homes, sustain patients' life in hospitals, and automate our factories [210].

While the digitalisation of our society offers tremendous advancements and opportunities, it also introduces new risks to our cyber security, as well as to our physical safety and data privacy [17, 258]. The explosion of connectivity, fuelled by the adoption of new technologies and the ever-larger digital footprint of many organisations, have introduced weaknesses in infrastructures [86, 266]. Criminals have adapted their "physical" *modus operandi* to exploit these vulnerabilities in the digital world [245, 252] and found opportunities to monetise their cyber crimes [7]. They generate profits in multiple ways, for example by reselling stolen data and ransoming systems. The growth and maturity of online black markets facilitate their trading activities of goods and services, allowing them to barter malware or zero-day exploits, and to propose services like bulletproof hosting or hacking-as-a-service [9, 247]. The cyber black market has evolved into a network of highly organised groups, often linked to traditional crime entities (e.g., drug cartels, terrorist cells) and nation-states [4]. In fact, cyber crimes are frequently associated with nation-states' military activities (e.g., [74]), as traditional international conflicts have shifted into the cyber space [90]. In order to maximise damage and financial gain, cybercriminals are increasingly attacking critical infrastructures (e.g., Colonial pipeline [181]) and health-

care institutions (e.g., Irish Health Service [100]) with malware such as ransomware, or attacks like Distributed Denial Of Service (DDoS) [20, 222]. The introduction of IoT and embedded devices has also opened new doors for cybercriminals [17]. These systems are notorious to have weak security mechanisms and to be often left unmanaged on Internet-facing networks [72]. They can be found with online services like Shodan[1] and accessed remotely [8]. Once compromised, they can be used in numerous ways, for example, as stepping stones to infiltrate networks [51, 91, 249]. Additionally, they can be leveraged to spy on people [81, 89, 209] or to launch massive distributed cyber attacks on core network infrastructures [122, 125].

In this current digital landscape, cyber attacks can impact various levels of our society, from entire countries to companies, down to single individuals. Consequences of attacks can range from disruptions to total shutdown of business operations, and loss of critical information [114]. As observed recently with the Irish Health Service, a ransomware outbreak in critical infrastructures can cripple a nation for weeks [21, 52]. Such examples also illustrate that cyber attacks can nowadays significantly impact our physical safety and digital security. In fact, with the deployment of Cyber-Physical Systems (CPS) and IoT devices connecting virtual and physical world, cyber-physical attacks [136] targeting these systems can adversely affect physical space and people's safety [16, 135]. Such devices have sensing and actuating faculties, used to monitor and control the physical world with minimal human interaction [111, 187]. Examples of cyber-physical attacks have been demonstrated on cars [31, 159] and medical devices [99, 193, 195], highlighting the critical impact they can have on the health and safety of the user(s). Our digital security is also at stake as cyber attacks often result in abuse of data [54]. Data has become the lifeblood of our modern economy [259] and many decisions impacting us are made based on documents stored and processed on electronic systems, such as payroll, social security records and medical files [112, 245]. Managing and protecting data are challenging tasks for organisations [191], and mistakes can create opportunities for cybercriminals to abuse this data [247]. Perpetrators behind data breaches steal and monetise information by, for instance, reselling it on black markets [20], or by extorting targets with the threat of releasing sensitive private documents [222]. Companies, as well as individuals, can be heavily impacted by data leaks [13, 191]. In this context where the functioning of our economy and society relies on modern technologies and networked devices, it is imperative to address the risks to our *safeness*, i.e., our physical safety and digital security.

Safeness-critical environments

We introduce the term *safeness-critical* to qualify environments where cyber attacks can have critical impact on the *safeness* of the users, i.e., their physical safety and digital security. The term *users* here covers anyone interacting with the environment's CPS with physical proximity (e.g., the driver of a car) or anyone having his or her personal (sensitive) data being collected, processed and stored on the networked systems of the environment (e.g., a patient of a clinic). We cover below the need for a new term, and the characteristics of safeness-critical environments.

[1] `https://www.shodan.io`

The issues of devices' (physical) safety has been well studied in the literature and usually relates to "safety-critical" systems [94, 96]. This term traditionally refers to devices whose malfunction or failure can lead injury or loss of life, as well as harm to other equipment or to the environment [267]. However, this definition does not entirely fit our context since the term "safety" covers topics such as reliability or fault-tolerance [58, 255] while we, in contrast, focus on cyber security.

The term "critical infrastructure" has been introduced to refer to "those systems and assets (both physical and cyber) so vital [to a country] that their incapacity or destruction would have a debilitating impact on national security, national economic security, and/or national public health and safety" [165]. So far, the security research conducted on such infrastructures focused on the measures and technologies deployed to guarantee their robustness and resilience [14]. This term does not fit our context very well either, since it lacks the privacy issues discussed above.

As neither of these terms is sufficient to fully describe the environments we mentioned previously, we use in this thesis the term "safeness-critical environment" (SCE) as the intersection between safety-critical and privacy-critical environment. This term captures the nature of modern infrastructures where cyber security is a matter of protecting the safeness of their users.

Characteristics

SCE have specific characteristics which, together, set them apart from other environments.

- **C1 - Diversity of devices:** SCE are comprised of a broad range of devices that are diverse in multiple ways. Their type and the function(s) they deliver can vary greatly, depending on their usage or the industry for which they are developed. Examples of devices found in SCE networks range from operational technology (OT) devices (e.g., building automation systems), IoT (e.g., camera, smoke alarm), industry specific IoT such as Internet of Medical Thing (IoMT, e.g., infusion pump), CPS (e.g., robot arm), to the more traditional information technology (IT) devices (e.g., computer, server). These systems are also diverse in terms of technology and performance, embedding various hardware and software components (e.g., operating system, network protocol stack(s)), which can be proprietary and/or open source.
- **C2 - Ecosystem of devices:** The devices in SCE work together as an ecosystem [148]. Despite their differences mentioned above, they are connected and orchestrated to operate and deliver value to their users. Technological advances enable certain devices to interact autonomously with each others through machine-to-machine communications, removing the need for human intervention. This results in a broad ensemble of interconnected devices, some of which being operated manually while others function autonomously, and they communicate together over diverse network protocols.
- **C3 - Devices spread in space and time:** Devices in SCE can be dispersed across large geographical areas which can span over multiple countries, even when they are operated by the same entity [32]. Some of these devices can be reached from anywhere

in the world, enabling remote workers to access company servers and resources, or third-party suppliers to perform maintenance operations. Additionally, devices in SCE can span over a relatively long time frame. Brand new, cutting-edge devices often coexist alongside older systems which can remain functioning until break down.

- **C4 - Safeness critical:** As mentioned above, cyber attacks on SCE can impact the safeness of users. Adversaries can exploit vulnerabilities and tamper the functioning of CPS to physically arm users. Due to the high sensitivity of data collected and processed, it makes SCE a target of choice for criminals.

Security challenges

The characteristics of SCE bring certain security challenges. The nature of these environments calls for security needs that go beyond those of "traditional" ICT environments or critical infrastructures. Similarly to the latter, SCE rely on security principles aiming at guaranteeing business continuity and protecting the confidentiality, integrity and availability (CIA) of digital and physical assets. However, it is unclear whether these established principles are sufficient by themselves to address the security requirements of SCE. Moreover, the characteristics of SCE listed above bring certain complications with regard to cyber security : the diversity of devices, their heterogeneity, and their varying longevity bring together challenges with regards to 1) inventory & patch management, 2) security update & patching, and 3) blurred boundaries in infrastructures.

- **Inventory & patch management:** One cannot easily maintain an inventory of all the devices connected at a given time, leading to problems regarding patch management and vulnerability identification [83, 247]. Security tools traditionally used for device detection have difficulties coping with newer IoT devices due to specificities in their design. Consequently, various systems are scattered throughout the network with associated risks unbeknownst to the network manager [7].
- **Security update & patching:** Devices are commonly supported by their manufacturer for a limited amount of time, and no security patches will be issued afterwards. Yet, customers of these devices are not inclined to decommission and replace unsupported systems that can still operate. Consequently, it is common to see devices outliving their "support lifespan" and running outdated software for long period of time. The result is that, combined with the inventory issue, long forgotten devices can remain connected and operating on SCE networks, while being exposed and vulnerable to attacks.
- **Blurred boundaries in infrastructures:** The architectures of SCE can lack clear network boundaries, which contrasts sharply with the traditional IT network architecture where devices were physically located on site. Cyber security principles were initially developed with the idea that everything on premise could be trusted and ought to be protected from the untrusted outside. However, the adoption of new technology such as cloud computing changed this paradigm and blurred the trust boundaries. Moreover, certain principles like Bring Your Own Device (BYOD) or guest Wi-Fi networks can allow unmanaged (and potentially malicious) devices to access SCE networks and resources, with all the security concerns that brings.

1.1. Scope and Motivation

In this thesis, we investigate two environments that present characteristics of SCE: Healthcare Delivery Organisations (HDO) (e.g., hospitals, clinics) and modern cars. Both have undergone much technological changes over time, making them drastically different than what they used to be. Recent security research and real-life incidents in these environments highlight the risks to safeness for their users, and therefore the urgent need for appropriate security measures.

The first SCE that we investigate are HDO. Healthcare organisations have a broad range of devices connected to their networks, consisting in a combination of IT and OT systems such as computers and building automation, as well as connected medical devices (IoMT) delivering treatment and care (e.g., infusion pump), tracking patients' vitals (e.g., patient monitor), and performing analysis on samples (e.g., blood analyser). All of those devices digitalise patients' health information and send them to various systems in the hospital. HDO collect, store and share highly sensitive data such as Personally Identifiable Information (PII) and Protected Health Information (PHI). This data can be accessed by external third parties such as general practitioners or insurance companies. In addition, HDO have a rather "open nature" from a physical security point of view: patients are coming in and out, family relatives and friends visit them, and technicians come to maintain or replace devices. It is also common practice for hospital staff to share devices and workstations. Moreover, Internet access is commonly provided via open Wi-Fi networks to guests and patients for entertainment.

The second SCE we research is modern cars. Every year millions of vehicles are sold around the world, embedding multiple CPS and systems responsible for performing various operations such as accelerating, breaking, and steering. [242]. In the automotive context, these systems are called electronic controller units (ECU). Modern vehicles consist of more than 100 ECU communicating with each other over automotive-specific protocols, forming multiple networks within the same car [120]. Additionally, cars are increasingly interconnected with the outside world [138]: the adoption of technologies such as Vehicle-to-Everything (V2X) communications enable data exchange with other vehicles, infrastructure (e.g., road sign units and traffic lights) and even pedestrians. There is currently an effort from manufacturers to have all their new vehicles connected to the Internet [241]. Modern cars also collect up to terabytes of users' data every year [120]. Manufacturers can collect it for remote diagnostic and monitoring, while insurance companies can use this information to reconstruct driver behaviour to evaluate, for example, how eco-friendly is the driving [84]. Personal data is also stored internally, including current and past locations, phone call history and contacts information such as addresses, emails and photos among other things [82, 107].

We consider HDO and modern cars to be good examples of SCE as they share the four main characteristics discussed previously. First, they both are complex environments, consisting of a plethora of systems, including CPS, using diverse technologies (proprietary and open source) and communicating with each other over various networks and protocols (C1 & C2). Second, while they were considered closed environments in the past, they are nowadays increasingly interconnected with the outside world, offering multiple entry

points to access them (C3). Third, the presence of CPS and the vast amount of sensitive data collected can lead to threats to users' safeness (C4).

Recent research and incidents have highlighted the cyber security risks in HDO and cars with regards to users' safeness. Cyber attacks on HDO have been increasing in number and sophistication [105]. These attacks are primarily in the form of ransomware in which the perpetrators prevent the usage of workstations until the ransom is paid [140]. Hospitals under attack see all their operations put to a halt, affecting patients' health as they suffer from delay in receiving care [100]. Recent examples include a woman who died after being unable to receive emergency treatment due to an ongoing ransomware attack in a German hospital [93]. Missed diagnoses can also occur, with doctors being unable to access previous scans to assess the evolution of patients' condition [100]. Next to ransomware attacks, cybercriminals have been observed recently to focus on exfiltration of sensitive data [20]: they monetise the theft by selling patients' and doctors' files on black markets or by threatening to publicize the data unless ransom is paid. Spotting and stopping illegal activities in HDO network is challenging; breaches span on average over 329 days from initial compromise to remediation [191]. While the reported attacks on HDO seem to mostly target the IT part of the network, medical devices themselves could also become an attack vector. A growing body of security research conducted on IoMT demonstrate the risks of cyber-physical attacks for patients (e.g., [11, 124, 198, 232, 262]). Examples include the ability to manipulate the amount of drug being delivered by infusion pumps [1], or the interception and modification of patients' medical data such as real-time vitals [151] and even CT scans [161].

Modern cars have also been increasingly targeted by cyber attacks [241]. The technologies and connectivity embedded into modern vehicles result in a large threat surface, exposing them to risks to driver's and passengers' safeness [242, 246]. Previous research have demonstrated the feasibility of exploiting multiple entry points from long- or short-range distances to gain access to cars, whether to physically enter the car or virtually access the network to attack it further [31, 158, 197]. The motivations behind the incidents reported over the last 10 years can be categorised into three main groups: 1) car systems control, 2) car theft, and 3) data theft [242].

As the name suggests, the first category relates to the physical control of certain systems within the car such as the engine or brakes. Researchers have demonstrated their ability to take over the control of certain vehicles at speed, enabling them to perform actions such as steering or braking [159, 177]. Entire fleets of cars can also be exposed to multi-vehicle remote control. Some manufacturers use command-and-control servers to communicate with their fleets, send remote commands and collect diagnostic information [242]. Vulnerabilities on such servers can allow an attacker to control any cars in the fleet, as demonstrated with Tesla [128].

The second category covers car theft, which occurrences are rising around the world [251]. Due to weak cryptographic algorithms in key fobs, criminals can perform a keyless entry relay attack which tricks the targeted vehicle to believe the legitimate key is close by, enabling them to open it and drive away [219]. They can then resell the stolen car, parts of it, or ask for a ransom [241]. Research has shown how pervasive this problem was among most car makers [6].

The third category relates to data theft, being of growing interest for cybercriminals targeting the automotive sector. According to the FBI, the large amount of data collected by Internet-connected vehicles is highly valued [87]. Privacy issues in modern cars are twofold. First, as cars store private information internally, it is a concern in the context of the sharing economy [120]. This data can be exposed in car rentals and car-sharing gigs where individuals rent out their own vehicle to strangers [29, 82]. Second, manufacturers retrieve and store this data on their corporate network [23]. The vast amount of data collected can be used to infer sensitive information, including drivers' identity [84]. In case of a breach, customers' private information ends up exposed [241].

From the discussion above, we see that it is imperative to protect SCE such as HDO and cars from cyber attacks in order to protect users' safeness. Safeguarding these SCE requires to be *in control* of the environment and assets. In this context, being in control means two things. First, it means to have a thorough awareness of the environment/infrastructure to protect. This awareness translates into the capability to know what is on the network(s) and what is going on, at any given point in time. Second, being in control means to have the ability to detect and respond to threats. Depending on the events being observed on the network(s), one can take specific actions and apply corrective measures in order to mitigate threats and handle security incidents according to procedures and policies in place. The concept of *Network Security Monitoring* (NSM) helps to strive toward the two aforementioned capabilities, and therefore to be in control.

NSM has been defined by Bejtlich in [22] as "the collection, the analysis and the escalation of indication and warnings to detect and respond to intrusions". According to the author, NSM is a strategy backed by tactics focused on visibility, instead of being a blocking, filtering or denying technology. To be effective, NSM requires to have both extensive visibility into networks, and the ability to collect and analyse intelligence related to the intent and capability of threat actors [200]. Meeting these two requirements enables to shift from *seeing* to *understanding* what is going on. The term "seeing" refers in our context to the ability to collect (relevant) observables on the network in order to obtain a clear picture of the environment at hand. "Seeing" allows us to answer questions such as "What is on the network?" (e.g., devices, software, ...) , "How are devices communicating?", or even "What device is exposed to the Internet?". However, seeing the picture is one thing; understanding it is another. The term "understanding" here is the ability to incorporate contextual information and meaning to this picture. It allows us to answer questions such as "What are the risks introduced by this device/technology?", "Is this behaviour legitimate?", or "Is this communication flow allowed?". Ultimately, it provides the means to spot illegal actions and detect intrusions. Having this understanding enables actionability, where appropriate and effective decisions can be taken based on the answers to those aforementioned questions.

NSM is a concept well-established in the IT world, where its implementation commonly relies on a set of technologies (e.g., Security Information and Event Management, or SIEM), people and procedures (e.g., Security Operation Center, or SOC). A number of solutions have been developed based on security principles fitting the specificities of that domain. However, it is unclear how the current NSM solutions, approaches and paradigms can perform in the context of SCE. Three observations already come to mind. First, the "old" IT-based security paradigm, based on ad-hoc solutions developed and deployed specifically

for each system, cannot scale with the diversity of devices in SCE. Second, our current security models based on separating a trusted "inside" from an untrusted "outside" is no longer applicable as this distinction blurs away. Finally, since the goal of IT security is to guarantee the confidentiality, the integrity and the availability of systems and data, the solutions aiming at fulfilling this goal may not be enough to also guarantee users' safeness in SCE.

This situation leads us to question the applicability of NSM in SCE. With the goal of shifting from *seeing* to *understanding*, two questions already emerge: How much do we actually see? And how much do we understand? It is yet unclear how big is the gap to be filled. In an attempt to protect SCE from cyber attacks, we need to identify and address this gap.

1.2. Objective & Research Questions

In this thesis, we create a basis to perform NSM in SCE and show how to provide methods to bridge this gap. The goal of this work is to analyse the feasibility of NSM (i.e., how much can we see and understand) and, where possible, enable effective monitoring in HDO and modern cars. More precisely, our objective is to identify the requirements for NSM and intrusion detection, and provide means to create and evaluate intrusion detection systems (IDS) for these SCE.

We articulate our efforts around the following main research question (MRQ):

> **MRQ:** *To what extent can we perform network security monitoring in safeness-critical environments?*

In order to answer this question, there are two main requirements to be fulfilled. First, we need to have a clear picture of the current situation in a given SCE with regards to cyber security. More precisely, it is crucial to start by observing and learning about the technological landscape of an SCE and its related threats. Second, we have to identify the gaps with regards to NSM. It requires to investigate the potential limitations of current solutions and the impediments to perform effective monitoring in a given SCE. After having these two requirements met, we can then move on to conducting further research to bridge the gaps and enhance the monitoring capabilities of SCE.

Fulfilling these two requirements necessitates data. For this reason we collaborate with key industrial partners, enabling us to access the information and resources needed. This allows us to conduct two distinct studies on SCE, namely HDO and modern cars, helping us to answer the MRQ. To further clarify the research direction, we divide the MRQ into two sets of three concrete sub-questions. The first set of questions (RQ 1.1, 1.2 and 1.3) covers healthcare security, while the second set (RQ 2.1, 2.2 and 2.3) relates to automotive network security. In both studies, the first research questions RQ 1.1 and RQ 2.1 are used to identify our starting points. Answering them fulfils the two aforementioned requirements and sheds light on the technologies and security challenges of these SCE. From these starting points we can address the remaining research questions. We answer them by proposing and developing novel approaches suited for the specificities of HDO and cars. These sub-questions give a coherent structure to our work, and we detail them below.

1.2.1. Healthcare security

Starting with the healthcare sector, we investigate the current status of network security in HDO. As discussed previously, there has been an increase in the number and sophistication of cyber attacks targeting HDO in recent years. It is not well understood how exposed are these organisations. Hence our first research question to be answered is:

RQ 1.1: *What is the technical state of readiness of HDO in the face of the current threat landscape?*

To answer RQ 1.1, we conduct several studies on HDO networks. The key findings indicate security gaps such as usage of insecure protocols, weak encryption, and private-to-public network communications which can directly expose patient data to attackers. Filling these security gaps is challenging due to the very nature of HDO outlined previously. This situation calls for new solutions that address the characteristics of HDO.

In particular, we observe that current solutions for network monitoring and device classification deployed in HDO are ill-suited to encompass the specificities of these organisations' networks. Device classification is an important concept as it enables the creation and maintenance of a device inventory upon which certain security controls rely (e.g.,vulnerability assessment, network access control) [247]. Current tools used in HDO to classify connected devices are either based on manually-defined fingerprints, or on black-box machine learning techniques, which both come with their respective limitations. The main problem with classification tools in HDO is the significant number of unclassified devices, or "unknowns", which could introduce risks as mentioned previously. For this reason, we investigate how to address these limitations by asking ourselves the following question:

RQ 1.2: *Can we overcome the current limitations of device classification?*

To answer RQ 1.2 we research and analyse device classification methods. We focus on clustering techniques and explore their effectiveness to classify connected systems. Such techniques have been applied in the past to classify network traffic [66, 217]. These algorithms are designed to find similarity between entities and are well-suited to device classification by considering the semantic of the features and defining a proper similarity measure [261]. This enables us to complement fingerprinting-based solutions by using unsupervised learning techniques with a semantic-driven feature selection. By collecting specific data from devices on a network and applying a clustering algorithm, we can group similar devices together and ultimately infer their type.

Another challenge highlighted by the conclusions of RQ 1.1 is the use of insecure communication protocols in HDO and presence of (potentially) vulnerable medical devices. Our analysis points out that certain IoMT devices interact with other systems and share sensitive data over healthcare communication protocols. Some of these devices and network protocols are known to be vulnerable to attacks which could impact patients' safeness. We also notice the presence of other protocols for which no attacks have been demonstrated yet. It leads us to consider the following question:

RQ 1.3: *Are there unpublished security vulnerabilities in healthcare network protocols or medical devices which could impact patients' safeness?*

To answer RQ 1.3 we conduct security analysis on a selection of healthcare protocols and medical devices to assess the presence of vulnerabilities. Performing such security research is important as the discovery of weaknesses can help enhancing IDS in multiple ways. We can update IDS to detect attacks targeting the newly found vulnerabilities, and create new attack datasets to help with the design and testing of HDO-specific IDS or other NSM tools. We start our assessment by building a lab with connected medical devices, aimed at replicating a section of a HDO's network. Our protocol analysis and device security assessment lead to the discovery of new vulnerabilities. We demonstrate in our lab a number of attacks impacting the safeness of patients. These findings enable us to have a more accurate picture of HDO' exposure to cyber attacks and understand the risks that come with connected medical devices and their respective communication protocols.

1.2.2. Automotive network security

Moving on to the automotive sector in the second part of our study, we investigate the security of in-vehicle networks. Modern cars being literally computer networks on wheel, we want to identify to what extend network security measures are applicable to protect them from attacks. More precisely, we investigate the efficiency of Network Intrusion Detection Systems (NIDS) on automotive networks. To do so, we need to first understand how attacks on cars work before analysing the systems designed to detect them. We therefore start by answering the following question:

RQ 2.1: *What are the capabilities of an adversary using in-vehicle network attacks against a vehicle?*

To answer RQ 2.1, we first investigate automotive architectures and technologies to understand the specificities of modern vehicles, before analysing the capabilities of an attacker in a car network. The predominant protocol in automotive networks is the Controller Area Network (CAN). It is used to interconnect certain ECU, including the ones responsible for safety-critical operations (e.g., breaking and steering). Previous research has shown that an attacker can abuse the specifications of CAN to alter the functioning of ECU and influence their behaviour. With these cyber-physical attacks, an adversary can, for instance, steer the car to an arbitrary direction or switch the engine off.

Following the publications of such attacks, researchers have proposed a number of NIDS for CAN networks. They provide a non-invasive security measure that is well-established in other fields (e.g., IT domain). Their interest in the automotive sector is growing as they seem potentially suitable to detect attacks on CAN. In this context comes the following question:

RQ 2.2: *What kind of coverage do NIDS provide against attacks on CAN networks?*

To answer RQ 2.2 we survey the literature on CAN bus NIDS and categorize the proposed solutions. Such task commonly calls for the use of a taxonomy, as often done in NIDS surveys in other fields. However, due to the technical specificities of CAN NIDS, no existing taxonomy is suitable for this purpose. We therefore introduce a new taxonomy specifically for CAN NIDS in order to get a more informative and accurate categorization.

While a taxonomy can provide a clear picture of the coverage of the NIDS published at a theoretical level, it does not tell much about their practical performance. In fact, their capability to detect different attacks on the CAN bus is uncertain based on the papers alone. Too often the evaluations provided (if any) are disparate and limited (in terms of, for example, data used or attacks implemented). Consequently, this situation makes it impossible to compare the effectiveness of the solutions published, and leaves us with the following question:

RQ 2.3: *How can we evaluate CAN bus NIDS on equal ground?*

To answer RQ 2.3 we develop a unified framework for evaluating CAN bus NIDS. We introduce performance metrics to evaluate these systems and create datasets covering normal driving data from different cars, as well as several attacks datasets (in line with RQ 2.1). In order to demonstrate our framework, we implement and evaluate different CAN bus NIDS from our survey, covering the diverse categories of approaches identified in RQ 2.2. Our results shed light on the difficulties of CAN bus intrusion detection due to the challenging nature of the automotive setting.

Overall, these six research questions enable us to answer the MRQ. In fact, the conclusions drawn from each of them contribute to the greater understanding of the extent to which network monitoring can be performed in the SCE we investigated. The knowledge resulting from this work can in turn serve as a foundation to address monitoring challenges in other SCE.

In order to answer these questions, we collaborated with the company Forescout Technologies (formerly known as SecurityMatters). Its position as market leader in device visibility and intrusion detection on industrial and enterprise networks allows us to access substantial amount of data and key resources. Forescout's technology granted us invaluable insights into real-world environments, enabling us to fully grasp the specificities and challenges of the aforementioned SCE.

1.3. Thesis Outline and Contributions

The objective of this work is to provide an answer to the MRQ and to understand the extend to which we can perform NSM in SCE. In addition to understanding, these studies of healthcare and automotive security presented in this thesis contribute to the security of SCE by laying the foundations that enable NSM in SCE. More precisely, we explore how to set up the ability to perform effective intrusion detection in these environments. To achieve this, we conduct our research on healthcare and automotive network security, and structure our results in this dissertation in two parts. Each part consists in three chapters,

where each chapter is dedicated to answer one of the research questions mentioned above and provides concrete contributions to meet the objectives of this thesis.

We begin in Chapter 2 by exploring the current state of network security in HDO. To answer RQ 1.1, we identify gaps via a literature study and two observational studies. This work displays a unique overview of HDO from a technical point of view and provides novel insights into their security posture. The scale and the depth of the observational studies make them arguably one of the most comprehensive security research on HDO networks. Starting with the literature study, this chapter proposes as first contribution a description of a network architecture and examples of threats on typical HDO. Then, we present in the first observational study a large-scale analysis based on data from 2.3 million devices in 67 HDO networks. As second contribution, the results of this analysis demonstrate that HDO are large ecosystems of devices that are difficult to manage and secure as they comprise a variety of software, vendors, and protocols. The second study is an in-depth analysis based on captured network traffic from 5 HDO. It contributes by shedding light on the presence of insecure and clear-text protocols, as well as weak encryption schemes, which can directly expose patient data to attackers and enable them to launch cyber-physical attacks that can adversely affect patient health. Part of this chapter has appeared in a refereed conference publication [63].

In Chapter 3 we address the limitations of device classification solutions based on traditional device fingerprinting and black-box machine learning. We answer RQ 1.2 by proposing a semantic similarity-based clustering method. We evaluate our solution and compare it to the state-of-the-art fingerprint-based classification using data from 20,000 devices. Our novel approach contributes to the improvement of the performances of the existing classification engine in terms of coverage (i.e., how many unknown devices can be classified?), accuracy (i.e., are there devices misclassified?) and granularity (i.e., is the type of a device precise enough?). The results show that we can successfully classify a good amount of unknown devices with 99.5% consistency and 95.6% high granularity. Also, the validation of our approach using real-world databases demonstrated its effectiveness to solve industry-wide problems. Contrary to supervised learning, unsupervised algorithms overcome the extensive need for labelled data, which helps to keep up with the growth of the number and diversity of devices in HDO. Part of this chapter has appeared in a refereed conference publication [64].

In Chapter 4 we answer RQ 1.3 by identifying new weaknesses in healthcare network protocols and medical devices that were not yet published. We show how to set up a small healthcare lab with a number of devices commonly found in HDO (for instance, patient monitor, blood analyser). We demonstrate the practical exploitation of the vulnerabilities we found by implementing cyber-physical attacks which could impact patients' safeness. This work contributes to the enhancement of NSM capabilities in HDO in two ways. New signatures for IDS can be created to detect our attacks and protect patients, and new attack datasets can also be made to assist the development and testing of HDO-specific IDS. Part of this chapter is to be published in a refereed journal publication.

In Chapter 5 we investigate the state of automotive security, with a specific focus on CAN network. The contribution of this chapter is twofold. In order to answer RQ 2.1, we first

lay out the necessary background knowledge about automotive network, the CAN protocol and its vulnerabilities. Then we provide a threat analysis for the CAN bus and we detail the capabilities of an adversary and the attacks that can be launched against a vehicle's components. This chapter's contribution represents a solid foundation required to understand the unique specificities of a modern vehicle and their implications for network security measures in automotive networks, as we explore in the following chapters. Part of this chapter has appeared in a refereed conference publication [62].

In Chapter 6 we focus on network intrusion detection systems (NIDS) proposed for the CAN bus. To answer RQ 2.2, we start by covering key concepts regarding intrusion detection before investigating the kind of coverage CAN bus NIDS provide against attacks described in Chapter 5. As contribution, we introduce in this chapter a novel taxonomy for CAN bus NIDS and propose an organised inventory of the NIDS present in the literature. Part of this chapter has appeared in a refereed conference publication [62].

In Chapter 7 we answer RQ 2.3 by proposing a unified evaluation framework, enabling to test the performance of CAN bus NIDS. The contributions of this chapter consist in our methodology, combined with the datasets of attacks we implemented and made publicly available, which allows to evaluate NIDS on equal ground. To demonstrate our framework, we select a number of NIDS and evaluate them. This test enables us to draw a number of conclusions regarding intrusion detection on CAN systems. These results challenge the design of CAN bus NIDS as they have been proposed until now. We finish by suggesting a number of considerations that could enable effective intrusion detection algorithms for the CAN bus. Part of this chapter has appeared in a refereed conference publication [61].

In Chapter 8 we finally conclude this thesis by answering the MRQ and summarising the learnings drawn from the analysis of these two SCE. We also propose considerations to pursue the development of security measures for these SCE.

1.4. Publications

Our research led to the publications of the following:

1. **G. Dupont**, J. den Hartog, S. Etalle & A. Lekidis, *A survey of network intrusion detection systems for controller area network*, IEEE International Conference on Vehicular Electronics and Safety (2019)

2. **G. Dupont**, J. den Hartog, S. Etalle & A. Lekidis, *Evaluation framework for network intrusion detection systems for in-vehicle CAN*, 8th IEEE International Conference on Connected Vehicles and Expo (2019)

3. **G. Dupont**, J. den Hartog, S. Etalle & A. Lekidis, *Technical report - Network intrusion detection systems for in-vehicle network*, arXiv eprint 1905.11587 (2019)

4. **G. Dupont**, D. R. dos Santos, E. Costante, J. den Hartog & S. Etalle, *A Matter of Life and Death: Analysing the Security of Healthcare Networks*, ICT Systems Security and Privacy Protection - 35th IFIP TC 11 International Conference (IFIP Advances in Information and Communication Technology; vol. 580 IFIP) (2020)

5. **G. Dupont**, C. Leite da Silva, D. R. dos Santos, E. Costante, J. den Hartog & S. Etalle, *Similarity-based clustering for IoT device classification*, IEEE International Conference on Omni-layer Intelligent systems (COINS) (2021)
6. **G. Dupont**, D. R. dos Santos, S. Dashevskyi, S. Vijayakumar, S. P. Murali, E. Costante, J. den Hartog & S. Etalle, *Attacks on Healthcare Network Protocols*, To be published

1.5. Additional research output

Over the course of the PhD, the research also enabled the following:

Patent application: Device Fingerprinting Assisted by Similarity-based Semantic Clustering (Application No. 63/181,908 ; filed April 29, 2021)

Disclosure of 2 new vulnerabilities: Performing vulnerability research on a medical device (Siemens DCA Vantage blood analyser) led to the discovery of two previously unknown vulnerabilities. We disclosed them to Siemens and worked together for remediation[2]. Both have been attributed a CVE number (Common Vulnerabilities and Exposure), and are publicly reported as "Hard-coded password" (CVE-2020-7590) and "Improper Privilege Management" (CVE-2020-15797) respectively.

[2] https://www.siemens-healthineers.com/support-documentation/security-advisory

Healthcare Security

2

The State of Cyber Security in the Healthcare Industry

Healthcare Delivery Organisations (HDO) are complex institutions which have a broad range of devices increasingly interconnected. This situation brings security concerns, as we observe a rising number and sophistication of cyber attacks on hospitals. Additionally, a growing body of research focusing on connected medical devices demonstrate the risks of cyber-physical attacks which could impact patients' safeness. It is unclear how prepared are HDO to face this current threat landscape. For this reason, in this first chapter, we take a broad look at these environments and explore the current technical state of readiness of HDO with regards to cyber security (RQ 1.1). We conduct various analysis over 67 instances of them, enabling us to look at these infrastructures from different perspectives. Our results demonstrate the large scale and challenging nature of HDO, as well as the limits of some network security monitoring tools commonly used in these environments. This work lays out the foundations for the following two chapters.

The results reported here have been published in [63] (paper 3 of Section 1.4).

2.1. Introduction

HDO, such as hospitals and clinics, are complex institutions where a broad range of Information Technology (IT), Operational Technology (OT), and Internet of Things (IoT) devices are increasingly interconnected [178]. IT devices and enterprise systems process and exchange highly sensitive data (e.g., patients' health records and financial information), whereas OT and IoT devices are used for diverse functions such as building automation, and guest entertainment. Specialized IoT devices, referred to as Internet of Medical Things (IoMT) [88], are connected medical devices supporting clinical care and they can generate and exchange patient data with other devices, such as Electronic Health Record (EHR) systems [131]. These new technologies and increased connectivity can help improve the efficiency and quality of care.

However, this reliance on such technologies can also introduce new security and privacy risks [10, 116]. We are witnessing an increase in the number and sophistication of cyber attacks on hospitals [105]. So far, these attacks are mainly in the form of ransomware [140], targeting mostly the IT part of the network. But the increased connectivity is not restricted to the IT systems as it also applies to the OT systems. Attacks already seen in different domains like Building Automation Systems (BAS) [167, 185, 196] show that OT devices may be targeted. Specialized tools (e.g., Shodan) for finding exposed systems and potential exploits can aid attackers in finding vulnerable targets and launching attacks. Additionally, IoMT devices can also be vulnerable to cyber attacks which may have devastating consequences for patients and HDO (e.g., [151, 161]). All of this makes it essential to be prepared for attacks that exploit the complexity of HDO environments.

Various components of HDO's environments have been studied by the security research community. Security assessment for IT infrastructures is a well covered topic [152], and work like [123] looks at the human factor in HDO. Most research on cybersecurity in healthcare focuses on connected medical devices (e.g., [11, 124, 232, 262]), with special attention on implantable devices because of their potential direct harm to patients [198]. These works mostly ignore other kinds of devices present in an HDO's network and that can be used during cyber attacks. Some studies discuss the security threats not only to medical devices, but also to medical data (e.g., [127]). Jaigirdar et al. [116] analysed the trust that physicians place in secure end-to-end communication of healthcare data. Kune et al. [76] surveyed medical and non-medical protocols used in HDO and analysed their security properties. Wood et al. [257] introduced a method to capture network traffic from IoMT devices and automatically detect cleartext information that may reveal sensitive medical conditions.

In this chapter we investigate the current technical state of readiness of HDO with respect to cyber attacks targeting their networks and aiming at, for example, stealing or altering patients' data or even harming their health. To achieve this aim, we address the following research questions (RQ):

- **RQ1** How is an HDO's network organised?
- **RQ2** What are some potential threats to an HDO's network?

- **RQ3** What kinds of devices and software are present in an HDO's network?
- **RQ4** What security vulnerabilities are linked to HDO's network protocols?

2.1.1. Summary of Contributions and Outline of the Chapter

We answer RQ1 and RQ2 by investigating existing literature to give an overview of the network architecture and examples of threats on typical HDO (Section 2.3).

To answer RQ3, we conduct in the first observational study a large-scale investigation of 67 HDO networks (Section 2.4). Its results confirm that HDO are large ecosystems of devices that are difficult to manage and secure as they comprise a variety of software, vendors, and protocols. Moreover, certain devices found in HDO are not only running network services often exploited by malware and malicious actors (e.g., SMB and RDP) but also legacy operating systems no longer supported by vendors, thus providing potential access to attackers.

To answer RQ4, we perform a network security assessment of 5 of these HDO networks as second observational study by analysing captured network traffic (Section 2.5). The outcome highlights the presence of insecure and clear-text protocols, as well as weak encryption schemes, which can directly expose patient data to attackers and allow cyber-physical attacks that can adversely affect patients' safeness.

2.2. Methodology

The methodology used to collect the data and run our analyses is depicted in Figure 2.1. The figure is divided in four parts (or phases): *Data Collection*, *Literature Study*, *Large-Scale Study*, and *In-Depth Study*. The *Data Collection* part depicts how our datasets were captured. The data comes from many HDO; the elements in the box labelled HDO_i are repeated across HDO. Within HDO_i, network traffic is collected and analysed by (one or more) network monitoring appliances that are connected to (one or more) network switches in the HDO. (The Figure shows only one switch and one appliance per HDO for the sake of readability.) The appliance collects data both by passively listening and by actively interacting with the devices on the network (e.g., by running Nmap[1], p0f[2], and other network scanning tools). The data is then analysed by the appliance to find *attributes* of devices (e.g., their MAC address, the device's operating system, etc.). Some of these attributes, called raw attributes, can be directly obtained from the network traffic. Other attributes are obtained by classifying devices using a *Device Profile Library*. The Device Profile Library is a set of rules which assign a profile to a device once a given combination of raw attributes have been detected for that device. A profile is a triple of attributes (*vendor, OS, function*), where the first two elements are self-explanatory and the third element represents the function of a device in the network (e.g., 'OT/Healthcare/X-ray machine'). The classification of devices is not the focus of this work and we assume that the Device Profile Library is always correct. We elaborate in the next chapter about device classification.

[1] `https://nmap.org/`
[2] `http://lcamtuf.coredump.cx/p0f3/`

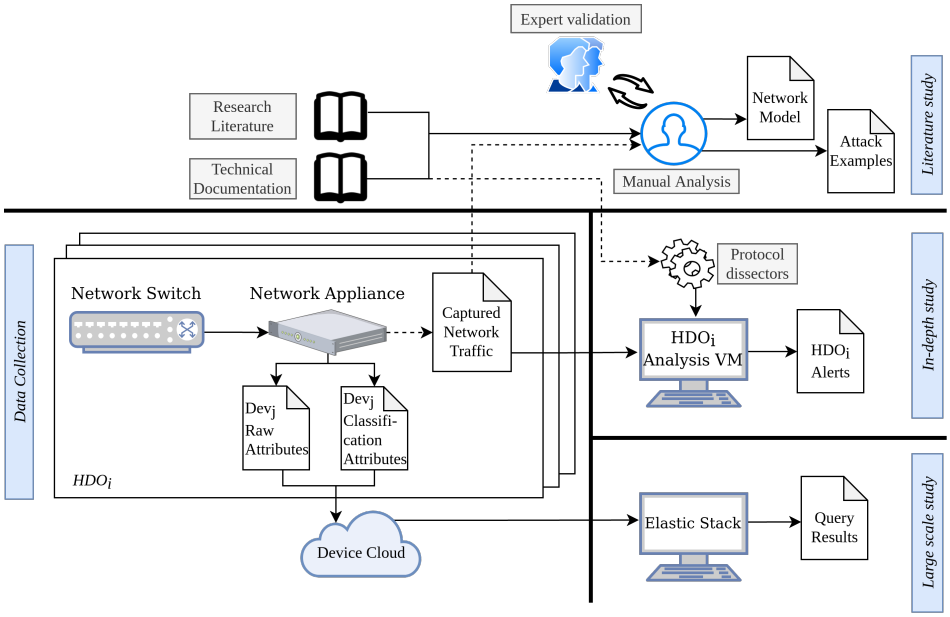


Figure 2.1: Methodology for data collection and analysis.

Overall, the appliance thus outputs three types of data:

- *Raw attributes*, such as *MAC address* and a list of *open ports*, for devices seen on the network. These are attributes obtained directly from the gathered network traffic.
- *Classified attributes*, for example 'OT/Healthcare/X-ray machine' or 'IT/Printer', which are linked to the function of devices found on the network.
- *Captured Network Traffic* in the form of raw network packets saved in PCAP files[3].

The first two types of data are anonymised and sent to a *data lake*, which aggregates the data collected in all HDO. We employ this data as input for the *Large-Scale Study* (Section 2.4), in which we execute a number of queries to retrieve information about devices in HDO. This allows us to understand the types and some characteristics of devices in several HDO networks. The output of this analysis is a series of *Query Results* and an accompanying explanation. The tool used for analysis in this phase is Elastic Stack[4], containing the ElasticSearch search engine and the Kibana visualization tool.

While the Large-Scale Study gives us an overview of the devices connected to HDO' networks, to know whether they would be susceptible to attacks targeting medical protocols requires knowing which protocols are actually used and by which devices. We therefore also perform an *In-Depth Study* (Section 2.5). We take the third type of data, captured network traffic, and perform various analyses in which we look at network activities and

[3] https://wiki.wireshark.org/Development/LibpcapFileFormat
[4] https://www.elastic.co/what-is/elk-stack

communication protocols, among other things. We leverage several tools in order to analyse the network traffic, such as Wireshark[5], NetworkMiner[6] and Forescout's SilentDefense[7] solution, enhanced with our *Protocol Dissectors* we created for the key medical protocols identified during the *Literature Study* (see below). These dissectors allow us to identify the protocols' presence in traffic and parse the contents of their packets. Since collecting all network data from all HDO is clearly not feasible, the study takes instead as input datasets from five HDO captured at key locations in their network. These HDO are referred to as HDO_{1-5} in the rest of this chapter. As output, the study provides a detailed view on the network security of HDO, identifying insecure protocols and susceptibility to attacks.

