Design, Implementation, and Evaluation of Secure Cyber-Physical and Wireless Systems

Submitted by

Daniele ANTONIOLI

Thesis Advisor

Dr. Nils Ole TIPPENHAUER (Asst. Prof. at SUTD until 07-2018)
Dr. Pawel SZALACHOWSKI

Information Systems Technology and Design (ISTD)

A thesis submitted to the Singapore University of Technology and Design in fulfillment of the requirement for the degree of Doctor of Philosophy

2019

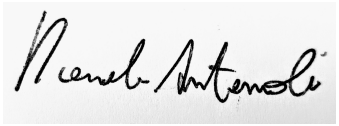
PhD Thesis Examination Committee

TEC Chair:	Prof. Zhou Jianying
Main Advisor:	Dr. Pawel Szalachowski
Co-advisor(s):	Dr. Nils Ole Tippenhauer
Internal TEC member 1:	Dr. Sudipta Chattopadhyay
Internal TEC member 2:	Dr. Alexander Binder

Declaration

I hereby confirm the following:

- I hereby confirm that the thesis work is original and has not been submitted to any other University or Institution for higher degree purposes.
- I hereby grant SUTD the permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created in accordance with Policy on Intellectual Property, clause 4.2.2.
- I have fulfilled all requirements as prescribed by the University and provided 1 copy of my thesis in PDF.
- I have attached all publications and award list related to the thesis (e.g. journal, conference report and patent).
- The thesis does contain patentable or confidential information.
- I certify that the thesis has been checked for plagiarism via ithenticate. The score 49% where 43% are matches against my own research papers and the remaining 6% is mostly related to matching bibliographic entries.



Name and signature: Daniele Antonioli

Date: 02-Aug-2019

Abstract

Information Systems Technology and Design (ISTD)

Doctor of Philosophy

Design, Implementation, and Evaluation of Secure Cyber-Physical and Wireless Systems

by Daniele ANTONIOLI

The first part of the dissertation presents our contributions in the area of *cyber-physical system security*. CPS are composed or sensors, actuators and controllers, and they are used to manage different processes such as industrial plants. Securing CPS is an open challenge and attacks such as Stuxnet and TRITON have reiterated the importance of securing CPS. In this work we focus on three (intertwined) problems to advance the security of CPS: technologies and processes, multi-disciplinary communities, and threat models and incentives. To address these problems we present the design, implementation, and evaluation of *MiniCPS*, an open-source toolkit for lightweight and real-time simulation of CPS. MiniCPS is built on top of Mininet, a network emulator based on Linux containers. We explore the usability of MiniCPS to develop defense mechanisms for CPS, with a particular emphasis on honeypots. A honeypot is a virtual or physical replica of a real system deployed to detect, mitigate and counteract cyber attacks. We leverage MiniCPS to design, implement and evaluate a novel high-interaction honeypot for industrial control systems. In addition, we test the effectiveness of MiniCPS as an educational and experimentation tool to help cybersecurity researchers and professionals. In particular, we used MiniCPS to design novel cybersecurity challenges based on real-time simulations of CPS. The challenges were proposed in a gamified security competition that we designed called SWaT Security Showdown (S3).

The second part of the dissertation presents our contributions in the area of *wireless systems security*. Wireless systems are used to transmit (sensitive) information and to manage and monitor systems remotely. In this work we focus on three problems to advance the security of wireless systems: effectiveness of deployed physical layer features as defense mechanisms, complexity and accessibility of wireless technologies, and security evaluations of wireless protocols. Firstly, we present a comparison between b/n/ac amendments of IEEE 802.11 (WLAN) where we theoretically estimate and empirically measure that recent physical layer features, such as MIMO and beamforming, could be used to mitigate passive eavesdropping attacks. These features are already present in commercial devices and they are complementary to the other (upper-layer) security mechanisms. Then, we present the first security analysis of Nearby Connections, an API for proximity-based services developed by Google. The API uses a combination of Bluetooth and Wi-Fi, and it is included in all Android devices since version 4.0 and all Android Things devices. Our analysis uncovers the proprietary (security) mechanisms of Nearby Connections and it is based on our reverse-engineering of its implementation. We demonstrate attacks where we maliciously manipulate Nearby Connections, and we extend the connection range to devices that are not nearby. Prior to publication we disclosed our findings to Google and we suggested them effective countermeasures. Finally, we describe how we found and exploited an architectural vulnerability of Bluetooth BR/EDR. We show how an attacker can downgrade the entropy any Bluetooth BR/EDR encryption key to 1 byte without being detected, and brute force the low entropy key in real time. We call our attack the Key Negotiation Of Bluetooth (KNOB) attack. We implement the attack and evaluate it on 21 Bluetooth vulnerable devices and we recommended to the Bluetooth SIG effective countermeasures.

Publications

- Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen. *The KNOB is broken: Exploiting Low Entropy in the Encryption Key Negotiation of Bluetooth BR/EDR*. In Proceedings of the USENIX Security Symposium 2019. https://francozappa.github.io/publication/knob/paper.pdf Repository: https://github.com/francozappa/knob

- Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen. *Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android*. In Proceedings of the Network and Distributed System Security Symposium (NDSS) 2019. https://francozappa.github.io/publication/rearby/paper.pdf Repository: https://github.com/francozappa/rearby

- Daniele Antonioli, Sandra Siby, Nils Ole Tippenhauer. *Practical Evaluation of Passive COTS Eavesdropping in 802.11b/n/ac WLAN*. In Proceedings of the Cryptology and Network Security (CANS) conference 2017. https://link.springer.com/chapter/10.1007/978-3-030-02641-7_19

- Daniele Antonioli, Hamid Reza Ghaeini, Sridhar Adepu, Martin Ochoa, and Nils Ole Tippenhauer. *Gamifying ICS Security Training and Research: Design, Implementation, and Results of S3*. In Proceedings of the Workshop on Cyber-Physical Systems Security and Privacy (co-located with CCS), November 2017 https://dl.acm.org/citation.cfm?id=3140253 Repository: https://github.com/scy-phy/minicps/tree/master/examples/s3-2017

- Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. *Towards High-Interaction Virtual ICS Honeypots-in-a-box*. In Proceedings of the Workshop on Cyber-Physical Systems Security and Privacy (co-located with CCS), pages 13–22. ACM, 2016. https://dl.acm.org/citation.cfm?id=2994493 **Research excellence award by ST Electronics FIRST workshop 2017**

- Daniele Antonioli and Nils Ole Tippenhauer. *MiniCPS: A toolkit for Security Research on CPS Networks*. In Proceedings of the Workshop on Cyber-Physical Systems-Security and/or Privacy (co-located with CCS), pages 91–100. ACM, 2015. https://dl.acm.org/citation.cfm?id=2808715 Repository: https://github.com/scy-phy/minicps

Acknowledgements

I'd like to thank my former supervisor, Nils Ole Tippenhauer, for his guidance and support during these years. I'm indebted to him for providing me the opportunity to work as a research assistant, PhD student, and teaching assistant at the Singapore University of Technology and Design (SUTD) and at the Helmholtz Center for Information Security (CISPA). Nils allowed me to independently explore exciting research directions that have gradually shaped my scientific interests. Nils did teach me how to do research, write and review papers, and assist a teacher. Nils helped me to design and implement elegant solutions to complex problems that resulted in significant research contributions. Nils introduced me Kasper.

I'd like to express my gratitude to Kasper Rasmussen for his guidance and support during the last part of my PhD. Kasper allowed me to work at the University of Oxford as a researcher and teaching assistant. Kasper did teach me how to become an independent researcher and provided me an excellent research topic, in line with my interests, that resulted in high impact contributions. Kasper sharpened my skills as a reviewer, writer and thinker. The role of Kasper in my PhD was pivotal.

I'd like to thank my current supervisor Pawel Szalachowski. Pawel was very helpful in the last part of my PhD because he allowed me to keep working with Nils and Kasper overseas without any issue. I would like to express my gratitude to all the members of my PhD thesis committee: Zhou Jianying, Pawel Szalachowski, Nils Ole Tippenhauer, Sudipta Chattopadhyay, and Alexander Binder. Their valuable comments greatly contributed to improving the quality of this dissertation.

I'd also like to acknowledge several professors and collaborators that I met in this journey. Srdjan Capkun introduced me Nils. Martin Ochoa introduced me to advanced topics in system security. Tony Quek let me revisit wireless communications from a different perspective. Nicholas, Pierre, Anand, John, Hamid, Sandra, Ahnaf, and Giuseppe have been good collaborators and friends.

I would like to thank Aurora (Goizane), my girlfriend, for her love, and patience during my PhD. I thank my family for their support. I'd like to thank Paolo (PVag) that was instrumental in my academic decisions, Alessandro (K=D) a loyal friend who is available during good and bad times, and Daniel, Ragav, Francesco, Giovanni, Giorgio, Elena, Stefano, and Adit that helped me with many things in Singapore. I'd like to extend my gratitude to the many friends that I found in Singapore, Oxford, and Saarbrücken.

Contents

List of Figures

List of Tables

To the ones I love.
Ai miei cari.

Part I

Cyber-physical systems security

Chapter 1

Introduction to Cyber-Physical Systems Security

1.1 Problem Statement

Cyber-Physical Systems (CPS) are composed of heterogeneous devices that control and monitor a physical process and talk to each other over a network. Control devices, such as Programmable Logic Controller (PLC) and Remote Terminal Unit (RTU), are programmed to periodically query sensors to read values from the physical process, and drive actuators to alter the state of the physical process. CPS are managing a variety of physical processes including critical infrastructures. Attacks on CPS can cause severe damages to the people and the environment. Securing CPS is still an open problem for various reasons. In this thesis we focus on three (intertwined) problems to *advance the security of cyber-physical system*:

1. *Technologies and processes*
2. *Multi-disciplinary communities*
3. *Threat models and incentives*

1.1.1 Technologies and processes

CPS technologies are complex to manage. A CPS composed of expensive, heterogeneous, and interconnected components that are designed for high availability. CPS's equipment has to reliably work for a long time span (e. g. years and sometimes decades) and it is subject to minimal upgrades. That said, introducing new equipment almost always result in inter-operability problems with already deployed legacy systems. Legacy systems have a slow phase out, they are difficult to patch and expensive to replace. The CPS market is dominated by the private sector, meaning that different functionalities might be implemented by different (competing) vendors. A typical outcome of this complexity is over-engineering, where a CPS is designed to provide more functionalities than needed resulting in a system that is more difficult to secure [182].

Physical processes are diverse and hard to model. Industrial control systems, self-driving cars, and building automation systems can be considered classes of CPS. Each class of CPS has subclasses. For example an industrial control system (ICS) controls and monitors an industrial application. If the underlying industrial application (physical process) is considered of vital importance we can label the ICS as a critical infrastructure (subclass of ICS). Water treatment, water distribution and power grid facilities are

examples of critical infrastructures. Each CPS subclass has its own set of unique challenges that have to be well understood first, and then secured. For example a water distribution system has generation/extraction, transmission/distribution and storage phases and each phase has unique associated threats [182].

Cyber-physical and IT systems are converging. Cyber-Physical Systems were used to be developed by and for trained engineers using custom (and often proprietary) devices, software, and protocols. In the last decade we have seen a convergence between the Information Technology (IT) space and the Operational Technology (OT) space. The IT space used to be confined to business and consumer electronics and the OT to industrial applications. This convergence resulted in the development of OT hardware and software that is able to communicate with the IT world. For example, control devices resemble general-purpose computers and industrial protocols have been adapted or redesigned to be compatible with Internet technologies [63]. Industrial Internet of Things (IIoT) and Industry 4.0 are the perfect incarnation of this new trend where industrial devices such as PLC, RTU, Supervisory Control and Data Acquisition (SCADA), Human-machine interface (HMI), sensors and actuators are reachable (and attackable) over the Internet.

1.1.2 Multi-disciplinary communities

CPS communities are diverse. The diversity of Cyber-Physical System results in an huge number of professionals working on them such as engineers, researchers and policy makers from different technical disciplines. It is notoriously hard to put in the same room groups of people with different technical backgrounds and let them agree on something. That is why we have (even within the same CPS domain) different technical languages, regulating bodies, stakeholder interests, specification formats, programming methodologies, hardware architectures to mention a few. Hence, it is almost infeasible for a single professional to acquire all the skills to deal with a Cyber-Physical System. Delegation and shared liability are the typical solutions. This problem is amplified in a security context where even the mismanagement of a minimal detail might result in catastrophic consequences.

CPS communities are fragmented. Fragmentation creates silos in the research community, opposed to an holistic and inclusive research approach. We need to find a way to let CPS (security) experts talk more often and more constructively. In our experience most of the hard technical challenges are already solved by the relevant technical community. Unfortunately, the best solution does not always reach other technical communities for various reasons such as patents, marketing, miscommunication, and laziness. As a direct result of this problem we have seen frequent usage of sub-optimal solutions to (already solved) technical problems e. g. custom broken crypto, custom useless authentication, poor network segmentation.

CPS regulations are not effective. Effective security policies and standards are one step in the right direction, however the diversity and complexity of cyber-physical systems bite again. Typically safety and process regulations, such as IEC61508, are done vertically by sector e. g. electricity vs. water. Cyber-security regulations, such as NIST SP800, are horizontal (cross-sectoral) e. g. cryptography. A CPS defender has to "muster and master" multiple standards affecting the same system from different perspectives. These perspectives might even conflict with each other: one standard could say: "for

better performance please use X" but another one says: "for security reasons please do not use X". In our experience CPS regulations are used mostly for compliance, leading to a (checkbox-style) false impression of security. We believe that this is an ineffective way to address CPS security.

1.1.3 Threat models and incentives

CPS were not designed after a threat model. Threat (attacker) modeling is the key part of any security assessment. Saying that something is secure does not make any sense if it is not stated in a specific threat model. Without a threat model is it impossible to specify and claim any security guarantee. Unfortunately, CPS were not designed after a threat model but for safety. At this time of writing there are no standard threat models for CPS. What is said is that the sophistication of a CPS attack is related to the attacker knowledge of the target physical process. However the relation is neither quantified not directly measurable. Another approach to CPS threat modeling is to re-use IT security concepts such as the CIA model [1] and Dolev-Yao [50]. For example, the CIA model for CPS involves ignoring authentication and turning around the importance of its metrics. That is, availability first, then integrity and finally confidentiality, but it is still not clear if it is a good idea or not.

The attacker surface of a CPS is hard to confine, segment and model. Typically, we decompose the CPS attacker surface in two intersecting surfaces: cyber and physical. A cyber attack involves software and hardware vulnerabilities of the CPS equipment (including the network). A physical attack involves physical tampering of the CPS equipment (including sensors and actuators). A cyber-physical attack uses vulnerability in the cyber part to cause damages in the physical world. The Aurora Generator Test [195] is an example of cyber-physical attack dating back to 2007 where researchers from Idaho National Laboratories showed how to destroy a diesel generator's circuit breaker using a computer program. We can even think of a physical-cyber attack where tampering with the physical process triggers a vulnerability in the IT side. This rich attacker surface is problematic during threat modeling. As defenders we have to specify from whom we want to protect, however if we select a surface that is too narrow we might end up with a limited defense mechanism e. g. focus only on network monitoring. On the other hand, a too broad attacker surface might result in impractical (often theoretically secure) defenses. To complicate even further the situation CPS are subject to cascading attacks where the attack surface is not limited to the target CPS but it is extended to the system directly connected to the target CPS e. g. attacking electricity distribution might disrupt water treatment because of a power outage. Cascading attacks still not well understood and modeled because they are difficult (and dangerous) to reproduce in a controlled environment.

CPS attackers' incentives are expanding. As said, the services provided by cyber-physical systems are essential to our society e. g. distribution of water and electricity, and CPS and IT system technologies are converging. This fact has severe consequences on CPS security. Firstly, CPS are becoming appealing to any type of attacker, from the script-kiddie to state-sponsored villains. Secondly, the attacker (malicious actor) can assume any role: competitor, insider, supplier, user, consultant, and unknown party.

[1] The CIA model uses confidentiality, integrity, and availability as his main metrics.

Thirdly, the attack might be highly targeted to a single system or launched to a wide range of systems. These are hard problems to be solved by CPS security researchers. Recent high-impact CPS attacks confirm this trend, here we list five of them in a timeline:

- 2003: Ohio's nuclear facilities (Slammer worm) [144]
- 2007: Maroochy's water breach [163]
- 2010: Iran's nuclear facilities (Stuxnet) [59]
- 2015: Ukraine's power grid [32]
- 2017: Middle East's safety instrumented systems (TRISIS) [89]

1.2 Our Vision, Research Directions and Questions

In the previous section we explained what are the main problems related to securing cyber-physical systems. In this section we present our "vision" and the related research directions and questions to be answered to reach our goals.

We believe that *enabling realistic low-cost simulations of Cyber-Physical Systems is a stepping stone to help solving the problems presented in Section 1.1*. By *realistic* we mean a simulation environment that takes into account all the constituent part of a CPS, namely: the control devices, the physical process and the network. By *low-cost* we mean that the users of our simulations could run them in their machines, have free access to its source code, documentation, and user guides. We envision short and long term benefits for researcher and professionals deriving from the possibility to accurately simulate a CPS. Some examples: decreasing the costs and the risks associated with CPS security experiments, encouraging sharing of data, models and (reproducible) results, and allowing to fast-prototyping and testing of new (security) solutions. A shared and open simulation platform would also help cross-sectoral collaborations between (and not limited to) public and private sectors, and it will lower the entry barrier to newcomers.

We admit that realistically simulating the behaviours of interconnected control devices and the physical process is challenging. Nevertheless, we think that even providing the basic building blocks in the short-term is beneficial for the long-term vision. In other words, we believe that once a minimum viable product is available (e. g. the CPS simulator) then if many people will use it we can benefit from the knowledge of the inter-disciplinary CPS communities to improve it. We know that many simulators already exists as open-source project and (expensive) closed-source products. Our goal is to be inclusive and provide a playground where different tools and programming languages can coexist. As an analogy: we don't want to reinvent the wheel, we want to enable a scientist to use different types of already developed wheels and engines in different contexts.