The *Literature Study* phase, presented in Section 2.3, takes as input the relevant identified *Research Literature* plus *Technical Documentation* from vendors and industry participants (e.g., standards and technical bodies). The outputs of this phase are a *Network Model*, including devices and protocols, and a set of *Attack Examples* for HDO, which are obtained via manual analysis. This output is validated by conducting interviews with stakeholders in HDO cybersecurity and by looking manually into the *Captured Network Traffic*. The medical protocols identified at this stage form the basis of the attack examples we describe, and this knowledge allows us to recognize this particular type of communication in the network traffic of HDO.

2.3. Network model and threats

In this section we conduct a literature study to answer the first and a second research questions, namely "How is an HDO's network organised?"(RQ1) and "What are some potential threats to an HDO's network?"(RQ2). We address RQ1 by providing a network model of a typical HDO and RQ2 by listing examples of threats to HDO. In addition, we validate the attacks in a laboratory setting (see Chapter 4).

2.3.1. Organisation of HDO's network

The major distinction between HDO networks and typical enterprise networks comes from 1) the type of devices deployed, and 2) the communication protocols used, both of which are described below.

Network devices

HDO are generally divided into several departments, delivering specific clinical care (e.g., radiology) or organisational services (e.g., administration). We represent on Figure 2.2 a simplified model of typical HDO networks, including two departments in the plain-line boxes, as well as some of the IT, OT, and IoMT devices commonly found. While some departments can have specialized equipment related to their operations (e.g., imaging modalities in radiology department), there are also certain devices that can be found in multiple

[5] https://www.wireshark.org/
[6] https://www.netresec.com/index.ashx?page=NetworkMiner
[7] https://www.forescout.com/platform/silentdefense/

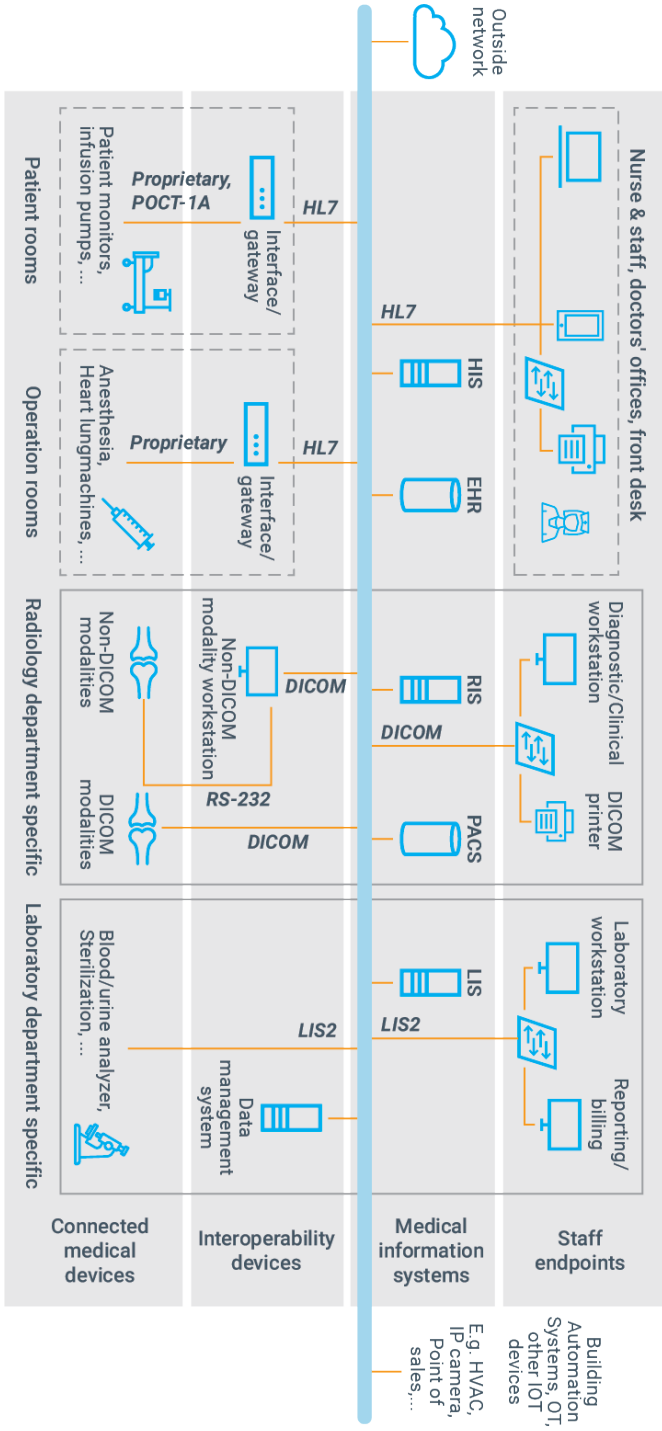


Figure 2.2: Simplified network architecture of a typical HDO.

departments. In addition, there are systems that can be found ubiquitously across an HDO such as IT devices, as depicted in the upper left side of the figure.

We classify HDO's networked devices into 4 categories. The *connected medical devices* support clinical care, while *interoperability devices* assure communication for some devices on the network. Then, *medical information systems* store and manage clinical data and finally *staff endpoints* provide human interfaces to information systems. *Connected medical devices* can be further divided into active or passive devices [115]. Active medical devices are meant to deliver medical treatment and sustain patient life (e.g., drug pumps). Passive devices monitor patient information such as vital signs or test results, and report events or need for treatment to clinical staff (e.g., patient monitors and laboratory equipment).

Depending on the network protocol used by the aforementioned devices, they may be connected to *interoperability devices*, which will convert network data into an interoperable format, allowing it to be further processed and/or stored by *medical information systems*. Such systems can be seen as the backbone of an HDO, as they collect, store and manage various types of healthcare data. For example, health, radiology, and laboratory information systems (respectively HIS, RIS and LIS), will manage electronic medical records, radiology pictures from imaging modalities and laboratory analysis results, respectively. Finally, HDO also have other types of devices represented together under building automation systems, OT and other IoT devices.

Communication protocols

Medical devices in HDO transmit data using standard or proprietary protocols. Table 2.1 summarises the most important medical protocols we identified during our research. Depending on the protocol, specific information about a given device can be found in packets' payload such as the firmware version and hardware version for that device.

HL7v2 is the most widely used interoperability and data exchange protocol in medical networks. This messaging standard allows the exchange of patient, clinical and administrative information. DICOM defines both the format for storing medical images and the communication protocols used to exchange them. As de-facto standard, it is implemented by all major vendors of devices involved in medical imaging processes (e.g., modalities and diagnostic workstations). POCT01-A and LIS02-A2 are used for point-of-care and laboratory devices, respectively. These protocols can issue test orders with patient information and transfer the results of tests to a Data Management System (DMS). The proprietary protocols Philips Data Export [190] and GE RWHAT [151] are used to control patient monitors

Table 2.1: Main medical protocols identified.

Protocol	Type	Devices
DICOM	Standard	Imaging modalities, PACS
HL7v2	Standard	Connected Medical Devices, Medical Information Systems, Interoperability Gateways
POCT1-A	Standard	Point of Care Testing
LIS2-A2	Standard	Laboratory devices
Data Export	Prop. (Philips)	Patient monitors
RWHAT	Prop. (GE)	Patient monitors

of their respective vendors. They allow patient monitors to communicate the vital readings of patients to a central monitoring system (CMS).

While supporting critical operations in HDO, these medical protocols support neither encryption nor authentication (or support them without enforcing their usage, in the case of DICOM), a situation similar to what is found in other cyber-physical systems, e.g., Industrial Control Systems (ICS) [24] and Building Automation Systems (BAS) [38]. We also identified other protocols such as HL7 FHIR, as well as other proprietary protocols. However we choose to ignore them in this chapter as they are not as widely deployed as the ones in Table 2.1.

2.3.2. Potential threats to an HDO's network

Malicious actors may have various motivations to attack HDO [105, 115]. All reported attacks on HDO (see, e.g., [74, 226, 227]) seem to have been motivated by financial gains directly via ransomware and cryptomining, or indirectly via stolen information and use of infected computers in botnets.

However in the light of the security research done on medical devices and their protocols [30, 50, 59, 101, 151, 161, 193, 195, 264], one can wonder how an attacker could leverage vulnerabilities on such devices. We provide below some examples of attacks, considering an attacker on the network. Such foothold can be established in various ways [105]. These attacks can be the final step in a multi-step attack [115].

Attack examples

Security research in healthcare focuses either on devices or network protocols. Vulnerabilities in specific medical devices have been found over the past years (see, e.g., [99, 193, 195]), and the number of security advisories in the medical space has been growing [243, 264]. Currently, there is a trend of research into protocol insecurity [73, 79]. Vulnerabilities of the protocols below have been demonstrated.

HL7 standards, which are used to exchange patient data between systems, can be abused in several ways and are often insecurely implemented [50, 59, 101]. As HL7 data is sent over unauthenticated communications, attackers can intercept and modify information in transit, which may lead to life threatening consequences. Similarly, unauthenticated and unencrypted DICOM communications also allow attackers to tamper with medical images, misleading medical staff to wrong diagnostics. The DICOM standard supports user authentication and message encryption, however while their implementations and usage are left to product vendors and HDO, we observe in a number of HDO that these security mechanisms are not implemented. To demonstrate the possible consequences of this situation, researchers implemented a proof-of-concept to add or remove tumours from CT scan images being transferred over the network, leading to dramatic consequences for patients [161]. While attacks against unprotected protocols such as POCT1-A and LIS2-A2 have not yet been published, we demonstrate in Chapter 4 how they can also be abused.

Proprietary protocols have also caught the attention of security researchers, who have shown how one could intercept a patient's vital signs sent by a GE patient monitor over their

Table 2.2: Potential attacks on the main medical protocols identified.

ID	Protocol	Target	Attack	Description
A1	HL7v2	Patient data	Data theft	An attacker can retrieve sensitive patient data such as clinical and financial information as the data is sent unencrypted
A2	HL7v2	Patient health	Tamper with EHR	An attacker can modify arbitrarily the electronic health records of patients (e.g., change the allergies or medication prescription)
A3	DICOM	Patient health	Tamper with test results	An attacker can tamper with medical images by virtually adding or removing tumours for respectively healthy or sick patients
A4	POCT1-A	Patient health	Tamper with test results	An attacker can change the results of point of care equipment (e.g., blood glucose analysis)
A5	LIS2-A2	Patient health	Tamper with test results	An attacker can modify the test results of laboratory equipment (e.g., blood analysis)
A6	Data Export	Patient health	Tamper with vitals	An attacker can tamper with patients' vital signs read by Philips patient monitors
A7	RWHAT	Patient health	Tamper with vitals	An attacker can tamper with patients' vital signs read by GE patient monitors

RWHAT protocol [151]. Once intercepted, a malicious actor could modify the patient signs arbitrarily. We reproduced in our lab a similar attack with a Philips patient monitor. As described later in Chapter 4, such monitors send information over the Data Export proprietary protocol, which can be intercepted, decoded and modified on the fly.

Our results show that those issues are prevalent in healthcare networks. In Table 2.2, we summarize seven examples of attack against these protocols.

2.4. Large-Scale Study

In this section we answer the third research question, "What kinds of devices and software are present in an HDO's network?" To this end we leverage data from various HDO, providing us insights into the devices connected on their networks. We present the charts resulting from our analysis, alongside our conclusions.

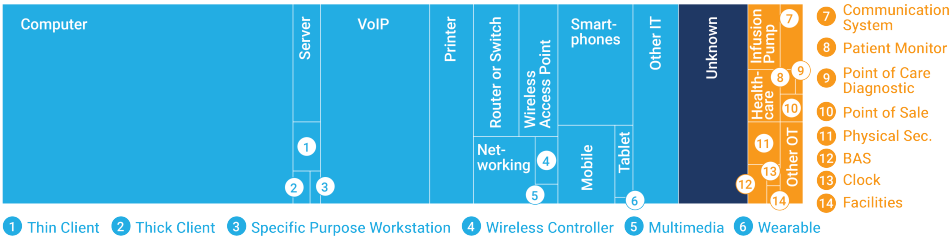


Figure 2.3: Average distribution of IT, OT and unknown devices found in HDO networks.

2.4.1. Sample description

The dataset comprises a total of 2.3 million devices. The amount of unique devices per HDO, regardless of their type, ranges from 597 to 234,305 with an average of 50,078 and a median value of 12,766. We see a wide range of sizes across the sample, but most are in the thousands to tens of thousands of devices. To help better understand the composition of HDO networks we provide below different perspectives through our data analysis.

2.4.2. High-level device overview

Figure 2.3 represents the three main classes of device found in HDO, namely IT, OT and unknown devices. On average, these classes correspond to respectively 84%, 7% and 9% of the total number of devices. *IT devices* include personal computers, VoIP devices, network printers, mobile devices, and various networking equipment, among other things. *OT devices* are comprised of not only healthcare devices and infusion pumps, but also BAS devices, points of sale, physical security and other facilities-related devices such as IP security cameras. Finally, the devices that cannot be classified are referred to as *unknown devices*.

Figure 2.3 also shows the average distribution of device types in HDO. One can observe that more than roughly a third of the connected devices are computers (36.4%), followed by VoIP devices (13.8%) and smartphones (5.7%). Understanding the distribution of devices is important because many networks still operate in organisational silos, where different departments are responsible for different sections of the network. This situation tend to leave gaps in security [149].

2.4.3. Types of medical devices

Since connected medical devices are especially critical for HDO, it is important to understand the distribution of these devices in a finer granularity. Figure 2.4 shows the most common types of connected medical devices. Per-patient devices, such as infusion pumps and patient monitors represent the majority of healthcare devices on HDO networks, as well as per-personnel devices like communication systems. This makes sense as they are the devices deployed mostly on a 1:1 ratio. Devices such as those used in laboratory diagnostics or medical imaging represent a smaller number because they are shared devices. The "healthcare" device type on Figure 2.4 refers to medical devices that cannot be further categorized into a more specific type.

2.4.4. Diversity of vendors and operating systems

We now look at the diversity of the device ecosystem in HDO in terms of vendors and Operating Systems (OS). Our analysis shows that on average, HDO have a total of 152 different device vendors. When looking at the number of unique vendors for specific device types in HDO, we observe for example that IT computers have on average 51.5 unique vendors and networking and VoIP equipment have respectively 25.5 and 7.2 unique vendors. Regarding medical devices, infusion pumps, patient monitors and point of care diagnostics devices have respectively 2.5, 2.2 and 2.6 unique vendors on average.

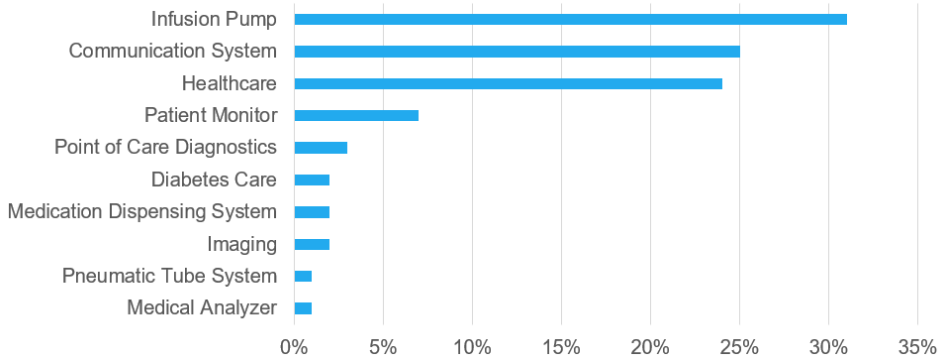


Figure 2.4: Top 10 connected medical devices in HDO networks.

The complexity of device management is linked to the number of unique vendors whose devices are deployed on a network. Vendors have different support, maintenance, and patching programs, which can affect the time between the disclosure of a vulnerability and the patching of the related systems. As an example, consider the recently disclosed set of vulnerabilities on the IP stack of the VxWorks real-time OS [208]. Some medical devices run this particular OS, but it is not immediately clear to the users whether a particular device is affected, if there is a patch available and how it can be applied. Contacting each vendor for inquiry would be very time consuming.

Additionally, it is important to consider the diversity in OS as it can bring some security concerns as well. Figure 2.5 shows the OS variants of devices on HDO networks. For each OS, its proportion relative to the others is given and, for some of them, a breakdown of the version in use. Windows is the most common OS across HDO's devices (41%). Windows 7, 10 and Windows Server 2012 represent respectively 11%, 8% and 1% of the OS, while we still observe a non-negligible amount of other variants such as Windows XP, Windows Server 2008 and 2003.

Our analysis reveals that 40% of networks have more than 20 different OS. We see that 0.4% of devices are running an unsupported version of Windows and 70% of devices a version of Windows for which Microsoft support is planned to expire by January 14, 2020[8], such as Windows 7, Windows 2008 and Windows Mobile. Running unsupported OS is a well known security issue. HDO networks will most likely continue to have medical devices running legacy OS since updating can be too costly or even infeasible, due to unacceptable downtime required or (software) compatibility issues. Consequently, many devices would have to keep operating while remaining potentially vulnerable. This situation calls for additional protections, such as appropriate segmentation of systems, which can be achieved through Virtual Local Area Networks (VLAN) for example.

[8] https://bit.ly/38e9QXc

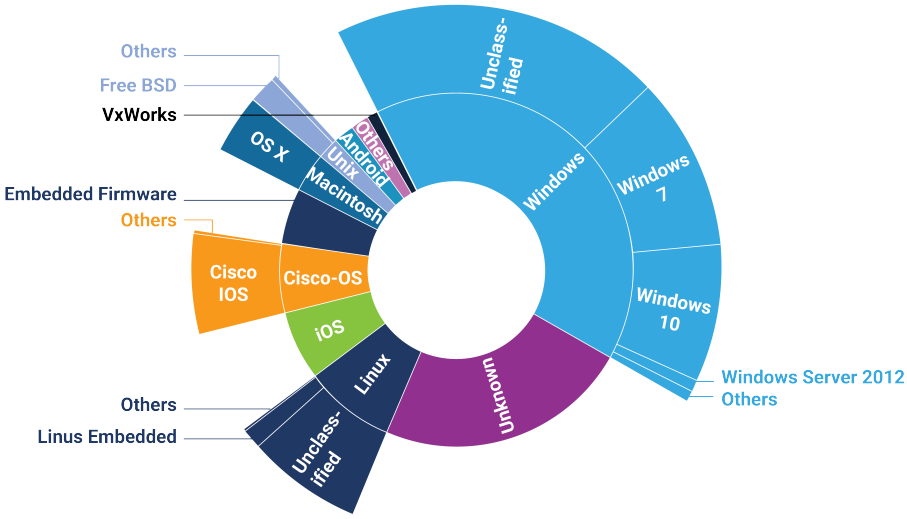


Figure 2.5: Distribution of OS variants in devices in HDO networks.

2.4.5. VLAN analysis

Network segmentation is a commonly advised security measure [152]. VLAN can segment the network by effectively isolating critical systems, segregating similar devices by function, and limit access to data and other assets in a segment. In this context, isolating medical devices in VLAN could help to keep them separated from the rest of the network.

However, our study shows that, on average, less than 20% of the medical devices are deployed in a VLAN and, as Figure 2.6 shows, 86.5% of the HDO have medical devices outside of VLAN. In addition, we also found that 61 of these HDO have at least one VLAN with a combination of medical devices and other OT devices, thus undermining the segmentation that use of a VLAN may provide. Examples of such cases that we saw in the data are VLAN containing both medical imaging modalities and IP cameras or HVAC systems, or even blood glucose monitors with points of sale.

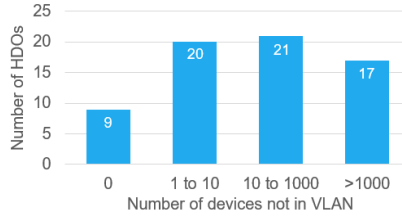


Figure 2.6: Occurrences of HDO with devices not in VLAN.

This observation confirms the statement of the ISE regarding HDO's network being improperly segmented [115].

2.4.6. Enabled common services

Some common network services are often targeted by recent malware and malicious actors [163]. Table 2.3 shows the amount of devices that have the given service's port open.

Server Message Block Protocol (SMB) is the transport protocol used by Windows machines for a variety of purposes such as file sharing and access to remote Windows services. WannaCry and NotPetya are two examples of ransomware that exploit vulnerabilities in SMB. Remote Desktop Protocol (RDP) is another common protocol exploited by modern automated threats [222]. Secure Shell (SSH) may be abused by brute-force attacks to log remotely onto machines. Telnet and File Transfer Protocol (FTP) are often-exploited vectors: these protocols do not secure nor encrypt network sessions.

Overall, after analysing the kinds of devices and software present in an HDO's network, we can conclude that a large number of devices on HDO networks have high-risk services turned on. The access requirements of medical vendors and outsourced suppliers often demand devices to have services like RDP enabled. Other times, the network ports are left open by default without the knowledge of IT and security staff. In the next section we look closer at HDO networks to better understand the security vulnerabilities linked to network protocols.

2.5. In-Depth Study

This study described in this section aims at answering our fourth research question, "What security vulnerabilities are linked to HDO's network protocols?" We analyse network traffic of HDO in order to provide a detailed view of their network security posture, including the identification of insecure protocols and susceptibility to attacks.

2.5.1. Sample description

The datasets used in this study correspond to raw network traffic of five different HDO. For HDO_1 and HDO_2 the data was captured over a period of 2 days and it comprises a total of respectively 2,207 and 1,513 devices. For HDO_3 the capture lasted one day and it collected data from 12,289 devices. Finally, for HDO_4 and HDO_5 the captures ran over four days and account for 11,051 and 4,423 devices respectively.

2.5.2. Overview and network activities

In the datasets, we found the healthcare network protocols presented on the left side of Table 2.4. Recall that these protocols are used by diverse device types in HDO as described previously in Section 2.3.1.

Table 2.3: Common enabled network services.

Network service (port number)	Devices (%)	Devices (absolute)
SMB (445)	26%	573,455
RDP (3389)	14%	318,634
SSH (22)	8%	187,135
Telnet (23)	5%	113,071
FTP (21)	5%	107,719

Table 2.4: Findings of network traffic analyses in five HDO.

	Healthcare protocols						Susceptible to attacks							Weak proto.			Certificates	
Dataset	HL7v2	DICOM	POCT1-A	LIS2-A2	Data Export	RWHAT	A1	A2	A3	A4	A5	A6	A7	SSLv3	TLSv1.0	TLSv1.1	Issuer not whitelisted	Expired
HDO_1	✓	✓				✓	✓	✓	✓				✓		✓	✓	✓	
HDO_2		✓			✓				✓			✓			✓		✓	
HDO_3	✓	✓		✓			✓	✓	✓		✓			✓	✓		✓	✓
HDO_4	✓	✓	✓				✓	✓	✓	✓				✓	✓	✓	✓	✓
HDO_5	✓	✓		✓			✓	✓	✓		✓				✓	✓	✓	

The presence of these protocols indicates that these HDO are susceptible to some of the attacks described in Section 2.3.2. We propose in Table 2.2 a list of attacks leveraging these protocols' weaknesses that an attacker could execute once having access to the HDO's network. As one can see in Table 2.4, all five HDO in our analysis are susceptible to be vulnerable to at least two attacks on medical device protocols. We also found obsolete versions of other protocols such as SNMPv1 and v2 and NTPv1 and v2. We do not elaborate on those findings because they do not fit the attack examples that we defined in Section 2.3.2.

2.5.3. Weak encryption

As presented on the right side of Table 2.4, we found that SSLv3, TLSv1.0 and TLSv1.1 are still used. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols used to secure network communications of higher-level protocols, such as HTTPS or FTPS. SSLv3, TLSv1.0/1.1 are known to be insecure. They are impacted, for instance, by the POODLE and BEAST attacks [211], in which an attacker can downgrade a connection and decrypt the traffic. Our analysis shows that these weak protocols are used internally in HDO, where one could still argue that other additional security measures could compensate. However, they are also used externally, even to connect to organisations such as Microsoft and Google.

Additionally, our analysis revealed issues with the SSL/TLS certificates used in HDO. Such certificates play a critical role in authentication and data encryption. We found that all HDO use certificates with non-whitelisted issuers. For security purposes, it is usually recommended to only use certificates delivered by trusted issuers (i.e., whitelisted issuers). Also, two of the HDO displayed are still using expired certificates, both for healthcare applications and network equipment.

2.5.4. External interfaces and communication

As discussed, HDO networks are complex and use many different protocols, including ones dedicated to healthcare. For operational reasons, some medical applications can be reached from the outside of the network, sometimes using healthcare protocols. This adds to the complexity of managing such systems and increases the probability of sensitive information or systems being exposed. For example, we found in one HDO a system containing an EHR application exposed on the public Internet.

In addition, in all HDO except for HDO_2, we observed communications between public and private IP addresses using HL7v2. As these communications are unencrypted, they can be easily read, and leak sensitive patient information such as names and addresses, employment status, phone number, allergies and also test results. Other information regarding the care provider can also be found such as the doctor's name in charge of the patient, with his or her license number.

Moreover, there were also in two HDO medical devices communicating over non-medical protocols with external servers. For example, a medical information system was seen to communicate over SSH, and another to reach a web server over HTTP. In one instance, a communication with an external file server over FTP was observed, and we confirmed (using Shodan) that this external machine contains up to 25 vulnerabilities. If exploited, it could potentially lead to the compromise of such a server and create an entry point into the HDO's network.

Additionally, we also observed in an HDO a machine behaving suspiciously. In our sample, this computer was trying to reach a number of public IP addresses over various ports. We noted a number of port scans executed and other host discovery attempts. Finally, it was communicating internally with 11 other machines over Telnet. These signs of compromise require further investigation, and we are validating our hypothesis with the network's owners.

2.5.5. Firmware Versions and Vulnerabilities

Certain firmware are known to be vulnerable, as reported in *ICS Medical Advisories* [243]. To determine whether HDO have medical devices running known vulnerable firmware, we employ the protocol dissectors we developed (see Section 2.3.1). We find that in HDO_2 Philips IntelliVue patient monitors deployed in intensive care units have a firmware which could potentially be abused. If successfully exploited, the vulnerabilities could allow an attacker to read and write the memory of the device, and force it to restart, potentially leading to delays in diagnosis and treatment of patients[9].

In HDO_4, we find vulnerable Roche Accu-Chek Inform II blood glucose meters. This model, popular and commonly found in HDO, presents multiple vulnerabilities in which attackers could execute arbitrary code on the device by crafting POCT1-A packets and change the instrument configuration. This could lead to false analysis results and inaccurate diagnosis[10].

2.6. Chapter Conclusion

In this chapter we explore the technical state of readiness of HDO through studies across 67 organisations (RQ 1.1). This research indicates gaps such as insecure protocols, weak encryption schemes, and private-to-public network communications which can directly expose patient data to attackers. Filling these gaps is challenging: as our study show, HDO are

[9] `https://bit.ly/2E8wCC2` and `https://bit.ly/2YFCwnE`
[10] `https://www.us-cert.gov/ics/advisories/ICSMA-18-310-01`

large SCE, with diverse ecosystems of devices, including legacy and safety-critical systems, processing sensitive data. Their devices are also difficult to manage and secure because of the variety of software, vendors, and communication protocols. New network monitoring solutions that address the combination of these characteristics will be needed. More specifically, our analysis highlights two main issues with regards to performing network monitoring in HDO: 1) the limitation of device classification tools used in HDO, and 2) the lack of situational awareness with regards to threats to patients' safeness introduced by healthcare protocols.

The first issue comes from the design of classification tools that are not suited to the characteristics of HDO. Common tools rely on fingerprint-based classification techniques which are not suited for the ever-increasing diversity of devices in SCE. As a result, there is a substantial amount of unknown devices in network inventory (9% of devices on average in HDO, as seen in Section 2.4). This is a concerning issue since unclassified systems can introduce risks to the environment. Chapter 2 is dedicated to address this issue: we propose a novel classification approach to improve both the coverage (which devices are covered), and the granularity (how detailed is the information about them) of the classification. Moreover, the methodology we develop can be applied to other SCE.

The second issue comes from the limited knowledge available regarding certain healthcare network protocols we observed being used in HDO. They could potentially introduce risks if they happen to have vulnerabilities. In fact, previous security research conducted on other healthcare protocols have uncovered weaknesses which, if exploited, could impact patients' safeness. The usage of these protocols for which little information is available leads us to question whether they could have unknown vulnerabilities. For this reason, we conduct in Chapter 4 security analysis on selected protocols with the goal of identifying previously unknown weaknesses. Conducting such research is important as the information obtained can be leveraged to improve threat detection capabilities of network security tools such as Intrusion Detection Systems (IDS).

3

Semantic Similarity-based Clustering for Device Classification

Identifying and classifying devices connected to a network is a fundamental security control. It allows the creation of an inventory of all devices connected, which in turn enables other security controls such as vulnerability assessment or network access control. Most importantly, it helps detecting rogue devices maliciously deployed on the network, and unmanaged systems that could introduce threats. However, device classification seems to be challenging in HDO as we observed in the previous chapter an average of 9% of devices being unknown after application of current state-of-the-art tools. This situation can be explained by the limitations of such tools relying on fingerprints. Their design prevents them from being effective in HDO (and possibly other SCE) as they are unable to keep up with the broad diversity of devices. In order to address these limitations , we consider the techniques proposed in another domain known as network traffic classification, leveraging clustering methods. The question is whether the current limitations of device classification can be overcome with clustering techniques (RQ 1.2). To answer this question, we propose in this chapter a semantic similarity-based clustering method. Combined with a state of the art fingerprint-based classification engine, our results show that we can successfully classify up to half of the unclassified devices with a high accuracy. We also validate our approach with domain experts to demonstrate its effectiveness to solve industry-wide problems. Additionally, our novel method can be applied to solve device classification challenges faced in other environments.

The results reported here have been published in [64] (paper 4 of Section 1.4).

3.1. Introduction

There is a growing number and variety of devices connecting to enterprise networks due to trends such as BYOD and the rapid deployment of IoT [19]. This situation makes it critical for organisations to have an accurate inventory of these devices, including information such as device type (e.g., computer, IP camera, server), vendor, and operating system. This inventory enables security controls with monitoring, vulnerability assessment, and network access control and allows organisations to detect attacks leveraging rogue devices, among others [47, 57, 161, 179].

Creating and maintaining such an inventory requires the ability to classify devices based on information that can be observed on the network (e.g., their traffic patterns). However, tools commonly used to identify connected devices are not suited to the current enterprise environment. While traditional device classification relies on manually-defined fingerprints, it has limitations such as scalability and low performance in terms of *coverage/accuracy* and *granularity*. The first is the ability to classify/correctly classify a device, and the second requires it to be specific enough to be useful.

Machine learning-based alternatives have been proposed to overcome these issues. However, most of these approaches are "black-box", i.e., the features and classification algorithms used do not allow straightforward interpretation and actionability of the results. These solutions also require a large amount of labelled data for training and use models that require retraining when adding new types of devices. Moreover, related works adopting these solutions suffer several limitations, such as the narrow focus on IoT devices, thus missing the enterprise-wide picture, and evaluating limited datasets, both in terms of the number of devices and in the variety of their types.

3.1.1. Summary of Contributions and Outline of the Chapter

In this chapter we address the limitations of manual fingerprint-based and black-box machine learning-based device classification by proposing a similarity-based clustering method. Our solution uses unsupervised learning with a semantic-driven feature selection aiming to improve the coverage, accuracy, and granularity of an existing fingerprint-based classification engine. Unsupervised algorithms overcome the need for labelled data, which helps to keep up with the growth of the number and diversity of devices.

The underlying idea behind our approach is that devices with similar attributes have a similar type, allowing us to cluster them together and give them the same labels. Our solution can be broken down into the following steps. First, we select semantically meaningful attributes of devices as features. Second, we define distance functions to evaluate the similarity between devices based on these features. Third, we cluster devices which are similar based on the notion of distance we defined. Finally, the clusters are analysed, resulting in a number of classification suggestions that can be automatically applied or manually reviewed by an expert.

We evaluate our solution and compare it to a state-of-the-art fingerprint-based classification engine using data from 20,000 devices. The results show that we can successfully classify around half of devices that were left unclassified by the other engines. We also validate

our approach with domain experts to demonstrate its effectiveness to solve industry-wide problems.

The rest of this chapter is organised as follows. Section 3.2 discusses the background and related work. Section 3.3 states the classification problem that we aim to solve, while Section 3.4 describes our methodology to solve this problem. Section 3.5 shows the implementation and experimental results. Finally, Section 3.6 concludes this chapter.

3.2. Background

Device classification (also known as "device fingerprinting" [220, 229, 248]) works by applying a classification function on data about networked devices [71]. This data can be collected passively or actively. Passive techniques are non-intrusive, capturing packets as they move on a network. Active techniques interact with devices by sending them packets and observing responses or change of behaviour. Most commonly used classification approaches (e.g., Nmap[1], which is active, or p0f[2], which is passive) are rule-based, comparing data collected from a device against a set of fingerprints (or signatures) [3, 104, 263] that are manually designed. However, fingerprint-based classification is limited in terms of coverage, accuracy, and granularity [71].

Device classification is a well-researched problem, and recently the focus has been on classifying IoT devices [12, 18, 71, 110, 141, 155, 156, 186, 201, 215, 216, 218, 225, 233, 265]. While many proposed approaches intend to address the same problem, they diverge in multiple ways. First, the granularity of classification ranges from simply distinguishing IoT and non-IoT devices [180] to identifying the specific model and software version of a device [156]. Second, data collection ranges from passive techniques using network flows [110, 141, 218], packet headers [18, 154, 156], application layer data [155, 186, 215], or a combination of those methods [12, 201, 233, 265], to active techniques [19, 216]. Some research also investigates how to enrich network data with information found online [71]. A common limitation of these works is the scope of devices considered, which usually consists in smart home IoT devices and other consumer products. This type of setting does not reflect the broader range of devices found in enterprises, such as connected medical devices or operational technology equipment. In addition, the evaluation of classification methods demonstrated are often conducted in a lab setting with little or no actual user operation, which does not accurately represent "real-life" network deployments.

Recent solutions leverage machine learning to circumvent the expensive process of designing fingerprints manually. The majority of the work implements supervised learning algorithms [110, 141, 154, 155, 201, 215, 218, 233], including deep learning [18, 265]. These models not only require training with a sufficient amount of data for the devices to be classified, but they also require to be retrained each time a new device is introduced. This constraint limits such approaches to scale adequately as new devices are introduced. Additionally, these algorithms function as a black box, providing little insights into the rationale for classification. A network operator as an end-user would have limited information

[1] `https://nmap.org/`
[2] `http://lcamtuf.coredump.cx/p0f3/`

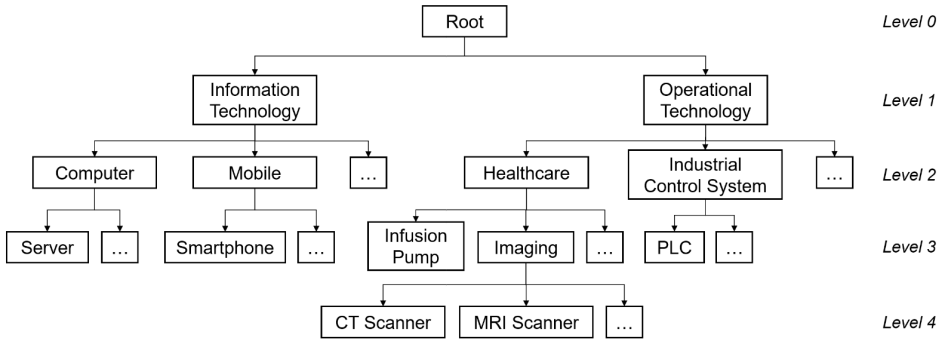


Figure 3.1: Excerpt of a taxonomy of device types.

about why a certain device has been classified as such, and no means of troubleshooting if the classification was inaccurate.

Techniques like clustering have been applied in the past to classify network traffic [66, 217]. Such algorithms are ideal to find similarity between entities and are well-suited to device classification by considering the semantic of the features and defining a proper similarity measure [261]. Our approach using a dissimilarity function follows the same principle used in Few-Shot Learning (FSL) algorithms for computer vision [205]. However, our dissimilarity function is not based on neural networks as it is required to solve the image recognition problem. Instead, we define it manually by using specific distance metrics applied to the selected features. As a result, it provides greater control by allowing the selection of any meaningful features for the distance function calculation, while also adding value by enabling its use with other problems than classification and image recognition. From the discussion above and to the best of our knowledge, there no other work that focuses on applying clustering with semantic-driven feature selection to classify devices.

3.3. Terminology

A *device* is a networked-enabled system fulfilling certain functions and delivering value to an organisation. Let us assume a set of devices D and a set of types are given. A *type* characterises the main function of a device, such as printer, or IP-camera. Device types are organised in a *taxonomy*, which is a finite directed tree with nodes labelled with strings, as shown in Figure 3.1. The label of the root is the empty string and no two children of a node share the same label. We also interpret the taxonomy as a partial order on the nodes, with the root being the smallest element. The *level of a node* is its depth in the tree, i.e., the length of the path to the root. We denote a specific node by the sequence of labels from the root, e.g., "/IT/Networking/Router" denotes a router. Let us assume a taxonomy of devices types, which we call DT, is fixed.

Definition 3.3.1. A *device classification function* $c : D \to DT$ assigns a device type in DT to each device in D, which we call the label of the device.

Given a classification function, the *level of a device* is the level of its label. A device is called *unlabelled* if its level is 0 and *labelled*, otherwise. Depending on its level, a labelled device can be qualified as *weakly-labelled* (the level is 1) or *well-labelled* (the level is at least 2). In the taxonomy DT, a level 0 label gives no information about a device, and a level 1 label gives very limited information. Hence devices with a label of level 0 or 1 (i.e., unlabelled and weakly-labelled devices) are referred together to as *to-be-classified* devices.

The fitness of a device classification function is measured in terms of *coverage* (percentage of well-labelled devices), *granularity* (the levels of the devices) and *accuracy* (percentage of correctly labelled devices compared to some ground truth).

Definition 3.3.2. We define the *device classification problem* as finding a device classification function that optimizes some fitness metric(s).

3.4. Methodology

We use similarity-based clustering to solve the device classification problem stated in Section 3.3. Figure 3.2 shows an overview of the methodology.

First, we capture semantically meaningful features describing the devices we want to classify. Second, we compute the distance between devices using distance metrics tailored for each feature. Third, we use a clustering algorithm that groups similar devices based on their distances. Fourth, we analyse the *quality* of resulting clusters and assess their *actionability*. Finally, we present the results to an operator. This white-box approach allows using the suggestions in different ways to improve the existing device classification function. Below, we explain each step in detail.

3.4.1. Feature selection

A device d can be represented as a tuple of features $(f_1, \ldots, f_n)$, where each feature f_i in a domain V_i captures an aspect of d. The device classification results are expressed in term of features to a user. They should have meaning and be directly understandable by a user. We call them semantically meaningful features.

3.4.2. Dissimilarity functions and distance

A dissimilarity function $dissim_i : V_i \times V_i \mapsto [0, 1]$ computes the dissimilarity between two devices d and d' with respect to a single feature f. A dissimilarity score of 0 represents the highest similarity between two feature values (i.e., they are identical) and 1 the lowest. For each feature, we define a dissimilarity function based on the feature's domain V_i.

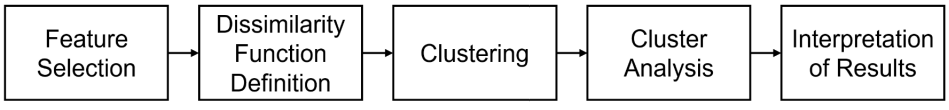


Figure 3.2: Overview of our methodology.

We then calculate the distance between two devices as follows. For a given feature vector $(f_1, \dots, f_n)$ along with their respective dissimilarity functions $(dissim_1, \dots, dissim_n)$, we define the distance D between two devices $d = (f_1, \dots, f_n)$ and $d' = (f'_1, \dots, f'_n)$ as the average dissimilarity per feature:

$$D(d, d') = \sum_{i=1}^{n} (dissim_i(f_i, f'_i))/n$$

with $D \in [0, 1]$, where 0 means that two devices have exactly the same values for every feature.

3.4.3. Clustering

In this step of the methodology, we input the distance D for a multiset of devices into a clustering algorithm. Its goal is to aggregate similar objects together while separating dissimilar ones in different clusters [126]. We adopt density-based clustering [26]. Density-based clustering searches for areas with a high density of objects separated by regions of lower density [126]. Additionally, it can find clusters of different sizes and shapes while also handling noise (i.e., objects that cannot be clustered) [268].

In our device classification context, density-based clustering is well suited as it follows a non-parametric approach. It does not require the specification of total clusters in advance, nor makes assumptions regarding their shape, unlike the parametric approaches. We leverage density-based clustering to find clusters of similar devices comprising both well-classified and to-be-classified, allowing us to classify the latter.

3.4.4. Cluster analysis

Using clustering algorithms requires a way to evaluate the quality of the results [97]. This evaluation, known as *cluster validation*, has been the topic of extensive research [15, 97, 98, 134, 147]. Clustering quality can be measured by a Clustering Validation Index (CVI) [98], such as the Silhouette Index or Davies-Bouldin Index [189]. Most CVI estimate the cohesion (also known as "compactness" or "tightness") and separation of each cluster, then combine these values into a quality measure [15]. However, these CVI consider clustering as an application-independent mathematical problem and provide little information about the *utility* of the clustering for a given use case [250].

For this reason, we also introduce a new cluster validation index designed to assist in our classification problem. The first component of this validation evaluates the quality of a cluster. The second component measures the actionability of a cluster from the point of view of a user.

Structural quality of a cluster

To evaluate the structural quality of a cluster, we look at the dissimilarity between its devices. We first look per feature how similar they are for that feature and consider them good if they are similar on at least some features. The feature dissimilarity AD_i of a Cluster C is

computed as the average of the dissimilarity $dissim_i$ between all devices in the cluster (with $avg(\emptyset) = 1$):

$$AD_i(C) = avg(dissim_i(f_i, f_i')|d, d' \; in \; C|f_i, f_i' \neq NaN)$$

We consider the feature similar if it does not exceed a threshold t_{diss}. We say a cluster is structurally *bad* if the count of similar features ($\#\{AD_i(C) < t_{diss}|i = 1 \ldots n\}$) does not reach a minimum threshold t_{good}.

Cluster actionability

The actionability of a cluster is assessed by performing a *heuristic-based cluster analysis*. To evaluate the actionability of a cluster we look at the specific use case; what can the user do with the cluster. For our use case of classification we consider the following *suggested actions* that may be assigned to a cluster:

- **Classify to-be-classified devices**: suggests a label for to-be-classified devices in the cluster.
- **Increase from level n to level $n + x$ label**: suggests an improved level $n + x$ label for level n ($\geqslant 2$) devices in the cluster.
- **Fix misclassification**: suggests the label of specific devices in the cluster may be incorrect.

Taking the suggested action from an actionable cluster improves the cluster, either with regards to coverage (first action), granularity (second action) or accuracy (third action, assuming the indicated devices were indeed classified incorrectly). While we consider that the action 'classify to-be-classified devices' can be taken automatically, the other two would be presented to a user. Of course, we still need a procedure that actually assigns these labels, and we discuss the one we use next.

If labelled devices in a cluster share the same parent at Level-2, we consider this a good indication that this Level-2 label would also apply for to-be-classified in the same cluster. Formally, we call a clusters *level-2-coherent* when (the cluster has well-classified devices and) the level of the greatest lower bound (GLB) of the labels of all well-classified devices in the cluster is at least 2:

$$level(GLB(\{label(d)|d \in C \wedge level(d) \geqslant 2\})) \geqslant 2$$

For instance, a cluster with only well-classified devices labelled as "/Information Technology/Mobile/Smartphone" or "/Information Technology/Mobile/Tablet" is level-2-coherent, since these devices share the same Level-2 label, "/Information Technology/Mobile". If a cluster is level-2-coherent, we assign 'classify to-be-classified devices' as an action and suggest the level-2 ancestor of the well-classified devices as the label for to-be-classified devices.

When a cluster is level-2-coherent, we look at the number of devices for each label in that cluster. We characterise as *main label* the highest label that has at least t% devices of the well-classified devices under it (with $t > 50$). If the main label's level is greater than 2 and

Table 3.1: Selected features, including their source, example after preprocessing, and similarity function.

Source	Features	Example	Function
DHCP	Class	'Printers'	Levenshtein
	Options	(53,61)	Jaccard
	Param.Request List	(1,121,3)	Jaccard
	Vendor	'Cisco'	Levenshtein
	Operating System	Windows Vista/7	Levenshtein
HTTP	Banner	(Mozilla/4.0, Microsoft)	Jaccard
	Headers	(Server Microsoft-IIS)	Jaccard
MAC	Vendor (24 bits)	'Hewlett Packard'	Levenshtein
	OUI	'cc9891'	Levenshtein
Nmap	Banner	(23/tcp telnet, 80/tcp http)	Jaccard
	OS	(Windows 7, Windows)	Jaccard
	Function	'Windows'	Levenshtein
	Open Ports	(139/tcp, 445/tcp)	Jaccard
p0f	Fingerprint	(Windows NT kernel 5.x)	Jaccard
Other	Group	(Network, Non Corporate)	Jaccard
	Network Function	'Windows Machine'	Levenshtein

if there are devices in the cluster with a lower level label, we assign 'increase from level n to level $n + x$ label' as action and suggest this label for the latter devices, thus increasing their granularity.

When a cluster is not level-2-coherent but does have a main label of at least Level-2, we suggest a 'Fix misclassification' action. The devices whose labels are not under the main label are the ones suggested to be misclassified.

3.5. Experimentation

This section details the implementation of the methodology outlined in Section 3.4, with explanations about the experimental setup and results obtained.

3.5.1. Implementation

We detail below the features and similarity functions chosen, as well as the clustering algorithm and cluster validation.

Features and similarity functions

The features we used in our implementation are presented in Table 3.1, with references motivating their use in device classification. Based on the value types of the selected features (strings and sets of strings), we choose two well-known similarity functions, namely the Levenshtein Index[175] and the Jaccard Index [117].

The Levenshtein Index, also called edit distance, computes the minimal number of insertions, deletions, and substitutions to make two strings the same [175]. It allows comparison of two strings of arbitrary length, thus it is suited to measure the distance between feature values of string type.

The Jaccard Index measures the similarity between two sets A and B as the cardinality of the intersection divided by the cardinality of the union of the sets, i.e., $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. The Jaccard Index is well-suited for features whose values are sets of strings. We implement a similarity function for those features (see Table 3.1) that first transforms the set of strings into two sets of tokens and then computes the Jaccard Index.

For each feature, its similarity function is used to compute a pair-wise similarity matrix as described in Section 3.4.

Clustering

We selected the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm [26, 27] for clustering. Like other density-based clustering algorithms, HDBSCAN keeps parameter tuning to a minimum, not requiring specification of a number of clusters nor assuming certain cluster shapes, and it handles noise (i.e., outliers). HBSCAN stands out by its capacity of finding clusters of varying densities and by being more robust to parameter selection [150]. HDBSCAN is also suitable for this use case because it does not require the distance between each pair of points, only expecting a connected graph. It allows us to use NaN values for uncomputed distances of missing features as defined in the previous subsection.

We use the implementation of HDBSCAN available at [150], which supports pre-computed distance matrices. This allows us to feed the algorithm with our devices' distances computed with our distance metric (see Section 3.4). We only have one parameter to tune: the minimum number of devices that should be together to be considered as a cluster. We set it to a value of 3. In the context of device classification, there is no requirement regarding the size of a cluster. Keeping this parameter low increases the chance for devices to be clustered with similar ones, which could ultimately help with classification.

Cluster analysis and interpretation of results

We analyse the clusters by implementing the cluster analysis presented in Section 3.4. The evaluation of the goodness of clusters is done as follow: With the 16 features we selected, we define a good cluster as one that has least 3 features ($t_{good} = 3$) with an average dissimilarity below 0.2 ($t_{diss} = 0.2$). These thresholds are chosen based on our empirical experimentation.

Regarding the heuristic-based analysis, we implement a function that identifies the type of clusters as described in Section 3.4. It analyses the labels of the devices, and based on the type of the cluster, it results in one or more suggested actions. For example, let us consider a cluster containing 20 devices: 15 labelled as /Information Technology/Mobile/Smartphone,

3 as /Information Technology/Mobile/ and 2 unlabelled (unknown). The resulting suggested actions for that level-2-coherent cluster are 1) Classify the two unlabelled devices and 2) Change Level-2 to Level-3 label for the three Mobiles.

3.5.2. Experimental setup

Dataset

For our experiments, we used datasets containing data up to 20,000 devices from a healthcare enterprise network. The devices found on these networks comprise a wide variety of types, ranging from traditional computers and servers, to IoT devices such as IP cameras. On some networks, industry-specific devices can also be found, such as patient monitors and other connected medical devices in hospitals.

Within the network, we capture device data by passively monitoring their communications with network monitoring tools such as p0f and actively scan them with Nmap and other tools. This data is fed to Forescout's existing classification engine, which assigns for each device a unique ID and applies a set of fingerprints against the data to assign a label to these devices. The classification engine has the limitations of fingerprint-based systems discussed in Section 3.2.

To create a dataset, we extract and store the devices and respective data as a collection of key-value pairs in a JSON file. Each device is represented as a tuple consisting of its unique ID, its label (if found by the classification engine, otherwise "Unknown") and its values for the 16 features selected (see Table 3.1).

In the end the dataset comprises 15,760 well-classified and 4,240 to-be-classified devices.

Evaluation metrics. To evaluate the fitness of our approach, we measured the coverage, granularity and accuracy of the clustering results as follows:

- **Coverage**: Number of to-be-classified devices having a suggestion of classification.
- **Granularity**: Number of level-2 devices with a classification suggestion to level-3 or higher.
- **Accuracy**: Number of devices potentially misclassified.

In our evaluation, we consider the initial classification from the fingerprint-based classification engine described above as baseline for the coverage (i.e., how many to-be-classified devices are in the dataset). It is non-trivial to obtain a comparable baseline for the two other metrics.

3.5.3. Validation

Before running the main experiment, we performed a 20-fold cross-validation to assess the solution's effectiveness by checking if the automatically assigned labels are correct. For this test, we used a sample of 20,000 well-classified devices with an accurate primary classification (i.e., ground truth). Using a database of 4.4 million devices from more than 200

different companies operating in various industry sectors, we observed that both in the overall database and in these companies' specific networks, there are around 21.2% of to-be-classified devices. With that in mind, we removed the label of 4,240 devices randomly selected from the sample to reproduce the same distribution of to-be-classified devices expected in a typical database. The tests resulted, on average, in 1387 clusters (1349 good clusters and 39 bad clusters) and 6,609 devices unclustered (i.e., noise points).

As shown in Figure 3.2, a Level-2 label in the taxonomy represents the minimum information about the type of the device (i.e., a computer, a mobile device, an ICS, etc) and gives a good amount of details about the device. Correctly suggesting a label up to at least Level-2 means that it achieves a consistent label, referred to in this cross-validation as a Consistent Level-2 Classification. In case that the suggestion is also correct to its full extent (i.e., the highest level of the primary label), it means that it achieved a consistent classification with a good granularity level; in this case, it will be called a High-Granularity Classification. For example, suggesting the label 'IT/Computer' to a device with the removed primary label 'IT/Computer/Workstation' means a Consistent Level-2 Classification, while suggesting 'IT/Computer/Workstation' means a High-Granularity Classification.

The 20-fold cross-validation results in an average of 2174 devices with a new suggested label out of the initial 4,240 to-be-classified. By comparing the label suggested in the experiments for the clustered devices with the primary classification removed from them, the methodology achieves, on average, 99.5% of Consistent Level-2 Classification and 95.6% of High-Granularity Classification during the tests. These results indicate that the methodology classifies devices with reliable labels and excellent precision, validating the following results.

3.5.4. Results

We execute our program with the dataset created and analyse the results from the main experiment described on Section 3.5.2. Among the 20,000 devices, 12,595 devices are grouped into 1,310 different clusters, while the remaining 7,405 devices are considered as noise points. From the 1,310 clusters, 1,285 are actionable and suggestions are provided which can help classifying a total of 1,539 to-be-classified devices (out of the initial 4,240). The amount of 1,539 to-be-classified devices candidate to classification can be broken down into 802 unlabelled devices (out of 2,234 unlabelled) and 737 Level-1 devices (out of 2,006). Moreover, the output of our experimentation suggests 36 Level-2 devices (within 29 clusters) that can be improved with a Level-3 classification, and 533 devices (from 120 clusters) that could be potentially misclassified.

The results can be used in two different use cases as mentioned in Section 3.4. In the case of automatic classification, 1,539 to-be-classified devices (802 unlabelled and 737 Level-1 devices) can be labelled automatically. In the case of manual classification, the end user is presented with the 249 clusters and their respective classification suggestions mentioned above. Assuming all the suggestions being validated by the user, it results in a total of 2,108 devices with an improved classification.

3.6. Chapter Conclusion

This chapter addresses the challenges of device classification based on traditional device fingerprinting and black-box machine learning by proposing a semantic similarity-based clustering method (RQ 1.2). We evaluate our solution and compare it to the state-of-the-art fingerprint-based classification using data from 20,000 devices. The results show that we can successfully classify a good amount of the to-be-classified devices among these with 99.5% consistency and 95.6% high granularity. Also, the validation of our approach using real-world databases establishes its effectiveness to solve industry-wide problems.

The results demonstrate that the method is effective. The proposed solution delivers consistent labels with high granularity to unclassified devices automatically while still allowing a user (e.g., a domain expert) to perform manual verifications on these labels. Thus, the method can reduce a significant portion of the work involved in manual and fingerprint-based classification. Another added value over black-box ML approaches is that our method provides to a user insights into the classification mechanisms due to the methodology's feature selection and distance function definition steps. Since we are in control of the whole process, we can better understand the results provided and fine-tune/adjust it if required. This white-box approach adds actionability and increases the results' reliability by combining automatic classification with manual verification.

Our solution shows that we can successfully remediate (in the most part) to the device classification problem in HDO identified in Chapter 2. Such novel approach enables more accurate device visibility and therefore helps improving the security posture by enhancing NSM capabilities in HDO. Our method also has the advantage of being industry-independent, and can therefore be easily used in other SCE or any modern environments.

While good device visibility is a significant first step toward effective NSM, addressing the cyber threats to HDO requires the ability to detect intrusion. Similarly to device classification, intrusion detection is a well-established concept in IT, which aims at detecting attacks and other illegitimate activities. It requires situational awareness with regards to the threats to an environments, such as knowledge about attack patterns targeting certain weaknesses of systems or protocols. As observed in the previous chapter, we identified a number of network protocols in HDO for which such awareness is lacking. We investigate in the next chapter a selection of healthcare protocols with the objective of providing the awareness and the knowledge to implement effective intrusion detection in HDO.

4

Security Analysis of Healthcare Protocols and Medical Devices

Previous security research conducted on healthcare communication protocols have led to the discovery of weaknesses which, if exploited, could impact patients' safeness (see Chapter 2, Section 2.3.2). Our analysis in Chapter 2 reveals the usage of other healthcare network protocols for which limited knowledge (from a network security perspective) is available. This leads us to wonder whether these protocols used in HDO networks could also have weaknesses that could impact patients if exploited. For this reason we conduct in this chapter vulnerability research on three healthcare protocols for which no vulnerability has been yet published. We find a number of vulnerabilities and we demonstrate in our lab four novel attacks exploiting them, highlighting the risks faced by HDO using the protocols selected.

The results reported here will be published (paper 6 of Section 1.4).

4.1. Introduction

Effective network security monitoring (NSM) requires both extensive visibility into network and knowledge of threats to that network. In the previous chapter, we addressed the first premise by developing a novel classification method providing greater network visibility. In this chapter, we focus on the second premise by identifying new threats to HDO's networks.

Our analysis in Chapter 2 highlights the usage of certain healthcare networks protocols for which we have limited knowledge, and no security analysis can be found in the literature. This can represent an issue, as these technologies may have weaknesses which could be leveraged by attackers to compromise networks' systems and data, ultimately putting patients' safeness at risk. For this reason, we conduct a security analysis on selected protocols and medical devices to identify previously unknown vulnerabilities. This research provides greater knowledge of threats relevant to HDO, helping us to better understand these environments and the risks introduced by certain technologies deployed.

Vulnerability research, a process aiming at discovering weaknesses in systems and communications, can provide the knowledge required to enhance the security posture a given environment. As an ongoing process, it is relevant to safeness-critical environments (SCE) due to the increasing diversity of technologies (devices and protocols), demanding constant effort to find and fix weaknesses. Thanks to the growing body of research conducted on connected medical devices (e.g., [11, 124, 198, 232, 262]) and healthcare network protocols (e.g., [73, 79, 151]) we are aware of (some of) the risks to patients' safeness introduced by the technologies deployed in HDO.

Vulnerability research on healthcare protocols and connected medical devices contributes in multiple ways to the improvement of NSM solutions such as IDS for HDO. It enables the creation of signatures for detection systems, giving them the ability to detect attacks targeting the newly found vulnerabilities. It also allows the creation of attack datasets relevant to HDO, which can be used for testing novel environment-specific IDS and other security tools. In fact, new datasets are needed to evaluate security measures and develop new ones. As shown in [106], current datasets available are often limited or ill-suited to reflect the modern threat landscape (e.g., lack of real-network attacks, or large number of deprecated threats).

Performing vulnerability research on critical environments such as HDO can be challenging. Contrary to IT networks, for instance, where practices such as penetration testing or red team engagements are common, experimentation in HDO's "live environments" is not feasible for two main reasons. First, security testing can introduce risks for HDO's operations by altering the functioning of devices. Common operations such as port scanning can crash systems that are connected to patients and deliver health care [75]. Second, obtaining "real" data to conduct research (e.g., for protocol security analysis) can be difficult, especially when it contains sensitive patients' information. It is therefore essential to have dedicated labs to perform security research in safe and controlled environments, yet realistic enough to guarantee the validity and accuracy of the experimentations.

4.1.1. Summary of Contributions and Outline of the Chapter

In this chapter, we perform security research on healthcare protocols and medical devices, and identify novel vulnerabilities that have not been published yet (to the best of our knowledge). The healthcare communication protocols considered are the *POCT1-A* and *LIS02-A* standards, as well as the proprietary protocol *Data Export* from Philips.

We start by detailing our experimental setup, including the selection of targets and how to build a medical lab representing a segment of a typical HDO's network (Section 4.2). We also provide the necessary background information about the functioning of the devices and protocols. Then, we demonstrate a number of attacks targeting these devices and protocols, highlighting the risks to patients' safeness in new ways (Section 4.3).

This research confirms the need for both better network inventory and device classification (see Chapter 3), as well as up-to-date security controls which would better fit HDO networks specificities and technologies used. Our work contributes to the improvement of IDS in two ways: 1) new signatures can be created to detect our novel attacks and protect patients, and 2) new attack datasets can be made to assist the development and testing of HDO-specific IDS.

4.2. Experimental setup

We explain in this section the objectives of our research and the selection of protocols and medical devices to analyse. Finally, we present the design and setup of our testing lab.

4.2.1. Objectives & Methodology

We aim at discovering vulnerabilities in a selection of healthcare protocols and medical devices, and demonstrating attacks exploiting them. More specifically, our objective is to create a number of novel network attacks which could impact the safeness of patients in HDO. The purpose of these attacks is to enable the creation of new IDS signatures and attack datasets relevant for this SCE, ultimately improving network monitoring capabilities [106]. While this work follows the examples of previous healthcare security research (e.g., [30, 73, 79]), we focus on different protocols than the ones already described in the literature (HL7 [101], DICOM [161], and RWHAT [151]). Our work contributes to the understanding of how prevalent are the risks linked to medical devices and healthcare protocols in HDO networks.

Based on our extensive research conducted on HDO networks (see Chapter 2), we identify and select healthcare protocols commonly found for which no attack has been yet published. We then acquire devices using those protocols. Our objective is to first understand the communication patterns between the devices and how the protocols operate. We specifically look at how the packets are formed, and what kind of data they convey, among other things. We then investigate whether these communications can be abused by, for instance, spoofing devices, intercepting packets and tampering values on the fly. Furthermore, we want to identify if the medical devices can be remotely controlled by sending commands via these protocols, similarly to other protocols like BACnet [185].

While it is important to understand how the devices and protocols work to find weaknesses, we limit the explanation of their functioning and specifications to the amount relevant for our study. We focus here on the demonstration and consequences of the novels attacks we found, and invite the curious readers to refer to the specifications of the devices and protocols for deeper understanding.

4.2.2. Targets

The protocols we investigate are POCT1-A2 [60], LIS02-A [166] and Data Export [190]. The two first protocols are used for interconnecting point-of-care and laboratory devices with information systems such as Laboratory Information Systems (LIS) or Data Management System (DMS) [60, 166]. As seen in Chapter 2, these protocols enable the exchange of clinical results and patient data between laboratory instruments and information systems [166]. Additionally, they can be used by clinical staff to issue test orders and remote requests to devices. Data Export is a proprietary protocol used by Philips bedside patient monitors [190]. It allows these devices to communicate vital readings to a Central Monitoring System (CMS).This protocol can be regarded as similar to the GE RWHAT protocol mentioned above. CMS are commonly used in hospitals to remotely display the vitals of several monitors, allowing nurses to watch over multiple patients from a single location.

We acquire several devices that use the selected protocols and are popular among the HDO we studied in Chapter 2. Obtaining these devices enables us to generate real data and observe "live" communications. This situation helps greatly in understanding how the protocols operate, and also to test attacks. We choose a Philips IntelliVue MP50 patient monitor for analysing Data Export, and a Siemens DCA Vantage blood analyser for both POCT1-A2 and LIS02-A. The patient monitor is used to track in real time the vitals of a patient, while the blood analyser measures glycaemia in patients with diabetes and detects early kidney disease.

Note: The LIS02-A standard is a revision of the former ASTM E1394-97 standard [166]. While the specifications of the Siemens DCA Vantage cover the usage of ASTM, this information applies to LIS02-A.

4.2.3. Healthcare lab design

We start our security research by setting up a lab designed to represent a segment of a typical HDO's network, as we observed in Chapter 2. It enables us to experiment with attacks in a controlled and safe environment. In this section, we present our lab by first introducing its architecture, then the tooling installed on the devices, and finally the basic functioning of the devices and protocols.

Architecture

The lab we build is depicted in Figure 4.1. It consists of the aforementioned medical devices, as well as a collection of computers, interconnected all together via a network switch. One computer is configured as a CMS to display the real-time readings of the patient monitor, and another as a LIS server that stores test results from the blood analyser. The devices

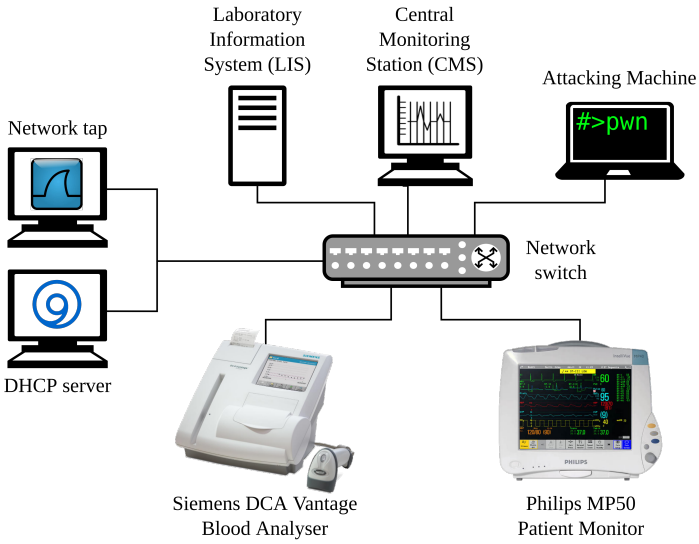


Figure 4.1: Network layout of our healthcare lab.

require to be provisioned an IPv4 address upon boot before being able to communicate. To fulfil this requirement we add a Raspberry Pi 3 to the network and configure it as a DHCP server. It is also important for protocol analysis to be able to observe real-time network traffic as it is being generated by the devices. For this reason, we configure one port of the switch as a SPAN port on which all network traffic is mirrored, and we connect a computer to it to be used as a network tap.

All devices are connected to the same network switch and there is no segmentation in place. Although it is an oversimplification, we discussed in Chapter 2 how, in practice, networks in HDO can be found improperly segmented. The consequence is that sensitive devices (and data) may be reachable not only by legitimate systems used by medical staff, but also by guests and adversaries that have access to the network. To replicate this situation, we connect a laptop (referred to as *Attacking Machine* on Figure 4.1) to the switch to represent an attacker present on the network. We elaborate on the capabilities of the attacker later in Section 4.3.

Software & Tooling

For the lab to be operational, we need to configure the devices and install a number of software. While the medical devices are deployed with their default configuration and the network switch is configured as described above, we install the necessary tools on the other devices as follow:

- **LIS:** Since no suitable open-source solution for LIS can be found, we implement a simple LIS02 server using Python ASTM and a POCT01 server according to the communication specifications of the Siemens DCA Vantage analyser [213].

- **CMS:** We install on the CMS the IxTrend Express software[1], which records medical signals from patient monitors. The program was selected for its support of the Data Export protocol.
- **Network tap:** We install the software Wireshark[2] to capture and analyse network traffic. This tool is a powerful protocol analyser, enabling us to inspect packets as they are passing on the network.
- **DHCP server:** We configure the Raspberry Pi 3 as DHCP server by installing Bind9[3]. The device answers to the DHCPDISCOVER messages broadcasted by the devices on the network and provides IPv4 addresses to them.
- **Attacking machine:** We install the tool Ettercap[4] for traffic redirection and packet modification, and Scapy[5] for packet crafting. Additionally, we deploy on this machine a copy of our custom LIS02 server which is used as a *rogue LIS server*. We elaborate further on how these tools are used when presenting the attacks and proofs of concept.

  Note: In a real situation, an attacker would likely use a broader collection of tools to guarantee the success of her objectives. However, we consider for our scenario only the tools required for the execution of the attacks we present in Section 4.3.

Basic functioning of the selected medical devices

We describe below how the devices in the lab interact with each other over the healthcare communication protocols we are interested in. In particular, we look at how the devices are configured to operate on the network, then how communications are established and how data is exchanged.

Philips MP50 Patient monitor: We first need to configure the patient monitor to communicate with the CMS over the Data Export protocol. Data Export is a message-based request/response protocol, and uses the Local Area Network (LAN) interface and the standard UDP/IP transport protocol [190]. The configuration is completed as follow. After being connected to the switch and booted up, the Philips MP50 begins by requesting an IP address on the network using the BOOTP protocol. The DHCP server answers to that request by supplying an address.The patient monitor then starts multicasting *Connect Indication Event* messages in a Data Export packet over UDP to port 24005 to signal its presence to a potential CMS on the network. These messages contain information about the patient monitor such as the serial number, product number, hardware revision, appliance software, and software revision. The Philips MP50 keeps on sending these messages until an association is established with a CMS.

The CMS on the network can establish an association with the patient monitor as described

[1] https://www.ixellence.com/index.php/en/home/17-default-en/products
[2] https://www.wireshark.org/
[3] https://gitlab.isc.org/isc-projects/bind9
[4] https://www.ettercap-project.org/
[5] https://scapy.net/

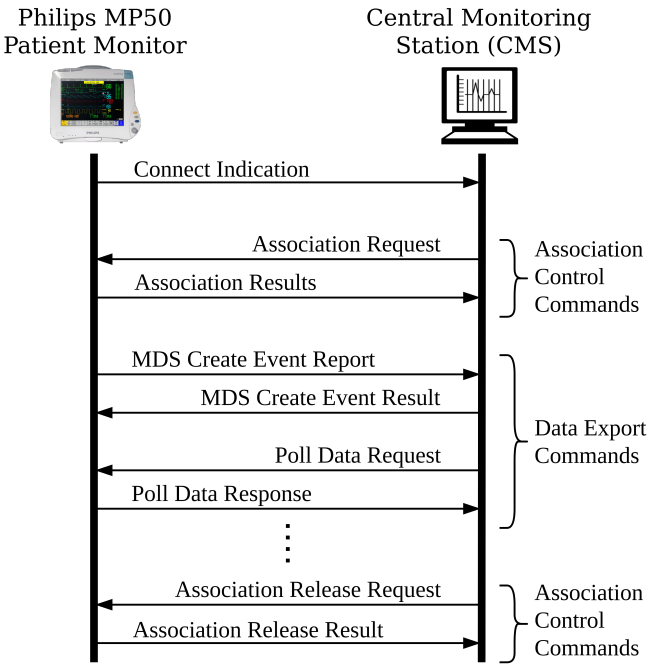


Figure 4.2: Communication sequence between the Philips MP50 patient monitor and the CMS used to establish an association, transfer data and close the connection.

in detail in [190] (p.25). The association is created (and terminated) according to a certain sequence of messages depicted in Figure 4.2. When the CMS receives a Connect Indication message, it can send an *Association Request* message. The Philips MP50 receives it and sends back an *Association Result* message, to either accept or refuse the association. If the association is accepted, the patient monitor sends next an *MDS Create Event Report* message, containing information about its system and its configuration. This message must be acknowledged by the CMS with an *MDS Create Event Result* message. From this moment on, the CMS can send *Poll Data Request* messages to the patient monitor to retrieve information such as vitals measurements, alerts (e.g., patient alarms), patient demographics or system attributes (e.g., version numbers). The Philips MP50 replies to these requests by sending *Poll Data Response* messages containing the data queried. The CMS can chose to close an association by sending an *Association Release Request* message to the patient monitor, which will be acknowledged in return with an *Association Release Result* message from the patient monitor. In case of a communication issue, the Philips MP50 can send an *Association Abort* message to the CMS, indicating that the association has been stopped (not shown on Figure 4.2).

Siemens DCA Vantage blood analyser: The configuration of the device is done manually by adjusting the settings directly on the device. In the Ethernet port settings menu can be found the relevant configuration options for our study. Particularly, the IP address of the

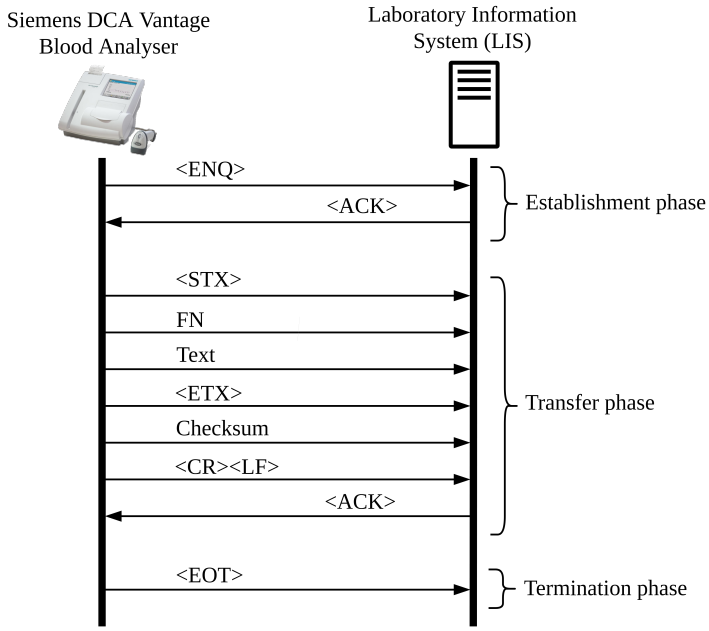


Figure 4.3: Communication sequence between the Siemens DCA Vantage and the LIS server over LIS02-A protocol.

blood analyser can be set to be either automatic (provided by the DHCP server), or static (entered manually on the device). Next, we specify the IP address and port number of the LIS server we want to connect to. Finally, we select which protocol to use: the Siemens DCA Vantage can communicate with the LIS server over the LIS02-A and POCT1-A2 protocols.

Since we use both in our study, it is important to understand their basic functioning described below:

- **LIS02-A**: When using LIS02-A, a *session* is created every time an operator wants to transmit data from the Siemens DCA to the LIS server, according to the procedure depicted in Figure 4.3. As described in the Siemens DCA's specifications [213] (p. 6), a session consists of three distinctive phases: *Establishment*, *Transfer* and *Termination* phase. In the Establishment phase, the Siemens DCA Vantage requests to establish the direction of communication by sending an enquiry character `<ENC>`, which is acknowledged by the LIS if it is ready to receive data. In the Transfer phase, the laboratory instrument can send messages as follow: it sends a Start of Text character `<STX>`, followed by the frame number FN (value from 0 to 7) and the Text corresponding to the content of the message to send. Finally it finishes the transmission by sending a End of Text character `<ETX>`, a Checksum and the end of frame characters `<CR><LF>`. The LIS replies back with an `<ACK>` if the checksum value is correct. Finally, in the *termination phase*, the blood analyser sends an End of transmission character `<EOT>` to the server, after which both devices return to

```
H|\^&|||DCA Vantage^01.00.00.00^A123456||||||P||20061002200015<CR>
P|1|987654|||Doe^Jane<CR>
C|1|I|age^39|G<CR>
O|1||333^9012<CR>
R|1|^^^HbA1c|2.5|%|4.0 to 6.0|<|||||20061002183420<CR>
C|1|I|1.000^0.0 %^NGSP|G<CR>
L|1|N<CR>
```

1 2 3 4

Figure 4.4: Example of a LIS02-A message containing the result of an HbA1c test. Source:[166]

their initial state.

Various information can be carried in LIS02-A messages: data about the laboratory instrument and the information system, patient (e.g., name, birthdate, address, phone number, known diagnosis, etc.), test order (e.g., specimen ID, priority, ordering physician, etc.), and results (e.g., measurement value, unit, reference ranges, result abnormal flags). Figure 4.4 depicts an example of a LIS2-A message containing the result of Haemoglobin A1C test (or HbA1c test): by measuring the amount of glucose in the blood, these tests may be used to diagnose diabetes in adults [153]. LIS02-A messages consist in multiple *records* (each line represents a different record), which carry specific information.

On Figure 4.4, some parts of the records of interest for our study are highlighted in red boxes. The first record is the *Message Header Record* (`H`), and contains sender information (product-code, software-version, serial number) (box 1), as well as the timestamp of message transmission (following the format YYYYMMDDHHmmSS) (box 2). The second record is the *Patient Information Record* (`P`), specifying the patient ID and name (box 3). The fifth record is the *Result Record* (`R`), displaying the universal test ID ("HbA1c"), the measurement ("2.5"), the unit ("%"), the reference range ("4.0 to 6.0"), and the timestamp of the beginning of the test analysis (box 4). A complete overview of LIS02-A messages can be found in [166].

- **POCT1-A2**: The Siemens DCA Vantage can communicate with the LIS over flows of POCT1-A2 messages referred to as *conversations*, which consist of a number of *topics* exchanged. As described in the Siemens DCA's specifications [213] (p. 36), two types of conversations are supported: *Basic profile*, where a conversation is initialised, data is transmitted and the conversation is terminated, and *Continuous Mode*, where the conversation remains open after initialisation, allowing the blood analyser to send status change, device events and observations (i.e., test results) as they occur.

  As depicted in Figure 4.5, a Basic profile conversation starts with the Hello topic, in which the Siemens DCA Vantage informs the LIS that it is ready to communicate. After receiving an acknowledgment of the LIS, the blood analyser sends a Device status message indicating its current level of readiness (e.g., ready, busy, locked). After acknowledgment, the conversation goes to the Observations topic where the LIS requests observations (i.e., test results). The Siemens DCA Vantage sends them back, the LIS acknowledges the reception, and the blood analyser finishes the transmission by sending an End of Topic message. The LIS then sends a directive to the device to

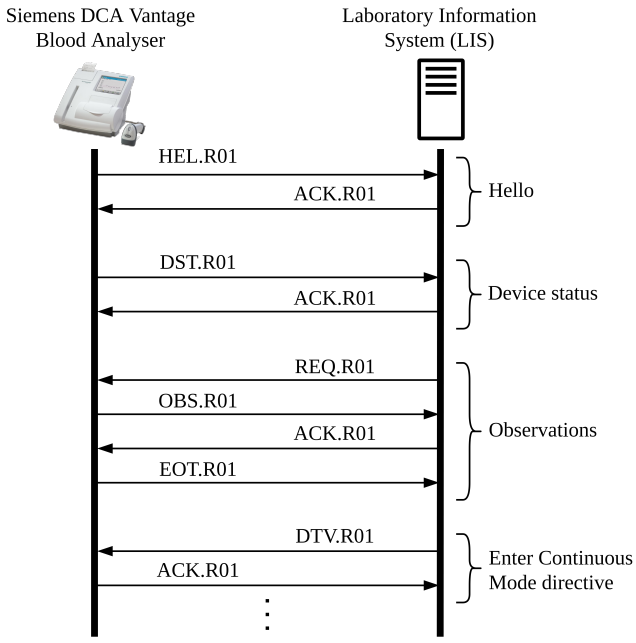


Figure 4.5: Communication between the Siemens DCA Vantage and the LIS server over POCT1-A2 protocol.

enter the Continuous Mode: according to the device's specifications, the Siemens blood analyser requires a Basic profile conversion to be followed by a Continuous Mode conversation [213]. From there on, the communication is maintained and a number of other topics can be used (see [213], p. 37). For our study, understanding the functioning of the Basic Profile conversation is sufficient. The details of the Continuous Mode conversation can be found in [213].

Once the conversation is established, there is a number of message types that are supported by the Siemens DCA Vantage, such as Request Observation Message (`REQ.R01`), Patient Observations (`OBS.R01`), Device Status (`DST.R01`) or even Remote Command Directive (`DTV.SIEM.DVCMD`). The exhaustive list and respective description can be found in [213] (p. 34). Patient Observation messages contain the results of a patient test. Such results are sent from the blood analyser to the LIS at the end of a test, or when a device operator performs a test recall. An example of an HbA1c result formatted according the POCT1-A standard is depicted in Figure 4.6. It represents the POCT1-A equivalent of the LIS02-A test result shown previously in Figure 4.4. The POCT1-A standard uses the XML format for the exchange of messages. The relevant parts of the message for our study are highlighted in red boxes on the figure. The first box contains the Service (SVC) information, indicating that the message is a patient test results (OBS), and it also includes the measurement timestamp and the sample sequence number. The second box displays the patient data (PT), which are the patient ID and name. The third box contains the actual obser-

```
<OBS.R01>
    <HDR>
        <HDR.control_id V="10003"/>
        <HDR.version_id V="POCT1"/>
        <HDR.creation_dttm V="2006-10-02T20:00:15-00:00"/>
    </HDR>
    <SVC>
        <SVC.role_cd V="OBS"/>
        <SVC.observation_dttm V="2006-10-02T18:34:20-00:00"/>
        <SVC.reason_cd V="NEW"/>
        <SVC.sequence_nbr V="333"/>                                    1
        <PT>
            <PT.patient_id V="987654"/>
            <PT.name V="Jane Doe">
                <FAM V="Doe"/>
                <GIV V="Jane"/>                2
            </PT.name>
            <OBS>
                <OBS.observation_id V="HbA1c" SN="SIEM" SV="1.0"/>
                    <TRANSLTN V="HbA1c"/>
                </OBS.observation_id>
                <OBS.value V="2.5" U="%"/>
                <OBS.method_cd V="M"/>
                <OBS.interpretation_cd V="L"/>
                <OBS.normal lo-hi limit V="[4.0;6.0]" U="%"/>    3
            </OBS>
        </PT>
        <RGT>
            <RGT.name V="DCA HbA1c"/>
            <RGT.lot_number V="9012"/>
            <RGT.expiration_date V="2008-05-31"/>
        </RGT>
    </SVC>
</OBS.R01>
```

Figure 4.6: Example of a POCT1-A2 message containing the result of an HbA1c test (similar to the test presented in Figure 4.4).

vation results (OBS): the unique test identifier (“HbA1c”), the test value and unit (“2.5%”), as well as the reference range (“[4.0;6.0]”) among other things.

4.2.4. Remarks

Before deploying the lab as described above, we first perform a vulnerability assessment of the two pre-owned medical devices which were obtained via an online auction site. While there existed no authoritative methodology for medical device assessment at the time of testing, we leverage various sources such as [102] to structure our work. Starting with a function evaluation, we set up and configure the devices to operate according to their “normal specifications” in order to understand how they work. We then perform a physical inspection to evaluate the physical attack surface of the devices. Based on the results, we then attack the exposed interfaces identified. We spot several issues with respect to the DCA Vantage and its configuration that we explain below. No vulnerability was found in the MP50 patient monitor in the time allotted for the analysis.

The Siemens DCA Vantage blood analyser runs a full-fledged Windows Embedded CE. Users operate the device via the restrictive user interface (RUI) called the kiosk app [30]. This application limits the actions that can be executed by a regular user (e.g., a nurse),

and provides an authentication mechanism to allow a privileged user (e.g., technician) to perform maintenance operation and change the system configuration. While users should not be capable of escaping from the kiosk app, we find that it is possible to break out from this application by connecting a keyboard to the exposed USB port and using specific keystroke combinations. This allows us to gain access to the underlying OS and file system, granting us the possibility to run arbitrary code and retrieve files stored on the system. This *Improper Access Control* vulnerability was reported to Siemens and assigned the number CVE-2020-15797.

We also find that the DCA Vantage contains several pieces of sensitive and confidential information. For instance, we find a database in the system which contains data such as administrator password, test results and device logs. The password corresponds to the four-digit pin required to authenticate a privileged user (for the authentication mechanism mentioned above). This vulnerability, commonly referred to as *Use of Hard-coded Password*, was assigned the number CVE-2020-7590.

Finally, our findings also reveal potential privacy issues. The test results stored in the database we retrieve are presumably from a previous owner. Although we did not investigate further the prevalence of sensitive data in second-hand medical devices, this shows that HDO should pay special attention to data confidentiality when disposing of medical devices. There should be a process in place to securely decommission devices, as well as functions implemented in medical devices to allow for the secure erasure of data.

4.3. Experimentation

In this section, we demonstrate in our lab four attacks which could impact the patients' safeness in HDO. These attacks leverage the weaknesses found in the healthcare protocols we investigated.

4.3.1. Attacker model

The attacker model we choose for our analysis is one of an attacker who wants to disrupt a healthcare network by launching cyber-physical attacks against medical devices to interfere with patient care. The attacker can be driven by various motivations, such as social and economical destabilisation, profit through ransom, or sheer "amusement". Although this model assumes the threat actor to be already inside the network, we mentioned earlier how such access can be trivial to obtain.

We consider an attacker having access to the network in one of the two following ways: via remote access or via physical access. Remote access to an HDO's network can be established through the compromise of a device reachable over the Internet or via phishing [105]. Physical access can be obtained by connecting a rogue device onto an exposed network sockets. The ease of obtaining such access is common in hospitals since sockets are often used in patient rooms to connect medical devices that interface directly with patients [115, 161].

An attacker having access to the network can leverage a man-in-the-middle (MITM) position. This consists in the attacker's ability to redirect communication flows to her, and

Table 4.1: Attacks on healthcare protocols implemented in the lab.

Attack #	Type of attack	Impact on CIA	Protocol	Description
1	Spoofing	Confidentiality	POCT1-A2	Retrieve test results stored in blood analyser
2	Tampering	Integrity	LIS02-A	Change test results sent from blood analyser to LIS
3	Denial of Service	Availability	Data Export	Abort connection between patient monitor and CMS
4	Tampering	Integrity	Data Export	Change real-time pulse reading shown in CMS

therefore allowing her to be *logically* in between devices. Such position enables her to intercept, read and/or modify packets on the network. Additionally, we consider an attacker who has already performed network reconnaissance activities [5] and identified the target devices and their IP addresses.