In this thesis we focus our attention on industrial control system, a subset of the whole Cyber-Physical System domain. Our decision has two main reasons the first is practical and the second is ideological. Firstly, Singapore University of Technology and Design (our institution) has three state-of-the-art interconnected ICS testbeds (water distribution, water treatment, and power grid). This is a privilege for us because we are able to validate our progresses comparing simulated results with expected ones. Secondly, we believe that industrial control system is the most interesting subset of cyber-physical

systems because it includes critical physical processes, unique devices software and protocols from the OT space, and standard equipment from the IT space.

Given the aforementioned research directions, the thesis aims to answer the following research questions:

- How could we design an open and extensible simulation toolkit for reproducible CPS security research?
  - What is the best way to simulate control devices, physical processes, networks and their interactions?
  - Is it necessary to develop new simulation and emulation environments?
  - Is it possible to do it in real-time?
  - Can we simulate multiple systems at the same time?
- Could we use our simulations to develop novel cyber-physical defenses?
  - Shall we focus on prevention or detection?
  - Could we take advantage of the low-cost and real-time aspects of our simulations?
  - Does it make sense to improve known defense mechanisms for the OT context such as Software-Defined Networking (SDN) and honeypots?
- Could we use our simulations for cyber-physical education and training purposes?
  - What is the best way to raise awareness and educate people about CPS (security)?
  - Shall we focus on applied or theoretical security?
  - Shall we design a targeted competition? If yes, who are the competitors? How can we effectively judge them?
- Could we use our simulations to develop novel cyber-physical attacks?
  - What are interesting goals enabled by the current CPS security landscape
  - Could we take advantage of the low-cost and real-time aspects of our simulations?
  - Does it make sense to improve known attack mechanisms for the OT context such as botnets?

1.3 Cyber-Physical Systems Security Contributions

The first part of the thesis makes contributions in the area of cyber-physical systems security, with results published in refereed venues. The contributions about wireless systems security are presented in Section 6.3.

- Chapter 2 presents *MiniCPS* a toolkit for Cyber-Physical System simulations build on top of mininet [6]. MiniCPS aims to provide an extensible, reproducible research environment for cyber-physical system. MiniCPS interfaces mininet (a network emulator) with with physical process and control device simulation and potentially actual CPS hardware. It provides a public API to program basic CPS interactions: `send`, `receive`, `get` and `set`. MiniCPS is open-source and a simulation can run on a single laptop with a Linux-based OS. Chapter 2 appeared in Proceedings of the ACM Workshop on Cyber-Physical Systems-Security and Privacy (CPS-SPC) 2015 [10].

- Chapter 3 presents the design and implementation of an honeypot for industrial control systems based on MiniCPS. Honeypots for ICS systems need to satisfy both traditional ICT requirements, such as cost and maintainability, and more specific ICS requirements, such as time and determinism. We propose the design of a virtual, high-interaction and server-based ICS honeypot to satisfy the requirements. The presented honeypot implementation is the first academic work targeting Ethernet/IP based ICS honeypots, the first ICS virtual honeypot that is high-interactive without the use of full virtualization technologies (such as a network of virtual machines), and the first ICS honeypot that can be managed with a Software-Defined Network (SDN) controller. To validate our idea we present an implementation of an honeypot mimicking a water treatment testbed. Chapter 3 appeared in Proceedings of the ACM Workshop on Cyber-Physical Systems-Security and Privacy (CPS-SPC) 2016 [7].

- Chapter 4 considers the challenges related to education and research about the security of industrial control systems (ICS). We propose to address those challenges through gamified security competitions targeted to researchers and professionals. Our gamification idea resulted in the design and implementation of the SWaT Security Showdown (S3), a Capture-The-Flag (CTF) event specifically targeted at ICS security. We developed ICS-specific challenges involving both theoretical and applied ICS security concepts. The participants had access to a real water treatment facility and they interacted with simulated components and ICS honeypots based on MiniCPS. We describe the phases of the S3, the scoring system. We present the results, statistics and lessons learnt from the S3 competition ran in 2016 and 2017 involving international teams from industry and academia. Chapter 4 appeared in Proceedings of the ACM Workshop on Cyber-Physical Systems-Security and Privacy (CPS-SPC) 2017 [13].

- In addition to the main contributions presented there are three research papers where the candidate is the main author or co-author that are related to this dissertation. In [8] we propose CPSBot, a framework to design and implement botnets to attack cyber-physical systems. In [33] we propose an authentication mechanisms for legacy (unsecure) industrial protocols based on lightweight cryptographic primitives, the paper appeared in Proceedings of the Conference on Applied Cryptography and Network Security (ACNS) 2017. In [64] we present an anomaly detection mechanism for industrial control systems that takes advantage of the state of the physical process, the paper appeared in Proceedings of Symposium on Applied Computing (SAC) 2018.

- Chapter 5 concludes the first part of this dissertation, summarizing our contributions, lessons learnt, and future works.

Chapter 2

MiniCPS: A toolkit for security research on CPS networks

Keywords: CPS, Mininet, Simulation, Emulation, Containers, SDN.

2.1 Introduction

Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) systems traditionally rely on communication technology such as RS-232 and RS-485, and field buses such as PROFIBUS [63]. Due to the long lifetime of industrial components in such settings, transitions to technology such as Ethernet, TCP/IP, and related protocols are often only implemented now. The adoption to the standard Internet protocol suite is expected to enhance interoperability of the equipment, and reduce overall communication costs.

The growing connectivity is also expected to introduce novel security threats, in particular when systems are communicating over public networks such as the Internet. While a great amount of research has been conducted on network security of office and home networks, recently the security of CPS and related systems has gained a lot of attention [34, 108, 182, 196, 197]. In CPS, physical-layer interactions between components have to be considered as potential attack vectors, in addition to the conventional network-based attacks. Examples for real-world CPS are unfortunately often not open to security researchers, and as a result few reference systems (i. e., physical process description, control systems, network topologies) are available. In addition, physical layer interactions between components need to be considered besides network communications. We believe that this will require novel simulation environments, that are specifically adapted to cater for the requirements of CPS and ICS. In particular, such environments could be used to co-simulate (in real-time) using different simulated and emulated components together with real hardware or processes.

In this work, we present *MiniCPS*, a toolkit intended to alleviate this problem. The goal of MiniCPS is to create an extensible, reproducible research environment targeted towards CPS. MiniCPS will allow researchers to emulate the Ethernet-based network of an industrial control system, together with simulations of components such as PLCs. In addition, MiniCPS supports a basic API to capture physical layer interactions between components. Based on MiniCPS, it is possible to replicate ICS in real-time, for example to develop novel intrusion prevention systems, or own software to interact with industrial protocols. While not all CPS systems are using Ethernet-based communication so

far, we see a general trend towards wide adoption of Ethernet as physical layer for the communication [63].

MiniCPS can also be used to share different system setup, and can be extended by standard Linux tools or projects. Due to our use of Mininet for the network emulation part, MiniCPS is also well suited to perform research on Software-Defined Networking in the context of Industrial Control Systems.

We summarize our contributions as following:

- We identify the issue of missing generic simulation environments for applications such as cyber-physical systems. In particular, such simulation environments should support physical interactions, parametric communication links, and specific industrial (ideally all) protocols that are used.
- We present MiniCPS, a framework built on top of Mininet, to provide such a simulation environment.
- We present an example application cases in which we use MiniCPS to develop and refine a specific attack, which we later validated in a real testbed.
- We propose the use of Software-Defined Networking for CPS networks to enable efficient detection and prevention of the attack presented earlier. We design and implement a matching SDN controller, and validate it in MiniCPS.

The structure of this work is as follows: In Section 2.2, we introduce Mininet and CPS networks in general. We propose our MiniCPS framework in Section 2.3, and provide an application example in Section 2.4. In Section 2.5, we show how MiniCPS can be used to develop a CPS network specific SDN controller. Related work is summarized in Section 2.6. We conclude the chapter in Section 2.7.

2.2 CPS Networks and Mininet

In this section, we will introduce some of the salient properties of industrial control system (ICS) networks that we have found so far. In addition, we will briefly introduce Mininet, the network simulation tool we use as part of MiniCPS.

2.2.1 ICS networks

In the context of this work, we consider ICS that are used to supervise and control system like public infrastructure (water, power), manufacturing lines, or public transportation systems. In particular, we assume the system consists of programmable logic controllers, sensors, actuators, and supervisory components such as human-machine interfaces and servers. We focus on single-site systems with local connections, long distance connections would in addition require components such as remote terminal units (see below). All these components are connected through a common network topology.

Programmable logic controllers. PLCs are directly controlling parts of the system by aggregating sensor readings, and following their control logic to produce commands for connected actuators.

Sensors and actuators. Those components interact with the physical layer, and are directly connected to the Ethernet network (or indirectly via remote input/output units (IOs) or PLCs).

Network Devices. ICS often use *gateway* devices to translate between different industrial protocols (e. g. Modbus/TCP and Modbus/RTU) or communication media. In the case where these gateways connect to a WAN, they are usually called *remote terminal units* (RTUs).

Network Topology. Traditionally, industrial control systems have seen a wide deployment of direct links between components, based on communication standards like RS-232. In addition, bus systems such as RS-485 and PROFIBUS have been used. In particular, focus on reliability led to a deployment of topologies such as rings [40], which could tolerate failure of a single component without loss of communications, with very low reaction time (typically in the order of milliseconds).

Industrial protocols. In recent years, industrial networks are transitioning to mainstream consumer networking technology i. e. Ethernet (IEEE 802.3), IP, TCP. Nevertheless, the need for reliability and interoperability with existing equipment leads to systems that are different from typical home and office networks, such as Ethernet rings, use of IP-layer multi-casting, and custom protocols such as the EtherNet/IP (ENIP) [129]. ENIP is a Real Time Ethernet (RTE) field bus based on TCP/IP with a custom application layer designed to meet typical industrial communications requirements like real-time response and packet determinism. Technically ENIP is an Ethernet based implementation of the *Common Industrial Protocol (CIP)* and it is defined in the IEC 61784-1 standard [63]. CIP messages can be used to query sensor readings from components, set configuration options, or even download new logic on a PLC. In that model, sensor readings or control values are represented by *tags*. CIP uses a request-response model where a client sends a request to a server (for example to read a tag containing a value read from a hardware component) and where the server then sends back a reply (e. g. with the requested value or an error code). Such requests can operate on tags and also on the meta-data associated with the tag, like access control and data type, which are stored in *attributes*. ENIP handles the selected aspects of the communication, for example with connected sessions (with handshake and tear-down messages) and unconnected sessions (without any handshake but with more contextual data in every CIP packet).

Topology layers. Networks for industrial control systems are often grouped in several layers or zones (more detail on such networks in [124, 142]). On the lowest layer (which we call layer 0 or L0), sensors and actuators are connected to controllers such as PLC. The sensors and actuators are either capable of connecting to a network directly (e. g. using ENIP), or they use basic analog or digital signaling, which has to be converted to Ethernet-based communications by *remote input/output* (RIO) devices. Only if actuators and sensors are physically very close to the PLC, the IO modules will be installed as part of the PLC.

The next higher layer (layer 1/ L1) will connect the different controllers (PLCs) with each other, together with local control such as Human-Machine-Interfaces (HMI), local engineering workstations, and Data historians. For simplicity, all these devices are often kept in the same IP-layer sub-network, although more complex topologies are possible. We also note that industrial Ethernet switches are often focused on electrical reliability, instead of IP-layer functionality (e.g. the Rockwell Automation Stratix 5900 switch). We provide the network topology of a generic ICS network as an example in Figure 3.1.

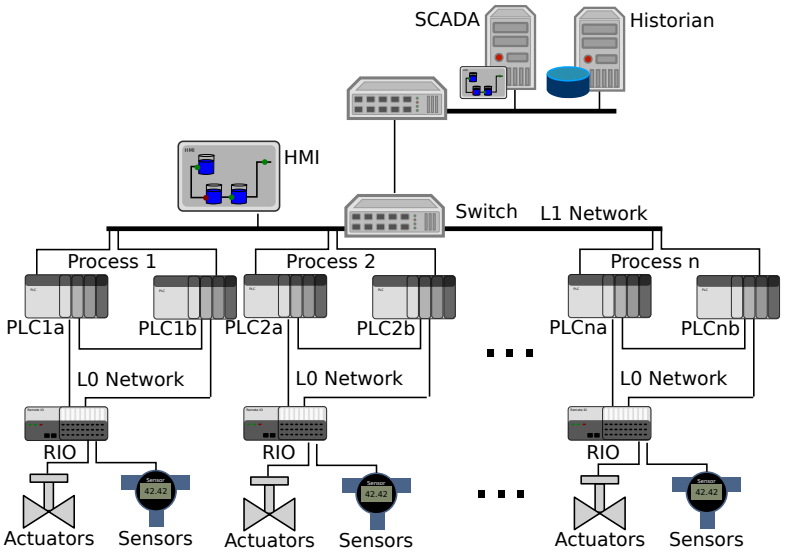


FIGURE 2.1: Example local network topology of a plant control network.

2.2.2 Mininet

Mininet [105] is a network emulator that allows to emulate a collection of end-hosts, switches, routers, middle boxes, and links with high level of fidelity. It enables rapid testing and prototyping of large network setups on constrained resources, such as a laptop. Furthermore, it was build around the Software-Defined Networking paradigm, facilitating SDN research and development [131].

Mininet exploits lightweight system virtualization using Linux *containers*. This approach presents various advantages over a full system virtualization: Mininet runs on a single kernel, its computational overhead is lower and the emulator can easily tolerate scalability issues (e. g. one thousand containers vs. one thousand dedicated virtual machines).

Each virtual host is a collection of processes isolated into a container. A *virtual network namespace* is attached to each container and it provides a dedicated virtual interface and private network data. Link are emulated using virtual Ethernet (`veth`) and they can be shaped through Linux Traffic Control (`tc`) to emulate link performance such as delay, loss rate and bandwidth. Each virtual host utilizes its virtual interface to communicate with other network devices.

Mininet can be used in multiple scenarios and can be easily adapted over time to track the evolution of CPS networks. It provides a realistic emulation environment to the user, and one can work with the same addresses, protocol stacks and network tools of a physical network, it is even possible to reuse helper scripts and configuration files from the emulated environment directly in the physical network.

Mininet ships with a set of common topologies, in addition the user can easily extend this collection through the provided public Python APIs. Dynamic interaction within any chosen topology can be achieved through a convenient command line interface. Mininet is free, open-source, well documented and actively maintained by a

strong and competent community. Furthermore, Mininet gives the user the opportunity to develop OpenFlow network architectures with direct integration of experimental code into production code.

2.3 MiniCPS

In this section, we present our proposed toolkit *MiniCPS*. MiniCPS combines a set of tools for real-time emulation of network traffic with CPS component simulation scripts, and a simple API to interface with real-time physical-layer simulations. The combined system will allow a) researchers to build, investigate, and exchange ICS networks, b) network engineers to experiment with planned topologies and setups, and c) security experts to test exploits and countermeasures in realistic virtualized environments.

In MiniCPS, components such as PLCs are represented by Python scripts that manage the decoding of industrial protocols and physical layer sensor and actuator signals. All network components (e. g. switches) are emulated using Mininet (as introduced in Section 2.2.2). Physical layer interactions are currently connected to suitable simulation software through a simple API (based on shared read/write to files). MiniCPS itself does not focus on providing physical process simulations – it rather connects components to their counterparts, and process simulation engines.

2.3.1 Goals of MiniCPS

In the following, we provide a discussion of our vision behind MiniCPS, in particular related to its focus on network emulation, overall simulation, and real-time properties. We also discuss co-simulation with real devices, integration of physical-layer simulation tools, and integration of software defined networking.

Network Emulation. MiniCPS focuses on high-fidelity network emulation, which allows simulated components to exchange very similar or the same traffic as seen in the real network, from Ethernet layer up to the application layer (based on Mininet, see Section 2.2.2). In particular, we aim to not only provide an abstraction of the network to perform simulations on (similar to network simulators such as NS2 [92], OMNet++ [179]), but we target a network emulation that is largely identical to a real network, without the cost or overhead of running a real network or a set of virtual machines. In particular, this would allow us to develop components that are directly using industrial protocols to communicate.

Simulation vs Emulation. We recognize that the difference between emulation and simulation might not be universally defined. In the context of this work, we use the terms *simulation* and *emulation* with the following intentions: components that are emulated will behave (in real-time) exactly at their real-world counterparts. The network emulation in Mininet is an example – there is essentially no difference for the OS between a virtual network adapter in Mininet, and a real physical network adapter. We note that the timing of the emulated components will be similar to the real counterparts, but might not match exactly. The emulated components will run in real-time, and cannot be sped up or slowed down in their operations. Emulated components should be able to interact directly with other emulated components, if their real-world counterparts are able to interact.

In contrast, simulated components operate on a reduced model to analyze selected properties of the target system. The simulation might run in real-time, or faster/slower. The simulated component will typically interact with other components through channels that do not replicate real-world interactions. In this work, we assume that physical processes can only be simulated. We also classify components as "simulated" if they are only approximating their real-world counterparts in behaviour or interactions. We note that in the context of MiniCPS, simulations will be restricted to real-time to allow interoperability with emulated components.

For example, if (in the future) software is available in MiniCPS that processes real-world PLC code (e.g. in ladder logic format), and replicates the behaviour of the PLC accurately, then we would call this a PLC emulation. For now, our tools that represent PLCs in MiniCPS are merely implementing a subset of PLC functionality, as manually extracted by us from the analysis of the real-world PLC code.

Time. While the traffic data will be mostly identical, the temporal properties of the traffic depend on the involved simulation scripts. In general, Mininet can emulate delays, but does not provide deterministic guarantees for timing. We recognize that this could lead to problems when simulating closed loop control over the network with very tight timing, but we do not face those problems in our current application (a relatively slow water testbed). We estimate that precision in the order of milliseconds could be possible.