4.3.2. Attacks

We implement four attacks using the protocols POCT1-A, LIS02-A and Philips Data Export, as described in Table 4.1. For each attack, we outline in the table its type and impact on the confidentiality, integrity and availability (CIA) of assets according to the STRIDE threat classification model [212]. This popular model is commonly used to identify weaknesses in technology [30] and has also been successfully applied to CPS [119, 146]. Following the STRIDE framework, the four attacks we demonstrate in this section are classified along the following three types:

- **Spoofing**: the attacker pretends to be a legitimate system.
- **Tampering**: the attacker violates the integrity of data by modifying it.
- **Denial of Service (DoS)**: the attacker violates the availability of data and system by disrupting system operation.

We present each attack below by outlining first the goal and impact. We then introduce the technical background relevant to the understanding of the attack, before explaining the scenario of the attack. We finish with a demonstration of our proof of concept where we implement the aforementioned scenario and execute the attack in our lab.

Attack 1: Retrieving test results

Goal and impact: This attack consists in the interception of blood test results from the DCA Vantage analyser. It impacts the confidentiality of patient data, where the attacker obtains a number of sensitive information contained in the POCT1-A packet. This attack highlights the risks to patients' data privacy, and can be reproduced with any two devices communicating over an unencrypted and unauthenticated POCT1-A channel.

Technical background: Let us first consider an example of a blood test result stored in the Siemens DCA Vantage as shown in Figure 4.7. It displays the result of an HbA1c test of

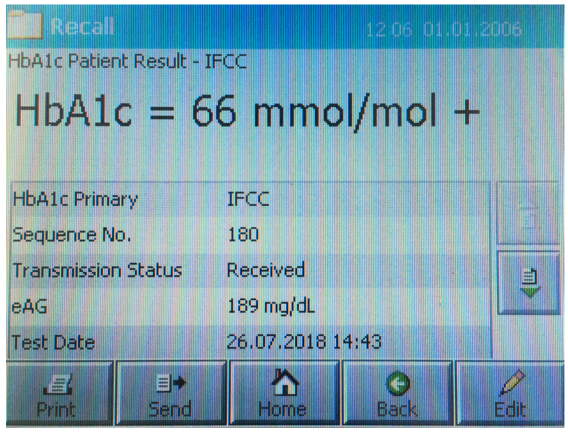


Figure 4.7: HbA1c test result displayed on the screen of the Siemens DCA Vantage.

66 mmol/mol, which could be indicative of diabetes. This result, dating from the 26th of July 2018, was already stored in the device when we received it (see Remarks above). We use this test result throughout our experimentation and attacks below. For sanitary reasons, we did not conduct blood tests ourselves in our lab. The interested reader can refer to the DCA Vantage Operator's manual [214] to learn how these tests are practically performed in an HDO setting.

When the operator (e.g., a nurse or a lab analyst) chooses to send a test result to the LIS, a POCT01-A packet containing the result is generated and sent over an established connection between the DCA Vantage and LIS. We execute the transmission of the aforementioned test (pressing the "Send" button on the Siemens DCA's screen shown in Figure 4.7). By using our network tap, we capture the packet and retrieve the payload in which we find the XML-formatted test result, as shown in Figure 4.8. The content of the payload is somewhat similar to the example shown in Figure 4.6. As expected, the data and test results contained in the payload are the same as the ones displayed on the blood analyser (see Figure 4.7).

Attack scenario: An attacker can retrieve test results in one of these two ways. The first way consists in passively intercepting POCT1-A packets as they are transmitted on the network. Since the packets are sent in clear text, the attacker can simply sniff the network traffic and examine the content. The limitation of this approach is that the attacker has to wait for an operator to manually initiate the transmission of the results to the LIS.

The second way an attacker can retrieve the test results is by sending a specific command via POCT1-A to the blood analyser (or any other POCT1-A devices). Knowing that a LIS server can send commands via POCT1-A to a device, an attacker can hijack the communications between a POCT1-A device and a legitimate LIS server (e.g., via ARP cache poisoning) and spoof the server. She can then send a limited set of remote commands to the target device. Depending on the commands supported by the device, the attacker will

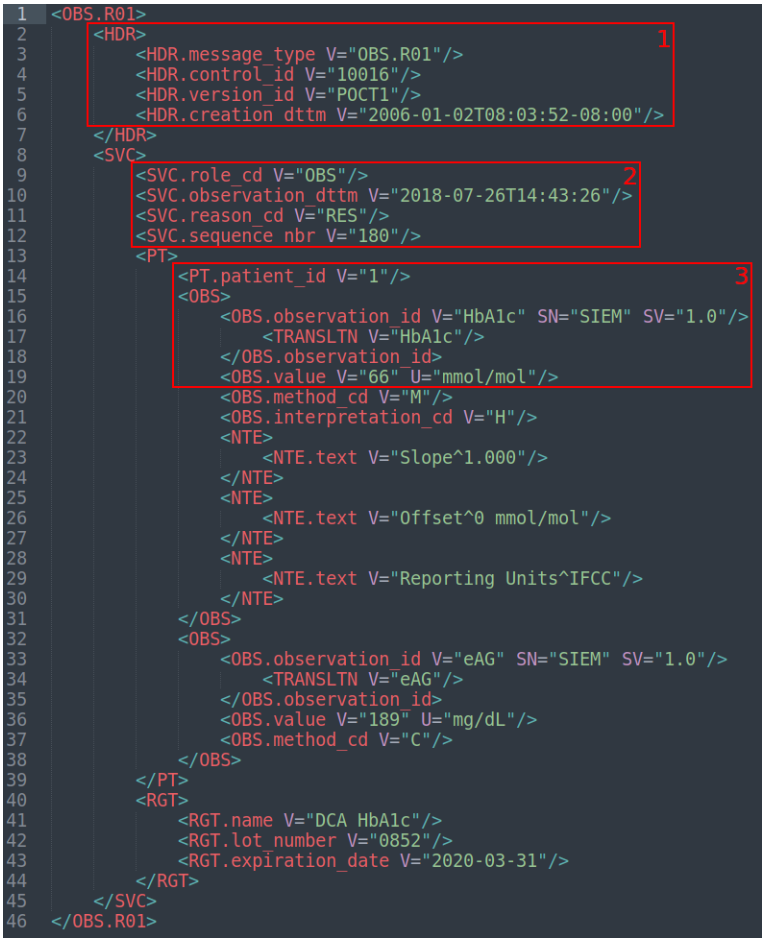


Figure 4.8: Details of the POCT1-A packet containing the HbA1c test results sent from the Siemens DCA Vantage to the LIS. The relevant parts of the message for our study are highlighted in red boxes. Box 1 contains the header of the message which specifies, among other things, the message type ("OBS.R01" for Patient Observations message), a control ID which uniquely identifies the message, and a timestamp corresponding to the date and time at which the message was sent. Box 2 highlights the Service section of the message, as introduced before. Box 3 contains the patient ID and the result value for the HbA1c test.

be able to, for example, force the device to send all pending test results to the LIS server (which she will intercept), or update the list of device administrators for example. These commands can be useful for the attacker depending on the motivations and/or objectives.

A malicious actor can easily pull such an attack in a hospital by connecting a small device like a Raspberry Pi embedding a rogue LIS server to the network via an exposed network socket. A similar attack targeting the DICOM protocol has been successfully demonstrated in a real hospital setting in [161].

Proof of concept: We demonstrate in our lab the second scenario described above by performing the following steps. From the attacking machine, we first run our rogue LIS server and perform an ARP cache poisoning attack with Ettercap to spoof the legitimate LIS server. This forces the DCA Vantage to communicate with our rogue server. Following the device's specifications [213], our server is configured to respond to the blood analyser's "Hello message" (`HEL.R01`) with an acknowledgement message (`ACK.R01`). Our server then requests pending tests results by sending a "Request message" (`REQ.R01`) to the target device. The DCA vantage replies by sending the test results. After a short conversation sequence (detailed in [213]) a continuous conversation mode is established between the DCA Vantage and our rogue device. In this mode, all further test results will be sent directly to our rogue LIS server. Moreover, the DCA Vantage will accept a limited set of commands from the server, such as to update the list of the device's operators (`OPL.R01`).

Attack 2: Changing test results

Goal and impact: The goal of this attack is to tamper with a test result being sent from the DCA Vantage analyser to the LIS over the LIS02-A protocol. With such ability, an attacker can modify healthy results to abnormal ones, or the other way around. This impacts the integrity of the patient data, which can ultimately affect the patient's health. This attack could be reproduced with any two devices communicating over unencrypted and unauthenticated LIS02-A.

Technical background: When the operator chooses to send a test result to the LIS via the LIS02-A protocol, a packet such as the one shown in Figure 4.9 is generated and sent over the network. When the LIS server receives the packet, it displays the results, as shown in Figure 4.10a and stores them internally.

Attack scenario: Let us consider a scenario where the attacker wants to tamper with this data flow and send incorrect results to the LIS. The procedure to implement this attack consists in the following steps. The attacker first executes a MITM attack to redirect the communication flow to her. When the test result is being sent by the laboratory device, she can intercept and drop the original packet. The attacker then crafts a new packet with a modified test result, and adjusts the checksum value by computing a new one before sending the packet. Finally, the crafted packet is received by and stored in the LIS.

Proof of concept: We demonstrate the aforementioned scenario by implementing an attack in which we change the result of an HbA1c test which could be indicative of diabetes

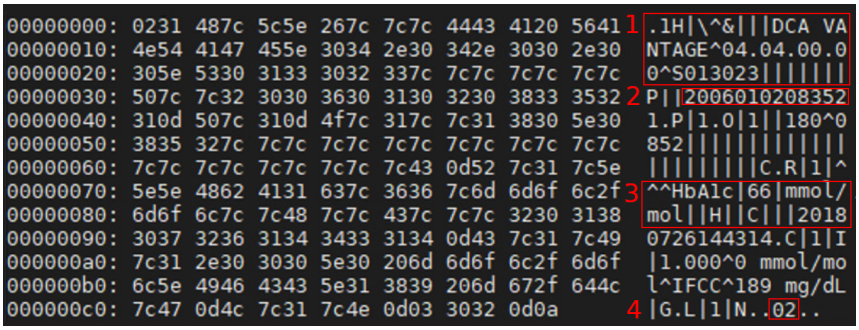


Figure 4.9: Details of the LIS02 packet containing the HbA1c test result sent from the Siemens DCA Vantage to the LIS. The format of the packet is somewhat similar to the example given above in Figure 4.4. We find relevant information for our study highlighted in red boxes, such as the Message Header record (box 1), the timestamp of transmission (box 2), the result value (box 3), and the checksum (box 4).

(66 mmol/mol) to a normal result (41 mmol/mol). We use the test result depicted in Figure 4.7. We follow the procedure outlined above by using Ettercap to establish a MITM position and create a custom filter that modifies test results and checksums accordingly. An example of filter creation is provided on the Ettercap's GitHub repository[6].

We begin the attack by running Ettercap from the attacking machine and hijacking the connection between the DCA Vantage and the LIS server. Next, we execute the test result transmission from the blood analyser using the test result already stored in the device (depicted in Figure 4.7). In a normal situation, when the analyser sends the test result (value of 66 mmol/mol), the LIS would receive the packet as shown in Figure 4.10a. However, after launching our attack, the Ettercap filter we developed intercepts and modifies the packet by setting the test result to the value of our choice (41 mmol/mol) and adjusts the checksum. The crafted packet is then sent to the LIS server and we can observe the success of the attack upon reception, as displayed in Figure 4.10b.

Attack 3: Disconnecting a patient monitor

Goal and impact: The goal of this attack is to close an ongoing association between the patient monitor and the CMS. Consequently, the hospital staff remotely monitoring a patient loses the real-time information about vital readings. As a DoS attack, the availability of the patient monitor's data is impacted and HDO operations are disrupted which could have adverse effect on patients' health.

Technical background: The Data Export protocol supports the transmission of commands. As described in its specifications [190], there is a number of commands that can be sent, allowing the monitor and the CMS to interact with each other in various ways. One of the commands is the *Association Abort* command. According to the documentation, a patient monitor can send this message in the case of communication problems to close the

[6] https://github.com/Ettercap/ettercap/blob/master/share/etter.filter.examples

```
----------------------------
RECEIVED!
----------------------------
HEADER: ['H', [[None], [None, '&']], None, None, ['DCA VANTAGE', '04.04.00.00', 'S0113023'],
 None, None, None, None, None, None, 'P', None, '2006010208352']

PATIENT: ['P', '1']

ORDER: ['O', '1', None, ['180', '0852']], None, None, None, None, None, None, None, None, No
ne, None, None, None, None, None, None, None, None, None, None, None, None, None, 'C']

RESULT: ['R', '1', [None, None, None, 'HbA1c'], '66', 'mmol/mol', None, 'H', None, 'C', None
, None, '20180726144314']

COMMENT: ['C', '1', 'I', ['1.000', '0 mmol/mol', 'IFCC', '189 mg/dL', 'G']]

TERMINATOR: ['L', '1', 'N']
```

(a) The test result containing a value of 66 mmol/mol is received by the LIS server as expected.

```
----------------------------
RECEIVED!
----------------------------
HEADER: ['H', [[None], [None, '&']], None, None, ['DCA VANTAGE', '04.04.00.00', 'S0113023'],
 None, None, None, None, None, None, 'P', None, '2006010208357']

PATIENT: ['P', '1']

ORDER: ['O', '1', None, ['180', '0852']], None, None, None, None, None, None, None, None, No
ne, None, None, None, None, None, None, None, None, None, None, None, None, None, 'C']

RESULT: ['R', '1', [None, None, None, 'HbA1c'], '41', 'mmol/mol', None, 'H', None, 'C', None
, None, '20180726144314']

COMMENT: ['C', '1', 'I', ['1.000', '0 mmol/mol', 'IFCC', '189 mg/dL', 'G']]

TERMINATOR: ['L', '1', 'N']
```

(b) Due to the tampering attack, the test result value received by the LIS server is 41 mmol/mol, instead of the 66 mmol/mol value sent by the blood analyser.

Figure 4.10: The HbA1c test results are displayed on the LIS server as they are received from the Siemens DCA Vantage blood analyser. The value of the tests is highlighted in red. (a) This result is the legitimate one, corresponding to the test result stored in the blood analyser (depicted in Figure 4.7). (b) This results is modified due to the successful tampering attack.

association between the monitor and the CMS [190]. The format of the command (i.e., the payload of the packet to send) is shown in Figure 4.11.

Attack scenario: An attacker can leverage this command to terminate the association between a patient monitor and the CMS, effectively causing a denial of service. She can spoof the targeted monitor and send an Association Abort message to the CMS. To do so, the attacker crafts a packet containing the Association Abort message as payload, and sets the source IP address to the one of the targeted patient monitor. As shown in Figure 4.11, the content of the packet's payload is the following:

```
0x19 0x2e 0x11 0x01 0x03 0xc1 0x29 0xa0 0x80 0xa0 0x80 0x30
0x80 0x02 0x01 0x01 0x06 0x02 0x51 0x01 0x00 0x00 0x00 0x00
0x61 0x80 0x30 0x80 0x02 0x01 0x01 0xa0 0x80 0x64 0x80 0x80
0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

The attacker can finally send the packet to the right port of the CMS (24105 as default). In case of another port being used instead, the attacker would first have to sniff the traffic to observe over which port the CMS and the monitor are communicating. Upon reception of

ASSOCIATION ABORT

The following building blocks can be used to format a Association Abort message:

AbortSessionHeader

```
0x19 0x2E
```

AbortSessionData

```
0x11 0x01 0x03
```

AbortPresentationHeader

```
0xC1 0x29 0xA0 0x80 0xA0 0x80 0x30 0x80
0x02 0x01 0x01 0x06 0x02 0x51 0x01 0x00
0x00 0x00 0x00 0x61 0x80 0x30 0x80 0x02
0x01 0x01 0xA0 0x80 0x64 0x80 0x80 0x01
0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

AbortUserData

This block is empty in the Abort message.

AbortPresentationTrailer

```
0x00 0x00 0x00 0x00
```

Figure 4.11: Association Abort message described in the Philips Data Export manual. Source: [190] p.337

this packet, the association between the CMS and the patient monitor will be closed until manually re-established. By repeatedly sending Association Abort commands, the attacker can prevent the staff from resuming to normal operations.

Proof of concept: We implement this attack in our lab by performing the following steps. Using Scapy, we create a UDP packet containing the aforementioned payload. We then set the packet's source IP address and source port to the patient monitor's values in order to spoof it. Finally, we set the destination IP address to the CMS's, and the destination port to 24105, the default port for Data Export protocol.

We send the packet from the attacking machine and we observe the result on the screen of the CMS, as shown in Figure 4.12. Upon packet reception, the CMS displays an error message informing that the patient monitor has closed the association. No data can be received any longer by the CMS, and the association has to be manually re-established.

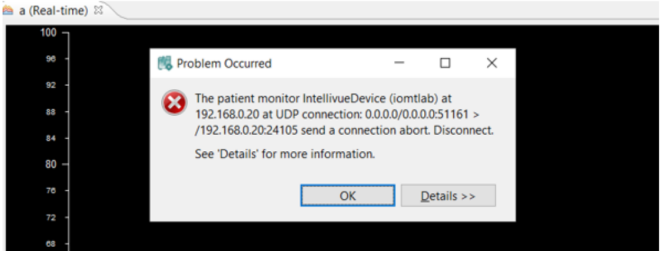


Figure 4.12: CMS displaying the result of an Association Abort message.

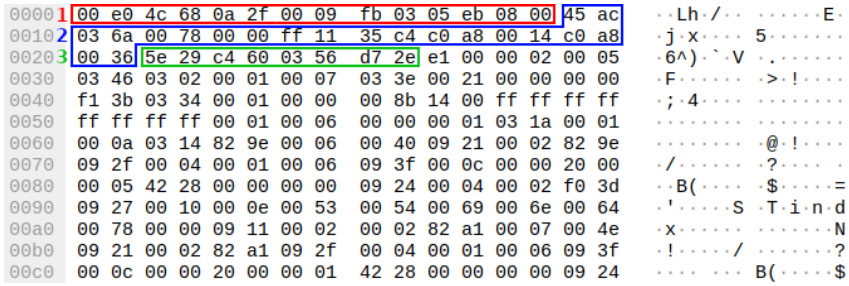


Figure 4.13: Example of one of the Data Export packets sent by the Philips MP50 patient monitor to the CMS captured with Wireshark. The packet is constructed as follow: the Ethernet envelop (box 1, in red), the IPv4 envelop (box 2, in blue), the UDP envelop (box 3, in green), and the remaining bytes represents the payload of the packet in the Data Export format. The packet is 888 bytes long in total (the payload depicted here is truncated).

Attack 4: Changing a patient's vital readings

Goal and impact: The goal of this attack is to tamper with the vitals readings sent from the patient monitor to the CMS over the Data Export protocol. As a result, the hospital personnel monitoring a patient remotely sees incorrect readings, unbeknownst to her. Similarly to Attack 2 above, tampering with the data conveyed in healthcare protocol packets violates the integrity of a patient's health information, which could ultimately lead to adverse effects on the patient's health.

Technical background: The patient monitor captures patient's vitals such as pulse, blood pressure and oxygen saturation through sensors connected to the patient's body. This information is displayed on the device's screen (see Figure 4.16 as example), and can also be sent to the CMS over the Data Export protocol. The patient monitor encodes the data according to the protocol specifications which can be found in the Data Export specifications [190]. An example of a Data Export packet sent by the monitor to the CMS is depicted in Figure 4.13. While certain clear-text key words can give an idea about the type of data present in the packet, one cannot directly understand the information in the payload. The protocol being a binary protocol, understanding the payload format requires a bit of reverse-engineering. The publication of the protocol specifications makes this task easier, as we show below.

Attack scenario: In this scenario, an attacker wants to modify the pulse rate of a single patient (this procedure can also be easily done for multiple patients). This attack is achieved by modifying specific bytes of the Data Export packets as they are sent from the monitor to the CMS. To do so, an attacker can follow a similar procedure as shown with Attack 2 by using Ettercap to establish a MITM position after having created a filter that captures and replaces the real-time vital readings in a Data Export packet with an arbitrary value. The challenge in this context is to first identify where and how the pulse rate is encoded in a packet, which requires a certain amount of reverse engineering.

The attacker can impact the patient's health and hospital operations in different ways by ar-

Pulse	Pulse Rate from Plethysmogram	
	Label:	
	NLS_NOM_PULS_OXIM_PULS_RATE	0x00024822
	Observed Value:	
	NOM_PLETH_PULS_RATE	0x4822
	Units:	
	NOM_DIM_BEAT_PER_MIN	0x0AA0

Figure 4.14: Patient's pulse rate encoding described in the Philips Data Export manual. Source: [190] p.118

bitrarily changing the pulse rate values. One way consists in modifying the pulse rate value to 0 to simulate a patient "flatlining", which would likely trigger an immediate emergency response from the staff. When executed on multiple monitors, the attack would be highly disruptive for the clinical staff. In another way, the attacker can simulate an arrhythmia condition by changing the pulse rate value of a patient to a rapid succession of high and low numbers. The clinical staff in charge of the patient would then likely engage in specific treatment(s) which could be detrimental for a patient not actually having this condition.

Note that, depending on the motivations, the attacker could also change other vital readings with the same procedure, such as blood pressure and oxygen saturation, as listed in the Data Export specification [190].

Proof of concept: To demonstrate this attack, we proceed with the three following steps:

1. We identify where and how are the pulse readings encoded in Data Export packets.
2. We create an Ettercap filter to intercept, modify and forward packets sent from the patient monitor to the CMS.
3. We execute the attack in two different ways, highlighting the risks to patients' health and HDO operations.

In the first step, we want to identify at which offset the pulse readings are located in the packets. To do so, we need to understand how the pulse rate is collected and processed by the patient monitor. In our lab setting, the pulse rate is measured via a finger plethysmograph connected to the monitor. We find in the Data Export documentation the information related to the pulse rate recorded by the plethysmograph and how this information is encoded according to the specifications as shown in Figure 4.14. In particular, we see the encoding for the label (`0x00024822`), the observed value, (`0x4822`) and the unit (`0x0AA0`). With this information at hand, we can search in Data Export traffic for packets with these byte values in the payload.

We generate and capture traffic by using our patient monitor's plethysmograph connected to one of our fingers. Using Wireshark, we find the packets sent by the patient monitor containing bytes with the values related to the pulse data (see the bytes highlighted in Figure 4.15) as identified in the protocol specifications. Moreover, by comparing the pulse value displayed on the patient monitor with the values displayed on the CMS, we identify the two bytes encoding the actual pulse rate value measured by the monitor, which are located 6 bytes after the pulse's unit (`0x0a 0xa0`). In the example given on Figure 4.15,

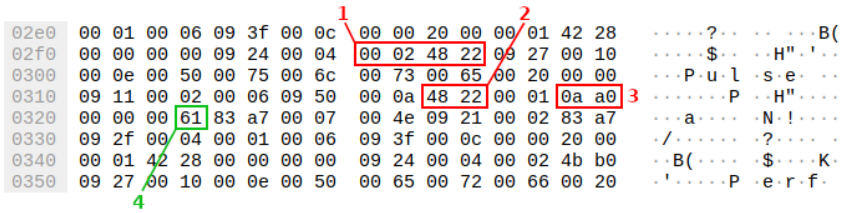


Figure 4.15: Data Export packet transmitted from the patient monitor and captured with Wireshark. It shows the patient's pulse rate in the payload of the packet. The bytes related to the pulse rate as found in the protocol specifications are highlighted in red (boxes 1, 2 and 3). The actual encoded patient's pulse rate can be seen in the green box (box 4).

4

where the value is 0x61 in hexadecimal, which translates in decimal into a pulse rate of 97 beats per minute. Having identified where the pulse value is located in the packet, we can now calculate the offset of the byte we want to change: offset 0x323.

In the second step, we create an Ettercap filter to capture and modify the packets containing patients' pulse readings. Similarly to Attack 2 presented before, Ettercap is used to establish a MITM position and intercept the packets of interest: the ones having a payload containing the values `0x00 0x02 0x48 0x22` (the pulse label), `0x48 0x22` (the observed value) and `0x0A 0xA0` (the unit). We then modify the pulse value to the desired value and forward it to the CMS to display this information. As we want to demonstrate the two attack scenarios described above, we create two different filters: one for the *flatlining spoofing attack* and the other for the *arrhythmia spoofing attack*.

In the final step, we execute the attacks as follow. First, we attach the plethysmograph to one of our fingers to generate data and simulate a patient being monitored. In Figure 4.16 is depicted the reading displayed on the patient monitor, which is a normal pulse of 80 beats per minute. We then execute the flatlining spoofing attack. The result can be observed in Figure 4.17a, as seen by hospital staff on the CMS. Right after the execution of the attack, the pulse suddenly drops from a legitimate range oscillating between 70 and 80 beats per minutes to 0. We execute then the arrhythmia spoofing attack. Figure 4.17b shows the result of the attack as seen on the CMS. Starting from the same legitimate value range as the first attack, the pulse suddenly increases after the execution of the attack to an accelerated value of 100 beats per minutes, and, after a short period, drops to a lower value of 50 beats per minutes. The attack repeats this pattern until stopped.

4.4. Chapter Conclusion

In this chapter we conduct vulnerability research on selected healthcare protocols and medical devices for which no security assessment has been yet published. Such research can help enhancing NSM capabilities in HDO as it enables the improvement of IDS by issuing signatures for newly found vulnerabilities and the creation of new attack datasets to be used for the design and testing of novel HDO-specific IDS. We select a number of targets based on our insights in HDO networks and published security research in healthcare, before

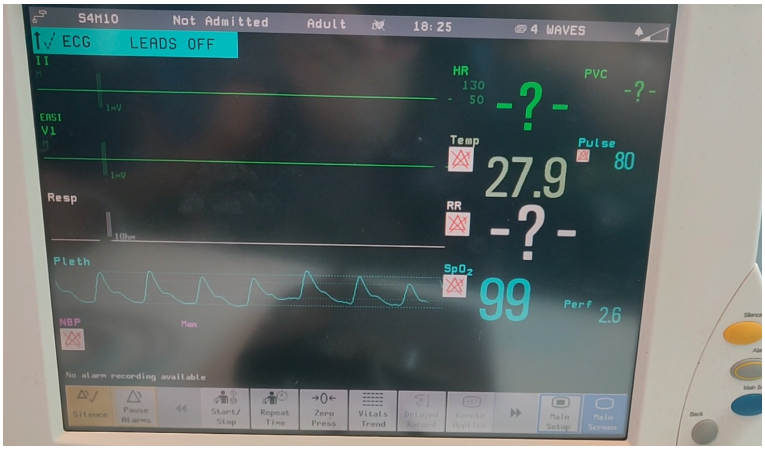


Figure 4.16: Patient vitals and other information shown on the display of the Philips MP50 patient monitor. The pulse rate reading (80 beats per minute) is displayed on the right-hand side of the screen. The pulse is measured by the finger plethysmograph that is attached to one of our fingers, simulating an actual patient being monitored. Some vitals are missing (value set to –?–) because the sensors are not connected to the patient monitor.

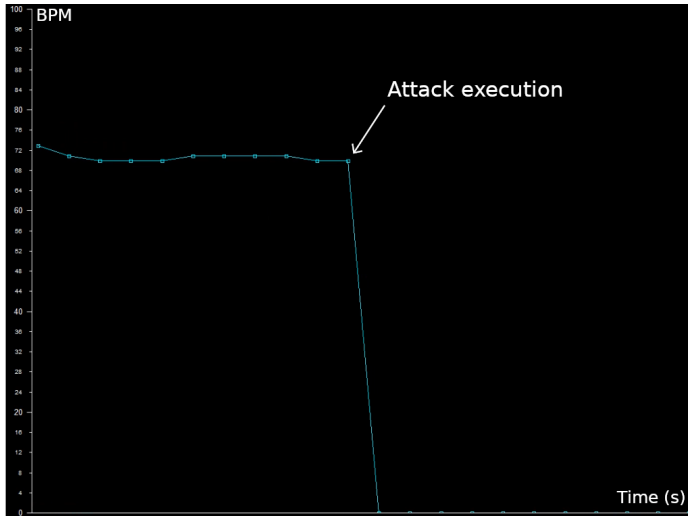
setting up a lab for our experimentations.

A number of weaknesses are found and we demonstrate four attacks exploiting them in our lab. These practical scenarios highlight the risks to patients' safeness by violating the confidentiality, the integrity and the availability of data transmitted between medical devices. The state of the network segmentation in healthcare networks that we observed in Chapter 2 let us assume that most HDO would be exposed to such threats. It is therefore necessary to protect these SCE by designing network monitoring tools suited to address the threats to patients' safeness.

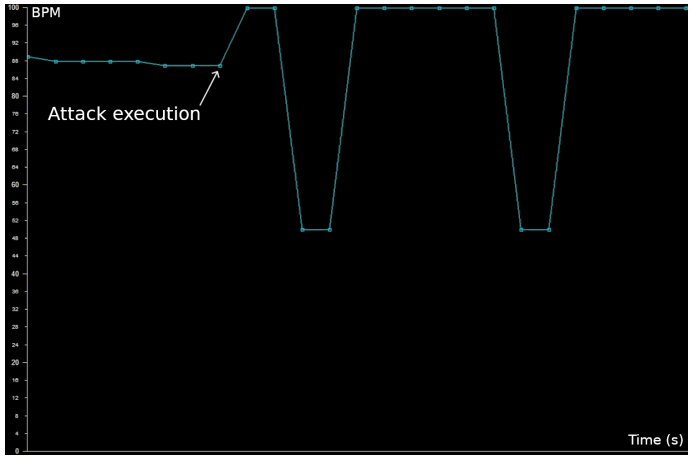
The research presented in this chapter enables such design. We create signatures to enhance the detection capabilities of Forescout's platform EyeInspect. Other IDS could be similarly improved, whether proprietary or open source. Additionally, new attack datasets can be assembled to test the signatures and assist with the implementation of novel IDS tailor-made for HDO. Finally, our methodology and blueprint for designing a healthcare lab allows researchers to reproduce our results and pursue vulnerability research efforts in the healthcare industry.

4.5. Part I Conclusion

This first part of the thesis explores the unique nature of HDO and (some of) their challenges with regards to NSM. HDO started a digital transformation in the past years that is profoundly changing how they deliver health care. The adoption of new technologies and the deployment of connected medical devices provide many benefits, yet also introduce new risks with regards to cyber security. It is unclear how well prepared these organisations are to defend themselves against modern threats.



(a) Flatlining spoofing attack observed on the CMS: before the attack execution, the pulse values oscillate between 70 and 80 beats per minute, corresponding to the actual pulse readings. After execution, the pulse value drops to 0 beats per minutes as long as the attack is executed.



(b) Arrhythmia spoofing attack observed on the CMS: before the attack execution, the pulse values oscillate between 70 and 80 beats per minute, corresponding to the actual pulse readings. After execution, the pulse readings fluctuate alternatively from 100 to 50 beats per minutes as long as the attack is executed.

Figure 4.17: Results of the two attacks tampering with the vitals readings sent from the patient monitor to the CMS over the Data Export protocol.

Our comprehensive analysis of the current state of network security in HDO in Chapter 2 shows both the complexity of these environments from an infrastructure management point of view, as well as security practices lagging behind well-established standards and principles. In particular, we observe the following two issues: 1) the current device classification tools used in HDO can leave a significant amount of unknown devices, and 2) the usage of some healthcare network protocols could introduce threats to patients' safeness. We used these two issues to pursue our investigation of HDO security in the subsequent chapters.

We explore in Chapter 3 the state of device classification, consisting in the ability to identify the type of devices connected to the network. Current solutions relying on manually-defined fingerprints or black-box machine learning techniques suffer from a number of limitations. We propose a semantic similarity-based clustering method to complement fingerprinting-based classification solutions, and we demonstrate that we can successfully classify up to half of the unclassified devices with a high accuracy. Our method can reduce a significant amount of work involved in manual and fingerprint-based classification by providing labels automatically. Moreover, compared to black-box machine learning approaches, our design allows clearer understanding of the inner-workings of the algorithm and its results.

In Chapter 4, we look at the usage of (potentially) insecure communication protocols used by certain medical devices in HDO. We select three protocols and conduct vulnerability research with the objective of identifying weaknesses which could put patients' safeness at risk. Such research can be considered an essential part of a good security strategy, as it enables the improvement of IDS and other NSM capabilities in various ways. In particular, the vulnerabilities we find and the attacks we demonstrate can be used to create new IDS signatures to detect those attacks, and new attack datasets can be created to assist with the evaluation of HDO-specific IDS and other NSM tools. Additionally, the blueprint of our lab we create for this study can help other researchers to pursue vulnerability research in HDO in a safe and realistic environment.

In conclusion, our study conducted on HDO security provides a thorough and deep understanding of these environments and their challenges with regards to NSM. The network security analysis in Chapter 2 shows limitations of a number of NSM capabilities and equips us with the knowledge needed to address some of these issues in Chapter 3 and 4.

In the next part, we keep on exploring the application of NSM in SCE by focusing on automotive security and intrusion detection in vehicular networks.

II

Automotive Security

5

Attacking the automotive Controller Area Network

Modern vehicles are complex networked systems that are connected to the outside world, with all security implications that brings. Attacks abusing the controller area network (CAN) protocol in cars have been demonstrated, which can impact greatly the users' safeness. This situation calls for effective network security monitoring (NSM) measures to protect the users. To fulfil our objective of identifying to what extend these measures are applicable in cars, we investigate in this second part of this dissertation the automotive security challenges with regards to the detection of these attacks targeting the CAN bus. More specifically, we study the design and the effectiveness of CAN-based network intrusion detection systems (NIDS) proposed in the literature. To do so, we first need to understand CAN specifications and the related attacks before analysing the different NIDS. For this reason, this chapter is dedicated to laying out the foundational knowledge for our study. In this chapter, we first cover the necessary background on CAN and related attacks, and we then define the scope of our research by identifying the capabilities of an adversary using in-vehicle network attacks against a vehicle (answering RQ 2.1).

The results reported here have been published in [62] (paper 1 of Section 1.4).

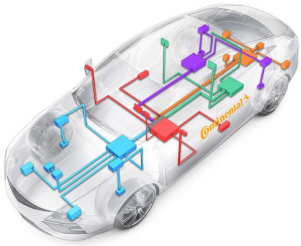
5.1. Introduction

The technological advances embedded into cars over time have drastically changed their very nature [49]. Vehicles are becoming more intelligent, offering increasing numbers of innovative applications covering different functionalities ranging from vehicle control to telematics and advanced driver assistance systems. The recent developments in the field of autonomous vehicle also promise a future where even driving would become obsolete. Overall, these advancements make cars more energy efficient, comfortable and safer. We arrive to a point where cars are no longer to be considered as mechanical devices, but rather as complex networked systems [246].

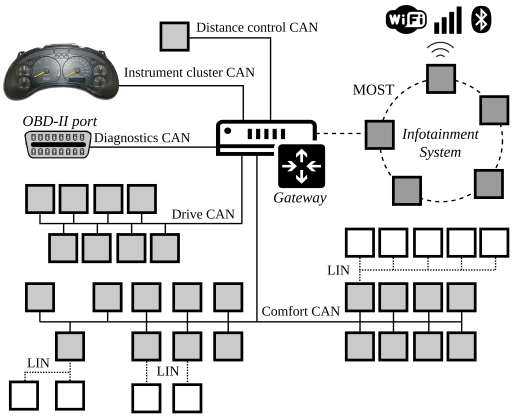
We can look at a modern vehicle from different perspectives to better understand its complex nature and the implications with regards to cyber security and users' safeness. Starting from a low-level perspective we call the "code perspective", we consider the software present in a car. One can find more than 100 million lines of code in a modern vehicle, running on micro controllers called electronic control units (ECU) [48].This code enables ECU to perform various operations such as accelerating, breaking, and steering. Additionally, it is also responsible to provide ECU with networked communication capabilities. The code loaded in a given ECU is developed by a third-party supplying that ECU and is proprietary. When assembling a car, automotive manufacturers rely on ECU provided by various contracted suppliers, and have actually little knowledge of and control over the embedded code [48].

The second perspective considered is the "architecture perspective", where we look at the collection of interconnected ECU within a car. One can count more than 100 ECU spread throughout a modern vehicle, communicating with each other over automotive-specific network protocols. Based on the function(s) of ECU and their communication requirement, car manufacturers choose the appropriate network where to deploy them, taking into consideration the safety level required, the data transmission rate and the costs [204]. This results in a typical automotive architecture as depicted in Figure 5.1, where one can find a gateway connecting multiple networks together within a single car, using specific protocols to guarantee proper ECU operation. It is important to keep in mind that automotive architectures are highly manufacturer specific, and even car model specific, which makes generalisation difficult.

The common automotive network protocols used are CAN, MOST and LIN [138]. Controller area network (CAN) is the predominant protocol used for communications between the mechanical sensors and systems such as transmission, braking, cruise control, power steering, windows and door locks for example [202]. Media-oriented system transport (MOST) provides high-bandwidth multimedia networking and is used for real-time audio, video and data transmission applications [204], interconnecting peripherals like CD/DVD players and GPS navigation systems [234]. Local interconnect network (LIN) can be seen as a lower cost supplement to CAN, and is typically used for non-critical functions such as air conditioning, door functionality and heating [46]. Additionally, other protocols can be present, such as Automotive Ethernet or FlexRay [138]. Automotive Ethernet is being increasingly adopted, as it offers higher bandwidth required by modern car systems such as advanced driver assistance systems (ADAS), infotainment systems or cam-



(a) Physical depiction of networks as they can be found within a vehicle. Source: [206]



(b) Logical depiction of networks and connected systems. Example inspired from [132]

Figure 5.1: Simplified representations of in-vehicle networks. These two representations are independent from each other.

eras. However, while it offers higher data transfer rates compared to CAN, it lacks some of the safety/performance features of the latter [45]. FlexRay is used in X-by-wire systems, such as steer-by-wire or brake-by-wire systems, although its cost makes its adoption in the automotive sector slow [174].

Some ECU and other vehicle's subsystems are also developed with wireless connectivity, which leads us to the third perspective: the "vehicle perspective", where we consider a car as a single system connected to a broader ecosystem of devices. Modern vehicles are increasingly connected to the outside world through multiple channels which can be broken down into three categories: infotainment, telematics, and vehicle-to-everything (V2X) technologies. Infotainment provides the driver and passengers with various connections, such as Bluetooth to pair their smartphone, and Wi-Fi access point. Telematics connect the car to cellular network, to the Internet, and to satellites. V2X technologies enable communications with other vehicles (V2V), with road infrastructure (V2I) and even pedestrian (V2P).

Looking at modern vehicles through these three perspectives highlights two important facts relevant for our study. First, they ought to be considered as complex cyber-physical networks rather than purely mechanical devices. Second, the increased connectivity and many functionalities, while offering many benefits, also come with evident security risks. The first fact is relevant to us because it enables the understanding of the challenging nature of modern cars and its implication with regards to NSM capabilities. Protecting cars from cyber attacks requires measures fitting the specificities of automotive architectures. The second fact is relevant because, in order to design effective monitoring measures, it is necessary to understand the risks linked to the technologies deployed in cars. More specifically, we need to consider the threats to vehicle users' safeness. These threats arise from the large attack surface resulting from the combination of the two aforementioned facts,

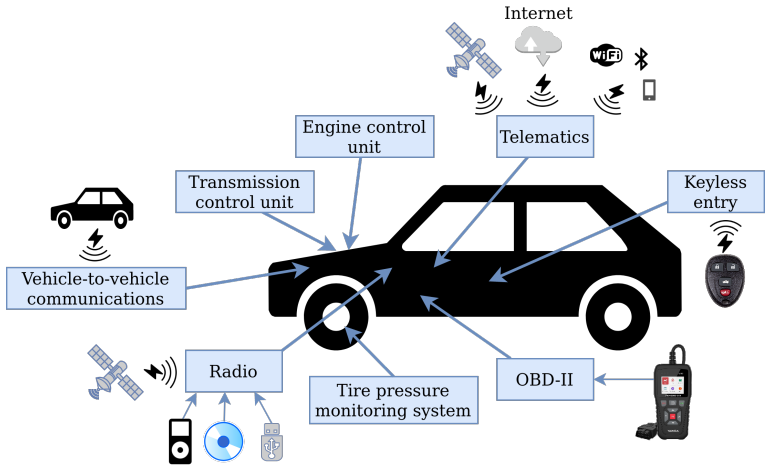


Figure 5.2: Example of some of a modern car's entry points. Depiction inspired from [31].

vehicle complexity and increased connectivity [31]. As depicted in Figure 5.2, a modern vehicle presents a number of entry points which make it potentially vulnerable to attacks. Researchers have already demonstrated the ability to abuse some of them to remotely take over the control of diverse vehicles and of certain subsystems, leading to serious consequences to users' safeness [31, 159, 160, 177].

The common point of these attacks is the abuse of the CAN protocol. In the vast majority of cars, one can find on the CAN bus ECU responsible for critical functions such as steering or braking. However, CAN having been developed initially for closed systems, the connectivity of modern vehicles exposes certain design flaws of the protocol, which in turn expose critical ECU [2]. Consequently, these ECU are vulnerable to CAN-based attacks abusing certain properties of CAN. For this reason, we focus in this dissertation on the study of CAN and the applicability of network monitoring techniques to detect CAN-based attacks.

5.1.1. Summary of Contributions and Outline of the Chapter

In this chapter we present our first contribution with regards to automotive security as we cover the necessary background related to CAN and the attacks abusing this protocol (Section 5.2). This contribution represents the body of knowledge upon which the next two chapters rely. More precisely, we first detail CAN specifications relevant for our study, before explaining CAN bus attacks and their functioning. Then we define the scope of our research on CAN-based NIDS by detailing the threat model that we use throughout our automotive study (Section 5.3). We finish by explaining the capabilities of an adversary using in-vehicle network attacks against a vehicle, answering RQ 2.1.