Co-Simulation. Based on the high-fidelity network emulation, it is also possible to connect simulated and real networks together, to enable cheap and easy evaluations of real devices in emulated network environments. This integration is possible because the network traffic is not simulated on some abstraction layer, but identical traffic running over an emulated physical network layer.

Physical-Layer Abstraction. The physical-layer integration into MiniCPS is mainly relying on our API to represent the limited number of *interaction points* between the physical process and the sensor/actuator (cyber) components. The API is intended to enable different simulated components to interact on the physical layer directly, without requiring complex specialized interfaces.

Reproducibility. In [82], the authors proposed to use tools such as Mininet to disseminate reproducible research results. In particular, researchers can publish the scripts to generate their network setups, which allows other researchers to reproduce the exact same environments for their experiments. We strongly believe that such dissemination of results would also be helpful in the context of security research, in particular when systems which are less mainstream are considered. While it is relatively easy to replicate office network settings as related software is well-known, specialized application setups such as ICS would be valuable to share.

Integration of SDN. The detailed network emulation will allow us to use novel concepts such as software defined networking in the context of CPS networks (see Section 2.5). We note that to achieve this compatibility, we will be constrained to real-time simulation instead of being able to simulate with arbitrary speedup.

What MiniCPS does not aim for. MiniCPS does not aim to be a performance simulator, or tool for optimizations. In particular, we consider that full-fledged physical process simulation is out of the scope of MiniCPS. Instead, MiniCPS focuses on connecting

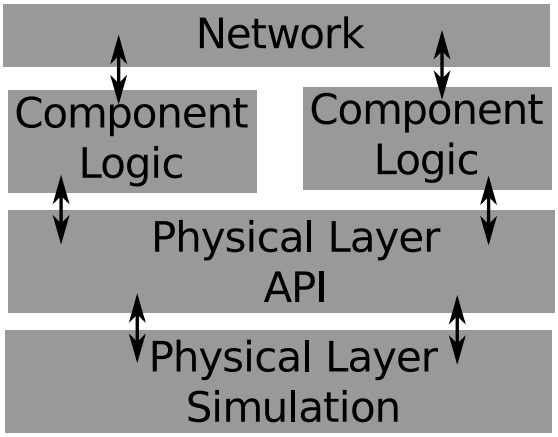


FIGURE 2.2: MiniCPS framework layers: CPS components are simulated as component logic, connected through the network emulation, and physical layer simulation.

component simulation/emulation scripts together in a virtualized environment, and enable simple connections to third party physical layer simulation tools. In addition, we currently put very little emphasis on GUI or visualization. We note that building on top of the physical layer API, and by extending the component logic scripts in general, it should be possible to easily create real-time charts of physical process parameters or controller states.

2.3.2 Design overview

Components in MiniCPS interact on several layers (see Figure 2.2). On the top layer, we have the network through which messages are exchanged on top of ENIP, or other protocols. Connected to this network are components, their logic is implemented in simple scripts or more advanced software packages. If the real-world counterpart of these components is interacting with the physical layer, the simulated components can also access specific physical layer properties through a second API, which abstracts the physical layer. To simulate chemical or physical processes, a selection of their properties can be made available through the API and updated in real-time by simulation scripts.

2.3.3 Network Communication

For the main network emulation layer of MiniCPS, we are using Mininet (see Section 2.2.2). We chose Mininet primarily because it allows lightweight emulation of a large number of hosts and their network connections (Mininet's SDN capabilities are a welcome addition). Mininet allows to directly use IP-based industrial protocols over the virtualized Ethernet connections. In contrast, other network simulation tools usually require an abstract re-implementation of the used protocols. Mininet allows to set basic link properties such as delay, loss rates, and capacity. In MiniCPS, we use this functionality to allow individual links to be configured with individual settings. As a

result, we can emulate wide area network connections and local area network connections with different properties.

Based on Mininet, the network communication in MiniCPS uses the default Linux networking stack based on Ethernet. All components have virtualized network interfaces that are connected to each other. In particular, this setup allows us to construct arbitrary topologies such as simple star topologies of switches connected to devices, intermediate routers and firewalls, and topologies such as Ethernet rings. Protocols such as the spanning-tree-protocol or other routing algorithms can be used to automatically avoid looping configurations, and to establish routes. All standard protocols such as ICMP, HTTP, NTP, etc. can be used right away. On top of that, specific industrial protocols can be used. In particular, we use the CPPPO Python library to provide fundamental EtherNet/IP (ENIP) services [104]. In addition to ENIP, CPPPO also supports protocols such as Modbus/TCP. In addition to CPPPO, we also use the pycomm library for ENIP communications [155].

2.3.4 Physical Layer Interactions

Physical layer interactions between different components in the systems are captured by our PHY-SIM API. This API is essentially a set of resources (currently files), that provide data in real-time. These resources can be read by components (i. e. a sensor reading some physical property), or written to (typically, by a script that emulates physical processes). The main purpose of the simple API is to allow different tools to interact with it as easily as possible. For example, such tools could be Matlab, Python scripts, or dedicated physics simulators. Representing the physical layer properties as file resources makes this API usable by a wide range of libraries and programming languages. We also envision that it is possible to connect these files to an actual physical process, i. e. to have the process *in the simulation loop* (if suitable interfaces to the physical system are provided). In the long term, the simple API could be extended to a more generic API, for example a RESTful API.

2.3.5 Implementation

MiniCPS is essentially a set of tools that extends Mininet, in particular by adding scripts to represent components such as PLC, HMI, and historian servers, and by adding the physical layer API and example physical process simulation scripts. Our class hierarchy follows Object Oriented design principles: every reusable, self-contained piece of code is wrapped inside a class (such as a topology, a topology manager or an SDN controller).

Our implementation contains three core modules: constants, topologies, and devices. The *constants* module collects data objects and helper function common to all the codebase. The *topologies* module is a collection of ad-hoc CPS and ICS topologies with realistic addresses and configurable link performance. The *devices* module contains a set of control logic applications (see Section 2.5). Each core module is mirrored with a testing module counterpart. Our class hierarchy design easily allows Test Driven Development [23] because each topology manager potentially can select a network configuration, a controller, the performance of the virtual links and even the CPU allocation for each virtual host. In other words, a topology manager it is a self-contained topology

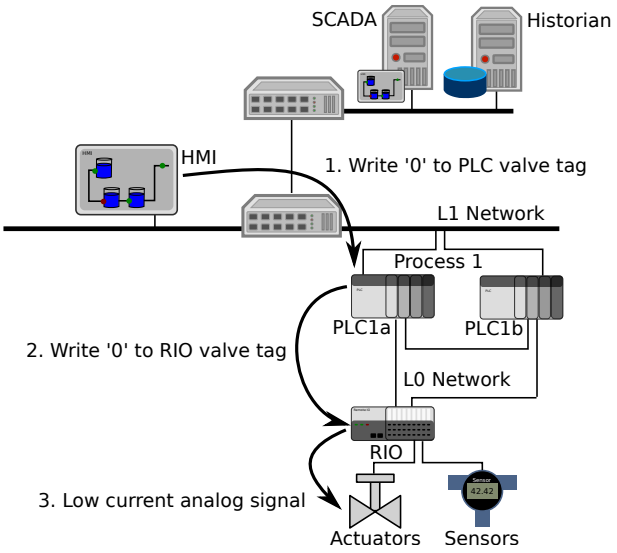


FIGURE 2.3: Normal control message flow in the CPS. We omit the acknowledgment reply from the PLC in this visualization.

test. Each test module is a collection of *test_Something* classes with appropriate fixtures. , e.g. to set the Mininet log level at setup.

We used the Python *nosetests* module to automate test design, discovery, execution, profiling and report. The *logging* module enables interactive code debugging/alerting and long time information storage. Each core module and its testing counterpart append information to the same log file, that rotates automatically through five time-sorted backups. SDN controllers log on separated files that are (over)written at runtime. SDN code integration is obtained by means of soft links using an initialization bash script.

We have implemented a first version of MiniCPS, and are currently in the process of testing and extending its functionality. We released the tool to the public under an open source license, with the main project website available at `minicps.net`. All extensions are using Python, and are documented using the Sphinx package.

2.4 Example Application: MitM traffic manipulations

We mainly use MiniCPS to model the communications and control aspects of a water treatment testbed at our institution (Singapore University of Technology and Design). While the testbed is intended for security research, we find it useful to have the MiniCPS emulation environment to replicate the network settings outside the lab. In addition to simulated interactions with PLCs and sensors, the MiniCPS model also allows us to experiment with different network topologies, and test SDN-related prototypes. In the following, we highlight two such projects based on the MiniCPS model of our testbed. The first application aims to provide on-the-fly manipulation of ENIP/CIP traffic to change commands and sensor values as exchanged between an HMI and a PLC. The second application (in Section 2.5) concerns SDN controller-based detection and mitigation of ARP spoofing attacks in the testbed.

2.4.1 System model

Let us assume the following setting (see Figure 2.3): the HMI is used to manually control the valve of a water feed line towards a water storage tank. The control decision is done on the HMI (e.g. operated by a human), based on the fill-level of the tank as reported by a sensor in the tank. In practice we found that such settings are common.

Based on that scenario, we modeled the network, HMI, PLC, and the physical layer interaction between the valve and the tank in MiniCPS. In particular, we modeled the valve as a Boolean value, and the fill-state of the tank (depending on the current volume of water held, and the inflow and outflow). The valve value is periodically read by a process simulation script. If the valve is open, the current fill-state of the tank is increased by an amount representing the inflow. Both the valve and fill-state are also used by the PLC simulation script, which periodically reads the fill-state and provides it as read-only CIP tag to the emulated network. The simulated PLC also provides a writable CIP tag for the valve control.

2.4.2 Attack scenario

The attacker has access to the local L1 network, his goal is to arbitrarily change the fill state of the tank. An example attack could aim to fill a water tank it over allowed maximal capacity (and thus damaging the tank or flooding the environment), without being detected.

As first naïve approach, an active attacker could potentially overwrite the valve control tag (as there is no direct access control in ENIP), but the HMI will continuously overwrite the setting to its intended state (in our system, with 10Hz). As a result, to continuously change the valve setting, the attacker has to send a large amount of traffic to compete with the intended control by the HMI, potentially interrupting normal operations.

To mitigate this unintentional countermeasure, we developed a Man-in-the-Middle (MitM)-based attack that does not increase the traffic load on either HMI or PLC, and which does not interferes with other data exchanged between PLC and HMI. The attack is based on ARP spoofing using the ettercap tool [58], and a custom ettercap script to manipulate captured traffic in real-time. We now provide a brief summary of ARP spoofing attacks, and then present the ENIP traffic manipulation attack in more detail.

ARP spoofing is a well-known attack in computer networks [186]. The attacker is connected to the same Link Layer network segment as two victims, that are exchanging messages. The attacker then sends specifically crafted address resolution protocol (ARP) packets to both victims to cause them to send their messages to the attacker, instead of each other. The attacker then forwards the redirected messages to the original recipient, which allows him to perform a stealthy man-in-the-middle attack. We will show a possible countermeasure against this attack in Section 2.5.

Using ARP-spoofing, an attacker in the Layer 1 network of an ICS system (as described in Section 2.4.1) can redirect all traffic between two victims, e.g. PLC1 and the HMI.

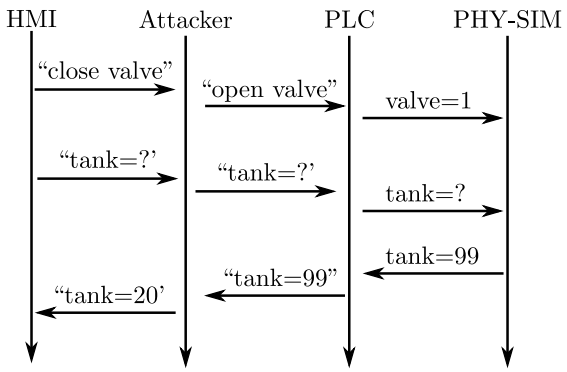


FIGURE 2.4: Abstract messages in the extended attack: in addition to the modification of the control messages, the affected measurements from the PLC are also manipulated to hide the attack. In this setting, PHY-SIM could either be a real physical process, or our simulation layer.

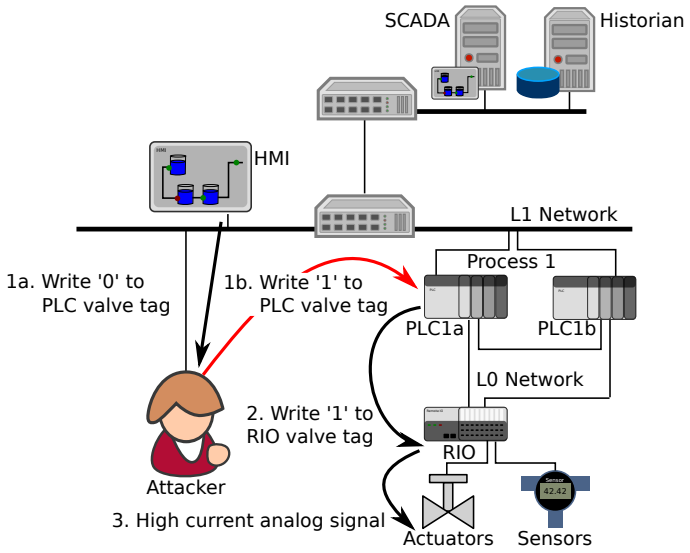


FIGURE 2.5: Control message flow during the ARP spoofing attack.

2.4.3 MitM Attack

In the MitM attack (see Figure 2.5), we used the ettercap tool [58] to run an ARP spoofing attack, and as a result install the attacker as MitM between the HMI and the PLC (see Figure 2.4). We wrote a set of ettercap filter rules to change the value written by the HMI to the valve tag at the PLC. As a result, each time the HMI sent a control message to the PLC to keep the valve closed, the attacker could then change this setting to “open”, without fearing the HMI from overwriting it again. We developed and deployed ettercap scripts to perform this attack in MiniCPS, and were able to successfully change the valve tag to arbitrary values.

2.4.4 Including physical layer interactions

In our MiniCPS setup, we then simulated physical layer interactions based on the MiniCPS physical layer API, and several scripts that updated physical layer properties based on a set of simple rules. As result, the valve opened by the attacker led to an increasing fill-state of the tank, which was in turn reported by the PLC when queried by the HMI. We note that this dependency is rather simple, and did not require complex simulations. In practice, this would allow the HMI to at least trigger an alarm condition after the tank is exceeding the maximal fill state.

To prevent such detection, we extended our attack with a second set of filter rules in the attacker. In addition to rewriting the valve control values, the attacker now also rewrote the value of the fill-state tag as reported from the PLC to the HMI. In particular, the attacker could set this value to a constant, or apply some noise to it if wanted. We successfully applied this attack in the MiniCPS environment.

After applying the attack in MiniCPS, we were able to validate the same attack to the real physical testbed, with only minor modifications. The modifications were necessary as the exact CIP messages exchanged between the HMI and PLC in the physical testbed are not yet fully identical to the ones exchanged in our MiniCPS environment. In particular, CIP supports different types of read operations, and our simulated PLCs do not support the variant that is used in the real PLCs (due to limitations in the used CPPPO library). Apart from these modifications, the same attack was successfully run.

2.5 Example Application: SDN

There are a number of known countermeasures against the ARP spoofing attack from the previous section (e. g. static ARP tables in the hosts, traffic monitoring with an IDS).

As a side-effect of using Mininet in MiniCPS, it is relatively easy to experiment with Software-Defined Network (SDN) technology. Given that, we were interested to see how a customized SDN controller could be used to detect and prevent the attack outlined in the previous section. In particular, SDN promises to allow a controller to manage CPS packet switching rules and to prevent or detect/mitigate malicious hosts packets. SDN-based solutions in the context of smart power grids were also recently proposed in [52]. In a more general context, related work was published recently in [194, 193].

We note that, while in many applications SDN is used to address highly dynamic network conditions, traffic in industrial control systems is usually quite predictable. In particular, topologies and the set of hosts remain static (until the system is updated with new components). In addition, we noticed that components exchange essentially the same traffic, with varying data payload. For example, tag values could be queried every 100ms, and control commands could be sent every second, resulting into regular traffic patterns. As result, we can use the SDN paradigm to extract and enforce these traffic patterns, which allows us to detect and prevent ARP spoofing attacks.

We now introduce SDN in general, the POX controller project in particular, and then show how we used MiniCPS to prototype a simple POX controller design to prevent such ARP spoofing completely in our testbed.

2.5.1 SDN Background

Software Defined Networking (SDN) is a novel architectural way to think about building networks and OpenFlow is the de-facto standard interface protocol between the SDN controlling logic and the network devices (physical and virtual). Both ideas were proposed by *M. Casado* and they derive from SANE [31], a protection architecture for enterprise networks.

SDN implementation defines a set of abstractions to provide separation of concerns at the *control plane*, in a similar way as the layering model that is used at the data plane. At the bottom of the stack there are network devices that form the physical topology. On top of that there is a Network Operating System (NOS) able to talk to each device and to serve a network view, in the form of an *annotated graph*, to the layer above. A virtualization layer is able to process this graph and provide only relevant details to the level above through an API. At the top of the stack there is the control logic that defines policy over the network assessing the processed graph. Communications between the control logic and the physical devices is bi-directional: network device messages will update the network graph and control plane messages will update the network policy. With this setting, the *end-to-end principle* is also applied to the control plane. The (complex) management of the network is shifted on the edges and central network devices merely act as relays, becoming an homogeneous set of forwarding objects referred as *datapaths*.

A Software-Defined network is managed by a control logic that dictates to the datapaths how to forward packets using a dedicated control plane protocol (e. g. OpenFlow). The control logic may be a centralized program running on a server (known as a SDN controller), a set of programs divided into modules by functionality (known as a Network OS) or even a set of equal or different distributed programs running on different servers. The control logic rules may act on single packets (micro-flow rules) or on a set of packets sharing some properties (aggregate-flow rules). The control logic strategy may be static and loaded at initialization time (proactive), dynamic and updated at runtime (reactive) or hybrid implementing both of them.