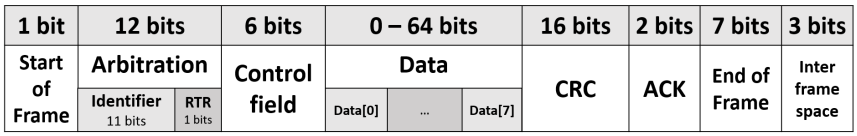


Figure 5.3: CAN frame structure.

5.2. Background

In this section we provide the background information relevant to our automotive study. We first give an overview of the CAN protocol and its application as automotive network protocol, followed by the CAN bus attacks published.

5.2.1. Controller Area Network

CAN is a message-oriented transmission protocol proposed by Robert Bosch in 1986. It was originally designed for the automotive industry, in an attempt to reduce the wiring complexity within cars [44]. Due to its low cost, simplicity, deterministic resolution of contention and resilience to electromagnetic interference, CAN is today not only the *de facto* standard protocol for in-vehicle communications, but also widely used in hospitals, factories and plant controls [56].

CAN is a multicast communication protocol, based on a multi-master access scheme [184]. The nodes connected to the bus broadcast their CAN frames. There is no addressing scheme with CAN: each frame is assigned a unique identifier, known as the *arbitration ID* or *CAN ID*, which defines both the content and the priority of the frame. With 11 bits in the identifier field, the CAN ID can range from `0x000` to `0x7FF`. CAN frames are composed of several fields, as depicted in Fig. 5.3.

According to the CAN specifications, four different types of frame are used to control message transfers on the bus [228]. *Data frames* are sent by nodes and can carry up to 8 bytes of data. Moreover a node on the bus can also send a *remote frame* to request information from another one. Data and remote frames can be dissociated with the use of the Remote Transmission Request bit (RTR): in the former case, the RTR bit is set to 0, while in the latter case it is set to 1. *Overload frames* (specified in the control field) are used by nodes to request an extra delay between two data or remote frames. Finally, any node on the bus can invalidate a frame being sent by sending an *error frame*, when it detects a problem.

While a node may broadcast multiple CAN ID, each CAN ID is bound to a single node; two nodes cannot send data frames with the same CAN ID. Every time a frame is transmitted, all nodes on the bus will receive it, and will determine, based on the CAN ID, whether they should accept and further process the message [173]. This mechanism allows a node to send a remote frame with a specific arbitration ID, and the node responsible for that ID will respond to the request.

The CAN protocol offers a robust arbitration mechanism that handles conflict when two or more nodes attempt to transmit a frame at the same time. As previously mentioned, the

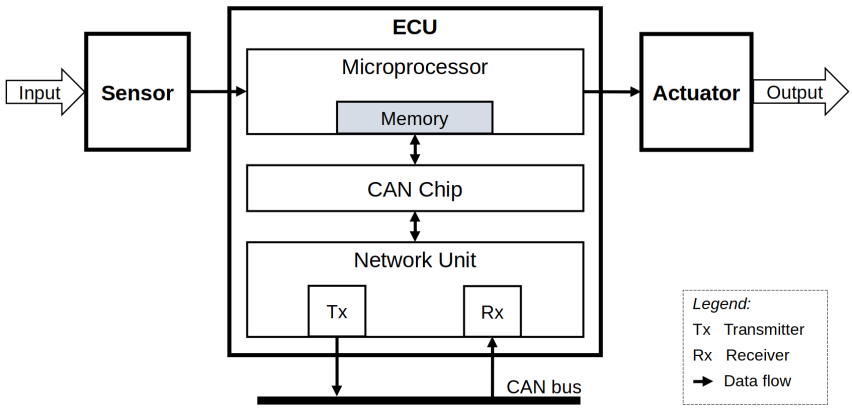


Figure 5.4: Simplified depiction of a typical ECU connected to an automotive CAN bus.

CAN ID of a frame determines its priority: the lower the ID value, the higher its priority. Consequently, the node sending the frame with the lowest CAN ID value wins the arbitration and has access to the bus. The other nodes wanting to transmit data wait until the bus is free before trying again. The advantage of the arbitration mechanism is twofold: it guarantees that neither information nor time is lost [228] and that eventually all nodes can have access to the bus [43].

Additionally, CAN includes a fault confinement mechanism which removes faulty nodes from the bus. A node can be in one of these three states: *error active* when functioning properly, *error passive* when suspected of faulty behaviour, or *bus off* when considered corrupted [56]. When too many errors are detected, nodes will change state and ultimately disconnect themselves from the network [173]. This mechanism protects the health of the bus by preventing faulty nodes from disturbing communications or impacting network performance [184].

CAN as automotive network protocol

In the automotive context, the nodes communicating over CAN are ECU. ECU typically consist in a microprocessor, sensor(s) and/or actuator(s), a CAN chip and a network unit, as depicted in Figure 5.4. The microprocessor performs the logic and input/output operations specified by the code stored in memory. The sensor(s) collects input data (e.g., the current pressure of the tires), and the actuator(s) performs a certain physical action (e.g., steering the wheels). The CAN chip interfaces the microprocessor and the network unit by converting the data to be sent and received over the CAN network. The network unit handles the transmission of data from the microprocessor to the other ECU on the network, and processes the frames received by determining whether to accept them or not.

Communications between ECU proceed as follow: depending on their function, ECU send data (e.g., the readings from their sensor(s)) and the other ECU can act upon the reception of certain messages and perform specific actions if necessary (e.g., controlling an actuator). An example of such communication is depicted in Figure 5.5. The format of the

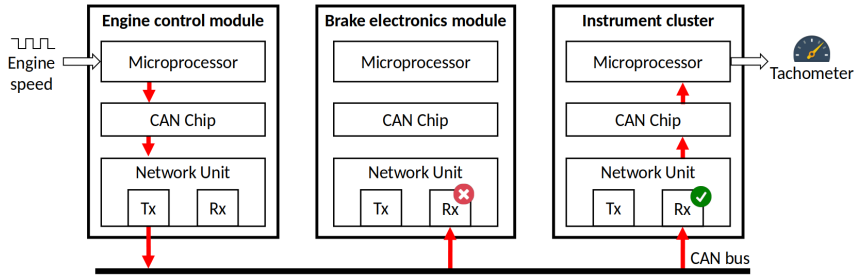


Figure 5.5: Example of ECU communicating over CAN. The engine control module ECU reads the engine speed from its sensor and broadcasts its value on the CAN bus. All other ECU on the network receive the message. The brake electronics module does not process further the message; it is programmed to ignore the associated CAN ID since it does not need it for its operation. The instrument cluster processes the message and uses the value contained in the payload to update the tachometer accordingly. Example inspired from [42].

data contained in CAN frames' payloads is defined by car manufacturers using higher level protocols on top of CAN. Generally speaking, two types of data frames are used on automotive CAN networks: *normal messages* and *diagnostic messages* [159]. Normal messages are transmitted by the ECU during regular car operations (e.g., driving), and they follow a proprietary format. An example of normal messages captured from the CAN network of an Opel Astra during normal operations is presented in Figure 5.6. The payload contained in a message is made of one or multiple values called signals, which correspond to information about certain car's systems [103]. Signals occupy a specific size and are located at a defined offset in the payload. Figure 5.7 depicts an example of a CAN frame containing the value of the current temperature and speed of the engine, both stored in 2 bytes and at a specific offset. The signals are often encoded by using a vendor-specific formula, which can consist in a combination of a scaling factor and an offset, among other things [84]. Additionally, manufacturers can use different endianness and units for storing signals. As an example, let us consider in Figure 5.7 the engine control module ECU sensing the engine temperature and broadcasting it alongside other readings, such as the engine's speed, over CAN ID `0x24E`. The temperature value start at byte[0], has a size of 2 bytes and uses the big-endian byte ordering. In our example, the formula used to encode the temperature uses a scaling factor of 0.0312 and an offset of -273 [173]. This encoding is usually confidential and considered by vendors as intellectual property [84]. It varies greatly from

```
1  (1536574931.814734) slcan0 12A#00F0707200003090
2  (1536574931.817030) slcan0 1F3#003C00
3  (1536574931.817835) slcan0 0C1#80260BC683EB4399
4  (1536574931.818045) slcan0 0C5#C0BFC74B80EBFD1B
5  (1536574931.818296) slcan0 0D1#00000000000000
6  (1536574931.818313) slcan0 185#1BA8
7  (1536574931.818692) slcan0 1C8#03FF000000003C01
8  (1536574931.818890) slcan0 0C9#800E950A00011000
9  (1536574931.819142) slcan0 1E9#400C000E000D0000
10 (1536574931.819344) slcan0 0D3#2BFE950A00011000
```

Figure 5.6: Excerpt of CAN bus traffic captured via the OBD-II port of an Opel Astra while driving. Each line corresponds to the following data (from left to right): timestamp (Unix epoch format), name of the network interface ("slcan0"), CAN ID and payload separated by # (e.g., "12A#00F0707200003090").

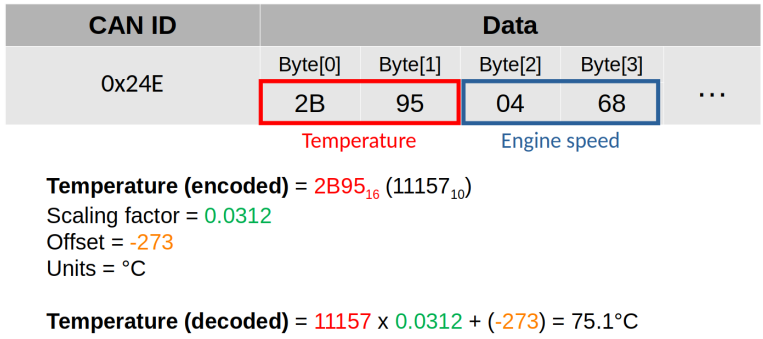


Figure 5.7: Example of a simplified CAN frame containing encoded signals as data. The fields irrelevant for this example have been omitted. Example inspired from [173].

manufacturer to another, and changes in the encoding schemes can also be observed in different car models of the same brand. Without the manufacturer's specifications, it is not trivial to understand ECU communications and the signals sent in CAN frames. One requires a significant amount of reverse-engineering efforts be able to do so [192]. The other type of data frames used by manufacturers are diagnostic messages, and are defined according to a diagnostic communication protocol such as Unified Diagnostic Services (ISO 14229-1). They are special messages normally sent by mechanic's tool (or "tester") during maintenance operations. Depending on the services implemented on an ECU, diagnostic messages can be used to perform various actions such as querying information or updating its firmware.

ECU transmit messages according to a communication scheme defined by ECU suppliers (recall the "code perspective" mentioned above), based on the function(s) fulfilled by the ECU. Most ECU in cars are programmed to follow a *regular* communication scheme, where they will send their messages at a regular time interval. ECU can also follow a *sporadic* communication scheme, transmitting messages under specific conditions, such as after a certain condition is met (i.e., event-based trigger). Figure 5.8 represents an example of the communication schemes of 10 different CAN ID from an Opel Astra. We capture the CAN network traffic of the CAN ID for 200 seconds while driving and plot the inter-arrival times in the figure. The results show that some of the CAN ID are sent regularly (e.g., `0x12A` and `0x160`), while some others have variation in timing (e.g., `0x4E9` and `0x52A`). Keeping in mind the arbitration mechanism of CAN based on CAN ID value, these variations can be explained by the fact that high CAN ID values (e.g., `0x52A`) have a lower priority to access the network compared to lower ID values (e.g., `0x12A`), resulting in delays before being able to be sent. One CAN ID (`0x140`) is sent irregularly, displaying a sporadic communication scheme.

The CAN protocol and its application in modern vehicles have a few important security implications. First of all, car makers use the reliable CAN protocol on high-speed buses inside their vehicles to interconnect safety-critical ECU. However, its simplicity and small packet size imply that many common security measures such as communication encryption or ECU authentication are not possible. In addition, the broadcasting nature of the protocol

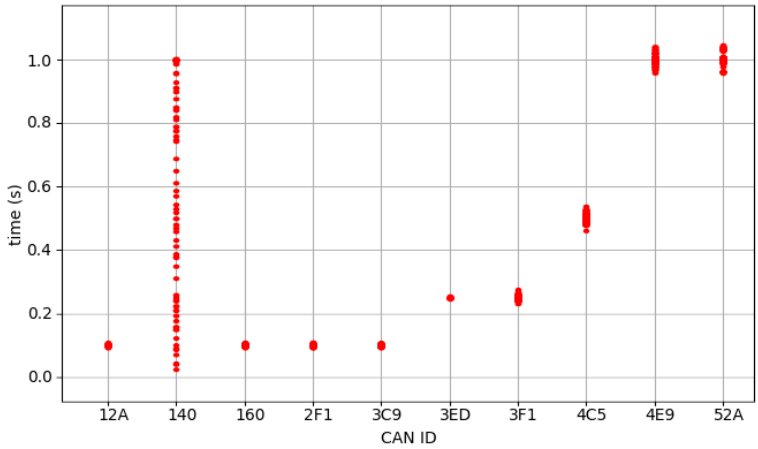


Figure 5.8: Inter-arrival time between messages of CAN ID of the Opel Astra while driving.

allows anyone on the bus to read and send messages, which can have critical consequences as explained in the next section.

5.2.2. CAN bus attacks

When targeting a vehicle, an attacker may try to make an ECU (or several of them) fail to function properly or try to influence their behaviour through manipulation of the CAN bus. As demonstrated in previous research [31, 159, 177, 197], an attacker has a board range of entry points which can be leveraged to gain access to the CAN bus and compromise an ECU, granting the attacker the possibility to read packets and send crafted ones. Several aspects in the context of in-vehicle CAN bus determines the types of attacks that can be performed once access to the bus is obtained. We shortly describe these aspects before explaining the attacks.

First, recall that the CAN bus is an unauthenticated and unencrypted broadcast bus where anyone having access to the network can read and send messages. Second, (some of the) ECU have *message confliction* resolution mechanisms. Such confliction occur when an ECU receives a legitimate packet followed by a spoofed one from a compromised ECU, and the difference between the signals' values in the two payloads received is too great to be realistic. When receiving conflicting values, an ECU will react differently depending on the resolution mechanism implemented [160] (discussed below). ECU also have a *diagnostic mode* used to evaluate their state and perform diverse operations. In this context, diagnostic messages are sent by mechanics' tools during maintenance to query ECU, for example to check the health of a controller or to update its firmware. Finally, the mapping from CAN ID to their respective ECU and functionalities varies across different cars. Car manufacturers design their vehicles in their own specific ways, making the automotive architectural ecosystem very diverse.

Considering the last aspect mentioned above, we see that when at first gaining access to the

CAN bus, it is not straightforward to know the architecture of the vehicle and which CAN ID is linked to which ECU without prior knowledge. Depending on the goal, an attacker must know what are the interesting CAN ID and their expected payload format and values. Such knowledge will enable more directed actions. Thus, during a reconnaissance phase, an attacker may perform a *fuzzing attack* by sending messages with random CAN ID and payload values and observing the reactions. This attack can allow an attacker to learn about the architecture and the behaviour of the vehicle and its ECU. It is arguable whether an attacker would actually fuzz her victim directly: the adversary would likely prepare her attack by first acquiring the same car as her victim and experimenting on it to gain in-depth knowledge [85].

As demonstrated in [159], an attacker can send a diagnostic message in order to open a diagnostic session on an ECU, allowing her to perform specific actions, depending on the services implemented on that ECU. Diagnostic messages are extremely powerful. They are, for example, leveraged by Miller and Valasek in [157] to perform various actions on a Toyota Prius and a Ford Escape.

Due to the broadcast nature of the CAN bus, an attacker can try to influence ECU by sending frames with arbitrary CAN ID and payload in two approaches. First, the attacker can *replay* messages observed on the bus. This action consists in crafting and sending packets with the same values than the messages transmitted by the legitimate ECU. Second, with sufficient knowledge of the car's CAN implementation (e.g., how the payload for a given CAN ID is encoded), an attacker could *spoof* messages by crafting and sending packets with an arbitrary payload to achieve the desired outcome (e.g., making the instrument cluster display an arbitrary speed). However, as the legitimate ECU would keep on sending its frames, such spoofed messages are likely to conflict with legitimate ones. The success of such attacks depends on the message confliction resolution implemented in the targeted ECU, as demonstrated in [160]. Depending on their implementation, ECU will handle conflicting values in different ways. Simple ECU may accept the messages that are the most predominant. Therefore, an attacker would have to send the spoofed frames faster (in practise 20 to 100 times faster [158]) than the legitimate messages to ensure the crafted frames are accepted by these simple ECU. However certain complex ECU may disregard conflicting messages and eventually shut their features off, rendering such *spoofing* attacks unsuccessful [160].

If the attacker aims to prevent ECU from communicating, a straightforward *Denial of Service (DoS)* attack can be executed. This attack consists in flooding the bus with frames with the CAN ID set to `0x000`. As mentioned in Section 5.2.1, the CAN ID determines a packet's priority. Therefore, sending repeatedly messages with CAN ID 0x000 (i.e., the highest priority) will prevent the other ECU from transmitting their frames on the network due to the arbitration mechanism of the CAN protocol. DoS attacks can put the car in an unstable state [157].

There are also low-level attacks relying on bus signal tempering, as demonstrated in [85], and researchers have been able to conduct *bus-off* attacks on diverse vehicles [33, 113, 183]. Unlike the attacks discussed previously, they do not require sending entire frames onto the bus. An attacker can abuse the error handling mechanism of the CAN protocol by sending

a few bits at the same time the targeted ECU is transmitting a frame. Consequently, the corruption of the message will trigger the fault confinement mechanism of CAN. After a certain number of errors, the targeted ECU will be put in "bus off" mode, preventing it from sending new frames. The attacker can then freely send crafted CAN frames, posing as the silenced ECU.

5.3. Scope of Research

In this section, we start by outlining the assumptions scoping our research. We then provide a threat model for the CAN bus in which we detail the capabilities of an adversary and the attacks which can be launched (answering RQ 2.1).

5.3.1. Assumptions

We assume that the attacker already has a foothold onto the CAN bus of a car. This access can be established either physically or remotely. In the first case, access to the CAN bus is trivially obtained if the attacker has physical access to the car. A laptop or another device (e.g., a Raspberry Pi) can be connected to the On-board Diagnostics (OBD-II) port through the use of a USB-CAN interface readily available on the market. The OBD-II port is present in modern cars to give easy access to mechanics for maintenance operations (even federally mandated in the United States). It is through this port that they would connect a tester and send diagnostic messages, as explained in Section 5.2.1. To guarantee easy access to the mechanics, OBD-II ports are located somewhere close to the steering column and require minimal effort to find and to connect to (the port location for most cars can also be found online). Once connected, the attacker could start transmitting crafted frames onto the CAN bus, however with possible restrictions. Depending on the architecture of the targeted car [158], the OBD-II port might be connected to the CAN network containing the ECU in which the attacker is interested in. However, as manufacturers seem to shift toward segregated architecture designs, the port can be granting access to a dedicated diagnostic network, with a gateway filtering the communications to others CAN networks (an example of such architecture is presented in Figure 5.9). In this case, the attacker would first have to compromise the gateway to access other networks of interest. Overall, the possibilities presented to an adversary with physical access to a car are concerning, especially in the context of the sharing economy.

In the second case, a foothold onto the CAN bus can be established remotely in diverse ways. Researchers have already demonstrated the feasibility of remotely compromising external interfaces [31, 80, 159, 177]. A possible scenario of remote attack similar to the research demonstrated would involve the following sequence of steps depicted in Fig. 5.9, as explained in [85]:

1. Exploitation of a vulnerability of the telematic unit over the cellular network, granting the attacker remote code execution capabilities and access to the CAN infotainment bus.
2. Pivoting onto the network by compromising the gateway ECU in order to gain access to the safety-critical CAN powertrain bus.

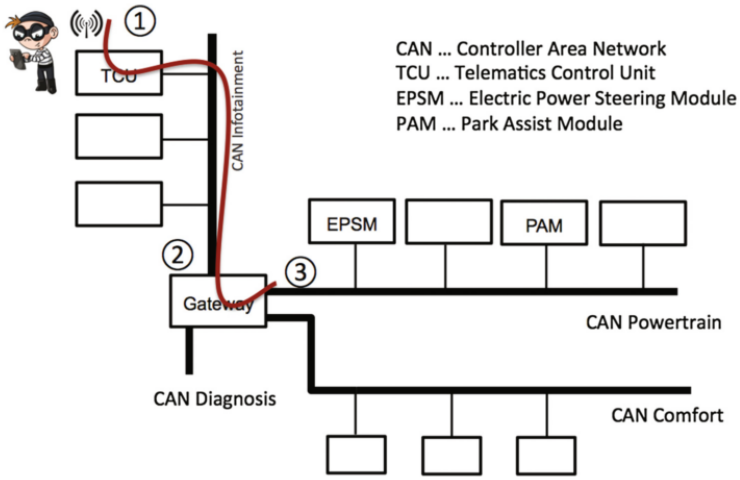


Figure 5.9: Stages in car hacking. Source: [85].

3. Finally the attacker can inject CAN frames onto the bus and perform diverse attacks, as described in Section 5.2.2.

For our research on automotive security, we focus on the third step of this scenario, where the attacker has access to the CAN bus. While a layered security approach is required to prevent an adversary to successfully perform these steps, we want to explore in this work the extend to which network security monitoring could be applied in an automotive CAN network.

5.3.2. Threat Model

We identify in this section the capabilities of an attacker with access to the CAN network. After previously explaining how such access can be obtained, we specify below the different CAN bus attacks that can be executed by the adversary. These attacks are the ones we consider in our research.

Capabilities

The simplicity of the CAN protocol, as mentioned in Section 5.2.1, offers a number of capabilities to an attacker once she has gained access to the network. Because of unencrypted communication, the attacker can read the messages out of the bus and learn about CAN ID and ECU communication patterns in order to reverse engineer the bus. This would allow the adversary to pose as a legitimate ECU by replaying or spoofing messages with arbitrary CAN ID and payload. Due to the broadcast nature of the protocol, all ECU on the bus will receive the crafted messages, and eventually perform certain actions accordingly. Depending on the attacker's motivations and objectives, she can send crafted packets to influence or manipulate the behaviour of ECU. For example, the adversary can turn off the engine

while the car is at speed, apply the brakes or even steer the vehicle to an arbitrary direction, as demonstrated in [157]. To do so, the attacker can leverage the attacks described next.

Attacks

With the capabilities described above, an adversary can perform a number of attacks presented in Section 5.2.2. We consider in our research the following CAN bus attacks:

1. Use of diagnostic messages
2. Fuzzing attack
3. Replay/spoofing attack
4. Denial of Service attack

All these attacks are based on the injection of frames onto the CAN bus. While there exist low-level attacks relying on bus signal tempering, we limit the scope of our research to the consideration of attacks at the CAN protocol level. Low-level attacks abuse the electrical properties of the bus, and therefore represent a separate topic of research. Although, we do consider a consequence of those attacks, namely that the ECU communication will be *suspended*. Therefore, we include a fifth attack also considered in the literature (e.g., [34, 231]) which we refer to as *suspension attack*. In this attack, an ECU suddenly stops emitting frames.

5.4. Conclusion

We explore in this chapter the complex nature of cars and implications with regards to cyber security and users' safeness. Modern vehicles embed a number of ECU which communicate over automotive networks. Particularly, CAN networks connect critical ECU together and its simplicity allows of fast and reliable communications between them. However, this simplicity comes at the price of security issues.

In fact, once an attacker obtains access to the CAN bus, she can abuse the specifications of the protocol to manipulate the behaviour of the ECU. Such access can be gained either physically or remotely by exploiting vulnerabilities in one of the various external interfaces connecting the car with the outside world. Once on the network, the adversary has the capability of reading and writing messages to the bus, allowing of the execution of CAN-based attacks which can result in threat to the safeness of users. We identify in this chapter five attacks leveraging CAN frame injection which can enable an adversary to take over control a car's system(s).

To address these attacks, researchers have proposed a number of network intrusion detection systems (NIDS) for CAN networks. We explore in the next chapter how these CAN bus NIDS function and we identify what kind of coverage do they provide against attacks on CAN network.

6

Network Intrusion Detection Systems for CAN Bus

Following the publications of attacks targeting the CAN bus of vehicles, a number of network intrusion detection systems (NIDS) have been proposed in an attempt to protect the safeness of car users. As non-intrusive security measure, their application seems well-suited to the automotive setting. These systems rely on certain characteristics of CAN communications to detect signs of attack (also called "indicators of attack"). In this chapter, we want to identify the coverage offered by these systems against attacks on CAN network (RQ 2.2). We start by conducting a survey of the systems to better understand their functioning and explain the principles they use to detect attacks. We also introduce a novel taxonomy for CAN bus NIDS, enabling the categorisation and comparison of NIDS at a theoretical level. This work allows us to evaluate them at a practical level in the next chapter.

The results reported here have been published in [62] (paper 1 of Section 1.4).

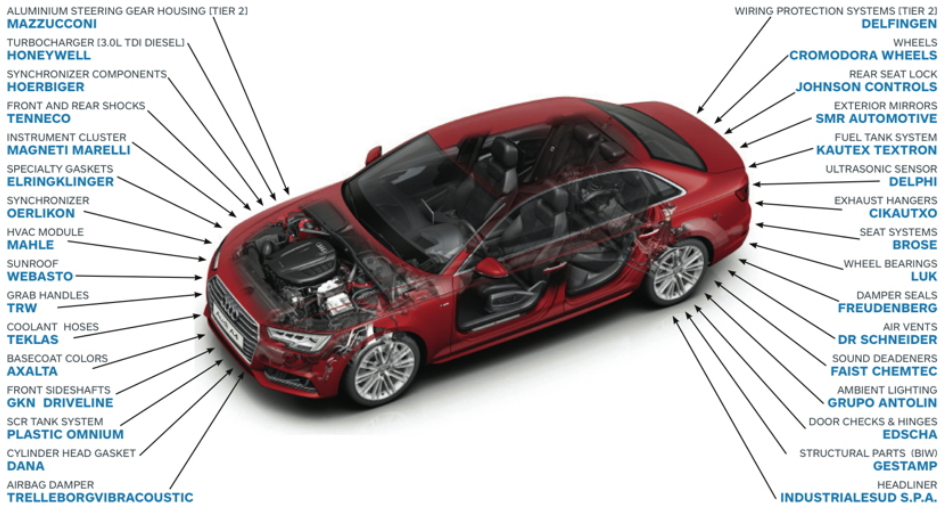


Figure 6.1: Example of some of the suppliers involved in the production of the 2015 Audi A4. Source: [40].

6.1. Introduction

In the previous chapter, we identified the capabilities of an attacker having access to the CAN bus (RQ 2.1). These capabilities allow of the execution of attacks with safeness-critical consequences to the users. This situation calls for NSM measures capable of detecting and preventing these attacks. We focus in this chapter on the monitoring measures proposed in the literature addressing the aforementioned attacks.

After exploring the automotive environment and the specificities of the CAN bus, a number of challenges already comes to mind which could hinder the design of security measures, and, at the same time, motivate the research on NSM for the CAN bus. First, the implementation of security measures in a safety critical, complex cyber-physical network is difficult as one has to guarantee that the measures do not impact the existing functionality. Invasive solutions that could affect the performance of safety critical functions or require a major redesign of the vehicle or its components will not be acceptable to the manufacturers. Second, recalling the "code perspective" explained in the previous chapter, it is difficult to embed security measures into ECU in an attempt to port well-established desktop-based solution (e.g., host-based intrusion detection, anti-virus, etc.). The measures should not interfere with ECU normal operation, and also the ECU may not have the necessary resources to embed additional security functions due to their limited computing power and hardware constraints. Third, as car makers assemble parts from various suppliers, managing the security level of each component can represent a daunting task (see Figure 6.1). Moreover, knowing that it takes on average 6 years for a car to go from design to production, and its life-cycle to be around 12 years [235], it is very likely that components would require security update over time. But handling a patch management process that is both cost-effective (e.g., avoiding costly vehicle recalls) and time-effective (e.g., updates installed as close as possible from their publication date) is complex. If fact, as ECU need to go through

lengthy re-certification process after code modification, car makers need the ability to provide timely security updates without leaving cars on the road unprotected for too long. The patching issues are notorious in other industries and it is common to observe systems running unpatched for months, if not years, after vulnerability disclosure (e.g., [77]). In this context, we need security measures both suited for the CAN bus and fitting the challenges of automotive environment.

Network-based intrusion detection systems (NIDS) provide a non-invasive security measure that is well-established in other fields (e.g., traditional IT, or industrial control systems) and thus seems potentially suitable for the automotive setting (e.g., [246]). In fact, conversely to host-based IDS (HIDS), NIDS are deployed directly on the network instead of being installed on ECU, which fits the challenges expressed above. From their vantage point, they offer the capability of monitoring the traffic and detecting signs of attacks.

After the publication of the CAN bus attacks, a number of NIDS have been proposed for CAN in the literature [2, 92, 108, 118, 129, 130, 142, 143, 158, 164, 169–171, 182, 221, 223, 230, 231, 238, 254]. We investigate in this chapter how do they function and on what kind of principles do they rely on to detect attacks.

6.1.1. Summary of Contributions and Outline of the Chapter

In this chapter, we analyse what kind of coverage the CAN bus NIDS published provide against attacks on CAN network that were identified in the previous chapter (RQ 2.2). To do so, we start in Section 6.2 by covering the required background and terminology regarding NIDS. We then introduce our two contributions for this chapter.

As a first contribution, we introduce in Section 6.3 a novel taxonomy designed for the classification of CAN bus NIDS. While such taxonomies are already available for NIDS in other fields (e.g., IT, or cyber-physical systems [162]), it is not the case for CAN bus NIDS, making comparison between them difficult. Our taxonomy offers an informative categorization which was previously lacking in the literature.

The second contribution consists in a reasoned inventory of published NIDS for CAN. We survey a total of 24 NIDS fitting our scope of research and detail their functioning in Section 6.4. At the time of publication, no such inventory was available in the literature.

As explained in [106], taxonomies and surveys are important as "they aim to improve both the efficiency of IDS and the creation of datasets to build the next generation IDS as well as to reflect networks threats more accurately in future datasets". This study presented below lays the foundations for the next chapter in which we develop an evaluation framework enabling the testing of CAN bus NIDS.

6.2. Network Intrusion Detection Systems

In this section we cover some background related to NIDS and their typical attributes used to categorise them, namely their detection method and depth of inspection. This knowledge is important to understand the principles upon which CAN bus NIDS rely and how

typical attributes are translated to fit the CAN network. Finally, we introduce some of the performance metrics commonly used to evaluate these systems.

6.2.1. Detection method

There are two main methods used to detect intrusions: *knowledge-based* and *anomaly-based* detection. The later method also encompasses another variant, referred to as *specification-based* detection. We introduce these three methods below.

Knowledge-based

Also known as *signature-based*, or *misuse detection*, a knowledge-based NIDS uses information about attacks, so-called *signatures*, as a pattern characterising a known threat [53, 162]. The NIDS will compare the signatures against observed events to identify possible attacks [203]. Upon detection (i.e., an event matches a signature) an action can be executed; typically an alarm will be raised (e.g., to notify the network administrator or the security team) and a counter-measure can be initiated (e.g., termination of the connection) depending on the rule specified for a given attack.

With their black-list approach, knowledge-based NIDS are effective to detect known threats with great accuracy. They generally present a very low rate of false positives (we elaborate on the terminology in Section 6.2.3 below) since the use of signatures guarantees that each match signifies that a malicious event has been successfully detected. Although, since the set of signatures cannot be exhaustive, such NIDS are unable to detect unknown attacks: attackers will find new vulnerabilities to exploit before a signature can be created to detect their attacks (so-called *zero day*). Moreover, some techniques can be employed in order to bypass knowledge-based detection systems [41], such as payload encoding.

Anomaly-based

An anomaly-based NIDS creates first a reference model (or "profile") of a system by recording and collecting normal/legitimate operations and communications. It then starts to monitor the current activities of that system. An alert will be generated anytime the NIDS identifies a significant deviation from the model [55]. Building such profiles can be done more or less automatically using a sample of historical data or even diverse machine learning techniques [78, 139, 244].

Put simply, anything that does not happen according to the normal behaviour previously learnt is regarded as an incident. The main advantage of the anomaly-based IDS is twofold: there is no overhead of maintaining a signature database up to date, and they provide the capability to detect unforeseen attacks. Any attack would (in theory) change the normal behaviour of a system by, for instance, accessing unusual resources or establishing connections with new machines outside of the trusted network. However, this capability comes at the price of false positives: these NIDS are prone to detect legitimate events as malicious if they have not been previously observed during the learning phase [203]. The quality of the model created will influence the risk of false positives. It is challenging to build accurate profiles because systems' activities are usually quite complex.

Specification-based

In the same fashion as anomaly-based detection, specification-based NIDS detect attacks by identifying deviations from a norm [207]. However, instead of creating the reference model during an initial learning phase, specifications of a system are manually developed to characterise legitimate program behaviours [240].

Compared to anomaly-based NIDS, the main benefit of this approach is its accuracy in distinguishing legitimate deviations from the malicious ones. If the profile is correctly developed based on the system's specifications, the rate of false positive events can be comparable to the one obtain with knowledge-based detection [240]. As a matter of fact, since the model is developed manually, it will be more exhaustive than the model created during a finite learning phase. However, depending on the system, it can be challenging to retrieve the complete set of specifications.

6.2.2. Depth of inspection

A NIDS can inspect network traffic at different "levels", and two levels of depth of inspection are commonly used: *flow-based* and *payload-based*. In a flow-based approach, a NIDS will look at a collection of packets presenting certain common characteristics. In the second approach, referred to as payload-based (or "packet-based"), a NIDS will analyse the payload of each packet passing on the network. We discuss these two methods below.

Flow-based inspection

A flow can be defined as a group of packets sharing common properties and passing an observation point during a certain period [39]. Even if the concept of *flow* is traditionally used in the IT context with TCP/IP communications [39], Japkowicz and Taylor adapted the term for the CAN bus [230]. In their paper, a CAN flow presents a collection of certain characteristics such as the CAN ID and the number of packets contained in that flow.

CAN bus NIDS researchers have been interested in flow-based detection as this approach fits well CAN traffic. The majority of ECU in a vehicle communicate at a very specific time (see Chapter 5). This regularity makes it easy (in theory) to create a model in which communication patterns for each CAN ID are defined. A flow-based NIDS would then look for deviations from these patterns, and would raise an alert when a packet arrives too early or too late compared to the norm.

Payload-based inspection

Payload-based NIDS focus on the payload content of a packet. As discussed in Chapter 5, Section 5.2.2, certain CAN bus attacks rely on crafting messages with a specific payload. An adversary can send a packet with a chosen CAN ID and a particular payload which can affect the behaviour of the other ECU receiving that message. If the attacker manages to send this frame without altering the communication patterns, the attack would be undetectable by flow-based NIDS.

Detecting these attacks requires to analyse the payload of messages and compare it with either a historical model considered to be legitimate, or with a signature of an attack. In an anomaly-based detection approach, one can create a normal model defining how legitimate payloads should look like, either by learning it or using vendor's specifications, as discussed in Section 6.2. Any deviation in payload values could indicate a potential attack. One could also rely on a knowledge-based method to compare the packet's payload against signatures of known threats, either with string matching algorithms, or by using regular expression matching algorithm [65].

6.2.3. Metrics for NIDS performance

Metrics such as the *accuracy* are traditionally used to evaluate a NIDS. According to Mitchell et al., the accuracy can be determined based on a NIDS's detection rate on a specific attack set [162]. In order to express the accuracy, the intrusion detection research community relies heavily on metrics such as *true positive* (TP) and *true negative* (TN), and *false positive* (FP) and *false negative* (FN). Table 6.1 formalizes this terminology.

Table 6.1: Attack detection terminology.

Attack Detected? \ Actual attack	Yes	No
Yes	TP	FP
No	FN	TN

When using a NIDS, each alert raised can be classified as either TP or FP: alerts which accurately detected an attack are considered TP, while if there actually was no attack, the alert would be regarded as FP. When the NIDS fails to detect an attack, the event is referred to as FN.

6.3. Taxonomy for CAN bus NIDS

During our study we realized that no taxonomy for CAN bus NIDS was available. Existing taxonomies in other domains are not easily adaptable to fit the specificities of CAN bus NIDS. As a result, it is difficult to compare them in a structured approach. For this reason we introduce a novel taxonomy for CAN bus NIDS and their different approaches for detecting CAN bus attacks. This unique taxonomy is the first to be published, and it enables us to identify the coverage provided by CAN bus NIDS against attacks (RQ 2.2).

To create a well-fitting CAN-specific NIDS categorisation, we use five dimensions, as depicted in Figure 6.2. The first three dimensions are: 1) the type of events considered, 2) the data used for detection, and 3) how the detection model is built. The fourth dimension corresponds to 4) the CAN bus attacks covered (according to the authors) and the fifth relates to 5) the validation strategy followed to test the NIDS. We introduce each dimension below.

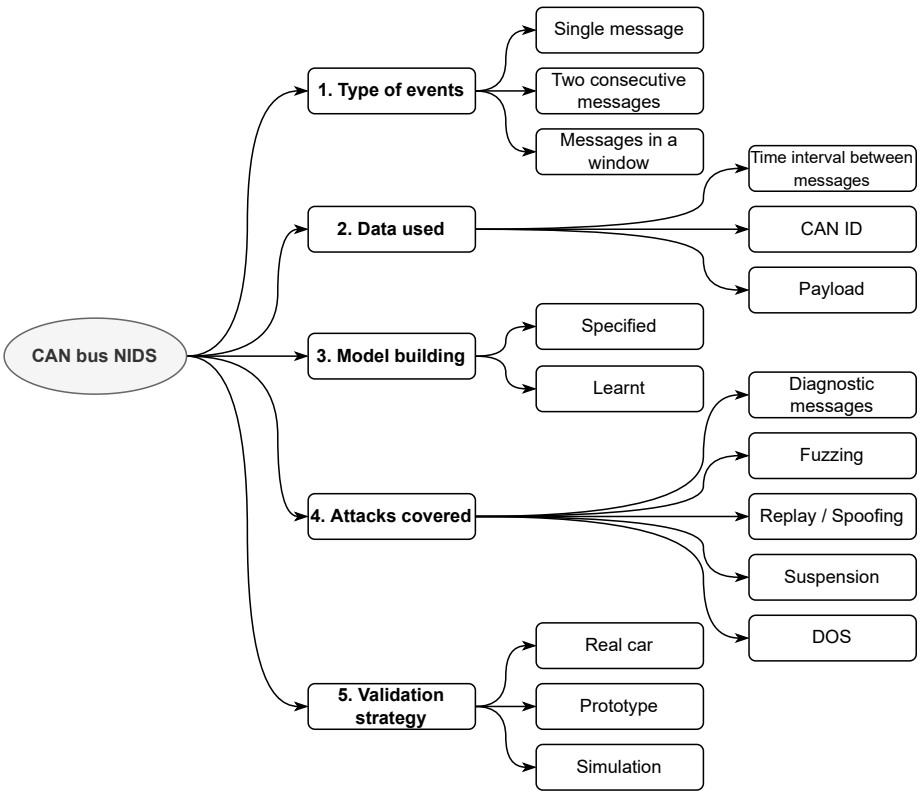


Figure 6.2: Our CAN bus NIDS taxonomy, introducing the five dimensions used for NIDS categorisation.

6.3.1. Dimension 1 - Number of frames

This dimension refers to the amount of CAN frames required by a NIDS to detect an attack on the network. Depending on the attack, the NIDS can detect it with a single CAN message, two consecutive messages, or with all the messages contained in a window.

Single message

By looking at CAN frames individually, a NIDS can detect attacks such as fuzzing or the ones using diagnostic messages. In the first case, an attacker will send packets with illegitimate CAN ID or packets with legitimate CAN ID but illegal payload values. To detect fuzzing attacks, certain NIDS create a white-list of legitimate (i.e., allowed) CAN ID and the possible values of their payload. Any message which does not match the white list will be flagged as malicious.

In the second case, NIDS such as [158] look at the CAN ID of each message. An alert will be raised anytime it sees a diagnostic message, since they should not be seen on the bus while driving [221].

Two consecutive messages

Some CAN bus NIDS leverage the regularity of ECU communications in order to detect attacks. As outlined in Chapter 5, Section 5.2.2, certain CAN bus attacks require to increase the rate of transmission of specific CAN frames for the attack to have the desired effect. As a result, the time interval between consecutive messages for a given CAN ID will be shorten. This principle has been applied in various papers such as [92] to detect CAN bus attacks influencing messages' frequencies: if the interval of time between two messages is smaller (e.g., possible replay or spoofing attack) or greater (e.g., ECU has been silenced) than normal, an alert would be triggered. The term *normal* refers to the expected value according to the model of normal behaviour (see Dimension 3 - Model building).

Messages in a window

As explained in the previous paragraph, variation of messages' frequency of transmission can be a sign of attack. Another way of detecting frequency deviations is to look at the messages for a given CAN ID within a certain window. Different approaches have been published, leveraging windows based on time [230, 254], or on a specific number of messages [130, 143, 171].

6.3.2. Dimension 2 - Data used

In order to detect an attack, a CAN bus NIDS will use different features of CAN frames, such as its arbitration ID, and/or its payload. In addition, the timing characteristic of CAN communications allows to detect attacks by looking at the time interval between messages.

Time interval between messages

As discussed in the previous section, certain NIDS leverage the regularity of CAN communications in order to determine whether the flow of messages is legitimate or not. Such approaches make sense under the assumption that all CAN frames will be sent at a fixed period.

CAN ID

In the case of attacks relying on using specific CAN ID such as fuzzing attacks, certain NIDS focus on the CAN ID of a frame to assert its maliciousness. The NIDS leveraging the CAN ID of a frame to detect attacks are usually performing analysis of single message, as presented in the previous section.

Payload

Another class of NIDS look at the content of a frame's payload for abnormal values. In traditional IT NIDS, this approach is often referred as *deep packet inspection*, as it analyses the payload instead of solely relying on the header's information to detect attacks. Different CAN bus NIDS using this principle have been published (e.g., [223]).