There are various interesting projects regarding SDN and OpenFlow, and it is relatively easy to find a platform that implements the core modules, namely the NOS and the virtualization abstractions. In our work we decided to use the POX [128] platform because it is targeted for the research community, it offers out of the box libraries and components, and it is object-oriented, event-driven with synchronous and asynchronous handling capabilities. In addition, POX is completely written in Python and it integrates well with our set of tools (Scapy, CPPPO, Mininet, MiniCPS).

In the POX framework, events represent communication from the network to the control logic (e. g. new datapath connections) and callbacks represent communication from the control logic to the network (e. g. install a new rule). Event handling is priority based: the same event can be handled sequentially by different callbacks, a callback can potentially stop the handling chain and the same callback can handle multiple events.

2.5.2 Preventing MitM attacks with a custom SDN controller

We now present our SDN controller design, which aims to prevent the ARP spoofing attacks as discussed in the previous Section. In particular, our controller will analyze all

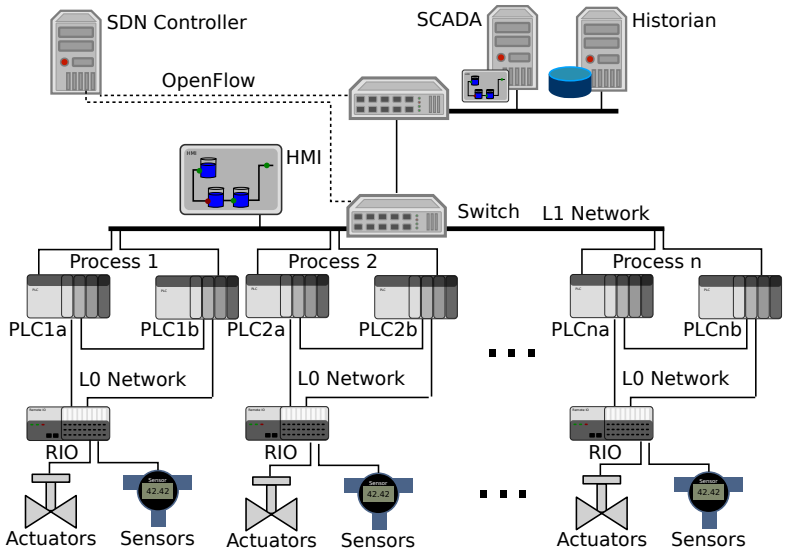


FIGURE 2.6: Extension of the generic ICS network with an OpenFlow switch and SDN controller.

ARP traffic, classify it as malicious or benign, and then update the SDN switches with suitable rules to prevent malicious attacks. Our attacker model consists of an attacker able to impersonate a network device (known or unknown to the other testbed hosts) that aims to mount a passive or active MitM attack using ARP poisoning over our real/emulated testbed L1 wired ENIP network.

Our POX controller implements a fully centralized SDN control plane with per-flow forwarding rules. Our control plane program uses both a proactive approach to perform a static pre-mapping and a reactive approach to adapt dynamically to the context. The detection and prevention code runs with higher priority than the management code and it is able to block the event handling chain.

Every time a new switch is connected to the network, our control logic will create a new reference to the *network state* accessible by the switch. According to OpenFlow protocol, when a switch doesn't know how to forward a packet it sends (a part of) it to the controller and it waits for instructions (that's when POX raises a PacketIn event). Our control logic process unknown ARP reply and ARP request packets, sent by the switches, verifying their consistency according to the network state. The controller is able to detect both *internal* and *external* ARP spoofing attempt and to prevent both *passive* and *active* ARP MitM attacks. In normal ARP request/reply circumstances, the controller dynamically updates the network state.

As shown in Figure 2.7, suspicious ARP request packets are signaled as warnings, and they are mirrored to a well-know port (potentially connected to a dedicated IDS for deep packet inspection), but the PacketIn handling chain is not halted. In contrast, suspicious ARP replies are actively managed: the control logic will install a permanent rule on the relevant switch to permanently block all the traffic coming from the attacker, the blacklist rule is based on the attacker MAC address and the used input port (but can be easily tuned according to the scenario). Finally, the controller will block the PacketIn handling chain.

In addition to this simple attack detection and prevention strategy, we are currently

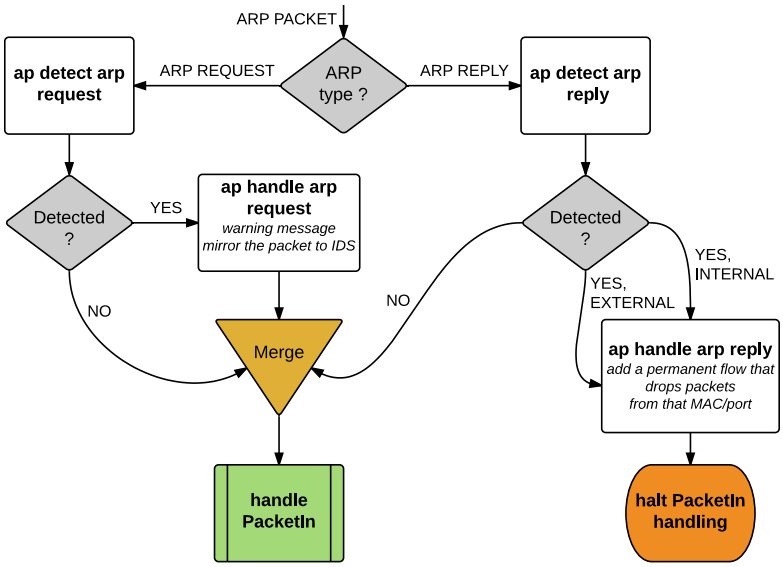


FIGURE 2.7: ARP spoofing prevention flow chart. The process either allows the PacketIn event handling (green) or blocks it (orange).

developing more elaborated ARP detection and mitigation techniques, in particular (i) an *ARP cache restoring* handler, and (ii) spoofing detection based on *static mapping* of MAC/IP pairs. The ARP cache restoring feature will periodically or asynchronously sends ARP replies to potentially every host in the network, forcing it to update its ARP cache with fresh and consistent data. The *strong static premap* feature will allow the controller to send to every new datapaths a set of predefined flow rules to speedup initial traffic congestion and policy establishment (e. g. who can talk to who). Eventually, this mechanism can be extended with a dynamic policy checker component, that is able to validate and restore the correct network state requesting and processing general and aggregated flow statistics directly from the datapaths. We think that those two add-ons will protect the network against (pre)mapped switches containing inconsistent rules and DoS attacks. We plan to extend our current centralized design into a more robust distributed scheme by using multiple synchronized controllers able to tolerate single point of failure in the control plane domain.

2.6 Related work

Security aspects of CPS have been discussed in [108, 182, 196, 197], in particular in the context of smart power grid infrastructure and control.

Process and network co-simulation. In [52], Dong *et al* propose a testbed that is similar to our MiniCPS platform in several ways. In particular, they propose to use Mininet as network emulation platform, a power grid simulation server, and a control center simulation server. The envisioned testbed uses Mininet to simulate delays related to dynamic network reconfigurations in the case of failures. In general, the authors just discuss the use case of the smart power grid, with component such as sensors and actuators connected to a central control via RTUs.

We note that MiniCPS differs from the testbed in [52] in several ways. Most importantly, MiniCPS' focus is on sharing reproducible CPS network topologies, in particular related to industrial control systems. MiniCPS focuses on using a set of PLC simulation tools, that directly interact with the network traffic, and the physical layer API. The physical layer API abstraction is not present in [52], as the authors propose the use of a powerful power-grid simulation tool (PowerWorld). In MiniCPS, the (generic) API allows to combine different types of physical layer simulations (e.g., combining water flow, mechanical levers, temperature transfer). Finally, the industrial protocol differs (ENIP vs. DNP3). From [52], it seems that the proposed testbed was not yet fully implemented.

In [34], a framework with similar intent as MiniCPS has been proposed. The framework uses OMnet++ as network simulation tool, and also features simulation of physical layer (e.g. a chemical plants). The authors simulated denial of service attacks on the sensor data, and the resulting control actions. As OMnet++ was used for network simulations, network communication was simulated as abstract messages that were routed through components, instead of simulating the full TCP/IP+industrial protocol stack. As a result, attacks such as our MitM ettercap manipulation could not be simulated in detail (i. e. considering all fields of the CIP/ENIP messages). On the other hand, simulations like [34] allow to use timescales other than real-time.

SDN. On the topic of SDN, SANE [31] represents one the first practical SDN-based solution for secure network design. The proposed implementation already included common SDN core concepts like centralized control logic, high level network policy design and easy network scalability.

SDN and OpenFlow projects involved from the beginning both academia and leading IT industries, that eventually found the Open Networking Foundation (ONF). There are several other recommended papers about SDN [60, 174, 133] and OpenFlow [120, 193].

Physical layer simulation tools. PowerWorld is a commercial (interactive GUI-based) power transmission grid simulation. PowerWorld does not provide capabilities to directly interact with industrial communication protocols and does not simulate communication network aspects. In [20], the authors present a test bed which combines physical layer simulation with PowerWorld, and abstract network simulation based on the RINSE [107] tool.

2.7 Conclusion

In this work, we proposed MiniCPS, which uses Mininet together with a physical layer API and a set of matching component simulation tools to build a versatile and lightweight simulation system for CPS networks. While currently the physical layer simulation is very simplistic, we believe that our general framework will (a) researchers to build, investigate, and exchange ICS networks, (b) network engineers to experiment with planned topologies and setups, and (c) security experts to test exploits and countermeasures in realistic virtualized environments.

MiniCPS builds on Mininet to provide lightweight real-time network emulation, and extends Mininet with tools to simulate typical CPS components such as programmable

logic controllers, which use industrial protocols (EtherNet/IP, Modbus/TCP). In addition, MiniCPS defines a simple API to enable physical-layer interaction simulation. We demonstrated applications of MiniCPS in two example scenarios, and showed how MiniCPS can be used to develop attacks and defenses that are directly applicable to real systems.

Chapter 3

Towards high-interaction virtual ICS honeypots-in-a-box

Keywords: Honeypot, High-interaction, Virtual, In-a-box, Detection.

3.1 Introduction

Industrial Control System (ICS) security is a promising research topic, because it combines traditional cyber-security threats with control system security ones [168, 169]. Attacks targeting ICS, such as the sophisticated stuxnet worm, are becoming more frequent, and can have serious consequences (e.g., economical damages, environmental catastrophes and loss of human lives [163, 59]).

It is fundamental to harden the security of an ICS, especially in this decade where often ICS devices are facing the Internet on a public IP. Typical defense mechanisms include the use of industrial firewalls to segment the network architecture, and Intrusion Detection Systems to monitor the network traffic, and react in case of suspicious activity.

In security research, *honeypots* are vulnerable systems that are set up by defenders with the intent to be probed and compromised by attackers. Monitoring systems will then record traces of the attacks and actions taken. In that context, honeypots are able to provide detailed information about the attacker's activities, and to defend-against, or slow-down, the ongoing attack. Honeypots are extensively used in traditional ICT systems, but they are rarely deployed in the ICS domain, mainly because of the very high associated costs, and maintenance's complexity. So far, little academic work has been done in the domain of ICS honeypot design and implementation.

In this work, we propose a design for realistic virtual ICS honeypots. Our design addresses the main challenges for ICS honeypots related to ICT and ICS requirements (e.g., time, determinism, and operating cost). We present an attacker model for the ICS honeypot that captures the goals, the skills, the resources, and the entry points of the attacker. According to the requirements of the attacker model we then propose our architecture design. We classify the presented ICS honeypot as server-based, and high-interaction honeypot (to satisfy the realism constraints), and virtual (to satisfy the cost and maintainability constraints).

We then present an implementation based on our ICS honeypot design. The implementation leverages the MiniCPS framework, which combines lightweight virtual network emulation with physical process, and ICS devices simulation, to help researchers

simulating Cyber-Physical Systems. To the best of our knowledge, the presented honeypot implementation is the first academic work targeting EtherNet/IP based ICS, the first ICS virtual honeypot that is high-interactive without the use of full virtualization technologies, such as a network of virtual machines, and the first ICS honeypot that can be managed with a Software-Defined Network controller.

To show the effectiveness of the implemented honeypot, the chapter presents the evaluation of a honeypot that is mimicking a water treatment testbed. The attacks on that system were conducted in the context of a cyber-security Capture-The-Flag event. The evaluation confirms that honeypots are a potential solution also in the ICS domain and that they can be integrated in an ICS defense-in-depth scheme.

Honeypot development is a broad topic, and the chapter is focusing on the honeypot's core functionalities such as: the network, the physical process, the physical devices and the data retrieval. In particular, the chapter is not focusing on the data post-processing part (e. g. no data analytics).

The rest of the chapter is organized as follows: in Section 3.2, we introduce ICS networks, ICS honeypots, and the MiniCPS framework. The honeypot's requirements, attacker model, and proposed design are presented in Section 3.3, and the honeypot's implementation core components, and additional benefits are presented in Section 3.4. In Section 3.5, we present the evaluation of the implemented system. Related work is summarized in Section 3.6. We conclude the chapter in Section 3.7.

3.2 Background

We now briefly summarize ICS networks, and ICS honeypots Then, we introduce the MiniCPS framework.

3.2.1 ICS Networks

Industrial Control Systems (ICS) are used to supervise and control systems such as critical infrastructure (electric power, and water), and public transportation systems (trains, and planes). In this work, we assume the system consists of supervisory components, such as human-machine interfaces and servers, programmable logic controllers, sensors, and actuators. All those components are interconnected through a network with a specific topology. We provide the network topology of a generic ICS network as an example in Figure 3.1.

Programmable logic controllers. PLCs are the core controllers of an ICS. Each device runs a program, also known as control logic, that is able to perform many tasks such as: reading values from a sensor, requesting specific values from other PLCs, and driving an actuator. If the ICS can be divided into stages, then each PLC typically controls one of these stages.

Sensors and actuators. Those components interface with the physical process, and they are either directly connected, or indirectly connected, via remote input/output units (RIOs) or PLCs, to the network.

Network Devices. An ICS uses different types of network devices. Industrial switches and firewalls are deployed to segment the network into layers (e. g. DMZ, and control

network). Gateway devices are used to translate one protocol into another (e. g. Modbus into Modbus/TCP). Remote Terminal Units (RTUs) are deployed on the filed to collect and send data back to the SCADA system. We also note that industrial Ethernet switches are often focused on electrical reliability, rather than IP-layer functionality (e.g. the Rockwell Automation Stratix 5900 switch).

Network Topology. Traditionally, ICS follows standard like RS-232 to connect together different components. Additionally, alternative field bus schemes, such as RS-485 and PROFIBUS, have been used. In specific situations where reliability is a major concern, ring [40] topologies are widely used in practise due to ease of deployment, and single-point-of-failure tolerance with very low reaction time.

Industrial protocols. In recent years, ICS networks are transitioning to traditional ICT technology like Ethernet (IEEE 802.3), and TCP/IP. However, the need for reliability, and interoperability with existing equipment led to the development of customized industrial protocols, such as Ethernet rings, and EtherNet/IP (ENIP) [129]. We now discuss the main ENIP features as an illustrative example of an industrial protocol. ENIP is a Real Time Ethernet (RTE) field bus based on TCP/IP, with a custom application layer designed to meet typical industrial communications requirements, like time constraints, and packet determinism. Technically, ENIP is an Ethernet based implementation of the Common Industrial Protocol, and it is defined in the IEC 61784-1 standard [63]. A PLC uses CIP messages to obtain sensor readings, to query another component, to set configuration options, to update its firmware, and even download a new control logic. In that model, sensor readings and control values are represented by tags, that are like variable names in a programming languages. CIP uses a request-response model, and such requests can operate on tags, and on the meta-data associated with the tags.

Topology layers. ICS networks typically are layered in different zones (more details in [124, 142]). On the lowest layer, controller devices are connected to sensors and actuators, or to remote input/output (RIO) devices, capable of converting raw sensors and actuators signals into Ethernet-based packets. The next layer will connect together the controller devices, and the additional devices such as: Human-Machine-Interface (HMI), engineering workstation, and historian server. For simplicity, all these devices are often kept in the same IP-layer sub-network, although more complex topologies are possible.

3.2.2 ICS Honeypots

A honeypot is a system *intended* to be probed, attacked and compromised [165]. Historically, the idea behind the development of modern honeypots comes from the nineties, where skilled computer programmers and system administrators were playing in real time with the attackers, to gain information about their targets, techniques, exploited vulnerabilities and ultimately for fun [37, 167].

Honeypots can be classified by means of different orthogonal features. *Real* honeypots use real physical devices to replicate the target system. They are the most realistic solution, however in the ICS domain their deployment is too expensive in terms of money, maintenance and space. On the other hand, *virtual* honeypots utilize virtualization technologies, such as PLC emulators, to reproduce a system, and they offer a

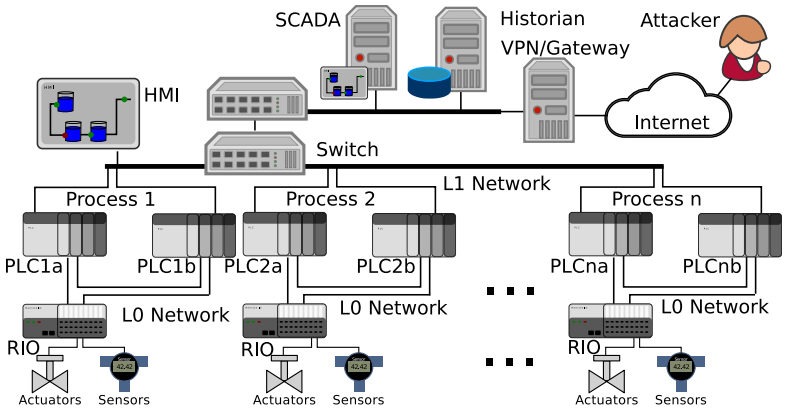


FIGURE 3.1: Example local network topology of a plant control network.

compact, and low-cost solution. Hybrid honeypots include a mixture of virtual and real devices, and they might be an interesting cost-effective solution for the ICS domain.

One of the core aspect of an honeypot is its level of realism. *Low-interaction* honeypots simulate only specific systems services (such as a telnet daemon), providing a narrow attack surface. Indeed they are easy to develop, configure, and secure against the attacker. However, the effectiveness of a low-interaction honeypot in the ICS domain is questionable, mainly because the ultimate targets of an ICS attack are the physical process and the ICS devices, and these parts are not simulated by low-interaction honeypots. In contrast, *high-interaction* honeypots use real services running on real Operating Systems, such as a webserver running on Linux listening to port 80, or simulate the services and the relevant parts of an Operating System. High-interaction honeypots provides a realistic environment for the attacker, a large attack surface, and they are tricky to implement, and secure against a motivated attacker. To the best of our knowledge, in the context of ICS honeypot there is no standard definition regarding high-interaction honeypots. The chapter defines an high-interaction ICS honeypot as an honeypot able to simulate both the physical process, and the ICS devices' control logic, and to emulate the ICS network using industrial protocol stacks, and network topologies.

Honeypots have different roles with respect to the attacker. *Server-based* honeypots expose, over an insecure channel, a number of vulnerable services, that are passively listen to well-known ports. In simple words, a server-based honeypot is passively waiting to be attacked. In contrast, a *client-based* honeypot acts as vulnerable client application, such as a web browser, and it actively looks for an attack from a malicious webserver.

Honeypots are used in different contexts. *Research* honeypots implement well-known vulnerabilities to lure attackers, and to study their behaviours, or (less often) for educational and security training purposes. *Production* honeypots are supposed to be more secure, and they are deployed to defend a system against an attacker. In the best case, the production honeypot will prevent the attacker to sabotage the real system, in the average case it will slow-down the attack, and hopefully will increase the attacker frustration, and in the worst case it will help the attacker to complete his job.

It is important to emphasize that ICS honeypots present additional requirements

compared to traditional ones, most importantly *time* and *determinism* constraints. An ICS device has to complete a sequence of tasks within a critical time interval, and in a precise order, that's way it uses a Real Time Operating System with a deterministic scheduler. In the same manner, ICS packets are sent over the network with a specific order, and they had to reach their destinations within a time period, that's why industrial protocols, such as EtherNet/IP, are extended with special application layer features to address these requirements. An ICS honeypot has to take into account these factors with great care, otherwise the attacker can easily detect the honeypot with simple tests.

There are many academic and industrial projects involving honeypots. In the domain of traditional network security we have *honeynets*, that are networked honeypots able to communicate among themselves in a NIDS fashion [78]. In the context of ICS and SCADA the most well known (still active) project is Conpot [173], an open-source ICS/SCADA honeypot, that is part of a large-scale project called The Honeynet Project [166].

3.2.3 MiniCPS Framework

MiniCPS [10] is a toolkit for security research on Industrial Control System (ICS) security. It builds on top of a lightweight Linux network emulator called Mininet [105], and it extends its application to the ICS domain.

MiniCPS combines network emulation, physical process simulation, and ICS devices simulation to build a real-time, ICS simulation in-a-Box. It is a framework written in Python, and it provides an high-level, object-oriented, public API. MiniCPS is developed using modern and agile techniques, like distributed source version control, test-driven development, build and documentation automation, and it is free and open source (MIT license) [6].

In this work, we propose to extend MiniCPS in the context of ICS honeypots. Those honeypots traditionally lack a realistic network sub-systems or focus on the simulation of a single ICS device, like a PLC. With the help of MiniCPS, we can reproduce the exact ICS network topology with PLCs, HMI, middle boxes, etc.. We can use the same network configuration as the real ICS, to take care of the attacker host enumeration, and fingerprinting phase (e. g. same IP, MAC, and net masks). From the emulated network the attacker may discover real ICS services, listening to standard ports, such as ssh or VPN servers. Furthermore, MiniCPS supports link shaping, meaning that each host can be configured with a custom link bandwidth, packet loss rate, and time latency.

MiniCPS's public API can reduce the honeypot's development time, and increase the portability of the developed code across different simulation experiments, involving different physical processes and industrial protocols. The public API is built against four core methods: `set`, `get`, `send` and `receive`. Each device in the simulated ICS inherits (a subset of) these methods according to its functionality. For example, a PLC is able to `get` (read) a sensor value, and `set` (write) an actuator command, additionally a PLC can `send` (serve) a packet over the wire, or `receive` (request) a packet from the wire.

3.3 High-Interaction, Virtual ICS Honeypot Design

3.3.1 ICS Honeypots

A honeypot is a system *intended* to be probed, attacked and compromised [165]. Historically, the idea behind the development of modern honeypots comes from the nineties, where skilled computer programmers and system administrators were playing in real time with the attackers, to gain information about their targets, techniques, exploited vulnerabilities and ultimately for fun [37, 167].

Honeypots can be classified by means of different orthogonal features. *Real* honeypots use real physical devices to replicate the target system. They are the most realistic solution, however in the ICS domain their deployment is too expensive in terms of money, maintenance and space. On the other hand, *virtual* honeypots utilize virtualization technologies, such as PLC emulators, to reproduce a system, and they offer a compact, and low-cost solution. Hybrid honeypots include a mixture of virtual and real devices, and they might be an interesting cost-effective solution for the ICS domain.

One of the core aspect of an honeypot is its level of realism. *Low-interaction* honeypots simulate only specific systems services (such as a telnet daemon), providing a narrow attack surface. Indeed they are easy to develop, configure, and secure against the attacker. However, the effectiveness of a low-interaction honeypot in the ICS domain is questionable, mainly because the ultimate targets of an ICS attack are the physical process and the ICS devices, and these parts are not simulated by low-interaction honeypots. In contrast, *high-interaction* honeypots use real services running on real Operating Systems, such as a webserver running on Linux listening to port 80, or simulate the services and the relevant parts of an Operating System. High-interaction honeypots provides a realistic environment for the attacker, a large attack surface, and they are tricky to implement, and secure against a motivated attacker. To the best of our knowledge, in the context of ICS honeypot there is no standard definition regarding high-interaction honeypots. The chapter defines an high-interaction ICS honeypot as an honeypot able to simulate both the physical process, and the ICS devices' control logic, and to emulate the ICS network using industrial protocol stacks, and network topologies.

Honeypots have different roles with respect to the attacker. *Server-based* honeypots expose, over an insecure channel, a number of vulnerable services, that are passively listen to well-known ports. In simple words, a server-based honeypot is passively waiting to be attacked. In contrast, a *client-based* honeypot acts as vulnerable client application, such as a web browser, and it actively looks for an attack from a malicious webserver.

Honeypots are used in different contexts. *Research* honeypots implement well-known vulnerabilities to lure attackers, and to study their behaviours, or (less often) for educational and security training purposes. *Production* honeypots are supposed to be more secure, and they are deployed to defend a system against an attacker. In the best case, the production honeypot will prevent the attacker to sabotage the real system, in the average case it will slow-down the attack, and hopefully will increase the attacker frustration, and in the worst case it will help the attacker to complete his job.

It is important to emphasize that ICS honeypots present additional requirements compared to traditional ones, most importantly *time* and *determinism* constraints. An

ICS device has to complete a sequence of tasks within a critical time interval, and in a precise order, that's way it uses a Real Time Operating System with a deterministic scheduler. In the same manner, ICS packets are sent over the network with a specific order, and they had to reach their destinations within a time period, that's why industrial protocols, such as EtherNet/IP, are extended with special application layer features to address these requirements. An ICS honeypot has to take into account these factors with great care, otherwise the attacker can easily detect the honeypot with simple tests.

There are many academic and industrial projects involving honeypots. In the domain of traditional network security we have *honeynets*, that are networked honeypots able to communicate among themselves in a NIDS fashion [78]. In the context of ICS and SCADA the most well known (still active) project is Conpot [173], an open-source ICS/SCADA honeypot, that is part of a large-scale project called The Honeynet Project [166].

3.3.2 Problem Statement

In this work, we address the problem of designing an ICS honeypot with the following requirements:

- Realistic, with multiple services supported.
- Low cost, in terms of hardware, software, and deployment time.
- Reconfigurable, to allow extensibility, scalability, and secure maintenance.
- Targeting the ICS domain, dealing with physical processes, physical devices, industrial protocols, time, and determinism constraints.
- Usable both for research, and production.

To the best of our knowledge, there exists no related work that presents a solution able to satisfy the outlined requirements. Given the ICS honeypot classification criteria, and related tradeoff presented in Section 3.3.1, we are proposing the design of a *virtual*, *high-interaction*, *server-based* ICS honeypot, to solve our problem. In the remaining part of the section we will present the reference attacker model, and the related system architecture.

3.3.3 Attacker Model

In this section we present the reference attacker model. We assume that attacker reaches the honeypot over the Internet, e. g. finding an honeypot's Internet facing device using general purpose search engines, such as Google, or more targeted ones such as Shodan [118]. Once connected to the ICS internal network, the attacker is able to fingerprint the target ICS system, using tools such as `nmap`, and `xprobe2`. As result, the attacker is able to obtain basic system information, such as the number of devices, their addresses, their ports status, and the type of industrial protocol.

We assume that the attacker has a basic knowledge about not-well documented industrial protocols, such as EtherNet/IP, and extensive knowledge about well documented ones, such as Modbus, and DNP3. The attacker may also be familiar with the underlying physical process, and control logic.

We assume that the attacker connects to the honeypot only through the intentionally vulnerable interfaces, and that the provided interfaces define the initial attacker surface. For example, if the attacker connects to a VPN server, using default credentials, then he can only interact over the network. Nothing prevents the attacker from escalating his privileges within the honeypot, for example once connected to the internal network, the attacker may discover an intentionally vulnerable gateway device, and get a root shell on that device.

We assume that the attacker interacts with the physical processes, and the ICS devices both as a fair, and malicious device. For example, the attacker as a malicious device may send malformed packet, unauthorized commands, perform Man-in-the-Middle attacks (both passive and active), and try to Denial-of-Service the honeypot.

We are limiting the attacker model to what we think is a reasonable scenario. We understand that there are more powerful attacker models, such as the ones profiling a disgruntled employee or an insider threat, however we reserve the option to extend the presented honeypot design, and implementation to deal with these kind of attacks in future work.

3.3.4 System Architecture

The main contribution of the chapter is the design, implementation, and evaluation of a realistic, low-cost, and reconfigurable honeypot targeted to ICS. In this section we present our design points according to the requirements presented in Section 3.3.2, and the attacker model presented in Section 3.3.3.

We classify the presented honeypot as follows:

- Virtual (lightweight virtualization).
- High-interaction.
- Server-based.
- Targets ICS requirements.
- Research and Production usage.

The use of virtualization allows us to implement a low-cost solution, that is easy to configure, reproduce, deploy, and maintain. High interaction is crucial to support multiple services and to keep the attacker busy as long as possible. Our honeypot is server-based, because it has to expose realistic services, listening on standard ports that are accessible from outside the ICS internal network perimeter. We envision that our honeypot could be used both in research and production environments. Researchers could use the honeypot system to learn about novel threats in the wild, while plant operators could use the honeypot system to detect specific threats targeted to their system, or mitigate ongoing attacks.

Figure 3.2 shows an high-level comparison between the presented honeypot architecture and a real ICS architecture. As dictated by the attacker model, our attacker comes over the Internet, and he can access the fake ICS internal network using two different vulnerable interfaces that he may discover during the attack reconnaissance

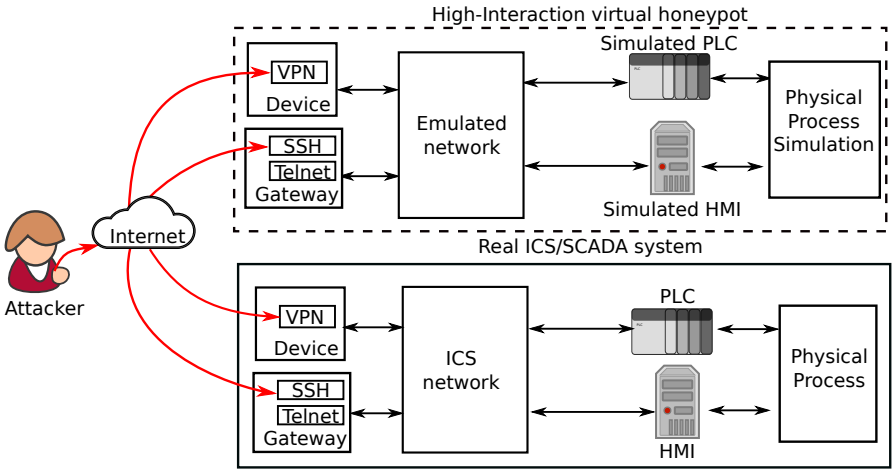


FIGURE 3.2: High-Interaction Virtual ICS Honeypot vs. Real ICS Architecture.

phase. The first interface will give the attacker access to the honeypot over the network, in the figure we are using a vulnerable VPN server as an example. The second interface will give the attacker a command line interface on an ICS device connected to the internal network, in Figure 3.2 we are using a vulnerable gateway device as an example.

An emulated network enables us to reproduce the same network topology as the real ICS, with the same number of hosts, addresses, and link characteristics. The emulated hosts send packet over the virtual network using real protocol stacks (e. g. EtherNet/IP or ARP). A set of simulated devices reproduces the control logic of the ICS system, and a physical process simulation mimics the real physical process. With those modular settings, we can separate the individual device control logics, and the physical process simulation in different sub-systems, allowing to reuse the blocks according to the honeypot initial configuration.

The ICS block is represented with dashed lines, to underline the fact that the proposed design *physically* separate the honeypot network from the real ICS one. In contrast, traditional honeypots are deployed inside the internal network, using unallocated IP addresses, and their separation from the real system is logical, typically by means of a firewall, a router or an ARP proxy. The physical separation between the honeypot and the real ICS provides an additional layer of security for free, meaning that an attacker who gained (privileged) access to the honeypot is not connected to the real ICS network, but to a virtualized emulated replica.

We explicitly focus on the design, and implementation of the core honeypot functionalities such as physical layer and network layer interaction, or data collection. In particular, we are not focusing on the data post-processing and analysis, and we refer to existing frameworks such as [138].

3.3.5 Qualitative Metrics

We stated that *realism* is one of our goals. Our assumption is that realism is required to attract interesting attackers. In particular, more knowledgeable attackers might recognize that they are interacting with a honeypot more quickly if the realism of the honeypot is lower. In practise, it can be hard to measure realism in a quantitative way. In the following, we propose some qualitative metrics to determine to which degree our honeypot represents a realistic system faithfully. To the best of our knowledge, there do not exist common metrics for such an evaluation so far.

We will use the following metrics later in the evaluation section to specify a summary of our honeypot prototype capabilities (e. g. honeypot provides feature 1, and does not provide feature 2). The following metrics are complementary to the proposed attacker model, and to the set of already presented requirements. We decided to divide the metrics into two categories: *network* and *physical*, and each category into subcategories.

Our network metrics:

- Network Parameters: IP, MAC and netmask addresses are identical to a real systems.
- Link Shaping: average packet loss, delay, and bandwidth can be set to comparable values as found in real systems.
- Infrastructure: The network topology is matching the real system exactly.
- Protocol: communications between devices in the honeypot use standard-compliant implementations of industrial, and common protocols (e. g. ARP, HTTP, DHCP).
- Advanced traffic properties: perfect sequence of messages and delay, matching a real system identically.

Our physical metrics:

- Process: The physical process simulation uses a realistic mathematical model of the process (with time steps < 1 minute).
- Devices: The honeypot provides real time simulation of sensor readings, actuator driving, and control logic.
- Human operator: the honeynet allows to simulate interactions of human operations.
- Advanced process: Simulation fast state change (e. g. transient, time steps < 1 s).

3.4 Honeypot Implementation with MiniCPS

Our honeypot implementation is based on the MiniCPS framework described in Chapter 2. To the best of our knowledge, the presented honeypot implementation is the first academic work targeting EtherNet/IP based ICS, the first ICS virtual honeypot that is high-interactive without the use of full virtualization technologies, such as a network

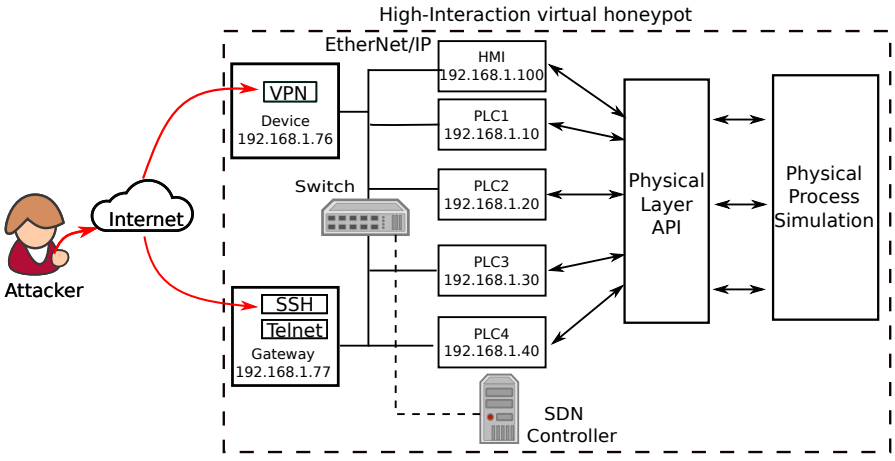


FIGURE 3.3: ICS Honeypot Implementation Block Scheme.

of virtual machines, and the first ICS honeypot that can be managed with a Software-Defined Network controller.