6.3.3. Dimension 3 - Model building

Finally, CAN bus NIDS using an anomaly-based detection approach will likely use a model of the system to be protected and will flag deviations from that model as attacks. There are two main ways to build such models: either by using system's specifications, or by learning it over a certain period of time.

Specified

Building a reference model can be done by manually describing the communication patterns, according to the vendors' specifications. The model is guaranteed to be complete and exhaustive enough so that the chances of having false positive is drastically reduced [240]. First proposed by Larson et al. in [129], this technique could greatly enhance the detection capabilities of NIDS, but it is unfortunately not trivial to obtain such specifications from manufacturers. They are often reluctant to disclose them and keep this information confidential.

Learnt

As specifications for cars are difficult to obtain, most CAN bus NIDS try to built their models during a learning phase. Over a period of time, the NIDS will observe the traffic and derive communication patterns. The challenge with this approach is to make sure that the learning phase encompasses a broad range of situations to guarantee that the model built is as close as possible to the reality. When failing to do so, it increases the risks of false positives.

6.3.4. Dimension 4 - Attacks covered

While designing their NIDS, researchers consider a number of attacks to be detected by their systems. By reviewing their papers, we found out that each NIDS aim to detect at least one of the five attacks we introduced in Chapter 5, Section 5.2.2: attacks using diagnostic messages, fuzzing, replay/spoofing, suspension and DoS.

6.3.5. Dimension 5 - Validation strategy

Researchers test and validate their NIDS with different strategies. We classify them in three main categories: *real car*, *prototype* and *simulation*. In the first category, the researchers implement their NIDS on a controller such as a Raspberry Pi or an Arduino board which they connect to the OBD-II port of their car. In the second category, they build a prototype of a CAN bus on a breadboard to which they attach a number of ECU (Raspberry Pi or Arduino) and emulate an in-vehicle CAN network. Finally, in the last category, they generated CAN traffic with a simulator such as Open Car Testbed and Network Experiments (OCTANE) [68]. We also indicate the approximate amount of data used for testing, expressed as dataset duration.

6.4. Survey of CAN bus NIDS

In this section we present our survey of NIDS for CAN bus. We start by introducing the scope before detailing the NIDS selected for our study. Finally we formulate a number of findings and discuss the insights resulting from our survey.

6.4.1. Scope

For our study we consider all NIDS approaches that perform CAN bus attack detection on a frame level. Some approaches [34–37, 121, 168], while referred to as NIDS, actually attempt to authenticate (or fingerprint) ECU on the bus in order to detect illegitimate sender(s) by using low-level signal characteristics, such as the shape of electrical signals on the bus. Such approaches use a different attacker model as we consider illegitimate traffic may come from the "correct" but compromised ECU. Thus, while such approaches are intelligent and seem promising, they would require another dedicated survey and we exclude them from this study.

Additionally there is a number of articles that did not fit our survey, such as the ones aiming at detecting anomalies like fault patterns [199, 236] for example, or malfunctioning ECU [25]. We also omit the papers proposing methods to solely create a system's behaviour model such as [109, 144, 169, 170, 194].

6.4.2. Survey

In this section we give an overview of the papers meeting our study requirements and briefly describe how their proposed system operates. We structure the following discussion based on the first dimension of our taxonomy. Finally, we present in Table **??** a consolidated overview of the NIDS surveyed according to our taxonomy.

Single message

A number of CAN bus NIDS aim at detecting attacks based on a single frame. Such systems may learn which frames are legitimate. For example, Kang et al. in [118] use a deep neural network to train the model based on the underlying statistical properties of normal and malicious CAN frames. It then extracts the low-dimensional features of CAN messages to discriminate normal packets from malicious ones. Bloom filtering techniques are used in [95] to assess the periodicity of frames, relying on CAN ID and certain parts of the payload. This approach allows for the detection of replay and modification attacks. The application of four different fuzzy algorithms on CAN payloads in [145] allows to classify CAN frames as being normal or injected.

Instead of learning, NIDS can rely on the specifications (see Section 6.2). A NIDS may, for example, look for CAN ID that are not part of a specified list of legitimate ID [133]. Abbott-McCune and Shay propose in [2] to detect the illegitimate use of CAN ID using the fact that only one ECU may send a certain CAN ID. By deploying detectors on the ECU in the CAN network, they can determine whether a given CAN frame has been sent by the legitimate ECU. Similarly, a detector deployed on a gateway could detect if CAN ID appear on the wrong subnetwork. A disadvantage of such a host-based approach is the need to modify these hosts. Redesigning ECU is impractical or at least costly for manufacturers.

Two consecutive messages

Some CAN NIDS leverage the regularity of ECU communications in order to detect attacks. Recall attacks like spoofing may require sending frames with a specific CAN ID at a much higher rate than normal for the attack to have the desired effect. As a result, the time interval between consecutive messages for that CAN ID will shorten. This principle has been used in various papers.

Gmiden et al. [92] propose a simple intrusion detection method for CAN network, based on the analysis of time intervals between consecutive messages. Every time a message with a certain CAN ID is sent on the bus, their algorithm calculates the interval time between this message and the previous message for the same ID. Assuming that each CAN ID has its own regular frequency, if the time interval calculated by the NIDS is less than half of the expected value, an alert is raised.

Moore et al. [164] also exploit the regularity in the timing of CAN communication. For each CAN ID, the NIDS stores the time differences between two successive messages and computes the mean arrival time. In addition, it will register the maximum time difference from the mean, which will be used to define a threshold. An alert is raised if the time between two packets differs from the expected time by more than the maximum time difference plus 15% of the mean.

Otsuka and Ishigooka [182] observed that the frequency of CAN messages may fluctuate due to collisions with other frames. As a result, NIDS relying on naive frequency analysis can be prone to false positives. In an attempt to overcome this situation, they propose a "delayed-decision cycle detection" method to be deployed on a gateway ECU. The idea can be summarized as follow: each CAN ID has a certain cycle of emission called T. If a frame

is received by the gateway at a time a, the legitimate next frame is expected at the time $a + T$. In the situation where a frame with the same CAN ID would be received before, the ECU will hold the frame and wait until $a + T$. If, by this time, a new frame with the same ID is received, the NIDS concludes that the previous message received was spoofed. By holding the frame, the risk of false positive is reduced by not raising an alert at the first sign of deviation.

Marchetti and Stabili [142] leverage the regularity of CAN bus traffic in a different way by noting that CAN ID tend to arrive in a certain order. Their solution learns the recurring patterns of transitions between CAN ID within sequences of frames and detects attacks by looking for unlikely transitions.

Attack frames will often have an illegitimate payload. Stabili et al. [223] consider the Hamming distance between two consecutive payloads to detect large changes in the payload. Their NIDS learns the normal rate of change per CAN ID as a range for the Hamming distance between successive packets' payload. Several attacks, like fuzzing, can cause large changes in this Hamming distance, enabling their detection.

Taylor et al. [231] propose an anomaly detector based on a long short-term memory (LSTM) recurrent neural network (RNN) to detect attacks. The idea is to train a neural network to predict the next packet's payload. Frames are considered malicious if they deviate from the predicated value.

Messages in a window

Even if communication is regular, fluctuations in the reception cycle may occur which could lead to false positives [182]. As attacks tend to influence frequencies for some time, considering sequences of multiple messages may be less sensitive to such fluctuations. Such sequences, referred to as windows, can be formed by taking a fixed number of messages with a given CAN ID, or all messages within a fixed amount of time (all together or per CAN ID). NIDS can extract features of such windows and use them to detect attacks.

Japkowicz et al. [230] propose a frequency-based anomaly detector which uses time-based windows per CAN ID. They use the term *flow* for a window and its features: the CAN ID, the number of packets, the average Hamming distance and variance of the Hamming distance between successive packet payloads, the average time difference and its variance between successive messages. They learn a "historical model" using one-second sliding windows and compute t test scores to compare new traffic with the model.

Waszecki et al. [254] note that CAN messages are not being sent perfectly periodically but are "periodic with jitter". Their NIDS uses arrival curves for each message stream, capturing the earliest and latest time each message in a window should arrive, thus accounting for the possible jitter. Messages arriving too late or early (or equivalently too many or too few messages having arrived at some point in time) signal an attack.

Lee et al. [130] describe an active approach where they query some ECU on the bus. Variation in the response time of the ECU may indicate that there is an attack on the bus. To detect such variations they consider the "offset" (how many messages occur on the bus be-

fore the response) and the "interval" (response time). Unusual combinations of offset and interval values are indicators of an attack.

Narayanan et al. [172] build a model based on Hidden Markov Models capturing the normal, "safe" state of the vehicle. For detection the NIDS maintains a state which is updated by activities (certain CAN frames) and looks for activities that are unlikely given the current and previous states.

Muter and Asaj [171] introduce entropy-based detection to the area of in-vehicle networks. They describe entropy as a "measure of how much coincidence a given dataset contains". They notice that automotive network traffic is regular and much more structured than traditional IT traffic: frames are simple, fixed format, use values with clear bounds and payload that follow some logic depending on the frame's function. With this characteristic in mind, change in entropy could signal an attack. The NIDS from [171] thus monitors the entropy in an automotive network.

Marchetti et al. [143] notice that the experimental evaluation of the entropy-based NIDS of Mutter et al. is rather limited, as it only considers about 15 seconds of CAN traffic for instance, containing only a single class of CAN messages. In order to evaluate the effectiveness of this approach, they implement it and perform some experiments with various parameters and two types of attacks, namely fuzzing and replay attacks, on the data collected from a 2011 Ford Fiesta.

In a similar approach Wu et al. [260] also develop an entropy-based detection method. It uses a sliding window comprised of a fixed number of messages. Compared to the other entropy-based systems, the authors leverage a Simulated Annealing sliding algorithm in order to find an optimal sliding window parameter. Additionally, Wang et al. [253] propose an entropy-based NIDS focusing on the detection of entropy changes of the CAN ID. While Muter's approach [171] considers the CAN ID as an inseparable vector of 11 bits, this system "analyses the entropy change bit by bit".

Combining approaches

CAN bus attacks can take different forms. Consequently, "using only a single algorithm is not enough when considering the need to be able to detect various types of malicious CAN messages" [238]. Miller and Valasek [158] implement such a hybrid NIDS. They note that all known CAN injection attacks rely on either CAN diagnostic messages or standard messages sent at higher rates. To detect these attacks they design a NIDS with two detection modules. The first module looks for diagnostic messages while driving. The second one focuses on the frequency of CAN messages.

A similar NIDS has been proposed by Song et al. in [221]. They do not discuss the knowledge part in depth, beside mentioning that diagnostic messages should not be seen on the bus while driving (i.e., obvious sign of attack). Regarding the behaviour module, they also select the message rate as a significant feature, as described in the previous paragraph. The NIDS monitors the time interval of messages and if an interval between two consecutive packets is shorter than normal, it considers the message to be maliciously injected.

Ujiie et al. [238] propose to combine learning white lists of normal CAN ID and payload

with algorithms to look for cyclic CAN messages sent outside of normal cycles, non-cyclic messages sent at an abnormal frequency, or messages otherwise not matching learnt historical statistical properties.

In [224], Studnia et al. derive from models of behaviour a set of attacks based on a list of forbidden sequences. Their system then checks whether a frame is compliant with the specifications and is consistent with the current state of the system.

6.4.3. Discussion

As seen above, the Dimension 1 (type of events) has implications on the types of attacks that can be detected. Dimensions 2 (Data used) and 3 (Model building) in our taxonomy also impact the performances of the NIDS, which we briefly discuss below.

Dimension 2 - Data used

Time interval between messages: As discussed in the previous section, certain NIDS leverage the regularity of CAN communications in order to determine whether the flow of messages is legitimate or not. Such approaches make sense under the assumption that all CAN frames will be sent at a fixed period. However, we saw in the previous chapter (Section 5.2.1) that this assumption may not hold (at least for certain cars).

CAN ID: In the case of attacks relying on using specific CAN ID such as fuzzing attacks, certain NIDS analyse the CAN ID of a frame to assert its maliciousness. Such NIDS use only a single message. Therefore, they can be blind to attacks using multiple packets (e.g., replay).

Payload: Another class of NIDS look at the content of a frame's payload for abnormal values. In traditional IT NIDS, this approach is often referred as *deep packet inspection*, as it analyses the payload instead of solely relying on the header's information to detect attacks. However, it is not trivial to confidently identify abnormal values in the payload without the car's specifications, which are themselves complicated to obtain (see below).

Dimension 3 - Model building

Specified: Building a reference model can be done by manually specifying the communication patterns, according to the vendors' specifications. The model is guaranteed to be complete and exhaustive enough, so that the chances of having false positives is drastically reduced [240]. As first proposed by Larson et al. in [129], this technique could greatly enhance the detection capabilities of NIDS, but it is unfortunately not easy to obtain such specifications from manufacturers, who consider this information confidential.

Learnt: Since specifications for cars are difficult to obtain, most CAN NIDS try to built their models during a learning phase. Over a period of time, the NIDS observe the traffic and derive communication patterns and other statistical information. The challenge with

this approach is to make sure that the learning phase encompasses a broad range of situations to guarantee that the model built is as close as possible to the reality. When failing to do so, it increases the risks of false positives.

Further considerations: While out of scope for this survey, appropriate response(s) to detected attacks is also an important issue that needs to be addressed when considering practical deployment of CAN NIDS. It is not trivial to choose a "safe" response to a possible attack and only a few possible options are mentioned in the surveyed papers. Abbott-McCune and Shay [2] propose blocking of a malicious message by sending several dominant bits (i.e., bits which value is 0) on the bus which will cause an error and invalidate the message. Miller and Valasek [158] suggest short circuiting the CAN bus when spotting an attack. This would result in putting the car into "limp mode", so the driver could safely stop the car. More work needs to be conducted in order to identify the right and safe response to attacks on cars.

6.5. Conclusion

We analyse in this chapter a selection of NIDS for the CAN bus which have been proposed to protect the users' safeness against CAN-based attacks. The survey we conduct helps us to understand how they function and the principles upon which they rely to detect attacks. Through our analysis, we observed that comparing these different approaches was difficult, as no taxonomy for CAN bus NIDS was available. In other fields, taxonomies help greatly in categorising NIDS approaches. Since existing taxonomies are not fitted to encompass the specificities of CAN bus NIDS, we propose in this chapter the first taxonomy for such systems.

We apply this taxonomy to compare the 24 CAN bus NIDS selected for our study. Although our taxonomy enables a categorisation at a theoretical level, we are left wondering about the practical implications of the NIDS design with regards to the different dimensions presented in the taxonomy. As we can see in Table 6.2, CAN bus NIDS authors use disparate validation strategies, varying in attacks covered, as well as in the nature and the size of datasets. More specifically, the results depicted in the table raise the following questions respective to the five dimensions:

1. How does the type of events considered affect the NIDS performance?
2. The NIDS proposed tend to rely on the timing of frames to detect attacks, but is it an effective indicator of attack?
3. The majority of the approaches build their model by learning, but what are the risks of false positives?
4. While most NIDS aim at detecting replay attacks, how do their design actually perform?
5. Based on the diverse validation strategies used by NIDS authors, can we assess how would a given NIDS perform in different automotive settings and architectures?

In order to answer these questions and draw conclusions regarding the performance of these systems, one would need to provide a quantitative comparison addressing standard NIDS performance metrics such as false positives and detection rate. In the next chapter, we explore how can we evaluate CAN bus NIDS (RQ 2.3) by proposing a practical testing methodology to assess their performance and effectiveness to detect attacks on the CAN bus.

Table 6.2: Inventory of CAN bus NIDS published in the literature.

	Number of frames			Data used			Model		Attacks covered					Validation data			
Papers	Single message	Two consec. messages	All messages in window	Time interval betw. messages	CAN ID	Payload	Specified	Learnt	Diagnostic messages	Fuzzing	Replay/Spoofing	Suspension	DoS	Real car	Prototype	Simulation	Dataset duration (min.)
[118]	✓				✓	✓		✓			✓					✓	-
[95]	✓				✓	✓		✓		✓	✓			✓		✓	-
[145]	✓					✓		✓		✓	✓		✓	✓			30∼40
[133]	✓				✓		✓	∘		∘						✓	-
[2]	✓				✓		✓			✓	✓				✓		-
[92]*		✓		✓	✓			✓		∘	✓			-	-	-	-
[164]*		✓		✓				✓			✓		✓	✓			-
[182]		✓		✓				✓			✓			✓	✓		-
[142]		✓			✓			✓		∘	✓			✓			240
[223]		✓				✓		✓		✓	✓			✓			-
[231]		✓		✓		✓		✓			✓	✓		✓			1140
[230]*			✓	✓				✓			✓	✓		✓			5
[254]			✓	✓			✓					∘	∘			✓	-
[130]			✓	✓						✓	∘		✓	✓			8
[172]			✓			✓		✓		∘				✓			-
[171]			✓	✓		✓		✓			✓		✓	✓			0.25∼5
[143]*			✓	✓		✓		✓		✓	✓			✓			240
[260]			✓		✓			✓			✓		✓	✓			8
[253]			✓		✓			✓		✓	✓			✓			-
[158]*	✓	✓		✓	✓			✓		✓	✓		✓	✓			-
[221]*	✓	✓		✓	✓		-	-	∘		✓		✓	✓			40
[238]*	✓		∘	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓		-
[224]	✓		✓		✓	✓	✓			✓	✓			✓			3.4
[237]	-	-	-	✓	✓	✓		✓		∘				✓			-

✓ Explicitly mentioned in paper ∘ Implicitly mentioned in paper
- Unclear/not mentioned * Implemented for evaluation

7

Evaluation Framework for CAN Bus NIDS

Although a number of NIDS have been published to detect attacks on the automotive CAN bus, their effectiveness and performance are unclear. This situation comes from the different approaches adopted and the disparate evaluation methods followed to assess them. Additionally, there is currently no mean to evaluate them in a standardised manner, conversely to NIDS in other domains. As a consequence, CAN bus NIDS are hard to assess and compare to each other. In this chapter, we address this challenge by proposing a unified evaluation framework for CAN bus NIDS and use it to test and evaluate a selection of them. This allows us to compare the different approaches and to draw a number of conclusions regarding intrusion detection on CAN networks. For instance, we observe that all these approaches refer to narrow indicators of attack that offer little guarantees that attacks will be detected. Finally, we argue that the implementation of CAN in the automotive setting makes it hard to distinguish between legitimate and malicious packets. One solution could be the design of a "meaning-aware" NIDS built with manufacturer's specifications.

The results reported here have been published in [61] (paper 2 of Section 1.4).

7.1. Introduction

In the previous chapter, we investigated the CAN bus NIDS developed to detect attacks targeting the CAN bus. We observed that they are designed based on various strategies, relying on different characteristics of CAN communications. However, we do not know how effective these characteristics are to detect attacks. While we provide the mean to compare the NIDS at a theoretical level with our taxonomy, it is not straightforward to infer any conclusion about their practical performance. The problem in comparing their effectiveness comes from 1) the lack of standardised testing framework, and 2) the disparate and often limited evaluation provided by the authors in their papers. We explore these two points below.

In other fields such as IT, NIDS are evaluated with established methodologies, allowing the benchmarking and assessment of their performance in a unified and replicable way [106]. These methodologies involve the usage of datasets of network traffic publicly available, containing legitimate and malicious traffic. By applying these methodologies, NIDS authors can evaluate new algorithms in an objective way. They can assert the effectiveness of their approach based on the metrics used and the results obtained after testing. Having an established evaluation methodology enables the straightforward comparison of NIDS against other designs. In the automotive CAN context, such a methodology does not exists. Consequently, the authors of novel CAN bus NIDS do not have the possibility to evaluate their systems in a standardised manner. It results in the literature in disparate performance assessments, leading to the impossibility to tell an effective design from another.

We observe the following issues with regards to the testing of NIDS proposed in the literature. First, there is a disparity between CAN data collected and used. Testing NIDS obviously requires vehicles' CAN traffic. As depicted in Table 6.2 in the previous chapter (see Attacks covered and Validation data), researchers in the literature collect normal CAN data (i.e., data without attacks) in one of the following ways: taping into a real car's network (mainly via the OBD-II port), using a CAN bus prototype or even a software simulating a CAN bus. Second, the durations of the datasets used (when explicitly stated) vary greatly across the different papers. This fact, combined with the observation that most NIDS approaches build their model by learning, makes it difficult to estimate how "complete" the model will be, and thus how prone to false positives a NIDS will be. Third, another disparity comes from the creation of attack datasets. As attacks are obviously required to assess a NIDS, authors in the literature do not always consider the same types of attack (presented in Chapter 5, Section 5.2.2), and also do not generate their malicious traffic the same way. The attack implementations, when explicitly stated, can consist in launching them on a car, on a CAN bus prototype or by modifying the CAN data previously collected by manually injecting packets in the file. Finally, the overall validation of NIDS algorithms is often limited in scope, with only one car considered. As we observed in Chapter 5, the diversity in car architectures on the market calls for testing conducted on multiple vehicles to reliably assess the effectiveness of a given NIDS. Overall, the issues above lead us to the following question: how can we evaluate CAN bus NIDS on equal ground? (RQ 2.3)

7.1.1. Summary of Contributions and Outline of the Chapter

In this chapter, we answer the need for a standardised testing methodology by proposing as a contribution a unified framework for evaluation of CAN bus NIDS. Our framework includes the data required for testing different systems in a consistent and replicable fashion, guaranteeing a uniform method for evaluating NIDS on equal ground (answering RQ 2.3). Our datasets we made publicly available cover normal driving situations for different cars, as well as a CAN bus prototype, and several different attacks for each dataset. As another contribution in this chapter, we also apply our framework to evaluate some of the NIDS presented in the previous chapter and draw a number of conclusion regarding the design of NIDS for CAN.

We start in Section 7.2 by introducing our methodology and explain the dataset creation procedure, followed by the selection and implementation of the algorithms we test. Then we present our experiments in Section 7.3, where we evaluate the algorithms for false positives and attack detection. Finally, we discuss the results obtained in Section 7.4 and provide a number of considerations for the design of effective NIDS for CAN.

7.2. Methodology

We present in this section the methodology followed to create our evaluation framework and the implementation of the NIDS selected for our tests. We begin with the collection of CAN data. Testing NIDS requires network traffic with and without attacks. As CAN implementations can vary a lot from one car to another, it is essential to include data from different cars. We record data from two cars, an Opel Astra and Renault Clio, and a CAN bus prototype we built.

Data covering the different types of attacks is needed. To obtain such data we use our datasets to create attack datasets which contain the attacks presented in Chapter 5. These datasets are developed in the same way than most papers on CAN bus NIDS. We also consider Kia Soul data provided online[1] by the Hacking and Countermeasure Research Lab (HCRL).

Then we select and implement a number of CAN bus NIDS from our survey in Chapter 6. This selection is based on the availability of sufficient information in the papers to guarantee an accurate implementation. Finally, we evaluate these NIDS by first conducting false positive testing and then attack detection.

Our datasets are available online on the TUe Security Group website[2].

7.2.1. Dataset creation

In this section, we outline the process we follow to create CAN datasets for both normal driving and the attack scenarios described in Chapter 5, Section 5.2.2.

[1] `http://ocslab.hksecurity.net`
[2] `https://security1.win.tue.nl` (under *Research/Software and Data*)

Driving data collection

We collect data from the two aforementioned cars and the prototype. For the cars we record CAN traffic through a CAN-to-USB interface (CANtact[3]) connected to the OBD-II port of the car. CAN bus traffic is logged on our laptop with the CAN sniffer `candump` from the *canutils* tool suit[4]. These vehicles do not have a firewall or gateway filtering CAN messages. We drive the Opel for approximately 20 minutes (analysis of this set shows the data is already comprehensive after a few minutes) and the Renault for about 4 minutes in urban environments (university campus and city). We believe this data to be sufficiently representative for normal urban driving for our purpose, though it obviously does not cover rare/specific events (e.g., empty tank). Such corner cases could provide additional challenges to NIDS, meaning that the results in our framework should be interpreted as best case scenario for the NIDS.

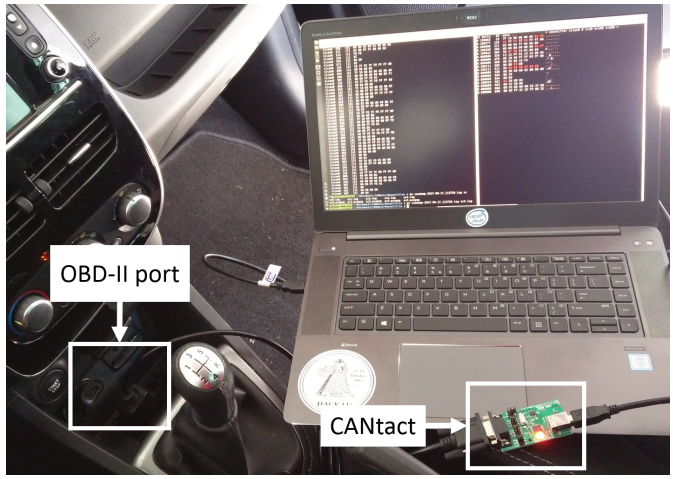


Figure 7.1: Driving data collection inside the Renault Clio.

The prototype we build is depicted in Fig. 7.2. It comprises two ECU (Arduino boards mounted with CAN shield), a Volkswagen instrument cluster and a joystick which is used to replicate a vehicle's throttle: when pushed forward, one can see the speed increasing on the instrument cluster. (The CAN/Wifi module is not used in this study.) We connect the CANtact to the bus and we record the data of the prototype for about 4 minutes, while using the joystick to emulate a driver accelerating and decelerating.

We partition the driving data in two parts, respectively accounting for 70% and 30% for the entire data trace. The first part is referred to as the *training dataset*, and it is used to train the NIDS. The second part is called the *testing dataset*, and it is used for evaluating their performance with respect to false positives. In Table 7.1 is given an overview of the data we collect for our different systems.

[3] `https://linklayer.github.io/cantact`
[4] `https://github.com/linux-can/can-utils`

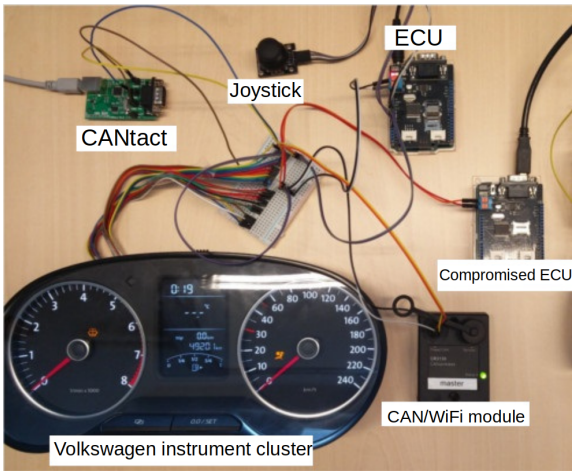


Figure 7.2: Data collection from home-made CAN bus prototype.

Attack dataset creation

To create attack datasets covering the attacks presented in Chapter 5, Section 5.2.2, we simulate them on the two cars and preform them on the prototype. This simulation is done by modifying normal driving data that is not in the training set in different ways. For the diagnostic attack, 10 packets with CAN ID greater than 0x700 [256] are injected at random position in the files. In a similar fashion, two different types of fuzzing attacks are performed: one consisting in injecting 10 packets with unknown CAN ID, i.e., CAN ID not present in the training dataset, and below CAN ID 0x700. The other fuzzing attack involves the replacement of the payload of 10 frames belonging to a legitimate CAN ID, without injecting new packets. The bytes of the payloads are set to illegitimate values, i.e, values not comprised within the range of values observed in the training dataset.

The replay attack for the two cars is replicated by injecting 30 times an arbitrary packet seen in the dataset. By adjusting the timestamps, we simulate the packets being sent 10 times faster than normal onto the bus. Since we do not know the functionality of that CAN ID and the semantic of its payload, we injected it without modification. On our prototype, we want to spoof the speed displayed by the speedometer. Since we know the

Table 7.1: Summary of the normal driving data collected.

	Opel Astra		Renault Clio		Prototype		Kia Soul	
Dataset	Train	Test	Train	Test	Train	Test	Train	Test
Duration (sec)	967.6	414.6	192.5	82.5	189.3	82.0	1,331.4	572.8
Number of packets	1,883,070	806,999	270,596	115,971	70,204	30,086	2,599,239	1,113,905
Number of unique CAN ID	85	85	55	55	17	17	28	27

payload semantic of the corresponding CAN ID, we can attack the speedometer to force it to display any speed value, as long as we send the spoofed frames faster than the legitimate ECU. We program the attacking ECU (called "compromised ECU" in Figure 7.2) to spoof the ECU which normally sends speed information to the instrument cluster. By replaying packets with this specific CAN ID and the payload value adjusted to our goal, we generate a successful attack which displays a speed of 220 Km/h in the instrument cluster for 10 seconds.

To simulate a DoS attack, we replace all messages within a 10-second window with messages having a CAN ID of 0x000 sent at a rate of 4 packets per milliseconds. This simulates flooding a 500 Kbps CAN bus (same bus speed as our cars). Finally, we simulate a suspension attack by deleting all messages with a certain CAN ID over a 10-second window.

HCRL online datasets: The HCRL data contain an attack-free log trace captured from a Kia Soul during 30 minutes. The researchers providing the data also address attacks similar to the ones we described previously, logging CAN traffic through the Kia's OBD-II port while message injection attacks are performed. They include a fuzzing attack, in which they inject frames with random CAN ID and payload every 0.5 milliseconds; an impersonation attack injecting frames with CAN ID 0x164; a spoofing attack, consisting in the injection of certain CAN ID related to RPM/gear information every 1 millisecond; and finally a DoS attack in which they inject messages with CAN ID 0x000 every 0.3 milliseconds. These data have also been used in previous NIDS research [145, 260], and some analysis of the data is presented in [95].

7.3. Experimentation

We use our evaluation framework to compare NIDS on equal footing. We implement all the NIDS from the papers published before mid-2018 which provide sufficient details to allow of a faithful reproduction (the ones marked with a star in Table 6.2). Some papers propose hybrid NIDS combining different modules, each with their own detection method. For such papers we implement the modules described in sufficient details. This results in 8 NIDS from 7 papers.

NIDS N1 reproduces the diagnostic messages detection module suggested by Miller and Valasek in [158], which simply looks for CAN ID above 0x700 as mentioned in [256]). N2 is a combination of the two pattern-matching algorithms proposed by Ujiie et al. in [238]: one algorithm looks for invalid CAN ID (also proposed by Abbott-McCune in [2]) and the second for payload values that do not match values previously observed. NIDS N3 [92], N4 [230], N5 [164], and N6 [221] leverage the regularity of CAN messages in order to detect CAN bus attacks. We also develop two entropy-based systems following the work of Marchetti et al. [143] which fills in details for an idea proposed by Muter and Asaj [171]: N7 computes the entropy for a window of CAN messages, while N8 calculates the entropy values of flows of CAN ID individually.

Setup and Training: We use the training datasets introduced in the previous section to train the NIDS (for the ones requiring a training phase). Following the authors' guidelines

Table 7.2: False positives testing: number of alerts generated for the attack-free datasets.

	Opel Astra		Renault Clio		Prototype		Kia Soul	
NIDS	# alerts	**%**	# alerts	**%**	# alerts	**%**	# alerts	**%**
N1	1657	0.20%	0	0%	229	0.76%	0	0%
N2	19917	2.46%	35288	30.428%	2444	8.12%	110949	9.96%
N3	411	0.05%	384	0.331%	8770	29.15%	302	0.027%
N4	0	0%	0	0%	0	0%	0	0%
N5	253468	31.40%	19008	16.39%	12	0.04%	4200	0.377%
N6	165	0.02%	163	0.14%	1806	6%	58	0.005%
N7	0	0%	0	0%	0	0%	4	0.0003%
N8	8	0.0009%	1	0.0008 %	2	0.006	7	0.0006%

for NIDS parameters as much as possible still leaves some choices to be made. N4 computes an anomaly score over a sequence of elements, but [230] does not specify the size of the sequences. As elements themselves are computed over a one-second window, we choose to set the sequence size to two. To interpret the resulting anomaly scores, we consider the maximum and the minimum scores as thresholds. For N7, a parameter k helps to adjust the detection threshold and requires to be adjusted for each car's dataset. For the Opel Astra, the Renault Clio and the Simulator and the Kia Soul, we follow [143] and select the lowest value which gives no false positive; k=4, k=2, k=5 and k=3 respectively. Regarding the Kia, avoiding the 4 false positives that are produced at this setting would require a k so large that no attack would be detected (see Table 7.2). N8 needs a window size (s messages) tailored to each dataset [143]. We choose the value producing the least number of false positives, giving s=40 for Opel Astra and the simulator, and s=50 for the Renault Clio and the Kia Soul.

For NIDS with CAN ID specific models (all but N1 and N7) one has to decide what to do with ID not seen in training. We assume an "unknown CAN ID" alert is raised, enabling detection of attacks using unusual CAN ID (Diagnostic, Fuzzing CAN ID and DoS). This assumption does not add any false positives as no such ID appear in the testing datasets.

False positives testing: Before evaluating the NIDS capability to detect different types of attacks, we test how many false positives they produce. Table 7.2 shows the false positives generated by the NIDS using the parameters above. Some authors motivate the use of frequency analysis for CAN bus NIDS by assuming that most (or all) traffic on the bus is regular: if each CAN ID is sent with a defined period, a change in message frequency could indicate an attack. However, our car data shows certain CAN ID are sent irregularly (see Chapter 5, Section 5.2.1).

To ensure that it is not simply the presence of irregular CAN ID that causes the false positives reported in Table 7.2, we repeat the experiment after removing the irregular CAN ID. We arbitrarily define irregular CAN ID as those having a variance of their inter-arrival time above 0.002. This leads to 21, 10 and 1 irregular CAN ID accounting for approximately 10%, 6% and 0.5% of the traffic for the Opel Astra, the Renault Clio and the Kia Soul respectively. As results are very similar to those in Table 7.2, we do not present them

Table 7.3: CAN bus NIDS testing results for various datasets.

	Opel Astra						Renault Clio						Prototype						Kia Soul				
NIDS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Replay	Suspension	DoS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Replay	Suspension	DoS	Diagnostic	Fuzz. CAN ID	Fuzz. Payload	Replay	Suspension	DoS	Fuzzing	Impersonation	Spoof RPM	Spoof drive gear	DoS
N1	✓	-	-	-	-	-	✓	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
N2	✓	✓	✓	-	-	✓	✓	✓	✓	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
N3	✓	✓	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓
N4	✓	✓	-	-	-	✓	✓	✓	-	-	-	✓	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓
N5	✓	✓	-	-	-	✓	✓	✓	-	-	-	✓	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓
N6	✓	✓	-	✓	-	✓	✓	✓	-	✓	-	✓	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓
N7	-	-	-	-	✓	✓	✓	✓	-	✓	✓	✓	-	-	-	-	✓	✓	✓	✓	✓	✓	✓
N8	✓	✓	-	-	-	✓	✓	✓	-	-	-	✓	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓

✓ Attack detected - Attack not detected

in detail and conclude that irregular CAN ID do not influence the false positive rate much.

The assumption made by N1 and several papers that CAN ID above 0x700 are reserved for diagnostic purposes and should never be seen on the bus while driving is apparently wrong. The Opel data shows CAN ID 0x772, 0x773, 0x778 and 0x788 are repeatedly sent every second and the Volkswagen instrument cluster in our prototype sends out CAN ID 0x727 among others.

A detection module in N2 looks for payload bytes not observed during the training phase. Our implementation raises an alert for each byte value that is not in range. This means that up to 8 alerts may be generated by a single CAN frame (see Chapter 5, Section 5.2.1). It is not accounted in the reported rate that multiple alerts may actually come from the same message.

Concluding, we see some approaches have very high false positive rates. Note that even a seemingly low false positive rate of 0.001% still translates to a false positive every minute.

Attack detection: We run our experiments using the attack datasets and report the results in Table 7.3. A tick in the table indicates the NIDS raises at least one alert when run on that attack dataset. We see that N1 by its nature is very limited in what it can detect. All others are able to detect the attacks on Kia Soul but on the other data sets they each miss several (types of) attacks. Attacks performed somewhat subtly may not be detected. We elaborate our findings in the next section.

7.4. Discussion

We can conclude from our survey of CAN bus NIDS and the tests performed that present NIDS work in one of these three ways. They try to detect either 1) flooding of the CAN

bus, 2) diagnostic messages, or 3) illegal payload values.

The reason behind the existence of the first method is a "safety feature" that is built in the CAN bus implementation and that has an interesting positive security side-effect. This feature is that during normal car operations, ECU keep broadcasting frames at regular intervals. Therefore, the recipient ECU receive a continuous flow of data, and (depending on their implementation) may be instructed to disregard inconsistent data (see "message confliction" in Chapter 5, Section 5.2.2). We call this *Safety Feature 1*. Therefore, in order for an attacker to force the ECU to listen to her data (in presence of the "normal" data which is still being broadcasted continuously) she may need to *flood* the targeted ECU, sending many more messages than it already receives. Eventually, the receiving ECU should have some quantitative reason to listen to the attacker and disregard the "legitimate input" (which is still being broadcast). Hence, a peak in the frequency of frames in the CAN bus could actually be a good indication of the presence of a spoofing attack. As explained below, however, this works only on ECU which do not implement specific safety features against inconsistent messages [160].

The second method detects the usage of diagnostic messages to compromise ECU. In 2011, Checkoway et al. were able to control their 2009 Chevy by sending diagnostic messages onto the CAN bus while driving [31]. Recent cars have a protection mechanism which prevents the ECU from accepting diagnostic messages while the vehicle is at speed. We call it *Safety Feature 2*. According to Miller and Valasek, most cars manufactured after 2010 are designed to disregard diagnostic messages if the vehicles are driving above a certain speed (8 km/h). However, they show in [160] that it is feasible to inject messages to trick ECU to believe that the car is idling. Consequently, diagnostic message based attacks may still be possible while the car is at speed.

We should acknowledge that the combination of Safety Features 1 and 2 has determined a noticeable increase in the security of cars, that are nowadays much more difficult to attack while in motion than they used to be. As pointed out by Miller and Valasek in [160], these safety features limit the criticality of CAN bus attacks: the attackers can only control the car at low speed, or the control was unreliable and sporadic [157]. However, new research demonstrated how these safety features could be bypassed, allowing an attacker to gain a more consistent control over the target [160, 183].

The third method of detection looks for illegal payload values. NIDS relying of this method require a model of the CAN system. While monitoring the network, they consider values outside of baseline ranges as indicator of attack. As one can see in Chapter 6, Table 6.2, all but one payload-based NIDS published learn their model. Our experiments show that this approach result in a significant amount of false positives (e.g., see NIDS N2 in Table 7.2).

Note the absence of a fourth category: standard signature-based detection, which is very common in general IT. Several papers [109, 170, 223] discuss why they do not adopt this approach. Below we also cover the limitation of this approach in the automotive setting.

Considerations on the NIDS limitations

Based on our research, we make a few considerations regarding the limitations of each of these methods.

Detecting flooding of CAN frames suffers from the following points. First, this approach is prone to false positives. Our experiments show that frequency-based NIDS produce too many false positives to be useful. In addition, in the papers we considered, CAN traffic is often assumed regular, but there are fluctuations in the baseline communication (depending on the car and its implementation) that make it difficult to draw the line between legitimate and non-legitimate traffic. This method detects only a relatively small class of attacks, relying on message injection. These attacks work only on some ECU, which are usually non-critical: as outlined in [160], the ones responsible for critical actions like braking or the power steering usually have additional safety features to handle conflicting messages (e.g., the brakes would brake and the power steering would shut off, reverting to "manual" steering).

The second method covers only a very specific type of attack and is also implementation dependent. In fact, there is no standard convention for identifying diagnostic messages from their CAN ID. While [256] states that diagnostic messages have a CAN ID above 0x700, our experiments show that such CAN ID can be found in regular operation of certain vehicles.

The third method aims at spotting attacks based on payload values. However, to look at payload without systems' specifications means that the model needs to be learnt, which makes a NIDS prone to false positives. A CAN frame's payload provides little information without knowing how to interpret it and this interpretation can vary a lot between cars or even between CAN ID. Having vendor specifications would provide what we call *meaning-aware* intrusion detection. We consider intrusion detection to be meaning-aware if it distinguishes malicious network traffic from normal traffic by its meaning (or semantic). Without being able to make such a distinction, one is limited to other indicators of attack, for example by trying to detect side-effects to specific attack methods. In the case of the approach tested, these indicators of attack are in our opinion rather limited. Meaning-awareness is thus essential to establishing general indicators of attack for detectors that are hard to circumvent and can give *actionable* alerts: ones that provide enough information to enable appropriate response.

Finally, we believe the fourth method also limited, as signature-based systems will never provide appropriate coverage in CAN IDS. The shortcomings of such systems for traditional IT systems have already been explored in the literature. They are nevertheless still used in IT, because they still have a reasonable business model, which is made possible thanks to the huge and uniform installed base, the thousands of honeypots and a true worldwide infrastructure that allows to detect attacks early and "quickly" generate and roll-out signatures for them. As already pointed out in [170, 171], diversity in CAN bus implementation and the lack of this infrastructure make, in our opinion, signature-based detection virtually hopeless, just like it is hopeless in ICS [67].

Considerations regarding CAN

In the light of what we have researched, we believe it is virtually impossible to do payload-based (let alone meaning-aware) intrusion detection on CAN bus, due to a design "shortcoming" of CAN bus. Doing intrusion detection on CAN bus is somewhat similar to intrusion detection on an encrypted channel: the lack of a uniform specification and semantics make it very challenging to distinguish between normal and malicious traffic. Our study shows that in CAN bus traffic, there are not many indicators of attack. Moreover, since each model of car presents a different architecture with a number of ECU designed and implemented by different parties, we believe it is virtually impossible to perform intrusion detection in a generic approach.