Figure 3.3 presents the basic building blocks of our implementation, using EtherNet/IP as the reference industrial protocol. The vulnerable, Internet-facing devices are connected to the internal network, and they are the attacker's baits. A virtual switch is distributing EtherNet/IP traffic in the internal network, enabling ARP poisoning attacks, packet sniffing, and malicious command delivery. Every simulated host is connected to the virtual network, and it is exposing real services. For example, PLC1 may expose an EtherNet/IP server for explicit messaging, listening on standard TCP port 44818, that is addressable with realistic tag names, and pre-loaded with realistic tag values. Another example is a Human Machine Interface (HMI), that exposes an HTTP configuration interface through a webserver listening on port 80.

We now provide details on the implementation of the vulnerable VPN endpoint, the vulnerable gateway device, the simulated ICS devices, the network emulation, and the data collection subsystem. Finally, we describe additional benefits discovered during implementation time.

3.4.1 Vulnerable VPN Endpoint

Virtual Private Network (VPN) are widely used in ICS network to establish a secure channel between a host located outside the control network, and a network interface inside the control network.

Our target hardware platform is an Allen-Bradley Stratix 5900 Router, with firewall capabilities. The target device runs an IPv4, OpenConnect (Cisco) VPN server, reachable from the Internet with weak credentials: user is `admin`, and password is `admin`. Given this vulnerable VPN server, the attacker is able to get an IP in the internal network, and interact with the honeypot through the virtual network interface associated with that IP.

We support the OpenConnect VPN server using one of its open source implementations: `ocserv`. Our emulated network has a dedicated firewall host with IP `192.168.1.76` that is listening on default port 443.

3.4.2 Vulnerable Gateway Device

Gateway devices are used in ICS control network to translate industrial protocols, such as back and forth from Modbus to Modbus/TCP, and extend the inter-device communication capabilities of the whole ICS (similar to NAT).

Our target hardware platform is Moxa OnCell IP gateway, that is able to connect to the testbed over cellular networks using different technologies such as: GPRS, EDGE, UMTS, and HSDPA. The target device has two configuration ports open: a telnet server is listening on port 23, and a ssh server is listening on port 22. The ssh service is configured with weak credentials: username is `admin` and password is `admin`. The telnet service is configured with plaintext-based unencrypted authentication, with the same weak credentials as the ssh server. Given such a configuration, the attacker is able to get a command shell on the gateway device, that is directly connected to ICS internal network.

We support ssh and telnet servers through `sshd`, and `telnetd`. Our emulated network has a dedicated 3G Gateway host, with IP `192.168.1.77`, that is listening on port 22 and port 23. The honeypot shell is chrooted, and the fake file system mimics the one of the gateway device. In the ssh case, the chroot jail is specified directly in the sshd's server configuration file, taking advantage of OpenSSH's convenient `ChrootDirectory` feature [46].

3.4.3 Network Emulation

The network emulation, and the virtual network hosts isolation is implemented by Mininet using a low-level feature of the Linux kernel, called container-based virtualization. Container-based virtualization takes advantage of Linux network namespaces, and virtual Ethernet links, known as veth, to isolate subsets of processes. Each collection of processes is called a container, and it has a complete virtualized Linux network stack associated (e. g. IP, ARP, and route tables). Each container interface is connected to the software switch's virtual interface through a veth. The net effect is an emulated virtual network, this is the reason why the proposed honeypot runs *in-a-Box*.

The link shaping feature is implemented using another low level Linux kernel functionality that can be accessed through the `tc` program. Tc allows to monitor, and manipulate the network traffic control setting for each active network interface. Indeed, it is easy for use to set custom bandwidths, delays, and packet loss for each container in our honeypot.

Finally, the proposed network emulation implementation is able to run any (industrial) protocol stack available for Linux. The chapter focuses on EtherNet/IP (ENIP), a modern object-oriented industrial protocol. ENIP is supported through the `cpppo` Python module [104].

3.4.4 Physical Process and Devices Simulation

The honeypot is simulating a water treatment physical process, an HMI, and four PLCs using a collection of python scripts. The PLCs logic mirrors the one described in the water treatment testbed operational manual, with the same control flow acting on real

tag names, values, and types. Interlocks are simulated as well: for example, PLC1's logic depends upon values stored on PLC2, and PLC3.

The physical process simulation script simulates only the hydraulic part of the system, in real-time. Technically, each water tank has an inflow pipe, and an outflow pipe, both are modeled according to the equation of continuity from the domain of hydraulics (pressurized liquids). Where present, a drain orefice is modeled using the Bernoulli's principle for the trajectories [126].

allows us to parametrize the simulation time of a water treatment simulation. We are not using this feature in the honeypot because the attacker may realize that the response time is too fast compared to the real water treatment. However, we used this feature during other types of experiments (e. g. Man-in-the-Middle attacks) to generate data faster than the usual. It is important to note that the speed-up factor is bounded by the capability of the Linux kernel scheduler to manage concurrent processes. In our case, the speed requirements are not very high as the physical process of the simulated water treatment system is relatively slow.

3.4.5 Data Collection Subsystem

The data collection subsystem involves different types of data acquisition, that depend upon the attacker activities inside the honeypot. In case of the vulnerable gateway device, that is providing a shell to the attacker, we use standard shell logging techniques: a log file is storing a set of records, and each record contains the timestamp, the username, and the issued command. We are monitoring every user to deal with an attacker able to escalate honeypot's privileges. The log file is periodically copied to a safe location, outside the honeypot.

For the vulnerable VPN server we use standard network traffic logging techniques. Multiple `tcpdump` daemons are attached to the vulnerable network interfaces, and they are generating pcap capture files. Eventually, these files can be post-processed using more sophisticated network analysis tools, such as `wireshark`.

Additionally, the honeypot includes a software keylogger program, running with root privileges, and masked from user-space memory. The keylogger is generating a log file with all the entered keystrokes, and it is able to deal with more motivated attackers, that for example might use obfuscation, and encryption techniques to transfer their malicious payloads. This is a key difference between the effectiveness of a honeypot versus a (signature-based) NIDS. The NIDS is not able to recognize an encrypted malware sent by the attacker to the target machine, in contrast the honeypot keylogger will log the decryption phase of the malware on the target machine, detecting the attack, and providing precious information about the attacker's tactics.

3.4.6 Implementation Benefits and Risks

It is important to notice that a good implementation yields additional benefits, that typically are not captured during the design phase. In this section we will present some of the additional benefits provided by the MiniCPS framework.

In the problem statement, we target an honeypot usable both for research, and production. MiniCPS is based on lightweight virtualization, and allows us to configure the security level of the honeypot *parametrically*, at start-up time. By security level, we

mean the amount of vulnerabilities deliberately included in our services, indeed a research honeypot will be pre-configured with a medium, or low security level, and a production honeypot will be pre-configured with a high security level. For example the latest version of an ssh server is installed in the high-security honeypot, and an older vulnerable version is installed in the low/medium-security honeypot. One can even think to patch a service, introducing trivial vulnerabilities, like default ssh admin credentials, for the low-security honeypot. Additionally, if the attacker manages to exploit the high-security production honeypot, we will most probably discover a new vulnerability, or a new exploitation technique.

MiniCPS allows to extend our honeypot from a virtual to an *hybrid* configuration. As discussed in Section 3.3.1, an hybrid honeypot presents a mixture of real and virtual devices and it is a cost-effective solution in the ICS context. Technically, a hybrid honeypot can be categorized as an hardware-in-the-loop simulation, and this setting increase the level of realism of the honeypot and also the complexity of its design and implementation. We have access to spare PLCs in our lab, and in the future we plan to perform experiments with a hybrid honeypot and compare the results against our virtual honeypot.

MiniCPS allows to connect different ICS instances together and to a real network. This feature enables the possibility to deploy a virtual ICS *honeynet*, that is a network of (virtual) ICS honeypots that can be accessed over the Internet and can collaborate to manage more advanced attack scenarios. For example, an honeynet might be able to manage multiple attackers attacking at the same time, redirecting each attack to a dedicated honeypot instance.

Finally, MiniCPS supports *Software Defined Network (SDN)* development natively [60]. In the default setting, the honeypot SDN controller is idle and it is not visible by the attacker. Nothing prevents us to actively using the honeypot SDN controller. For example, we may develop specific ICS control plane logics, able to extend the functionalities of our honeypot, such as data analytics, deep packet inspection, and detection mechanisms.

It is worth mentioning that the following implementation introduces the typical risks of a high-interaction honeypot. For example, if the attacker is able to escalate privileges inside the honeypot, we consider that honeypot useless (e. g. the attacker may send false values to the data collection subsystem). The attacker may also be able to penetrate the honeypot using a side channel, but this scenario is not captured by the presented attacker model, indeed is out of scope for this chapter. Finally, we understand that it is really difficult to protect the honeypot against a knowledgeable attacker, but still the physical isolation between the honeypot, and the real ICS system will protect the real ICS anyway.

3.5 Evaluation

3.5.1 Evaluation Context

In this section, we present a preliminary evaluation of our honeypot in the context of a Capture-The-Flag (CTF) competition. The competition was a part of broader ICS security event, called SWaT Security Showdown (S3), and hosted by Singapore University

of Technology and Design (SUTD) in July 2016.

CTF are educational cyber-security competitions, hosted, online and offline, by Universities, private companies, and non-profit organizations. There are two standard types of CTF: jeopardy-style and attack/defense. A jeopardy-style CTF involves a set of challenges, divided by category (e. g. reversing, exploiting, and cryptography), and each challenge is presented with a short description, a number of clues, and an amount of reward points. Each team scores points solving these challenges, and the solution usually consists in a message to be entered in the CTF's scoring system. An attack/defense CTF involves a set of machines running vulnerable services, given to the participating teams, and connected on the same LAN. To score points, each team has both, to defend its services from the other teams (e. g. by patching a vulnerable service), and to attack the services protected by the contender teams (e. g. by login as admin on a webserver). Usually the given machines settings are not known a-priori by the teams, this is a key point to augment the learning experience of the participants. In both cases, the CTF stars and ends at prescribed time, and the team that scores most points wins.

SWaT Security Showdown's CTF involved different instances of our honeypot, one for each participating team, pre-loaded with different CTF challenges. In particular, the honeypots were simulating the hydraulic part of a subset of the Secure Water Treatment (SWaT) testbed. SWaT is a state-of-the-art water treatment testbed located at SUTD, consisting of six stages managed by six control devices. There are many interesting details about the SWaT testbed, but they are out of scope, indeed they are omitted from the discussion. To understand the remaining part of this section, it is sufficient to know that: each honeypot included several simulated components, in particular: four Programmable Logic Controllers (PLCs), a Human Machine Interface (HMI), two water tanks (named Raw water tank, and Ultra-filtration tank), and a vulnerable gateway device, that EtherNet/IP was the spoken industrial protocol, and that we connected the simulated devices in a star topology, recreating one layer of the SWaT control network. Notice that, we exposed only one vulnerable interface over the Internet, because S3's CTF already assumed that the attacker knew the (vulnerable) ICS entry point.

3.5.2 CTF and Honeypot Setup

In this section we will provide a brief description abut the CTF's scoring system, and the honeypots' setup,

The CTF scoring system was implemented as a web application (webapp), using the flask Python framework [153]. The webapp authentication was based on username and passwords, and we used Let's Encrypt to encrypt the webapp's ingoing and outgoing traffic, via HTTPS [91]. Each challenge could be solved entering the flag using an HTML form field. We decided to log all the scoreboard's user input to understand common errors, and detect brute-force attempts.

The honeypot setup was the most complex task. For network security reasons, we decided to run all the honeypots "in the cloud", using Amazon Web Services' Elastic Compute Cloud (AWS EC2) instances. Each honeypot ran on a single Linux kernel, using an m3-type EC2 instance. We set up a single instance, tested it, and then replicated it, with minimal reconfiguration issues, to accomodate six teams.

We decided to use ssh as the vulnerable service on the gateway device, and we assumed that the attacker already had obtained the credential to access it. That is why we distributed a (different) private key for each team to login inside the honeypot as the `attacker` user. Each instance was running two different ssh servers, with different configurations. One server was used by us to access the virtual machine, and manage the honeypot. The other server was running inside a Linux container, and the attacker was chrooted to protect the honeypot file system and running processes. It is important to notice that, both servers were running on port 22 on their respective networks, however we had to use port forwarding (from port 2222 of the control network to port 22 of the honeypot network), to give the attacker a ssh connection inside the honeypot network.

3.5.3 Challenges Descriptions

This section describes in detail the five challenges involving our honeypot, mimicking a subset of the SWaT's testbed, more information about the settings may be found in Section 3.5.1. The challenges' design followed common best practices of jeopardy-style CTF: challenges were presented in increasing order of difficulty, and the solution of challenge x was providing useful knowledge to solve challenge $x + 1$.

Network warm up. The first challenge involves a basic understanding of the SWaT network topology. The challenge description is: "Can you eavesdrop what PLC2 has to say to PLC3?". The goal of that challenge is to perform a passive Man-in-the-Middle attack between PLC2 and, PLC3.

EtherNet/IP warm up. The second challenge involves some understanding of the EtherNet/IP industrial protocol. The challenge description is: "`cpppo` can be used in the testbed to communicate using the EtherNet/IP protocol. Can you read the README:2 tag." In this case, the attacker has to understand which PLC stores the README:2 tag, know its IP, and know how to query an EtherNet/IP server. Python's cpppo module is suggested because it is an easy to use library to do the job.

Overflow the Raw water tank. The third challenge involves a basic understanding of a water treatment industrial control system. The challenge description is: "PLC1 is controlling the raw water tank. It is reading the water level value addressed by the tag LIT101:1. PLC1 is controlling a motorized input valve, addressed by MV101:1, that can be turned ON/OFF using respectively 1/2. PLC1 is also controlling an output pump, addressed by P101:1, that can be turned ON/OFF using respectively 1/2. The maximum tank level in m from the ground is 1.2. The goal is to overflow the raw water tank." In this case, the attacker has to understand how to overflow a tank, based on a provided list of sensors and actuators.

Denial of Service HMI. The fourth challenge involves a basic understanding of Denial-of-Service (DoS) attacks. The challenge description is: "The HMI (set to manual mode) is constantly sending to PLC3 the keep alive value 2. You can access this value using the MITM:3 tag stored in PLC3. Can you change this value to 3? If you see that the tag value has a stable 3 value, wait a little bit to get the flag." In this case, the attacker has to find a way to disrupt the communications between HMI and PLC3. It is not sufficient for the attacker to write the value three in the MITM:3 tag.

Overflow the Ultra-filtration tank. The fifth challenge involves an advanced overflow attack on the ultra-filtration tank. The challenge description is: "The HMI (set to manual mode) is sending commands to PLC4. PLC4 is controlling the water level using MV301:3 and P301:3. Both can be turned ON/OFF using respectively 1/2. You can query LIT301:3 to discover the actual water level. The maximum tank level in m from the ground is 1.2. The goal is to overflow the ultra-filtration tank." In this case, the attacker could not reuse the same techniques used for the third challenges, and he has to find a smarter way to overflow the tank.

3.5.4 Challenges Results

In this section we present a number of interesting statistics gathered during the CTF event. We anonymized the names of participating teams. Team 6 is explicitly excluded from the analysis, because its members managed to exploit a side channel attack unrelated to the honeypot, which allowed them to bypass our honeypot challenges.

After the CTF event we post-processed the log files, and in Table 9.4 we present several interesting results. Only one team was able to solve all the challenges, and the average number of solved challenge per team was three. The majority of the teams used traditional reconnaissance tools such as: `nmap` and `ping`, expected attack techniques such as: Man-in-the-Middle attacks, and used `cpppo` for EtherNet/IP interaction, as suggested by us in the challenge description.

TABLE 3.1: CTF Results Summary.

Teams	Flags	Distinct Cmds	ExLOC	Rec/Att Tools	Most Used Tools*
Team 1	2	20	1074	3/1	{1, 2, 6, 8}
Team 2	5	30	2488	6/2	{1, 2, 3, 4, 5, 6, 7, 8}
Team 3	3	23	2045	5/2	{1, 2, 3, 4, 6, 7, 8}
Team 4	4	27	963	5/2	{1, 2, 3, 4, 6, 7, 8}
Team 5	1	3	52	1/0	{1}

ExLOC : Executed Lines Of Code
*{1: `ettercap`, 2: `nmap`, 3: `netstat`, 4: `tcpdump`, 5: `tshark`
6: `ifconfig`, 7: `cpppo`, 8: `ping`}

There are a number of lessons learnt by us during the CTF event, we will present two of them. Firstly, crash recovery management. It is important to implement an automated, and reliable honeypot's crash recovery sub-system, because an attacker may break the honeypot in (unexpected) ways. In that case, the honeypot has to shutdown gracefully, and restart after a reasonable time interval with the same settings. We experienced down-time issues because the attackers used simple bash scripts containing infinite while loops, resulting in a DoS of some of our honeypots. We were not able to automatically restart the targeted honeypots, and as result the offending teams had to wait a couple of hours before restarting to attack the system.

The second important lesson was related to side channel attacks mitigation. We believe that defending a system is generally more difficult than attacking it, because the attacker has to find one vulnerable hole, however the defender has to protect every holes (that he is aware of). During the CTF, we suffered a side channel attack on a

TABLE 3.2: Honeypot metrics evaluation summary.

Metric	By design	Implemented
Network		
IP, MAC and netmask	●	●
Packet loss	●	●
Packet delay	●	●
Bandwidth	●	●
Topology	●	●
Common protocols	●	●
Industrial protocol	●	◐
Advanced Traffic	●	◐
Physical		
Realistic math model	●	●
Sensor readings	●	●
Actuators driving	●	●
Control logic	●	●
Human operations	●	◐
Advanced Process	●	◐

Legend ●: full support, ◐: partial support.

machine that was not running on any of our honeypots. The attack disclosed (among other things) information about the event logistics, solutions to challenges, credentials for a web service. For future events, we will take into account every detail of the configuration process, and ensure that we protect sensible data with access control and delete unnecessary data from places accessible to the attacker. The last statement might sound trivial, but it can be tricky to implement in practice, especially when multiple people are configuring a complex virtualized system, running multiple services, connected over the Internet.

3.5.5 Evaluation using Qualitative Metrics

In Table 3.2, we present an evaluation summary of our honeypot prototype. As features, we use the metrics proposed in Section. 3.3.5. The table distinguishes between features that are enabled by the honeypot design, and the ones implemented in the evaluated prototype. For example, the honeypot design is capable of providing full industrial protocol support. However, if there is only a partial implementation of the protocol available for Linux (as in the case of EtherNet/IP) we have to indicate a partial support in the Implemented column.

Table 3.2 shows that the implemented honeypot satisfies all the basic metrics, and partially satisfies the more advanced ones (such the simulation of a human operation). Furthermore, the implemented honeypot supports different types of attack ranging from network attacks, such as: Man-in-the-Middle, port scanning, and service enumeration, to physical process related ones, like tank overflows and DoS attacks on devices.

3.6 Related work

It has been observed over the years Internet-facing ICS devices are vulnerable to cyber attacks and in respect to that security aspects of ICS devices have been discussed in [188, 189], in particular the author presents attack statistics and a robust attribution framework by deploying a honeypot architecture in the simulated virtualized ICS environment. We now review related work in detail, and position our work against it. To the best of our knowledge, our work and the *Conpot* share the most features. We summarize our findings in Table 3.3.

TABLE 3.3: Our Honeypot Features vs. Related Works.

Related Work	Real	Virtual	Low-Interaction	High-Interaction	Hybrid-Interaction	Research	Production	Live Project	Link Shaping	Light-weight	Emulated Network
Our work		●		●		●	●	●	●	●	●
Scott et al. [159]	●		●				●				
Buza et al. [29]		●		●			●				
Holczer et al. [85]		●		●			●				
Yin et al. [136]		●	●			●					
Conpot [173]		●	●			●	●	●	●		●
Provos et al. [146]		●	●			●	●				
Litchfield et al. [109]	●	●			●	●		●	●		●
Liljenstam et al. [107]		●				●			●		

Design and Implementation of ICS honeypot. In [159], Scott proposes a mapping and configuration of honeypot that is similar to our High-interaction Hybrid honeypot architecture in several ways. In general, for honeypots to work effectively, there are two major concerns. Firstly, it is needed to map the network attack surface of the target system, and choose which services to mimic in the honeypot. Secondly, The configuration and placement of the honeypot in the local network needs to ensure that controlling and monitoring of attacks can be performed and attacker activities can be logged by a secure channel.

PLC honeypot. On the topic of PLC honeypots, the authors of [29, 85] propose a high-interaction honeypot PLC solution for secure network design. The proposed implementation involves exploration and inspection of all the PLCs and the services (HTTP, HTTPS, ISO-TSAP, and SNMP) running on PLCs in a system that should be protected. Those services are then implemented and integrated in a Linux based Virtualized simulated environment acting as a honeypot. In contrast to our work, the authors do not consider any interactions with physical processes, and in general only network-based interactions with the honeypot (no shell or VPN access).

3.6.1 Honeypot frameworks

IoTPOT. In [136], a low interaction honeypot framework was proposed. The authors claim that telnet based attacks on IoT devices are increasing rapidly. The authors propose IoTPOT, a honeypot that emulates telnet interactions of IoT devices, in order to attract and analyze attacks against various IoT devices running on different CPU architectures such as ARM, MIPS, and PPC. In contrast to IoTPOT, our proposed honeypot is focusing more on ICS rather than IoT devices. Additionally, the proposed honeypot does not focus on a specific protocol, such as telnet and related attacks, but it deals with different industrial protocols and services, resulting in a much stronger attacker model, and much larger attacker surface.

Conpot. Conpot [173] is an open source project for industrial control system honeypots. The honeypots can be classified as low-interactive server side honeypots, that are implemented with following attributes in mind: a) Deployment of honeypot in network should be easy b) Modification and extension of it can be done easily. As Conpot is provided with a stack of several industrial protocols, it can interact with real devices such as HMI and PLCS, and is capable of emulating complex infrastructures.

In contrast to Conpot, our framework allows high-interaction honeypots. In addition, the Conpot project does not cover the host and network virtualization aspects of our system, or enable physical process simulation. Instead, Conpot is more related to providing libraries and tools to build simulated ICS components. Unlike our work, Conpot also provides analysis functionality of the communication and actions of the attacker. For the future, we believe that we can strengthen the device simulation aspects of our system with components from Conpot.

Honeyd. Honeyd [146] is an open source software framework to configure several virtual host/honeypots in an existing network. The honeypots can be configured with arbitrary services. The framework is useful for attack detection and to collect statistics about malware attack. Unfortunately, development of Honeyd seems to have stopped in 2013.

HoneyPhy. HoneyPhy [109] is physics-aware honeypot framework for Cyber-Physical Systems (CPS). The proposed work has a broader scope (CPS are a superset of ICS), and claims to provide realistic physical process, and devices simulation by means of a hybrid configuration. In contrast, our honeypot framework is purely virtual, and eventually can extended to a (more expensive) hybrid configuration.

3.6.2 Related ICS Simulation Frameworks

In [52], a framework similar to MiniCPS is proposed in the context of Software-Defined Networking (SDN), and network intrusion detection for ICS. In particular, the presented system also uses Mininet [105]. The authors do not discuss the use of Mininet framework in a honeypot setting, but they discuss about a virtual network layer built on the top of physical process, to monitor the network status with high level of granularity. Furthermore, they demonstrate how smart grids could be made resilient in catastrophic circumstances using SDN.

In [107], a simulator named RINSE (Real-time Immersive Network Simulation Environment for Network Security Exercises) is developed with similar intent of the proposed ICS honeypot. The simulator claims to provide realistic and scalable network

simulation by using multi-resolution traffic model, and routing protocols simulations. An extension of RINSE is proposed in [107], where a stronger attacker model is introduced. The new attacker's capabilities include: Denial-of-Service, computer worms, and similar large-scale attacks involving large numbers of hosts, and high intensity traffic. In contrast to RINSE, the proposed ICS honeypot does not simulate the network stack, but it uses network emulation to provide real packets and realistic network characteristics, such as delay, packet loss, and bandwidth. Furthermore, RINSE is developed as a training platform for network security, on the other hand the presented ICS honeypot is designed and implemented for both research and production purposes.

3.7 Conclusion

In this work, we presented the design of a virtual, high-interaction, server-based ICS honeypot, which aims to provide a realistic, cost-effective, and maintainable solution to observe and capture the activities of the attackers. Based on that design and the MiniCPS framework, we implemented parts of the SWaT testbed as honeypot.

To the best of our knowledge, the presented honeypot implementation is the first academic work targeting EtherNet/IP based ICS honeypots, the first ICS virtual honeypot that is high-interactive without the use of full virtualization technologies (such as a network of virtual machines) and the first ICS honeypot that can be managed with a Software-Defined Network controller.

We evaluated our implementation in the context of a capture-the-flag event targeted to ICS, called SWaT Security Showdown. During the event, six teams attacked six instances of our honeypot, producing interesting results. We were able to implement a realistic scenario, run multiple services, and generate realistic traffic over the virtual network. In the future, we plan to improve the crash management sub-system, the EtherNet/IP support, the logging capabilities over the network, and the SDN support.

Chapter 4

Gamifying ICS Security Training and Research: Design, Implementation, and Results of S3

Keywords: Gamification, CTF, Education, Research, ICS.

4.1 Introduction

Recently, it has been widely argued that one of the fundamental issues in securing industrial control systems (ICS) lies in the cultural differences between traditional IT security and ICS engineering [157, 112]. Therefore, education has been advocated as a means of bridging the gap between these cultures [111, 112]. However, recent surveys indicate that although general IT security education efforts have risen in ICS, there is still need for more targeted education combining both security and ICS specific knowledge [90].

Typically, those willing to do research on ICS security are facing severe problems, such as lack of understanding of a real ICS, and the inability to test (new) attacks and countermeasures in a realistic setup. ICS testbeds constitute a convenient environment to study ICS security, however their deployment is rare because of many (reasonable) costs, such as infrastructure and manpower costs [30, 184]. Another common issue in ICS security is resulting from the intrinsic inter-disciplinary nature of the subject. It is difficult to bring together people from different expertise domains, such as control theory, information security, and engineering.

In this work we propose a solution to the ICS security education problem, based on gamified security competitions. By education we mean both training of new ICS security professionals, and helping researchers to advance the state-of-the-art of ICS security. Our gamification idea evolves around four key points. Firstly, the competition has to be domain-specific (targeted education). Secondly it has to involve people from academia and industry possibly with different expertises (addresses the cultural differences). Thirdly, the contest has to be fun to play to motivate the participants (gamified). Finally it has to present interaction with real ICS components using real ICS tools (realistic attacks and countermeasures).

The result of our efforts is the *SWaT Security Showdown (S3)*, a Capture-The-Flag (CTF) targeted to industrial control systems security. This chapter focuses on the design, implementation and results from the first S3 edition of 2016. S3 was hosted by

our institution the Singapore University of Technology and Design. S3 is divided into two phases: an online training CTF, and a live attack-defense CTF. During the online phase the attackers participated in a Jeopardy-style CTF. The online CTF challenges included novel ICS-specific categories, involving for example real-time interactions with ICS simulations, and remote access and programming of real ICS devices. During the live CTF both the attacking and defending teams had access to our water distribution testbed (SWaT). They deployed a wide range of attacks, while two academic attack detection systems were in place.

We summarize our contributions as follows:

- We identify several issues that currently hinder industrial control systems security education and research.
- We propose a solution to address those issues, focusing on a gamified Capture-The-Flag (CTF) competition, using simulated and real ICS infrastructures.
- We present the design and implementation of the *SWaT Security Showdown (S3)* competition. S3 uses a combination of Jeopardy-style CTF and attack-defense CTF to provide a novel and hands-on learning experience for ICS security professionals.

This work organized as follows: in Section 4.2, we provide brief background on industrial control systems (ICS), the Secure Water Treatment testbed, and Capture-The-Flag events. In Section 4.3, we present the current challenges for ICS security education and research, our problem statement, and the design of S3. The details about S3 online and live phases are presented in Section 4.4 and Section 4.5. Related work is summarized in Section 4.6, and we conclude the chapter in Section 4.7.

4.2 Background

4.2.1 Industrial Control Systems Security

Industrial control systems (ICS) are autonomous systems composed of heterogeneous and interconnected devices. ICS are deployed to monitor and control different types of industrial processes, such as critical infrastructures (water distribution and treatment), and transportation systems (planes and railways).

ICS security is a major challenge for many reasons. Firstly, the complexity and diversity of devices involved in an ICS increases the attacker surface. For example, an attacker might attack the cyber-part, the physical-part or both parts of the ICS. Additionally, modern ICS are embracing standard Internet communication technologies, such as TCP/IP based industrial protocol, resulting in ICS that can be controlled (and attacked) from the Internet. Arguably, threats to ICS focus on impacting the physical world, instead of attacks on the confidentiality and the integrity of the information. As such, the damage by those attacks is expected to cause high financial and human costs due to destroyed property and decreased operational availability of commercial systems. Famous examples of high-impact attacks on ICS are the recent attack on the Ukraine power grid [32], the Stuxnet worm [59], and the attack on a wastewater treatment facility in Maroochy [163].

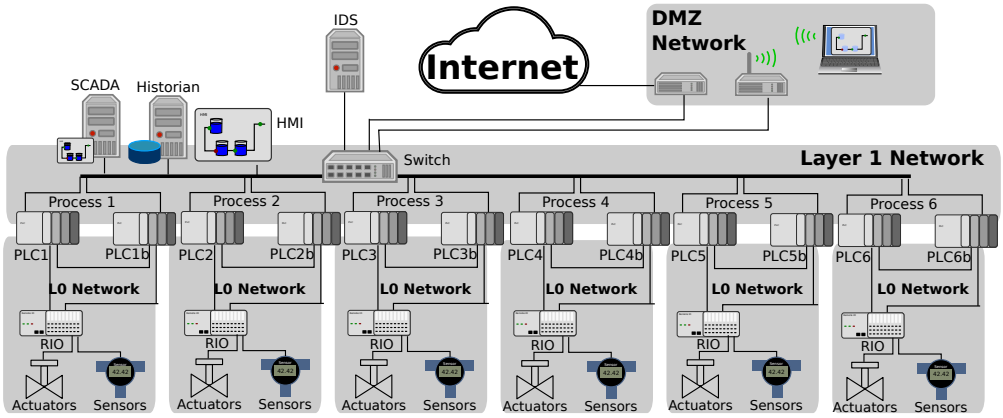


FIGURE 4.1: The Secure Water Treatment (SWaT) testbed architecture.

4.2.2 Secure Water Treatment (SWaT) Testbed

For the experimental part of this work we target the *Secure Water Treatment (SWaT)*. SWaT is a state-of-the-art water treatment testbed available at our institution since 2015 [119]. SWaT is composed of six stages and includes advanced filtering equipment such as: ultrafiltration and reverse osmosis sub-systems. We now briefly describe the six stages of SWaT:

1. *Supply and Storage* pumps raw water from the source to the Raw water tank.
2. *Pre-treatment* chemically treats raw water controlling electrical conductivity and pH.
3. *Ultrafiltration (UF) and backwash* purifies water using ultrafiltration membranes, collects ultra-filtrated water in the Ultra-filtration tank, and periodically cleans the UF membranes.
4. *De-Chlorination* chemically and/or physically (UV light) removes chlorine from ultra-filtrated water.
5. *Reverse Osmosis (RO)* purifies water using RO process, separates the result into permeate (purified) and concentrate (dirty) water.
6. *Permeate transfer and storage* store permeate water into the RO permeate tank.

Figure 4.1 shows a schematic view of SWaT architecture. Starting from the bottom we can see six gray boxes representing the six water treatment stages. Each stage involves two Programmable Logic Controllers (PLCs) configured in redundant mode, and a Remote Input-Output (RIO) device that interfaces the PLC with the sensors and actuators. The *field* networks (Layer 0) use an Ethernet ring topology. The rings are established and maintained using the device level ring (DLR) protocol. The data is exchanged using EtherNet/IP over UDP. Every PLC is connected to the *control* network (Layer 1). The control network has a star topology, and it includes the PLCs, a SCADA server, an HMI, and a historian server. Other network devices (e. g. in the DMZ network) access the SWaT control network through an industrial firewall. EtherNet/IP over TCP is used in the control network to carry data about commands, sensors, and actuators. EtherNet/IP is an object oriented industrial protocol. In particular, it is an implementation of the common industrial protocol (CIP) [129] on top of the TCP/IP protocol stack.

4.2.3 Capture-The-Flag (CTF) Events

Capture-The-Flag (CTF) events are cyber-security contests organized by universities, private companies and non-profit organizations. CTF competitions can be classified in two categories: *Jeopardy-style* and *attack-defense*. A Jeopardy-style CTF usually is hosted on the Web, and includes a set of tasks to be solved divided by categories (e. g. cryptography, exploitation and reverse engineering). Each task is presented with a description, a number of hints and an amount of reward points. The solution of a challenge comprises finding (or computing) a message (the flag) with a prescribed format, such as `ctf{foo-bar}`, and submitting it to the CTF scoring system. An attack-defense CTF, also called red team (the attackers) blue team (the defenders), is organized both offline and online. Each team is given an identical virtual machine containing some vulnerable services. The teams are connected on the same LAN, and their goal is both to have an high service runtime and to tamper with the services of the other teams. For example, finding and exploiting a vulnerable service has two benefits: it allows a team to patch its service to be more resilient to the attacks from the other teams, and to attack other teams vulnerable service. Both Jeopardy-style and attack-defense, CTF have time constraints (e. g. increase level of realism), and the team who scored most points wins the competition.

4.3 Gamifying Education and Research on ICS Security

We start this section by summarizing current challenge statements from academia and industry, and we leverage them to set the problem statement of this work. Then, we propose number of solution approaches. We focus on one of them, and discuss how it could be implemented.

4.3.1 ICS Current Security Challenges

In recent years, experts have argued extensively about the criticality of securing Industrial Control Systems (ICS)s. Many have pointed out that one fundamental challenge in achieving this task lies in cultural and educational differences between the fields of (traditional) information security and ICS security. According to Schoenmakers [157]: *"Differences in perspectives between IT and OT specialists can cause security issues for control systems. It is important for organizations to keep in mind that different values between groups can influence the perception of issues and solutions."*, which emphasizes the cultural clashes still existing between traditional IT security and ICS specialists.

Education and training have been advocated to bridge this gap, but there still work to do in this domain. Luijif [111] describes the security of ICS as a societal challenge, and recommends:*"Many of these challenges have to be overcome by both end-users, system integrators and ICS manufacturers at the long run: (...) proper education and workforce development"*.

Despite the problem of education being widely acknowledged, according to a recent report published by SANS Institute [90]: *"It is clear from our results that most of our respondents hold security certifications, but the largest number of these (52%) is not specific to control systems (...) IT security education is valuable, particularly with the converging technology trends, but it does not translate directly to ICS environments."*

In order to effectively improve the security of ICS it is thus crucial to educate researchers and practitioners such that they are able to understand the subtleties and domain-specific requirements and constraints of security and ICS. As recently pointed out by Luijif in [112]: *"(...) ICS and (office) IT have historically been managed by separate organizational units. ICS people do not consider their ICS to be IT. ICS are just monitoring and control functions integrated into the process being operated. ICS people lack cyber security education. The IT department, on the other hand, is unfamiliar with the peculiarities and limitations of ICS technology. They do not regard the control of processes to have any relationship with IT. Only a few people have the knowledge and experience to bridge both domains and define an integrated security approach. Organizations that have brought the personnel from these two diverse domains together have successfully bridged the gap and improved the mutual understanding of both their IT and ICS domains. Their security posture has risen considerably."*

4.3.2 Problem Statement

With the challenges from Section 4.3.1 as a high-level goal in mind, in the following we discuss the problem statement and the proposed solutions.