CAN bus has clearly been designed without thinking about indicators of attack, like anomalies in traffic. We believe this is an engineering "mistake", often present particularly in closed systems, i.e., systems which in theory should not be reachable from outside. With the automotive industry pushing towards Internet-connected vehicles and V2X communications, cars are no longer to be considered as closed systems.

In our opinion, to be able to do meaning-aware intrusion detection, it is essential that the underlying system allows the monitoring system to understand what is happening at each moment. Examples of systems that do allow easier distinction between normal and malicious traffic are BACnet systems [28, 70], some of the ICS protocols [69], and web applications to a lesser extent.

One possible way out of this impasse (that we can think of) is a combination of white box and specification-based [67, 240]. However, these approaches require significant information about the system being monitored. ECU behave and communicate differently in cars from different manufacturers, and even in different models from the same manufacturer. At this stage of our research we believe that the only hope to do content-aware intrusion detection on CAN is by having the manufacturer's specifications and analyse whether it is possible to design an effective specification-based NIDS.

7.5. Chapter Conclusion

In this chapter, we introduce a unified evaluation framework and use it to assess various CAN bus NIDS present in the literature. With the different datasets we provide, we can successfully test algorithms and evaluate their false positive rates and attack detection. This comparison enables us draw several conclusions regarding intrusion detection on CAN networks.

We observe that the automotive setting is challenging for NIDS. In fact, CAN messages provide little information and their meaning varies wildly from car to car. As a consequence, all the approaches proposed in the literature refer to indicators of attack that offer little guarantees of detecting an actual attack, let alone give much information about the attack if one is detected. As we observed in this chapter, the performance of the NIDS published may also come from the limitations of the model building strategy adopted (i.e., learnt).

We consider that none of these approaches can be regarded as a meaning-aware intrusion

detection system. In the light of our study, we believe that only such systems can provide effective and reliable detection capabilities for CAN. Building a meaning-aware NIDS requires manufacturers specifications, as they provide information about CAN communications such as the legitimate CAN ID, and data transmission patterns. These specifications can be converted into system's models, enabling the creation of effective NIDS for CAN bus.

Establishing the feasibility of a meaning-aware NIDS requires the cooperation of the manufacturers. This would need them to publicly share their cars' designs, yet considered today confidential. Their change of posture would represent the first step enabling effective NSM and other solutions capable of protecting the safeness of their vehicles' users.

7.6. Part II Conclusion

In this second part of the thesis, we investigate the state of NSM in the automotive setting with a focus on CAN bus and related NIDS. The design and conception of cars have evolved over time to the extent that modern vehicles are no longer to be regarded as mechanical devices, but rather as computer networks on wheels. Technological advancements and increased connectivity offer innovative features, yet also introduce new security risks. Following recent research demonstrating safeness-critical attacks abusing the in-vehicle CAN bus, a growing body of work has been focusing on the design of NIDS to detect such attacks. However, the performance of these systems and their effectiveness to protect users' safeness is unclear.

We begin in Chapter 5 with the study of in-vehicle networks and the CAN bus, which lays out the knowledge required for the analysis of NIDS in the subsequent chapters. Additionally, we identify the capabilities of an adversary and explain in detail how CAN bus attacks can alter the operations of a car. Overall, this study allows us to scope our research and establish assumptions to guide our work.

After understanding how an attacker may abuse the specifications of CAN, we focus in Chapter 6 on the detection of attacks. A number of NIDS designed for the CAN bus have been proposed in the literature over the past years. We investigate their functioning and study the principles upon which they rely to detect signs of attack. We introduce a novel taxonomy for these systems and propose the first organised inventory of NIDS published. This work enables us to identify the coverage CAN bus NIDS provide against attacks.

In Chapter 7 we research how to assess CAN bus NIDS practical performance. The evaluations provided by NIDS authors in their papers are often disparate and limited, making it difficult to draw objective conclusions about the systems' performance. For this reason, we develop a unified evaluation framework, allowing of the assessment of NIDS performance on equal ground. After introducing a testing methodology and creating diverse datasets, we evaluate a selected number of NIDS with our framework. Our results outline the difficulty of performing intrusion detection on the CAN bus and the limitations of some of the published NIDS approaches.

Overall, our results on CAN bus NIDS shed new light on the inner-workings and effectiveness of these systems. We demonstrate the challenges brought by the nature of automotive

settings, combined with the lack of clear indicators of attack. The lessons learned from our study enable us to propose a number of considerations for the design of effective NIDS algorithm for the CAN bus. We argue that, to overcome the aforementioned challenges and be effective, a NIDS should be "meaning aware", for example by leveraging manufacturer's specification to build a model of the system to monitor and detect attacks based on the semantics of the observed communications.

8

Conclusions

In this final chapter we summarise the results of our work presented in this thesis and we discuss how they contribute to answering the research questions formulated in Chapter 1. We draw a number of conclusions from our research, enabling us to fulfil our objective of identifying the requirements for network security monitoring (NSM) and intrusion detection in safeness-critical environments (SCE).

Let us first recall the scope of our work. We introduced the term *safeness* to conceptualise the combination of physical and digital safety, and *safeness-critical environments* to qualify environments where cyber attacks can have critical impact on their users' safeness. We identified in Chapter 1 a number of characteristics which make SCE challenging from a security point of view. First of all, these environments are comprised of a broad range of devices, diverse in type and function (C1). These systems are connected together and form ecosystems, with some of them directly controlled by human and others operating autonomously (C2). Additionally, these devices spread in space and time, meaning that the devices in a given SCE can be physically located in various geographical locations, and their lifecycle can span over multiple years, resulting in the coexistence of recent and older products on the network (C3). Finally, SCE are safeness critical: cyber attacks can physically arm users and impact their privacy (C4).

We investigated in this thesis two environments presenting the characteristics of SCE, Healthcare Delivery Organisations (HDO) in Part 1, and modern cars in Part 2, with the goal of analysing the feasibility of NSM and, where possible, enable effective monitoring in these environments. In this chapter we conclude these two parts in the following two sections by summarising their respective contributions and answers to the research questions. Additionally, we outline some of the limitations of our work, as well as future research directions to pursue the continuation of our endeavour. At the end of this chapter, we conclude our work by answering the main research question, and share some of the lessons we learnt.

8.1. Conclusion of Part 1

In the first part of this thesis, we started with the study of HDO. We lay out below the summary of contributions, the limitations of our research and future work.

8.1.1. Summary of Contributions

The digital transformation of HDO, combined with the deployment of connected medical devices, have increased their exposure to cyber threats. It is unclear what is their current security posture and how able are they to defend themselves from cyber attacks. Our first research question to answer is:

> **RQ 1.1:** *What is the technical state of readiness of HDO in the face of the current threat landscape?*

In the light of the rise of cyber attacks on HDO and disclosures of vulnerabilities on connected medical devices, we explored in Chapter 2 the current state of network security in these environments. These organisations are comprised of a broad range of interconnected devices, ranging from Information Technology (IT), Operational Technology (OT) to Internet of Things (IoT) and Internet of Medical Things (IoMT). New cyber security risks are introduced as the number and the diversity of devices increase. To better understand how exposed are HDO to those risks, we performed a comprehensive security analysis of multiple environments through two observational studies. These studies have such breadth and depth that it arguably forms the most comprehensive analysis of its kind to date. The two main contributions resulting from our work are the following:

- We demonstrated that HDO are large ecosystems of devices that are difficult to manage and secure as they comprise a variety of software, vendors, and protocols.
- We identified the presence of insecure and clear-text protocols, as well as weak encryption schemes, which can directly expose patient data to attackers and allow for cyber-physical attacks that can adversely affect patient health.

Our work answered RQ 1.1 by highlighting security gaps that are challenging to address due to the complex nature of HDO. In particular, our research pointed out limitations in current network security solutions deployed in HDO. These solutions, being initially developed for IT environments, do not fit well the specificities of HDO. Two issues stood out: 1) we observed that current device classification solutions leave HDO's network operators with a significant amount of unknown devices, and 2) our analysis shed light on the usage of healthcare network protocols for which no security research have been performed yet. As these two issues could introduce significant risks, we used them to frame our study in the subsequent research questions, RQ 1.2 and RQ 1.3 respectively.

Device classification is the process of identifying the type of a networked device. It is critical for network visibility as it enables the creation and maintenance of a device inventory required for other security controls like vulnerability assessment. Current solutions work either with the use of manually-defined fingerprints, or with black-box machine learning techniques. Both approaches come with their own limitations: on the one hand, fingerprinting presents issues such as scalability and low performance, and on the other hand, black-box machine learning techniques do not allow for straightforward interpretation and actionability of the results. This situation led us to consider the second research question:

RQ 1.2: *Can we overcome the current limitations of device classification?*

We proposed in Chapter 3 a semantic similarity-based clustering method to complement fingerprinting-based classification solutions. After implementing and testing our method, we demonstrated that we can successfully classify up to half of the unclassified devices with a high accuracy. This work contributes to the improvement of device classification in the following ways:

- Our method can reduce a significant portion of the work involved in manual and fingerprint-based classification. It automatically provides labels with high granularity while allowing a domain expert to manually verify these labels.
- Compared to black-box machine learning approaches, our method provides insights into the classification mechanisms, thanks to the methodology's steps Feature selection and Distance function definition. Being in control of the whole process, we can better understand the results provided and fine-tune any of the steps if required.

Our novel approach allowed us to answer RQ 1.2 and contribute to the improvement of state-of-the-art classification engines. This white-box approach adds actionability and increases the results' reliability by combining automatic classification with manual verification. After validation with domain experts, we also demonstrated its effectiveness to solve industry-wide problems.

Following on to the second issue with HDO identified earlier, we focused our attention in Chapter 4 on the usage of communication protocols and the presence of medical devices which could be potentially insecure. Our previous analysis in Chapter 2 revealed the usage of certain medical systems and healthcare protocols for which no attacks were published, leading us to consider our third research question:

RQ 1.3: *Are there unpublished security vulnerabilities in healthcare network protocols or medical devices which could impact patients' safeness?*

We argued in Chapter 4 that vulnerability research as an ongoing process is an essential part of a good security strategy. Finding flaws before bad actors enables proactive remediation and patching, and allows the improvement of detection capabilities of IDS. For this reason, we selected certain healthcare communication protocols (standard and proprietary) observed in HDO and we conducted security analysis to identify the presence of weaknesses. After building a lab replicating a section of an HDO's network with different (medical) devices, we identified weaknesses in these technologies. We demonstrated a number of attacks exploiting the newly found vulnerabilities, which can impact the safeness of patients. Our security research contributes to the enhancement of NSM capabilities in HDO in the following ways:

- We created new IDS rules to detect the attacks demonstrated leveraging the vulnerabilities we found. Knowing that IDS deployed in HDO are initially designed for enterprise networks (and therefore threats), they need to be tuned and adapted to

better fit the HDO environments and detect attacks leveraging HDO-specific protocols.

- New attack datasets can also be created to help with the evaluation of IDS. This contributes to the remediation of the lack of relevant datasets available for IDS testing purposes.

Our work answered RQ 1.3 and provides a more accurate picture of HDO's exposure to cyber attacks. Moreover, it helped us to understand the risks that come with certain connected devices and their respective communication protocols.

8.1.2. Limitations

During our research, we faced a number of restrictions and impediments which translated into the following limitations:

- **Traffic analysis:** We were not in control of the traffic captured in the HDO for the study presented in Chapter 2. We could not decide the location of the network appliances in charge of packet collection, nor could we choose the duration of capture. This lack of control has two main consequences which may have limited the extent of our analysis. First, since the location of the appliances has an impact on the traffic they can collect, we might have missed certain devices and protocols. Second, due to the different traffic capture durations, we may also have missed certain communication flows.
- **Device classification:** Similarly to the traffic analysis above, we were not in control of the data collected for the devices used in Chapter 3. The data was gathered by using various tools which, depending on their location in the network, could capture certain attributes of devices. Some devices in our dataset had no values for certain attributes we used in our classification method. As a result, it is challenging for our algorithm to classify these devices, and they may end up being considered as noise points (we elaborate on this point in Future work below).
- **Vulnerability research:** The scope of our vulnerability research performed on medical devices in Chapter 4 was focused on the networking part of the systems. A more thorough security assessment could include the evaluation of other attack vectors. For example, while looking at the printed circuit board of the Siemens blood analyser, we spotted the presence of a exposed JTAG interface which would be interesting to investigate further. Such interfaces can be used to manipulate runtime operations or even dump the firmware (which may contain vulnerabilities itself).

8.1.3. Future Work

Our research spawned a number of ideas which could contribute to the improvement of the state of security monitoring in HDO:

- **Device classification:** We proposed in this thesis a novel method for device classification based on semantic-driven feature selection. Future research could elaborate on our approach to increase the number of devices receiving a label suggestion while

maintaining the achieved consistency and granularity rates. In particular, more work is required to investigate the feasibility of reducing the amount of features used. As mentioned above, since certain devices have a limited number of feature values, the goal is to identify a subset of features that could still achieved similar (or even better) results. Additional research could also be done on the dissimilarity functions, especially regarding the investigation of different strategies to compute dissimilarity. While we treated all features as equally meaningful, one can ask whether including weights in the metrics could yield better results.

- **Privacy-preserving data sharing:** Our approach works thank to a substantial amount of data being shared by various organisations. Future improvements could focus on the analysis of the applicability of our method in environments where privacy-preserving data sharing technologies are used.
- **HDO-specific dataset of intrusion detection:** The specificities of HDO call for NIDS that are developed to address the uniqueness of their networks and their threats. The design and testing of such NIDS require relevant datasets. It has been shown [106] that datasets publicly available are not reflecting the current nature of the threat landscape (e.g., lack of real-network attacks). Therefore, additional research efforts should be invested in creating HDO-relevant datasets.

8.2. Conclusion of Part 2

In the second part of this thesis, we focused our study on the security of automotive CAN bus and intrusion detection systems. The summary of contributions is listed below, as well as the limitations of our research and future work.

8.2.1. Summary of Contributions

Modern cars are computer networks on wheels, embedding new technologies offering many innovative applications. However, the increased connectivity introduces new security risks, and researchers have demonstrated safeness-critical attacks targeting the automotive CAN bus. Following these publications, a growing body of research has been proposing different Network Intrusion Detection Systems (NIDS) to detect such attacks. Yet the effectiveness of these solutions to protect users' safeness is unclear. For this reason, we focused our study on the analysis of CAN bus NIDS.

We started our research with an investigation of the state of automotive security, focusing specifically on CAN network. In fact, CAN being the protocol abused in the attacks demonstrated on cars, we first needed to have a clear understanding of this protocol and its weaknesses before moving on to the application of NIDS. This led us to the fourth research question:

RQ 2.1: *What are the capabilities of an adversary using in-vehicle network attacks against a vehicle?*

Having a solid understanding of the threats to the CAN bus and adversary's capabilities is a prerequisite to investigate the design and effectiveness of NIDS for that network. The automotive security field being relatively new, information is scattered throughout the literature and acquiring the necessary background knowledge required extensive research. We started in Chapter 5 to investigate automotive architectures and technologies to understand the specificities of modern vehicles, before consolidating state-of-the-art information about automotive CAN and its related attacks. The contributions of this chapter represent the necessary foundations required to the study of CAN bus NIDS:

- We laid out a concise overview of CAN bus, its functioning in the automotive context and its vulnerabilities.
- We identified the capabilities of an adversary in a CAN network and the attacks that can be launched on the in-vehicle network, impacting the safeness of the users.

Our work answered RQ 2.1 and highlighted how an attacker can abuse the specifications of CAN to influence the behaviour of microcontrollers in an automotive network, and ultimately influence vehicle's operations. In order to protect cars from such attacks, researchers have been proposing a number of NIDS specifically designed for the CAN bus. This situation led us to consider the fifth research question:

RQ 2.2: *What kind of coverage do NIDS provide against attacks on CAN networks?*

NIDS are well established in other domains such as traditional IT, and their non-invasive nature could be well-fitted for the automotive domain. However, it is unclear how their concepts and principles could be applied to the CAN bus. After understanding how CAN bus attacks work in the previous chapter, we analysed the functioning of CAN bus NIDS and determine the coverage they provide against attacks. In Chapter 6 we surveyed the NIDS proposed in the literature for the CAN bus and we outlined their design principles upon which they rely to detect attacks. More specifically, the contribution of our work is twofold:

- We introduced a novel taxonomy for CAN bus NIDS, which enables their comparison at a theoretical level.
- We proposed the first organised inventory of NIDS published, highlighting the coverage they provide against CAN bus attacks.

This work allowed us to answer RQ 2.2. Moreover, it showed that their design principles are radically different from the NIDS in other fields. For this reason, we could not rely on existing taxonomies to categorise them, making it impossible to compare them. By creating a novel taxonomy tailor-made for these systems, our work provides the means to organise and compare them at a theoretical level. While having a clear picture of the NIDS coverage, we still needed a way to assess their practical performance. This led us to the sixth research question:

RQ 2.3: *How can we evaluate CAN bus NIDS on equal ground?*

The NIDS' capability to detect attacks on the CAN bus is unclear based on the papers alone. The evaluations provided by the authors are often disparate and limited. It is therefore impossible to assess the effectiveness of existing solutions. Conversely to NIDS in other fields, no benchmark or methodology is available to evaluate CAN bus NIDS. We developed in Chapter 7 a unified evaluation framework to test the performance of CAN bus NIDS on equal ground. We started by designing a testing methodology and creating diverse datasets. We then evaluated a selected number of NIDS with our framework. The two contributions of our work are the following:

- The evaluation framework we published can be used by the automotive security research community to test novel NIDS algorithms. Our work remediates to the lack of common methodology and available data, enabling the testing and comparison of NIDS in a unified manner as it is the case in other fields.
- In the light of our results, we proposed a number of considerations which could enable effective NIDS algorithm design for the CAN bus. In fact, our work highlights the difficulty of performing intrusion detection on the CAN bus and the limitations of some of the proposed NIDS approaches. The lack of clear indicators of attack, combined with the challenging nature of automotive settings, call for the implementation of meaning-aware NIDS leveraging the specifications of vehicles.

Our work in this chapter answered RQ 2.3 by enabling CAN bus NIDS evaluation on equal ground. It also demonstrated the challenging setting of automotive CAN bus to perform intrusion detection. These challenges could be overcome by designing meaning-aware NIDS with the use of manufacturer's specifications.

8.2.2. Limitations

The main limitations regarding the second part of the thesis consist of the following:

- **Costs:** The barrier to enter automotive security and conduct research can be substantially high: it requires a car. Or actually, multiple cars ideally. Modern vehicles with interesting (security-wise) features are expensive, which may not fit research budgets. Additionally, certain tools which can facilitate research such as testers containing ECU-specific information are also expensive and sometimes sold only to garages. While we had two cars at hand, the extend of the tests we were allowed to perform was limited (see next point).
- **Lab environment:** Not having a proper lab setting (e.g., a car with heavy-duty lift) limited the depth of research. For example, this situation hindered the creation of our attack datasets. Ideally, the dataset would be generated by launching CAN bus attacks on our cars at speed while recording the traffic on the CAN bus. However, it was not possible for safety and legal reasons. Furthermore, we did not manage to find collaboration partners with whom we could conduct such experiments.
- **Diversity of automotive architecture:** As seen in Chapter 5, there is a broad diversity of architecture among the automotive domain. Capabilities of attackers highly depend on the architecture of the targeted vehicle, and would likely vary from one

car to another. The implication for NIDS testing is that extensive data from multiple vehicles is required to evaluate a detection method and assess how the results generalise.

8.2.3. Future Work

The following ideas could be used as starting points to pursue our work:

- **Meaning-aware NIDS:** As we argued in Chapter 7, only meaning-aware NIDS could provide effective and reliable detection capabilities for CAN. Building such NIDS requires manufacturers specifications. They provide information about CAN communications, for instance the legitimate CAN ID, and data transmission patterns. However, due to the large variety of automotive architectures, different NIDS have to be built for each car. Future work in this domain could focus on automated model building, where vehicle specifications can be converted automatically into system's model, enabling the creation effective NIDS for CAN bus at scale.
- **Datasets:** While the research suggested above could help with the design of NIDS, these systems would still require to be tested and evaluated in a unified manner. Future work could build on top of our framework by providing more datasets. Additionally, as new attacks may be discovered for the CAN bus, the framework could be extended to reflect more accurately the threat landscape.
- **Automotive protocols:** We looked in this thesis at the CAN bus and identified the capabilities of an attacker. As the automotive industry could be moving toward other protocols for critical ECU such as CAN-FD, it is uncertain whether the attacker capabilities would remain the same. Certain protocols such as automotive Ethernet are also getting traction, and would require dedicated research to evaluate the application of network monitoring capabilities on such protocols.

8.3. Concluding Remarks

8.3.1. Answering the Main Research Question

Our research on HDO and modern cars helps us to better understand SCE and their challenges with regards to the application of NSM. The results of these two studies lead to invaluable insights which, combined, enable us to answer the main research question we formulated at the beginning of this dissertation:

> **MRQ:** *To what extent can we perform network security monitoring in safeness-critical environments?*

By taking HDO and modern cars as examples of SCE, we identified gaps for the application of NSM in SCE and addressed some of them. Based on our results, we can conclude that, for NSM to be successfully applied, certain needs have to be fulfilled and challenges to be overcome in relation to the characteristics of SCE outlined in Chapter 1. (Recall that the characteristics of SCE are: C1 Diversity of devices; C2 Ecosystem of devices; C3 Span

across space and time; C4 Safeness-critical.) Fulfilling these requirements determines the extent of the application of NSM in SCE.

The three major requirements for the application of NSM in SCE can be framed as follow:

1. The application of NSM in SCE requires a thorough *environment investigation*, which goal is to obtain a clear picture of the infrastructure, devices and protocols being used. This task calls for the collection and the analysis of a sufficient amount of data such as network traffic and host information. The outcome of this analysis should provide visibility into the devices (e.g., their type, operating systems) and communications patterns (e.g., network protocols used, communication statistics). We showed in Chapter 2 an example of environment investigation where we looked at multiple instances of HDO.

   The visibility one can have into an SCE's network may be hindered by the limitations of current tools (e.g., device classification tools, protocol analysers). Assuming the use of tools initially designed for IT network, they are likely to be ill-suited given the broad diversity of devices found in a given SCE (C1). Additionally, the deployment of sensors (already an issue in IT and worsened in SCE by cloud computing) (C1, C2, C3) and the restriction of usable techniques (e.g., no active probing on network segments containing critical systems) may also restrict visibility. These challenges may be addressed by designing new tools granting a better visibility into SCE infrastructures. We demonstrated in Chapter 3 how to (partially) address this challenge by proposing a novel approach to device classification.

2. The second requirement to be met for the application of NSM in SCE is the understanding of protocols and devices, as well as their related threats. The environment investigation of SCE will likely shed light on a number of blind spots, some of them linked to the usage of non-standard and proprietary network protocols, as we observed in Chapter 2 (C1). These protocols may not be supported by current NSM tools. Additionally, some IoT and embedded devices in SCE interact with each other through the use of machine-to-machine communication or other non-human-readable format, as observed in Chapter 5 (C2). Consequently, one may not understand the communication patterns and data being transmitted, which could ultimately lead to the inability to detect abnormal behaviours and intrusions.

   One approach to overcoming this limitation involves conducting security analysis of protocols and devices. The goal is to identify how they work, and this knowledge can then be used to find potential flaws and vulnerabilities. Discovering weaknesses provides a greater awareness of the threat landscape, which enables the improvement of NSM tools for a given SCE. We showed in Chapter 4 an example of security analysis conducted on HDO-specific protocols and devices. Analysing protocols starts with the aim of understanding how they are used by devices to communicate, how the packets are formed, the data conveyed, etc. This can be done by finding the specifications (if available) or by reverse-engineering the communication flows. The knowledge acquired allows to then look for vulnerabilities and ways to exploit them. Analysing devices can be done by conducting a security assessment, where a variety of interfaces will be identified and tested for weaknesses. It may not be possi-

ble/allowed to perform security research on "live" environments in SCE due to their "safeness-criticality" (C4). One will need the ability to build labs in order to replicate (part of) the network as faithfully as possible, allowing of experimentation in a safe environment, as we demonstrated in Chapter 4 and 7.

The resulting discovery of flaws and vulnerabilities during protocol and device security analysis contributes to the improvement of NSM solutions such as NIDS in multiple ways. First, it enables the creation of attack signatures for NIDS, giving them the ability to detect instances of attacks leveraging the newly found weaknesses. Second, it also allows of the creation of attack datasets relevant to a specific SCE for the purpose of testing new security tools. We demonstrated in Chapter 4 and 7 how to create attack datasets to be used for SCE-specific NIDS.

3. Finally, the third need required for the application of NSM in SCE is the ability to evaluate NSM tools such as NIDS in a unified manner. As new tools will be designed to addressed the specificities of a given SCE, it is essential to have an evaluation framework guaranteeing uniformity of testing and reproducibility of results. Such frameworks shall include a methodology and datasets that are shared across the security research community, enabling the evaluation of novel NIDS algorithms on equal ground. It shall rely on clear and well-defined metrics, allowing of straightforward comparison between algorithms. Additionally, the framework can be improved and updated over time, as new vulnerabilities are discovered and new attack datasets created. We demonstrated in Chapter 7 how to create such a unified evaluation framework.

   Having realistic and SCE-relevant datasets in an evaluation framework is essential. In fact, testing NIDS with simulated datasets can raise uncertainty with regards to the applicability of the results in real settings. Moreover, datasets shall be relevant to a given SCE, meaning that the types of devices, network traffic and attacks in the datasets should match those that can be found in instances of that SCE (C1). The datasets shall be a combination of both attack-free datasets to be used for the training of NIDS, and attack datasets for testing. Building a lab is essential for the creation of datasets. As previously outlined, it may not be feasible to create datasets from "live" environments (C4).

Having these requirements fulfilled represents a starting point from which effective network monitoring capabilities can be designed to better protect the environment and its users from cyber threats. How well these requirements are fulfilled largely determines the extend to which network security monitoring is possible and thus answers our research question.

8.3.2. Lessons learnt

During the past five years spent conducting security research in the healthcare and automotive domains, we encountered various obstacles to overcome. These challenges taught us a great deal, and this hard-earned experience can be useful for future research in SCE security. Generally speaking, the main learnings we can draw from both domains we in-

vestigated are the following:

- As relatively new fields of research, technical information about them can be scattered or lacking. It takes substantial research effort and time to start investigating these SCE from scratch, which should be considered during the planning phase of a project.
- Having stakeholder participation in the process is important to clarify misunderstandings and to validate assumptions. Information found online can be misleading or even inaccurate. Therefore, collaboration with domain experts helps greatly to lay the right foundations for the research.
- Due to the unique nature of the environments, it may require the acquisition of specific tools (software and hardware) in order to conduct research (e.g., for data collection). Such equipment can come at a significant cost, and its operation may also require knowledge and documentation not always easily accessible. A substantial budget should be allocated early on to cover tooling acquisition and eventual training.

8.3.3. Final Words

The digitalisation of our society and the deployment of connected devices have profoundly changed how we live. The current trends of technology adoption can be expected to accelerate. The size of computerised systems will continue shrinking, while their performances will keep on increasing, in accordance to Moore's law. We will continue to witness the relentless implementation of IoT devices and cyber-physical systems in various domains of our lives, from the cars that transport us, to the hospitals that treat us. While our technologies are becoming smarter, so are our adversaries. Our economy is growing dependent on systems which represent prime targets for cyber criminals. Protecting these devices and their infrastructures is critical to safeguard our society.

We hope to have shown in this thesis that, as technology evolves, so must our cyber defences. The old paradigms upon which our security solutions were built must be re-evaluated. New ones need to be developed to match both the characteristics of modern environments and the challenges of the current threat landscape. This thesis lays the foundations for future research on NSM applied to SCE. Taking as examples HDO and modern cars, we demonstrated some of the steps to be taken to enhance NSM capabilities. Our work is only the starting point, and we hope to inspire new generations of researchers to build a safer and more secure society.

References

[1] Wireless Infusion Pumps Could Increase Cybersecurity Vulnerability.

[2] **Abbott-McCune, S., and Shay, L. A.** Intrusion Prevention System of Automotive Network CAN Bus. In *IEEE International Carnahan Conference on Security Technology (ICCST)* (2016), IEEE, pp. 1–8.

[3] **Abdelnur, H. J., State, R., and Festor, O.** Advanced Network Fingerprinting. In *Recent Advances in Intrusion Detection (RAID)* (Berlin, Heidelberg, 2008), R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., Springer Berlin Heidelberg, pp. 372–389.

[4] **Ablon, L., Libicki, M. C., and Golay, A. A.** *Markets for Cybercrime Tools and Stolen Data: Hackers' Bazaar*. Rand Corporation, 2014.

[5] **Achleitner, S., La Porta, T., McDaniel, P., Sugrim, S., Krishnamurthy, S. V., and Chadha, R.** Cyber Deception: Virtual Networks to Defend Insider Reconnaissance. In *ACM 8th CCS International Workshop on Managing Insider Security Threats* (2016), pp. 57–68.

[6] **ADAC Vehicle Technology**. Motorcycles and Cars Tested by German Automobile Club ADAC. Tech. rep., ADAC Vehicle Technology, 2019.

[7] **Adams, P., Al-Shahery, O., and Chmiel, J.** 2020 Cyber Threatscape Report. Tech. rep., Accenture Security, 2020.

[8] **Albataineh, A., and Alsmadi, I.** IoT and the Risk of Internet Exposure: Risk Assessment using Shodan Queries. In *IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)* (2019), IEEE, pp. 1–5.

[9] **Allodi, L., Corradin, M., and Massacci, F.** Then and Now: On the Maturity of the Cybercrime Markets the Lesson that Black-Hat Marketeers Learned. *IEEE Transactions on Emerging Topics in Computing 4*, 1 (2015), 35–46.

[10] **Alsubaei, F., Abuhussein, A., and Shiva, S.** Security and Privacy in the Internet of Medical Things: Taxonomy and Risk Assessment. In *IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)* (2017), IEEE, pp. 112–120.

[11] **AlTawy, R., and Youssef, A. M.** Security Tradeoffs in Cyber Physical Systems: A Case Study Survey on Implantable Medical Devices. *IEEE Access 4* (2016), 959–979.

[12] **Ammar, N., Noirie, L., and Tixeuil, S.** Autonomous IoT Device Identification Prototype. In *Network Traffic Measurement and Analysis Conference (TMA)* (2019), IEEE, pp. 195–196.

[13] **Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., Moore, T., and Savage, S.** Measuring the Cost of Cybercrime. In *The Economics of Information Security and Privacy*. Springer, 2013, pp. 265–300.

[14] **Aradau, C.** Security that Matters: Critical Infrastructure and Objects of Protection. *Security Dialogue 41*, 5 (2010), 491–514.

[15] **Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M., and Perona, I.** An Extensive Comparative Study of Cluster Validity Indices. *Pattern Recognition 46*, 1 (2013), 243–256.

[16] **Ashibani, Y., and Mahmoud, Q. H.** Cyber Physical Systems Security: Analysis, Challenges and Solutions. *Computers & Security 68* (2017), 81–97.

[17] **Atlam, H. F., Alenezi, A., Alassafi, M. O., Alshdadi, A. A., and Wills, G. B.** Security, Cybercrime and Digital Forensics for IoT. In *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*. Springer, 2020, pp. 551–577.

[18] **Bai, L., Yao, L., Kanhere, S. S., Wang, X., and Yang, Z.** Automatic Device Classification from Network Traffic Streams of Internet of Things. In *IEEE 43rd Conference on Local Computer Networks (LCN)* (2018), IEEE, pp. 1–9.

[19] **Bajpai, P., Sood, A. K., and Enbody, R. J.** The Art of Mapping IoT Devices in Networks. *Network Security 2018*, 4 (2018), 8–15.

[20] **Bassett, G., Hylender, C. D., Langlois, P., Pinto, A., and Widup, S.** 2021 Data Breach Investigation Report. Tech. rep., Verizon, 2021.

[21] **BBC**. Cyber-Attack on Irish Health Service 'Catastrophic'. `https://bbc.in/3AQbAVs`, 2021. Accessed: 2022-02-02.

[22] **Bejtlich, R.** *The Practice of Network Security Monitoring: Understanding Incident Detection and Response*. No Starch Press, 2013.

[23] **Biba, E.** How Connected Car Tech is Eroding Personal Privacy. `https://bit.ly/3oeAOrB`, 2016. Accessed: 2022-02-02.

[24] **Bodungen, C., Singer, B., Shbeeb, A., Wilhoit, K., and Hilt, S.** *Hacking Exposed Industrial Control Systems: ICS and SCADA Security Secrets & Solutions*. McGraw Hill Professional, 2016.

[25] **Broster, I., and Burns, A.** An Analysable Bus-Guardian for Event-Triggered Communication. In *24th IEEE Real-Time Systems Symposium (RTSS)* (2003), IEEE, pp. 410–419.

[26] **Campello, R. J., Moulavi, D., and Sander, J.** Density-Based Clustering Based on Hierarchical Density Estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2013), Springer, pp. 160–172.

[27] **Campello, R. J., Moulavi, D., Zimek, A., and Sander, J.** Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 1–51.

[28] **Caselli, M., Zambon, E., Amann, J., Sommer, R., and Kargl, F.** Specification Mining for Intrusion Detection in Networked Control Systems. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX, 2016), USENIX Association, pp. 791–806.

[29] **Chaffin, G.** Street Justice: Huge Traffic Ticket Discounts and Theft-Inducing Car-Sharing Apps. https://bit.ly/3ASbZqs, 2020. Accessed: 2022-02-02.

[30] **Chantzis, F., Stais, I., Calderon, P., Deirmentzoglou, E., and Woods, B.** *Practical IoT Hacking.* No Starch Press, 2021.

[31] **Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T.** Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *20th USENIX Security Symposium (USENIX Security 11)* (San Francisco, CA, 2011), USENIX Association.

[32] **Chiang, M., and Zhang, T.** Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of things journal 3*, 6 (2016), 854–864.

[33] **Cho, K.-T., and Shin, K. G.** Error Handling of In-vehicle Networks Makes Them Vulnerable. In *ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 1044–1055.

[34] **Cho, K.-T., and Shin, K. G.** Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX, 2016), USENIX Association, pp. 911–927.

[35] **Cho, K.-T., and Shin, K. G.** Viden: Attacker identification on in-vehicle networks. In *ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1109–1123.

[36] **Choi, W., Jo, H. J., Woo, S., Chun, J. Y., Park, J., and Lee, D. H.** Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks. *IEEE Transactions on Vehicular Technology 67*, 6 (2018), 4757–4770.

[37] **Choi, W., Joo, K., Jo, H. J., Park, M. C., and Lee, D. H.** VoltageIDS : Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Transactions on Information Forensics and Security 13*, 8 (2018), 2114–2129.

[38] **Ciholas, P., Lennie, A., Sadigova, P., and Such, J.** The Security of Smart Buildings: a Systematic Literature Review. *arXiv preprint arXiv:1901.05837* (2019).

[39] **Claise, B., Trammell, B., and Aitken, P.** Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, Internet Engineering Task Force (IETF), September 2013.

[40] **Clark, J.** Suppliers to the new Audi A4. https://bit.ly/3LWLl43, 2016. Accessed: 2022-04-12.

[41] **Colajanni, M., Dal Zotto, L., Marchetti, M., and Messori, M.** The Problem of NIDS Evasion in Mobile Networks. In *4th IFIP International Conference on New Technologies, Mobility and Security* (2011), IEEE, pp. 1–6.

[42] **Coleman, D.** Automotive Communication Networks, Part II CAN Bus. https://bit.ly/3GkZHbm. Accessed: 2022-02-02.

[43] **Contemporary Controls**. CAN Tutorial. https://bit.ly/3IWBD0a. Accessed: 2022-02-02.

[44] **Corrigan, S.** Introduction to the Controller Area Network (CAN). *Application Report SLOA101* (2002), 1–17.

[45] **CSS Electronics**. CAN Bus Explained - A Simple Intro. https://bit.ly/3IYVH20, 2021. Accessed: 2022-02-02.

[46] **CSS Electronics**. LIN Bus Explained - A Simple Intro. https://bit.ly/3Hmn5GM, 2021. Accessed: 2022-02-02.

[47] **Cui, A., and Stolfo, S. J.** A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan. In *26th Annual Computer Security Applications Conference* (New York, NY, USA, 2010), ACSAC '10, Association for Computing Machinery, p. 97–106.

[48] **Dajsuren, Y., and den Brand, M. v.** *Automotive Software Engineering: Past, Present, and Future*. Springer International Publishing, Cham, 2019, pp. 3–8.

[49] **Dajsuren, Y., and van den Brand, M.** *Automotive Systems and Software Engineering*. Springer, 2019.

[50] **Dameff, C., Bland, M., Levchenko, K., and Tully, J.** Pestilential Protocol: How Unsecure HL7 Messages Threaten Patient Lives. Black Hat US 2018 - https://youtu.be/66x3vfac8rA, 2018.

[51] **Davis, J.** Nation-State Hacking Campaigns Targeting COVID-19 Research Firms. https://bit.ly/32VcMug, 2020. Accessed: 2022-02-02.

[52] **Davis, J.** Costs from Ransomware Attack Against Ireland Health System Reach $600M. https://bit.ly/3Hn7lTH, 2021. Accessed: 2022-02-02.

[53] **Debar, H., Dacier, M., and Wespi, A.** A Revised Taxonomy for Intrusion-Detection Systems. In *Annales des Télécommunications* (2000), vol. 55, Springer, pp. 361–378.

[54] **Deloitte Insights**. Making Data Risk a Top Priority. https://bit.ly/3IYqcVD, 2018. Accessed: 2022-02-02.

[55] **Denning, D.** An Intrusion-Detection Model. *IEEE Transactions on Software Engineering SE-13*, 2 (1987), 222–232.

[56] **Di Natale, M., Zeng, H., Giusto, P., and Ghosal, A.** *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Science & Business Media, 2012.

[57] **Donaldson, S. E., Siegel, S. G., Williams, C. K., and Aslam, A.** Enterprise Cybersecurity Capabilities. In *Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats*. Apress, Berkeley, CA, 2015, pp. 311–334.

[58] **Douglass, B. P.** Chapter 6 - Safety and Reliability Patterns. In *Design Patterns for Embedded Systems in C*. Newnes, Boston, 2011, pp. 357–423.

[59] **Duggal, A.** Understanding HL7 2.X Standards, Pen Testing, and Defending HL7 2.X Messages. Black Hat US 2016 - `https://youtu.be/MR7cH44fjrc`, 2016.

[60] **Dunka, Louis J. et al.** POCT01-A2 - Point-of-Care Connectivity, 2nd Edition. Standard, Clinical and Laboratory Standards Institute (CLSI), Wayne, PA, 2006.

[61] **Dupont, G., Den Hartog, J., Etalle, S., and Lekidis, A.** Evaluation Framework for Network Intrusion Detection Systems for In-Vehicle CAN. In *IEEE International Conference on Connected Vehicles and Expo (ICCVE)* (2019), pp. 1–6.

[62] **Dupont, G., den Hartog, J., Etalle, S., and Lekidis, A.** A survey of network intrusion detection systems for controller area network. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)* (2019), pp. 1–6.

[63] **Dupont, G., dos Santos, D. R., Costante, E., den Hartog, J., and Etalle, S.** A Matter of Life and Death: Analyzing the Security of Healthcare Networks. In *ICT Systems Security and Privacy Protection* (Cham, 2020), M. Hölbl, K. Rannenberg, and T. Welzer, Eds., Springer International Publishing, pp. 355–369.

[64] **Dupont, G., Leite da Silva, C., dos Santos, D. R., Costante, E., den Hartog, J., and Etalle, S.** Similarity-based Clustering for IoT Device Classification. In *IEEE International Conference on Omni-layer Intelligent systems (COINS)* (2021).

[65] **El-Maghraby, R. T., Elazim, N. M. A., and Bahaa-Eldin, A. M.** A Survey on Deep Packet Inspection. In *12th International Conference on Computer Engineering and Systems (ICCES)* (Dec 2017), pp. 188–197.

[66] **Erman, J., Arlitt, M., and Mahanti, A.** Traffic Classification Using Clustering Algorithms. In *SIGCOMM Workshop on Mining Network Data* (New York, NY, USA, 2006), MineNet '06, Association for Computing Machinery, pp. 281––286.

[67] **Etalle, S.** From Intrusion Detection to Software Design. In *European Symposium on Research in Computer Security (ESORICS)* (2017), Springer, pp. 1–10.

[68] **Everett, C. E., and McCoy, D.** OCTANE (Open Car Testbed and Network Experiments): Bringing Cyber-Physical Security Research to Researchers and Students. In *6th Workshop on Cyber Security Experimentation and Test (CSET 13)* (Washington, D.C., Aug. 2013), USENIX Association.

[69] **Fauri, D., dos Santos, D. R., Costante, E., den Hartog, J., Etalle, S., and Tonetta, S.** From System Specification to Anomaly Detection (and back). In *Workshop on Cyber-Physical Systems Security and PrivaCy, Dallas, TX, USA, November 3, 2017* (2017), pp. 13–24.

[70] **Fauri, D., Kapsalakis, M., dos Santos, D., Costante, E., Hartog, J., and Etalle, S.** Leveraging Semantics for Actionable Intrusion Detection in Building Automation Systems. In *International Conference on Critical Information Infrastructures Security (CRITIS)* (2018), pp. 113–125.

[71] **Feng, X., Li, Q., Wang, H., and Sun, L.** Acquisitional Rule-based Engine for Discovering Internet-of-Things Devices. In *27th USENIX Security Symposium (USENIX Security 18)* (Baltimore, MD, Aug. 2018), USENIX Association, pp. 327–341.

[72] **Fernández-Caramés, T. M., and Fraga-Lamas, P.** Teaching and Learning IoT Cybersecurity and Vulnerability Assessment with Shodan through Practical Use Cases. *Sensors 20*, 11 (2020), 3048.

[73] **Fiebig, T., Lichtblau, F., Streibelt, F., Krueger, T., Lexis, P., Bush, R., and Feldmann, A.** SoK: An Analysis of Protocol Design: Avoiding Traps for Implementation and Deployment. *arXiv preprint arXiv:1610.05531* (2016).

[74] **FireEye**. APT41 - Double Dragon. `https://bit.ly/38nj6bU`, 2019. Accessed: 2022-02-02.

[75] **Florek, C.** Medical Device Security, Part 2: How to Give Medical Devices a Security Checkup. `https://bit.ly/3s7Hlp4`, 2019. Accessed: 2022-02-02.

[76] **Foo Kune, D., Venkatasubramanian, K., Vasserman, E., Lee, I., and Kim, Y.** Toward a Safe Integrated Clinical Environment: A Communication Security Perspective. In *ACM Workshop on Medical Communication Systems (MedCOMM)* (2012), pp. 7–12.

[77] **Forescout Research Labs & JFrog Security Research**. INFRA:HALT : Jointly Discovering and Mitigating Large-Scale OT Vulnerabilities. Tech. rep., Forescout Research Labs, 2021.

[78] **Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A.** A Sense of Self for Unix Processes. In *IEEE Symposium on Security and Privacy (S&P)* (1996), IEEE, pp. 120–128.

[79] **Forshaw, J.** *Attacking Network Protocols*. No Starch Press, 2017.

[80] **Foster, I., Prudhomme, A., Koscher, K., and Savage, S.** Fast and Vulnerable: A Story of Telematic Failures. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)* (Washington, D.C., Aug. 2015), USENIX Association.

[81] **Fowler, G.** Alexa has been Eavesdropping on You this Whole Time. `https://wapo.st/3KXVRIJ`, 2019. Accessed: 2022-02-02.

[82] **Fowler, G.** What Does Your Car Know About You? `https://bit.ly/3uhLQjt`, 2019. Accessed: 2022-02-02.

[83] **Francois, J., Abdelnur, H., State, R., and Festor, O.** Machine Learning Techniques for Passive Network Inventory. *IEEE Transactions on Network and Service Management 7*, 4 (2010), 244–257.

[84] **Frassinelli, D., Park, S., and Nürnberger, S.** I Know Where You Parked Last Summer : Automated Reverse Engineering and Privacy Analysis of Modern Cars. In *IEEE Symposium on Security and Privacy (SP)* (2020), IEEE, pp. 1401–1415.

[85] **Fröschle, S., and Stühring, A.** Analyzing the Capabilities of the CAN Attacker. In *European Symposium on Research in Computer Security (ESORICS)* (Cham, 2017), S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., Springer International Publishing, pp. 464–482.

[86] **Furnell, S.** *Cybercrime: Vandalizing the Information Society*. Addison-Wesley London, 2002.

[87] **Gatlan, S.** FBI Warns of Cyber Attacks Targeting US Automotive Industry. `https://bit.ly/3HskuLl`, 2019. Accessed: 2022-02-02.

[88] **Gatouillat, A., Badr, Y., Massot, B., and Sejdic, E.** Internet of Medical Things: A Review of Recent Contributions Dealing With Cyber-Physical Systems in Medicine. *IEEE Internet of Things Journal 5*, 5 (2018), 3810–3822.

[89] **Gebhart, G.** Spying on Students: School-Issued Devices and Student Privacy. `https://bit.ly/3geT6Vj`, 2017. Accessed: 2022-02-02.

[90] **Geers, K.** Pandemonium: Nation States, National Security, and the Internet. *NATO CCDCOE* (2014).

[91] **Gheorghe, A.** How to Hack a Home through the Internet of Things. `https://bit.ly/3scWyVL`, 2016. Accessed: 2022-02-02.

[92] **Gmiden, M., Gmiden, M. H., and Trabelsi, H.** An Intrusion Detection Method for Securing In-Vehicle CAN bus. In *17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)* (2016), IEEE, pp. 176–180.

[93] **Goodin, D.** Patient Dies After Ransomware Attack Reroutes her to Remote Hospital. `https://bit.ly/349yxah`, 2020. Accessed: 2022-02-02.

[94] **Grimm, T., Lettnin, D., and Hübner, M.** A Survey on Formal Verification Techniques for Safety-Critical Systems-on-Chip. *Electronics 7*, 6 (2018), 81.

[95] **Groza, B., and Murvay, P.-S.** Efficient Intrusion Detection With Bloom Filtering in Controller Area Networks. *IEEE Transactions on Information Forensics and Security 14*, 4 (2019), 1037–1051.

[96] **Guiochet, J., Machin, M., and Waeselynck, H.** Safety-Critical Advanced Robots: A Survey. *Robotics and Autonomous Systems 94* (2017), 43–52.

[97] **Halkidi, M., Batistakis, Y., and Vazirgiannis, M.** On Clustering Validation Techniques. *Journal of Intelligent Information Systems 17*, 2–3 (dec 2001), 107–145.

[98] **Hämäläinen, J., Jauhiainen, S., and Kärkkäinen, T.** Comparison of Internal Clustering Validation Indices for Prototype-based Clustering. *Algorithms 10*, 3 (2017), 105.

[99] **Hanna, S., Rolles, R., Molina-Markham, A., Poosankam, P., Fu, K., and Song, D.** Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices. In *HealthSec* (2011).

[100] **Harrison, S.** Cyber Attack: When Will the Irish Health Service get a Resolution? `https://bbc.in/32VdeZw`, 2021. Accessed: 2022-02-02.

[101] **Haselhorst, D.** HL7 Data Interfaces in Medical Environments: Attacking and Defending the Achille's Heel of Healthcare. Tech. rep., SANS, 2017.

[102] **Heiland, D.** IoT Security Testing Methodology. `https://bit.ly/3umYVZ5`, 2017. Accessed: 2022-02-02.

[103] **Hennessy, B.** An Introduction to J1939 and DBC files. `https://bit.ly/3GrAVpL`, 2019. Accessed: 2022-02-02.

[104] **Herrmann, D., Fuchs, K.-P., and Federrath, H.** Fingerprinting techniques for target-oriented investigations in network forensics. *Sicherheit 2014–Sicherheit, Schutz und Zuverlässigkeit* (2014).

[105] **HIMSS**. 2019 HIMSS Cybersecurity Survey. `https://bit.ly/34v2q4u`, 2019. Accessed: 2022-02-02.

[106] **Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., and Bellekens, X.** A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *arXiv preprint arXiv:1806.03517* (2020).

[107] **Hoogendijk, F.** Reverse Engineering and Evaluation of Tesla Vehicle Logs. *29th Annual Congress of the European Association for Accident Research (EVU)* (2021).

[108] **Hoppe, T., Kiltz, S., and Dittmann, J.** Security threats to automotive CAN networks Practical examples and selected short-term countermeasures. *Reliability Engineering and System Safety 96*, 1 (2011), 11–25.

[109] **Hoppe, T. C., Kiltz, S., and Dittmann, J.** Applying Intrusion Detection to Automotive IT - Early Insights and Remaining Challenges. *Journal of Information Assurance and Security (JIAS) 4*, May (2009), 226–235.

[110] **Hsu, A., Tront, J., Raymond, D., Wang, G., and Butt, A.** Automatic IoT Device Classification using Traffic Behavioral Characteristics. In *SoutheastCon* (2019), IEEE, pp. 1–7.

[111] **Humayed, A., Lin, J., Li, F., and Luo, B.** Cyber-Physical Systems Security — A Survey. *IEEE Internet of Things Journal 4*, 6 (2017), 1802–1831.

[112] **IBM**. Closing the Data Privacy Gap: Protecting Sensitive Data in Non-Production Environments. Tech. rep., IBM, 2008.

[113] **Iehira, K., Inoue, H., and Ishida, K.** Spoofing Attack Using Bus-Off Attacks Against a Specific ECU of the CAN bus. In *15th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (2018), IEEE, pp. 1–4.

[114] **Interpol**. Cybercrime: Covid-19 impact. https://bit.ly/32TZuOF, 2020. Accessed: 2022-02-02.

[115] **ISE**. Securing Hospitals: A Research Study and Blueprint. https://bit.ly/3GkFp1v, 2016. Accessed: 2022-02-02.

[116] **Jaigirdar, F. T., Rudolph, C., and Bain, C.** Can I Trust the Data I See? A Physician's Concern on Medical Data in IoT Health Architectures. In *Australasian Computer Science Week Multiconference* (New York, NY, USA, 2019), ACSW 2019, Association for Computing Machinery.

[117] **Jiang, Y., Li, G., Feng, J., and Li, W.-S.** String Similarity Joins: An Experimental Evaluation. *Proc. VLDB Endowment 7*, 8 (Apr. 2014), 625–636.

[118] **Kang, M.-J., and Kang, J.-W.** A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security. In *IEEE 83rd Vehicular Technology Conference (VTC Spring)* (2016), pp. 1–5.

[119] **Karahasanovic, A., Kleberger, P., and Almgren, M.** Adapting Threat Modeling Methods for the Automotive Industry. In *15th ESCAR Conference* (2017), pp. 1–10.

[120] **Kiparoidze, M.** How Much Does Your Car Know About You — and Who Else Can Get Their Hands on Your Data? https://bit.ly/3olS4LD, 2021. Accessed: 2022-02-02.

[121] **Kneib, M., and Huth, C.** Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018), CCS '18, Association for Computing Machinery, p. 787–800.

[122] **Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J.** DDoS in the IoT: Mirai and other botnets. *Computer 50*, 7 (2017), 80–84.

[123] **Koppel, R., Smith, S. W., Blythe, J., and Kothari, V. H.** Workarounds to Computer Access in Healthcare Organizations: You Want My Password or a Dead Patient? In *Driving Quality in Informatics: Fulfilling the Promise*. IOS Press, 2015, pp. 215–220.

[124] **Kramer, D., Baker, M., Ransford, B., Molina-Markham, A., Stewart, Q., and Fu, K.** Security and Privacy Qualities of Medical Devices: An Analysis of FDA Postmarket Surveillance. *Public Library of Science ONE 7*, 7 (2012).

[125] **Krebs, B.** KrebsOnSecurity Hit With Record DDoS. https://bit.ly/3B0CqdK, 2016. Accessed: 2022-02-02.

[126] **Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A.** Density-based Clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1*, 3 (2011), 231–240.

[127] **Kumar, C.** New Dangers In the New World: Cyber Attacks in the Healthcare Industry. *Intersect: The Stanford Journal of Science, Technology, and Society 10*, 3 (2017).

[128] **Lambert, F.** The Big Tesla Hack: A hacker gained control over the entire fleet, but fortunately he's a good guy. `https://bit.ly/3IRTRQe`, 2020. Accessed: 2022-02-02.

[129] **Larson, U. E., Nilsson, D. K., and Jonsson, E.** An Approach to Specification-Based Attack Detection for In-Vehicle Networks. In *IEEE Intelligent Vehicles Symposium* (2008), IEEE, pp. 220–225.

[130] **Lee, H., Jeong, S. H., and Kim, H. K.** OTIDS : A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame. In *15th Annual Conference on Privacy, Security and Trust (PST)* (2017), IEEE, pp. 57–5709.

[131] **Lee, I., Sokolsky, O., Chen, S., Hatcliff, J., Jee, E., Kim, B., King, A., Mullen-Fortino, M., Park, S., Roederer, A., and Venkatasubramanian, K. K.** Challenges and Research Directions in Medical Cyber–Physical Systems. *Proceedings of the IEEE 100*, 1 (2012), 75–90.

[132] **Li, J.** Automobile Intrusion Detection. Hack In The Box Security Conference (HITB-SecConf) - `https://bit.ly/3L6ge6w`, 2016. Accessed: 2022-02-02.

[133] **Ling, C., and Feng, D.** An Algorithm for Detection of Malicious Messages on CAN Buses. In *National Conference on Information Technology and Computer Science* (2012/11), Atlantis Press, pp. 627–630.

[134] **Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J.** Understanding of Internal Clustering Validation Measures. In *IEEE International Conference on Data Mining* (2010), pp. 911–916.

[135] **Loukas, G.** A Cyber-Physical World. In *Cyber-Physical Attacks*. Butterworth-Heinemann, Boston, 2015, pp. 1–19.

[136] **Loukas, G.** Cyber-Physical Attacks on Implants and Vehicles. In *Cyber-Physical Attacks*. Butterworth-Heinemann, Boston, 2015, pp. 59–104.

[137] **Lueth, K. L.** State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time. `https://bit.ly/3ulLJDF`, 2020. Accessed: 2022-02-02.

[138] **Lukkien, J.** *Introduction to Cooperative Intelligent Transportation Systems*. Springer International Publishing, Cham, 2019, pp. 257–263.

[139] **Lunt, T. F., Tamaru, A., Gilham, F., Jagan, N. R., Jalali, C., and Neumann, P. G.** *A Real-time Intrusion-Detection Expert System (IDES)*. SRI International. Computer Science Laboratory, 1992.

[140] **Mansfield-Devine, S.** Ransomware: Taking Businesses Hostage. *Network Security 2016*, 10 (2016), 8–17.

[141] **Marchal, S., Miettinen, M., Nguyen, T. D., Sadeghi, A.-R., and Asokan, N.** AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE Journal on Selected Areas in Communications 37*, 6 (2019), 1402–1412.

[142] **Marchetti, M., and Stabili, D.** Anomaly Detection of CAN Bus Messages Through Analysis of ID Sequences. In *IEEE Intelligent Vehicles Symposium (IV)* (2017), pp. 1577–1583.

[143] **Marchetti, M., Stabili, D., Guido, A., and Colajanni, M.** Evaluation of Anomaly Detection for In-Vehicle Networks Through Information-Theoretic Algorithms. *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow, RTSI 2016* (2016), 1–6.

[144] **Markovitz, M., and Wool, A.** Field Classification, Modeling and Anomaly Detection in Unknown CAN Bus Networks. *Vehicular Communications 9* (2017), 43–52.

[145] **Martinelli, F., Mercaldo, F., Nardone, V., and Santone, A.** Car Hacking Identification Through Fuzzy Logic Algorithms. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (2017), IEEE, pp. 1–7.

[146] **Martins, G., Bhatia, S., Koutsoukos, X., Stouffer, K., Tang, C., and Candell, R.** Towards a Systematic Threat Modeling Approach for Cyber-Physical Systems. In *Resilience Week (RWS)* (2015), IEEE, pp. 1–6.

[147] **Maulik, U., and Bandyopadhyay, S.** Performance Evaluation of Some Clustering Algorithms and Validity Indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 12 (2002), 1650–1654.

[148] **Mazhelis, O., Luoma, E., and Warma, H.** Defining an Internet-of-Things Ecosystem. In *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2012, pp. 1–14.

[149] **McAdams, A.** Security and Risk Management: A Fundamental Business Issue. *Information Management 38*, 4 (2004), 36.

[150] **McInnes, L., Healy, J., and Astels, S.** HDBSCAN: Hierarchical Density Based Clustering. *The Journal of Open Source Software 2*, 11 (2017), 205.

[151] **McKee, D.** 80 to 0 in Under 5 Seconds: Falsifying a Medical Patient's Vitals. `https://bit.ly/2LJI8bB`, 2018. Accessed: 2022-02-02.

[152] **McNab, C.** *Network Security Assessment*. O'Reilly Media, 2016.

[153] **MedlinePlus**. Hemoglobin A1C (HbA1c) Test. `https://bit.ly/3up6FJV`, 2021. Accessed: 2022-02-02.

[154] **Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., and Elovici, Y.** ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis. In *Symposium on Applied Computing* (New York, NY, USA, 2017), SAC '17, Association for Computing Machinery, p. 506–509.

[155] **Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N. O., Guarnizo, J. D., and Elovici, Y.** Detection of Unauthorized IoT Devices Using Machine Learning Techniques. *arXiv preprint arXiv:1709.04647* (2017).

[156] **Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.-R., and Tarkoma, S.** IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (2017), IEEE, pp. 2177–2184.

[157] **Miller, C., and Valasek, C.** Adventures in Automotive Networks and Control Units. *Def Con 21*, 260-264 (2013), 15–31.

[158] **Miller, C., and Valasek, C.** A Survey of Remote Automotive Attack Surfaces. *Black Hat USA* (2014).

[159] **Miller, C., and Valasek, C.** Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA* (2015).

[160] **Miller, C., and Valasek, C.** CAN Message Injection - OG Dynamite Edition. `https://bit.ly/3gldqUX`, 2016. Accessed: 2022-02-02.

[161] **Mirsky, Y., Mahler, T., Shelef, I., and Elovici, Y.** CT-GAN: Malicious Tampering of 3D Medical Imagery using Deep Learning. In *28th USENIX Security Symposium (USENIX Security 19)* (Santa Clara, CA, Aug. 2019), USENIX Association, pp. 461–478.

[162] **Mitchell, R., and Chen, I.-R.** A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Computing Surveys (CSUR) 46*, 4 (2014), 1–29.

[163] **MITRE**. ATT&CK Tactic: Lateral Movement. `https://bit.ly/2qwuUaE`, 2019. Accessed: 2022-02-02.

[164] **Moore, M. R., Bridges, R. A., Combs, F. L., Starr, M. S., and Prowell, S. J.** Modeling Inter-Signal Arrival Times for Accurate Detection of CAN Bus Signal Injection Attacks. In *12th Annual Conference on Cyber and Information Security Research* (2017), pp. 1–4.

[165] **Moteff, J., and Parfomak, P.** *Critical Infrastructure and Key Assets: Definition and Identification*. Library of Congress Washington DC Congressional Research Service, 2004.

[166] **Mountain, Paul J. et al.** LIS02-A2 - Specification for Transferring Information Between Clinical Laboratory Instruments and Information Systems, 2nd Edition. Standard, Clinical and Laboratory Standards Institute (CLSI), Wayne, PA, 2004.

[167] **Mundt, T., and Wickboldt, P.** Security in Building Automation Systems - A First Analysis. In *International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)* (2016), pp. 1–8.

[168] **Murvay, P.-S., and Groza, B.** Source Identification Using Signal Characteristics in Controller Area Networks. *IEEE Signal Processing Letters 21*, 4 (2014), 395–399.

[169] **Müter, M., Groll, A., and Freiling, F. C.** A Structured Approach to Anomaly Detection for In-Vehicle Networks. In *Sixth International Conference on Information Assurance and Security* (2010), IEEE, pp. 92–98.

[170] **Müter, M., Groll, A., and Freiling, F. C.** Anomaly Detection for In-Vehicle Networks using a Sensor-based Approach. *Journal of Information Assurance and Security 6* (2011), 132–140.

[171] **Müter, M., and Asaj, N.** Entropy-based Anomaly Detection for In-Vehicle Networks. In *IEEE Intelligent Vehicles Symposium (IV)* (2011), pp. 1110–1115.

[172] **Narayanan, S. N., Mittal, S., and Joshi, A.** OBD SecureAlert : An Anomaly Detection System for Vehicles. In *IEEE International Conference on Smart Computing (SMARTCOMP)* (2016), IEEE, pp. 1–6.

[173] **National Instruments Corp.** Controller Area Network (CAN) Overview. `https://bit.ly/3rk1vwO`, 2020. Accessed: 2022-02-02.

[174] **National Instruments Corp.** FlexRay Automotive Communication Bus Overview. `https://bit.ly/3s8bowN`, 2021. Accessed: 2022-02-02.

[175] **Navarro, G.** A Guided Tour to Approximate String Matching. *ACM Computing Surveys 33*, 1 (mar 2001), 31–88.

[176] **Ngu, A. H., Gutierrez, M., Metsis, V., Nepal, S., and Sheng, Q. Z.** IoT Middleware: A Survey on Issues and Enabling Technologies. *IEEE Internet of Things Journal 4*, 1 (2016), 1–20.

[177] **Nie, S., Liu, L., and Du, Y.** Free-Fall : Hacking Tesla From Wireless To Can Bus. *Black Hat USA* (2017), 1–16.

[178] **O'Brien, G., Edwards, S., Littlefield, K., McNab, N., Wang, S., and Zheng, K.** Securing Wireless Infusion Pumps in Healthcare Delivery Organizations, 2018-08-20 2018.

[179] **Office of Inspector General**. Cybersecurity Management and Oversight at the Jet Propulsion Laboratory. `https://go.nasa.gov/3HonubK`, 2019. Accessed: 2022-02-02.

[180] **Ortiz, J., Crawford, C., and Le, F.** DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices. In *International Conference on Internet of Things Design and Implementation* (New York, NY, USA, 2019), IoTDI '19, Association for Computing Machinery, p. 106–117.

[181] **Osborne, C.** Colonial Pipeline attack: Everything you need to know. `https://zd.net/3gfJf1m`, 2021. Accessed: 2022-02-02.

[182] **Otsuka, S., and Ishigooka, T.** CAN Security : Cost-Effective Intrusion Detection for Real-Time Control Systems Overview of In-Vehicle Networks. *SAE Technical Paper* (2014).

[183] **Palanca, A., Evenchick, E., Maggi, F., and Zanero, S.** A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment* (Cham, 2017), M. Polychronakis and M. Meier, Eds., Springer International Publishing, pp. 185–206.

[184] **Parikh, B.** CAN Protocol - Understanding the Controller Area Network Protocol. https://bit.ly/3AUJZCM, 2021. Accessed: 2022-02-02.

[185] **Peacock, M., Johnstone, M. N., and Valli, C.** An Exploration of Some Security Issues Within the BACnet Protocol. In *Information Systems Security and Privacy* (Cham, 2018), P. Mori, S. Furnell, and O. Camp, Eds., Springer International Publishing, pp. 252–272.

[186] **Pêgo, P. R., and Nunes, L.** Automatic Discovery and Classifications of IoT Devices. In *12th Iberian Conference on Information Systems and Technologies (CISTI)* (2017), pp. 1–10.

[187] **Peng, Y., Lu, T., Liu, J., Gao, Y., Guo, X., and Xie, F.** Cyber-Physical System Risk Assessment. In *Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2013), IEEE, pp. 442–447.

[188] **Petrov, C.** 47 Stunning Internet of Things Statistics 2021. https://bit.ly/3rjb8Mo, 2022. Accessed: 2022-02-02.

[189] **Petrovic, S.** A Comparison between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters. In *11th Nordic Workshop of Secure IT Systems* (2006), pp. 53–64.

[190] **Philips**. Data Export Interface Programming Guide. https://philips.to/3eQQcWz, 2015. Accessed: 2022-02-02.

[191] **Ponemon Institute**. Cost of a Data Breach Report 2020. Tech. rep., Ponemon Institute, 2020.

[192] **Quigley, C., Charles, D., and McLaughlin, R.** Reverse engineering of CAN communication. *CAN Newsletter magazine* (2018).

[193] **Regalado, D.** Inside the Alaris Infusion Pump, not too much medicine, plz. DEF CON 25 IoT Village - https://youtu.be/w4sChnS4DrI, 2017.

[194] **Rieke, R., Seidemann, M., Talla, E. K., Zelle, D., and Seeger, B.** Behavior Analysis for Safety and Security in Automotive Systems. *25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* (2017), 381–385.

[195] **Rios, B.** Infusion Pump Teardown. S4x16 - https://youtu.be/pq9sCaoBVOw, 2016.

[196] **Roberts, P.** Let's Get Cyberphysical: Internet Attack shuts off the Heat in Finland. https://bit.ly/33XQgeK, 2016. Accessed: 2022-02-02.

[197] **Rouf, I., Miller, R., Mustafa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., and Seskar, I.** Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *19th USENIX Security Symposium (USENIX Security 10)* (Washington, DC, Aug. 2010), USENIX Association.

[198] **Rushanan, M., Rubin, A. D., Kune, D. F., and Swanson, C. M.** SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks. In *IEEE Symposium on Security and Privacy* (2014), pp. 524–539.

[199] **Salem, M., Crowley, M., and Fischmeister, S.** Anomaly Detection Using Inter-Arrival Curves for Real-Time Systems. *Euromicro Conference on Real-Time Systems 2016-Augus* (2016), 97–106.

[200] **Sanders, C., and Smith, J.** *Applied Network Security Monitoring: Collection, Detection, and Analysis*. Elsevier, 2013.

[201] **Santos, M. R., Andrade, R. M., Gomes, D. G., and Callado, A. C.** An Efficient Approach for Device Identification and Traffic Classification in IoT Ecosystems. In *IEEE Symposium on Computers and Communications (ISCC)* (2018), IEEE, pp. 00304–00309.

[202] **Sauerwald, M.** CAN bus, Ethernet, or FPD-Link: Which is best for automotive communications? `https://bit.ly/3uoA1Iq`, 2014. Accessed: 2022-02-02.

[203] **Scarfone, K., and Mell, P.** Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology. *NIST Special Publication 800-94* (2007).

[204] **Schmid, M.** Automotive Bus Systems. *ATMEL - Automotive Applications* (2004).

[205] **Schroff, F., Kalenichenko, D., and Philbin, J.** FaceNet: A Unified Embedding for Face Recognition and Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2015), IEEE, p. 815–823.

[206] **Schäfer, P.** NAV Alliance to Develop Future In-Vehicle Networking Architecture. `https://bit.ly/3J0VzPu`, 2018. Accessed: 2022-02-02.

[207] **Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., and Zhou, S.** Specification-Based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *9th ACM Conference on Computer and Communications Security (CCS)* (New York, NY, USA, 2002), CCS '02, Association for Computing Machinery, p. 265–274.

[208] **Seri, B., Vishnepolsky, G., and Zusman, D.** Critical Vulnerabilities to Remotely Compromise VxWorks, the Most Popular RTOS. Tech. rep., Armis, 2019.

[209] **Sethi, Amit**. Spies Among Us: Tracking, IoT and the Truly Inside Threat. `https://bit.ly/3AU3lb9`, 2018. Accessed: 2022-02-02.

[210] **Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., and Mustaqim, M.** Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access 8* (2020), 23022–23040.

[211] **Sheefer, Y., Porticor, Holz, R., TU Munchen, and Saint-Andre, P.** Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS), 2015.

[212] **Shevchenko, N., Chick, T. A., O'Riordan, P., Scanlon, T. P., and Woody, C.** Threat Modeling: a Summary of Available Methods. Tech. rep., Carnegie Mellon University Software Engineering Institute Pittsburgh United-States, 2018.

[213] **Siemens Healthcare Diagnostics**. DCA Vantage Analyzer Host Computer Communications Link. `https://bit.ly/3HqAkWU`, 2011. Accessed: 2022-02-02.

[214] **Siemens Healthcare Diagnostics**. DCA Vantage Operator's Guide. `https://bit.ly/3Holjoo`, 2012. Accessed: 2022-02-02.

[215] **Sivanathan, A., Gharakheili, H. H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V.** Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing 18*, 8 (2019), 1745–1759.

[216] **Sivanathan, A., Gharakheili, H. H., and Sivaraman, V.** Can We Classify an IoT Device using TCP Port Scan? In *IEEE International Conference on Information and Automation for Sustainability (ICIAfS)* (2018), pp. 1–4.

[217] **Sivanathan, A., Gharakheili, H. H., and Sivaraman, V.** Detecting Behavioral Change of IoT Devices Using Clustering-Based Network Traffic Modeling. *IEEE Internet of Things Journal 7*, 8 (Aug 2020), 7295––7309.

[218] **Sivanathan, A., Sherratt, D., Gharakheili, H. H., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V.** Characterizing and Classifying IoT Traffic in Smart Cities and Campuses. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2017), pp. 559–564.

[219] **Slater, Ben**. How easy is it to steal your car? `https://bit.ly/3IWBMAz`, 2019. Accessed: 2022-02-02.

[220] **Smart, M., Malan, G. R., and Jahanian, F.** Defeating TCP/IP Stack Fingerprinting. In *9th USENIX Security Symposium (USENIX Security 00)* (Denver, CO, Aug. 2000), USENIX Association.

[221] **Song, H. M., Kim, H. R., and Kim, H. K.** Intrusion Detection System Based on the Analysis of Time Intervals of CAN Messages for In-Vehicle Network. In *International Conference on Information Networking (ICOIN)* (2016), IEEE, pp. 63–68.

[222] **SophosLabs**. Sophos 2021 Threat Report: Navigating Cybersecurity in an Uncertain World. Tech. rep., Sophos, 2020.

[223] **Stabili, D., Marchetti, M., and Colajanni, M.** Detecting Attacks to Internal Vehicle Networks through Hamming distance. In *AEIT International Annual Conference* (2017), pp. 1–6.

[224] **Studnia, I., Alata, E., Nicomette, V., Kaâniche, M., and Laarouchi, Y.** A Language-based Intrusion Detection Approach for Automotive Embedded Networks. *International Journal of Embedded Systems 10*, 1 (2018), 1–12.

[225] **Suárez, J. N., and Salcedo, A.** ID3 and k-means Based Methodology for Internet of Things Device Classification. In *International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)* (2017), pp. 129–133.

[226] **Symantec**. New Orangeworm Attack Group Targets the Healthcare Sector in the U.S., Europe, and Asia. `https://bit.ly/3AUs9iZ`, 2018. Accessed: 2022-02-02.

[227] **Symantec**. Whitefly: Espionage Group has Singapore in its Sights. `https://bit.ly/3onx9HT`, 2019. Accessed: 2022-02-02.

[228] **Szydlowski, C. P.** *CAN specification 2.0: Protocol and Implementations*. SAE Technical Paper, 1992.

[229] **Taleck, G.** Ambiguity Resolution via Passive OS Fingerprinting. In *Recent Advances in Intrusion Detection* (Berlin, Heidelberg, 2003), G. Vigna, C. Kruegel, and E. Jonsson, Eds., Springer Berlin Heidelberg, pp. 192–206.

[230] **Taylor, A., Japkowicz, N., and Leblanc, S.** Frequency-based Anomaly Detection for the Automotive CAN Bus. In *World Congress on Industrial Control Systems Security (WCICSS)* (2015), IEEE, pp. 45–49.

[231] **Taylor, A., Leblanc, S., and Japkowicz, N.** Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks. *IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2016), 130–139.

[232] **Taylor, C. R., Venkatasubramanian, K., and Shue, C. A.** Understanding the Security of Interoperable Medical Devices Using Attack Graphs. In *3rd International Conference on High Confidence Networked Systems* (New York, NY, USA, 2014), HiCoNS '14, Association for Computing Machinery, p. 31–40.

[233] **Thangavelu, V., Divakaran, D. M., Sairam, R., Bhunia, S. S., and Gurusamy, M.** DEFT: A Distributed IoT Fingerprinting Technique. *IEEE Internet of Things Journal 6*, 1 (2019), 940–952.

[234] **The Clemson University Vehicular Electronics Laboratory**. Automotive Data Communication Buses. `https://bit.ly/3ul1Lxq`. Accessed: 2022-02-02.

[235] **The IFOD**. Car Facts: The Life Cycle of Cars. `https://bit.ly/34t3Z2U`, 2019. Accessed: 2021-08-18.

[236] **Theissler, A.** Anomaly Detection in Recordings from In-Vehicle Networks. *Big Data Applications and Principles* (2014).

[237] **Tian, D., Li, Y., Wang, Y., Duan, X., Wang, C., Wang, W., Hui, R., and Guo, P.** An Intrusion Detection System Based on Machine Learning for CAN-Bus. In *Industrial Networks and Intelligent Systems* (Cham, 2018), Y. Chen and T. Q. Duong, Eds., Springer International Publishing, pp. 285–294.

[238] **Ujiie, Y., Kishikawa, T., Haga, T., Matsushima, H., Wakabayashi, T., Tanabe, M., Kitamura, Y., and Anzai, J.** A Method for Disabling Malicious CAN Messages by Using a CMI-ECU. Tech. rep., SAE Technical Paper, 2016.

[239] **United Nations**. Digital Economy Report 2019 - Value Creation and capture: Implications for Developing Countries. `https://bit.ly/35zlLSH`, 2019. Accessed: 2022-02-02.

[240] **Uppuluri, P., and Sekar, R.** Experiences with Specification-Based Intrusion Detection. In *Recent Advances in Intrusion Detection* (Berlin, Heidelberg, 2001), W. Lee, L. Mé, and A. Wespi, Eds., Springer Berlin Heidelberg, pp. 172–189.

[241] **Upstream Security**. Upstream Security's Global Automotive Cyber security Report 2020. Tech. rep., Upstream Security, 2020.

[242] **Upstream Security**. Upstream Security's Global Automotive Cyber security Report 2021. Tech. rep., Upstream Security, 2021.

[243] **US DoH Cyber Security & Infrastructure Security Agency**. ICS-CERT Advisories. `https://bit.ly/369pLnZ`, 2019.

[244] **Vaccaro, H., and Liepins, G.** Detection of Anomalous Computer Session Activity. In *IEEE Symposium on Security and Privacy* (1989), pp. 280–289.

[245] **van Genderen, R. v. d. H.** Cyber Crime Investigation and the Protection of Personal Data and Privacy. Tech. rep., Discussion Paper, Economic Crime Division, Council of Europe, Strasbourg, France, 2008.

[246] **van Roermund, T.** *In-Vehicle Networks and Security*. Springer International Publishing, Cham, 2019, pp. 265–282.

[247] **Verizon**. Insider Threat Report: Out of sight should never be out of mind. Tech. rep., Verizon, 2019.

[248] **Veysset, F., Courtay, O., Heen, O., IR Team, et al.** New Tool and Technique for Remote Operating System Fingerprinting. *Intranode Software Technologies 4* (2002).

[249] **Vijayan, Jaikumar**. Target Attack Shows Danger of Remotely Accessible HVAC Systems. `https://bit.ly/3saJvEz`, 2014. Accessed: 2022-02-02.

[250] **Von Luxburg, U., Williamson, R. C., and Guyon, I.** Clustering: Science or Art? In *ICML Workshop on Unsupervised and Transfer Learning* (2012), JMLR Workshop and Conference Proceedings, pp. 65–79.

[251] **Wait, R.** Keyless Car Crime: How To Thwart The Thieves. `https://bit.ly/3ASdwg5`, 2021. Accessed: 2022-02-02.

[252] **Wall, D.** *Cybercrime: The Transformation of Crime in the Information Age*, vol. 4. Polity, 2007.

[253] **Wang, Q., Lu, Z., and Qu, G.** An Entropy Analysis Based Intrusion Detection System for Controller Area Network in Vehicles. In *31st IEEE International System-on-Chip Conference (SOCC)* (2018), pp. 90–95.

[254] **Waszecki, P., Mundhenk, P., Steinhorst, S., Lukasiewycz, M., Karri, R., and Chakraborty, S.** Automotive Electrical/Electronic Architecture Security via Distributed In-Vehicle Traffic Monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 36*, 11 (2017), 1790–1803.

[255] **Wolf, M.** Chapter 7 - System Design Techniques. In *Computers as Components*, M. Wolf, Ed., 4th ed., The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann, 2017, pp. 391–422.

[256] **Woo, S., Jo, H. J., and Lee, D. H.** A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems 16*, 2 (2015), 993–1006.

[257] **Wood, D., Apthorpe, N., and Feamster, N.** Cleartet Data Transmissions in Consumer IoT Medical Devices. In *Workshop on Internet of Things Security and Privacy* (New York, NY, USA, 2017), IoTS&P '17, Association for Computing Machinery, p. 7–12.

[258] **World Economic Forum**. The Global Risks Report 2019. `https://bit.ly/3uuF6Pg`, 2019. Accessed: 2022-02-02.

[259] **World Economic Forum**. A New Paradigm for Business of Data. Tech. rep., World Economic Forum, 2020.

[260] **Wu, W., Huang, Y., Kurachi, R., Zeng, G., Xie, G., Li, R., and Li, K.** Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access 6* (2018), 45233–45245.

[261] **Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S.** Distance Metric Learning, with Application to Clustering with Side-Information. In *15th International Conference on Neural Information Processing Systems* (Cambridge, MA, USA, 2002), NIPS'02, MIT Press, p. 521–528.

[262] **Xu, J., Venkatasubramanian, K. K., and Sfyrla, V.** A Methodology for Systematic Attack Trees Generation for Interoperable Medical Devices. In *Annual IEEE Systems Conference (SysCon)* (2016), pp. 1–7.

[263] **Xu, Q., Zheng, R., Saad, W., and Han, Z.** Device Fingerprinting in Wireless Networks: Challenges and Opportunities. *IEEE Communications Surveys Tutorials 18*, 1 (2016), 94–104.

[264] **Xu, Y., Tran, D., Tian, Y., and Alemzadeh, H.** Poster Abstract: Analysis of Cyber-Security Vulnerabilities of Interconnected Medical Devices. In *IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)* (2019), pp. 23–24.

[265] **Yang, K., Li, Q., and Sun, L.** Towards Automatic Fingerprinting of IoT Devices in the Cyberspace. *Computer Networks 148* (2019), 318–327.

[266] **Yar, M., and Steinmetz, K. F.** *Cybercrime and Society*. Sage, 2019.

[267] **Yeager, W., Leibham, M., Bartman, J., Delatorreleal, J., and Bivens, T.** Safety-Critical Systems. `https://bit.ly/3B6APTP`, 2012. Accessed: 2022-02-02.

[268] **Zhu, Y., Ting, K. M., and Carman, M. J.** Density-ratio Based Clustering for Discovering Clusters with Varying Densities. *Pattern Recognition 60* (2016), 983–997.

Curriculum Vitæ

Guillaume Dupont was born on the 23rd of February 1990 in Lagny sur Marne, France. Passionate about embedded system security, he focuses his work on vulnerability research on connected medical devices and intrusion detection systems for IoT.

Education

2017–2022	PhD in Computer Security Eindhoven University of Technology (NL) Department of Mathematics & Computer Science
2018–2019	General Management Plus Program (WHU Mini MBA Certificate) WHU Otto Beisheim School of Management, Düsseldorf (DE)
2013–2015	Master of Science degree in Computer Science University of Bordeaux (FR)
2011–2013	Bachelor of Science degree in Computer Science University of Bordeaux (FR)
2009–2011	Degree in Networks & Telecommunications University of Lorraine (FR)

Experience

2017–2022	Guest Researcher Forescout Technologies B.V., Eindhoven (NL)
2018–ongoing	Co-Founder Inspira Learning Centre, Pelwatte (LK)
2015–2017	IT Security Consultant Capgemini Netherland B.V., Utrecht (NL)

IPA Dissertations

Titles in the IPA Dissertation Series since 2019

S.M.J. de Putter. *Verification of Concurrent Systems in a Model-Driven Engineering Workflow*. Faculty of Mathematics and Computer Science, TU/e. 2019-01

S.M. Thaler. *Automation for Information Security using Machine Learning*. Faculty of Mathematics and Computer Science, TU/e. 2019-02

Ö. Babur. *Model Analytics and Management*. Faculty of Mathematics and Computer Science, TU/e. 2019-03

A. Afroozeh and A. Izmaylova. *Practical General Top-down Parsers*. Faculty of Science, UvA. 2019-04

S. Kisfaludi-Bak. *ETH-Tight Algorithms for Geometric Network Problems*. Faculty of Mathematics and Computer Science, TU/e. 2019-05

J. Moerman. *Nominal Techniques and Black Box Testing for Automata Learning*. Faculty of Science, Mathematics and Computer Science, RU. 2019-06

V. Bloemen. *Strong Connectivity and Shortest Paths for Checking Models*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2019-07

T.H.A. Castermans. *Algorithms for Visualization in Digital Humanities*. Faculty of Mathematics and Computer Science, TU/e. 2019-08

W.M. Sonke. *Algorithms for River Network Analysis*. Faculty of Mathematics and Computer Science, TU/e. 2019-09

J.J.G. Meijer. *Efficient Learning and Analysis of System Behavior*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2019-10

P.R. Griffioen. *A Unit-Aware Matrix Language and its Application in Control and Auditing*. Faculty of Science, UvA. 2019-11

A.A. Sawant. *The impact of API evolution on API consumers and how this can be affected by API producers and language designers*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2019-12

W.H.M. Oortwijn. *Deductive Techniques for Model-Based Concurrency Verification*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2019-13

M.A. Cano Grijalba. *Session-Based Concurrency: Between Operational and Declarative Views*. Faculty of Science and Engineering, RUG. 2020-01

T.C. Nägele. *CoHLA: Rapid Co-simulation Construction*. Faculty of Science, Mathematics and Computer Science, RU. 2020-02

R.A. van Rozen. *Languages of Games and Play: Automating Game Design & Enabling Live Programming*. Faculty of Science, UvA. 2020-03

B. Changizi. *Constraint-Based Analysis of Business Process Models*. Faculty of Mathematics and Natural Sciences, UL. 2020-04

N. Naus. *Assisting End Users in Workflow Systems*. Faculty of Science, UU. 2020-05

J.J.H.M. Wulms. *Stability of Geometric Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2020-06

T.S. Neele. *Reductions for Parity Games and Model Checking*. Faculty of Mathematics and Computer Science, TU/e. 2020-07

P. van den Bos. *Coverage and Games in Model-Based Testing*. Faculty of Science, RU. 2020-08

M.F.M. Sondag. *Algorithms for Coherent Rectangular Visualizations*. Faculty of Mathematics and Computer Science, TU/e. 2020-09

D.Frumin. *Concurrent Separation Logics for Safety, Refinement, and Security*. Faculty of Science, Mathematics and Computer Science, RU. 2021-01

A. Bentkamp. *Superposition for Higher-Order Logic*. Faculty of Sciences, Department of Computer Science, VUA. 2021-02

P. Derakhshanfar. *Carving Information Sources to Drive Search-based Crash Reproduction and Test Case Generation*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2021-03

K. Aslam. *Deriving Behavioral Specifications of Industrial Software Components*. Faculty of Mathematics and Computer Science, TU/e. 2021-04

W. Silva Torres. *Supporting Multi-Domain Model Management*. Faculty of Mathematics and Computer Science, TU/e. 2021-05

A. Fedotov. *Verification Techniques for xMAS*. Faculty of Mathematics and Computer Science, TU/e. 2022-01

M.O. Mahmoud. *GPU Enabled Automated Reasoning*. Faculty of Mathematics and Computer Science, TU/e. 2022-02

M. Safari. *Correct Optimized GPU Programs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2022-03

M. Verano Merino. *Engineering Language-Parametric End-User Programming Environments for DSLs*. Faculty of Mathematics and Computer Science, TU/e. 2022-04

G.F.C. Dupont. *Network Security Monitoring in Environments where Digital and Physical Safety are Critical*. Faculty of Mathematics and Computer Science, TU/e. 2022-05