Based on the literature and our experience, we think that traditional IT (security) professionals need more information about the following topics:

- Common device classes, network topologies, and protocols used in ICS.
- Design methodologies best-practices and operational objectives in ICS.
- Physical processes specifications.
- Control theory models.

Furthermore, training for ICS (security) professionals can be beneficial in the areas of:

- Common "modern" Internet communication technologies (Ethernet, IP, TCP, UDP, NAT).
- Common security challenges and standard solutions (MitM attacks, TLS, firmware and software update schedules).
- Standardization and specifications related to security products.

Problem statement How can we create an impactful educational experience that addresses the aforementioned gaps?

4.3.3 Proposed Solution Approaches

We now propose a number of approaches to alleviate the outlined problems. We then present a solution that covers several of the proposed approaches.

1. A set of common use cases for ICS. In particular, the use cases would include a fictional or real physical process and details on the communication network topology.
2. Interactive ICS testbeds, that allow users to familiarize themselves with the control devices and protocols used and interact with the underlying physical process.
3. Practical training on ICS and information security.
4. Testing of security solutions through external parties, and standard certifications.

4.3.4 Capture the Flag

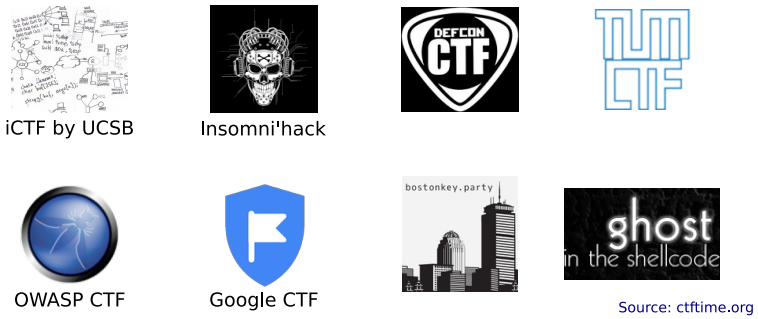


FIGURE 4.2: Popular CTF competitions.

Capture-The-Flag (CTF) are cyber-security contests organized worldwide by Universities, private companies and non-profit organizations. Figure 4.2 presents some of the most popular CTF such as DEFCON, iCTF, and Google CTF. CTF events can be classified as: *jeopardy-style* or *attack-defense*. A jeopardy-style CTF usually is hosted online and involves a set of tasks to be solved divided by categories, such as cryptography and reverse engineering. Each task is presented with a description, a number of hints and an amount of reward points. The solution of a challenge involves finding (or computing) a message (the flag) with a prescribed format, such as `CTF{my_flag}`, and submitting it to the CTF scoring system. An attack-defense CTF, also called Red (attacker) team/Blue (defender) team, is organized both offline and online where each team is given an identical virtual machine containing some vulnerable services. The teams are connected on the same LAN, and their goal is both to have a high service runtime and to penetrate the services of the other teams, e. g. finding and exploiting a vulnerable service has two benefits: it allows a team to patch a service to be more resilient against adversarial attacks, and to attack other teams vulnerable service. Both jeopardy-style and attack-defense, CTF have time constraints (realistic scenario) and the team who scores most points wins the competition. The presented work uses both online jeopardy-style and live attack-defense CTF styles to augment the learning experience.

Such events attract the attention of both industrial and academic teams and currently enjoy increasing popularity, as indicated by an established website in this community, listing CTF competitions worldwide [45]: this website lists 100 events being held worldwide, some of them with a long tradition such as the hacker-oriented DEFCON CTF [47] and the academic-oriented iCTF [38]. Of the 10,000 teams listed in CTF time in 2016, some are academic and others are composed of a heterogeneous mixture of security enthusiasts, many of them security professionals.

CTF-like gamified security competitions are expected to help the ICS security community in many ways [54, 147, 180]. A CTF is an hands-on learning experience and it can be used as an educational tool, research tool, and as an assessment tool. Ideally, both recruiters and candidates from academia and industry benefit participating in CTF events as they exercise key aspects of the ICS security domain such as knowledge of security (recent) threats, teamwork, analytical thinking, development of (new)

skills, and working in a constrained environment. The gamification aspect of a CTF allows the participant to express his or her full potential, e. g. attack/defend without fear of consequences or bad marks. CTF events have already been proposed as a means to enhance security education and awareness [54, 147, 180]. Although such events cover a wide range of security domains, to the best of our knowledge they do not include so far the security of ICS.

4.3.5 The SWaT Security Showdown

In this work, we focus on the aspect of training and validation of applied security skills for industry professionals and researchers. Gamification in education has been advocated as a means to enrich the learning experience [98]. In particular, within IT security, the implementation of CTF-like competitions have been argued to be advantageous for education and training [180]. Inspired by the gamified nature of CTF, we propose the following approach.

Our goal is to create a realistic environment where participants are encouraged to think out of the box. In real-life ICS settings, several intrusion detection mechanisms are in place to safe-guard critical operations. A successful attacker would have to bypass such systems in order to pose a threat, and simulating such settings would stimulate a participant's ingenuity to attempt creative attacks. On the other hand, if successful, such attacks will potentially unveil limitations of the defense mechanism. Therefore, we propose to divide participants in a training event into two categories: participants interested in developing defenses for ICS (defenders) and participants interested in testing the security of ICS (attackers).

In order to get the most out of an interaction with a real ICS testbed, it is important to learn fundamental concepts of ICS security. However, this learning phase should be as hands-on and gamified as possible. To this extent, we propose an *on-line* training phase, where attackers get familiar with ICS concepts and a particular critical infrastructure by means of a jeopardy-style CTF. Different from traditional CTFs, the challenges are tailored to highlight ICS concepts and use realistic simulations of ICS networks and remote interaction with ICS hardware. In this phase, attackers also should be aware of the internal workings of common defense mechanisms in place, and documentations there-of are shared with them.

After this preparation, in a *live* phase attackers phase interaction with a live system that is being monitored by defenders. In this setting, attackers should have concrete goals to achieve (or *flags* in CTF jargon), and their scoring should be influenced by the number of defenses triggered during their attack. In order to motivate attackers to perform more creative and difficult attacks, different attacker models can be suggested to them (i.e. insiders with administration capabilities, outsiders with network access) and the scoring can be adjusted according to the attacker model chosen.

Finally, participants will have access to statistics on their performance based on a unified scoring system taking into account both phases. Attackers will benefit from this experience since in order to solve the on-line and live challenges they will have to go through several of the topics discussed in the previous subsection. On the other hand, defenders will benefit by putting their solutions to the test against creative attackers.

We have implemented the proposed concepts at our institution in 2016, under the name SWaT Security Showdown. In the following two sections, we present the two

main phases of that event, which represent the two target systems we introduced earlier: a) online challenges using questions and simulated systems, and b) live events using a real physical ICS. Due to organizational constraints, and in order to maximize the learning experience, we decided to limit the participants to SWaT Security Showdown to selected invited teams from academia and industry, both for attacker and defender roles, for a total of 12 invited teams (6 attackers, 6 defenders, of which 3 academic and 3 industrial teams respectively). Teams were not limited in size, but only a maximum of 4 members could participate physically in the live event whereas remaining team members could join remotely.

4.4 Online phase of S3

In this section we will present the SWaT Security Showdown online event, and the details about its setup, and presented challenges. We will describe more in detail three set of challenges from the MiniCPS, Trivia and Forensics categories. We conclude the section with a summary of the collected results.

4.4.1 Online phase Setup and Challenges

The SWaT Security Showdown online phase involved six teams of attackers, three from industry, and three from academia. We presented a total of *twenty* challenges in preparation for this phase, and we offered to each team a limited time to access to Secure Water Treatment, and the required documentation to get familiar with the SWaT and EtherNet/IP. We organized two sessions, each one 48 hours long, where three teams at a time attempted to solve the challenges for a total amount of 510 points.

The S3 online phase was structured as a *jeopardy-style* CTF, and did not require physical access to the SWaT. The main goal of this phase was to provide an adequate training to the attacker teams (third goal from Section 4.3.3). Please refer to Section 4.3.4 for more information about Capture-The-Flag events.

Table 4.1 summarizes the proposed tasks. We presented twenty tasks divided into five categories: MiniCPS, Trivia, Forensics, PLC, and Misc, for a total of 510 points. Each category exercised several ICS security domains, such as Denial-of-Service, and Main-in-the-Middle attacks. It is important to notice that categories such as MiniCPS, Trivia, and PLC are *novel* in the domain of traditional jeopardy-style CTFs. Following CTF design best-practices we presented the challenges of each category in increasing order of difficulty e. g. solving challenge x helped to solve challenge $x + 1$, and when necessary, we gave hints e. g.: you could use `toolx` to accomplish a certain task. In general, we used the online event as a training session to prepare the attacker teams for the S3 live event that is described in Section 4.5.

We built a Webapp to run the S3 scoring system using the flask Python framework [153] (Figure 4.3 shows S3 online challenges' web page). The web pages were served over HTTPS, using Let's Encrypt [91], and a basic brute-force attempts detection mechanism based on user input logging was put in place on the backend side. A dedicated web page was showing a live chart with the scores from all the teams. We offered live help with two different channels: an IRC channel on `freenode.org`, and via email. The following is an example of user interface interaction with our Webapp:

TABLE 4.1: SWaT Security Showdown Online challenges summary: 20 tasks, worth 510 points.

Category	Tasks	Points	Security Domains
MiniCPS	5	210	network mapping, DoS, reconnaissance, MitM attacks in ENIP, tampering, tank overflow
Trivia	6	45	SWaT's physical process, devices and attacks
Forensics	4	105	packet inspection, processing and cryptography
PLC	3	60	ladder logic, code audit and development
Misc	2	90	web authentication, steganography

member of team A logs in to S3's Webapp (using the provided credentials), she navigates to challenge X's Web page, then enters the flag on an HTML form. If the flag is correct, she receives N reward points, otherwise a submission error appears on screen.

In the following, we focus on the tasks related to MiniCPS, Trivia, and Forensics categories, since they better illustrate the novel nature of the challenges proposed, and then we will present a summary of the results from the online event.

4.4.2 MiniCPS Category

The online phase presented five challenges in the MiniCPS category. MiniCPS is a toolkit for Cyber-Physical System security research [10]. MiniCPS was used to "realistically" reproduce (simulate) part of the Secure Water Treatment, including the hydraulic physical process, the devices and the network. Each simulated instance accessed by the attackers' teams was running on Amazon Web Services Elastic Compute Cloud (AWS EC2), using an m3-type virtual machine (one instance per team).

Figure 4.4 shows the simulation setup of a single instance, that was replicated for all the six teams. Each attacking team was provided with the credential to access an SSH server running on a simulated chrooted gateway device. The attacker had access to the emulated virtual control network that used the same topology, addresses (IP, MAC, net masks), and industrial protocol (EtherNet/IP), of the SWaT.

The attacker could interact with other simulated SWaT's devices in the star topology (four PLCs and an HMI), and alter the state of the simulated water treatment process affecting the two simulated water tanks (the Raw water tank and the Ultra-filtration tank). For example, an attacker might send a packet containing a false water level sensor reading of the Ultra-filtration tank to the HMI, or a packet that tells to PLC2 to switch off the motorized valve, that controls how much water goes into the Raw water tank. As a side note, Figure 4.4's setup is part of an internal project involving the development of novel honeypots for ICS [7].

The following five paragraphs summarize each of the MiniCPS challenges, with the attacker's goals and a reference solution:

Network warm up. The goal of the first challenge is to perform a passive ARP-poisoning MitM attack between PLC2 and PLC3. The attacker has to perform a network scanning to discover the hosts addresses and then use `ettercap` to read the flag on the wire.

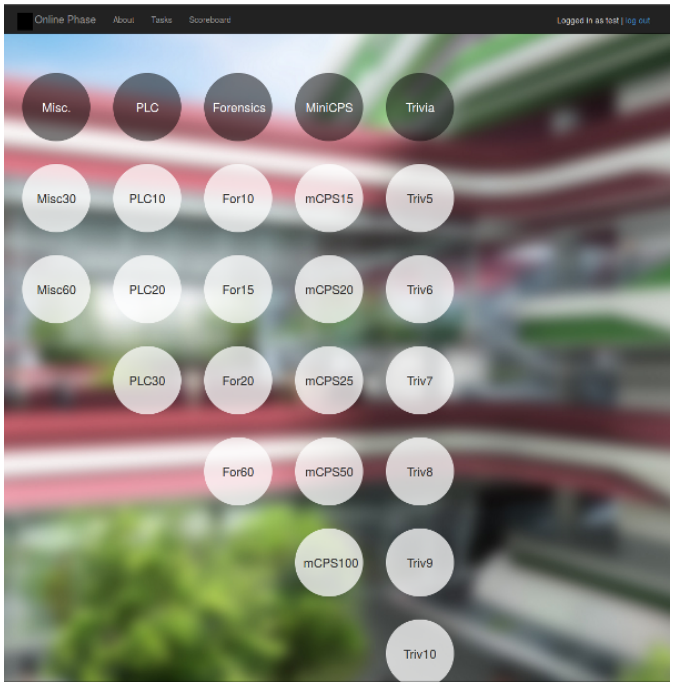


FIGURE 4.3: S3 online challenges' web page.

EtherNet/IP warm up. The goal of the second challenge is to read the flag stored in PLC2's EtherNet/IP server, and addressable with the name `README:2`. The attacker has to understand which PLC owns the `README:2` tag, and how to use `cpppo`, the suggested EtherNet/IP's Python library [104].

Overflow the Raw water tank. The goal of the third challenge is to overflow the simulated Raw water tank. The attacker has to understand the simulated dynamic of a water tank e. g. who drives inflow and outflow, and tamper with the correct actuators to increase the water level above a fixed threshold. Some hints were given to explain the binary encoding e. g. use m/n to switch ON/OFF a water pump and OPEN/CLOSE a motorized valve.

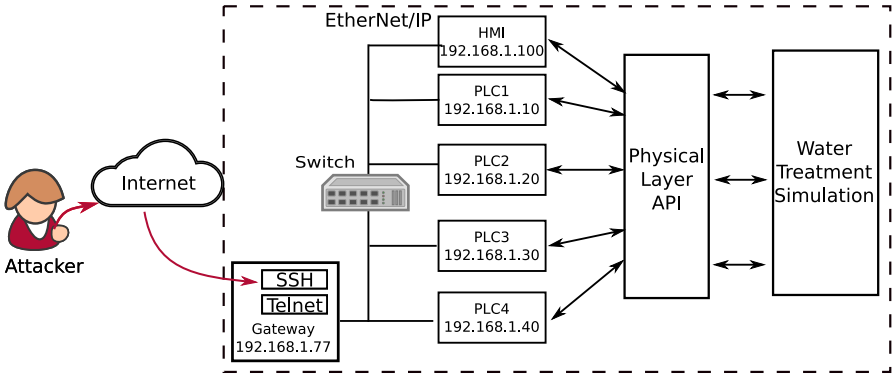


FIGURE 4.4: MiniCPS-based setup for online challenges.

Denial of Service HMI. The goal of the fourth challenge is to disrupt the communication (Denial-of-Service) between the HMI and PLC3, and then change a keep-alive tag value to 3 on PLC3 EtherNet/IP's server. In normal working condition the keep-alive tag is periodically set by the HMI to 2. Given the knowledge acquired from the previous three challenges, the attacker has to perform an active MitM attack that drops all the packets between the HMI and PLC3. Notice that, it is not sufficient to just write the required keep-alive tag value on PLC3 EtherNet/IP's server.

Overflow the Ultra-filtration tank. The goal of the fifth challenge is to overflow the Ultrafiltration water tank. The attacker has to reuse a combination of the previously used techniques to set up an active MitM attack using custom filtering rules e. g. use `ettercap` and `etterfilter`.

4.4.3 Trivia Category

The online phase presented six challenges in the Trivia category. The Trivia challenges were intended for the attackers to understand the plant structure, behavior, and the defense mechanisms. The knowledge gained from these challenges is expected to be of use to the attackers in other phases of the event. In the remainder of this section, we briefly describe the six challenges, their goals, and the steps needed to capture the flags.

The trivia challenges can be divided into two types, the first type involved the knowledge on SWaT, and the second type involved research papers on SWaT.

Knowledge on SWaT. Three challenges fall under this category. The goals of these challenges are to focus on the physical process of SWaT [119], the control strategy of SWaT, and the set points of the sensors and actuators. Following are the details regarding the challenges.

Trivia 1 The goal of the first challenge is to identify the analyzer that is used by the PLCs to control a specific dosing pump. In order to identify the device, the participant has to understand the control strategy of the particular dosing pump. As the PLC uses a number of different inputs to control the dosing pump, the participant has to trace the signals and identify the particular analyzer.

Trivia 2 The goal of the second challenge is to find out the set point that triggers the start of the backwash process. During the filtration process, small particles clot the Ultrafiltration membrane. To remove them and clean the Ultrafiltration membrane, a backwash process is started after reaching a specific threshold. In order to answer this challenge, the participant needs to revise and understand the backwash process.

Trivia 3 The goal of the third challenge is to identify the set point of the hardness analyzer used by a PLC to shut down the RO filtration. The hardness analyzer measures the water hardness in SWaT. The set point is a desired value of a particular sensor which is used by the PLC to control the plant. In the current scenario, when hardware analyzer exceeds desired value, PLC shuts down the RO filtration. In order to answer this challenge, attacker should understand the set points and control strategy of the RO process.

Research papers on SWaT. The remaining three challenges fall under this category. The goals of these challenges are to raise awareness about ICS attacks techniques and their

classification in the context of SWaT. We selected the following three papers: [1, 96, 2] as reference attack vectors targeting ICS. Following are the details of those challenges.

Trivia 4 The goal of this challenge is to familiarise the attacker with possible attacks on SWaT and potential impact of those attacks on SWaT. We provided a research paper [1] that presents an experimental investigation of cyber attacks on an ICS. In order to answer the challenge, the participant needed to read the paper and understand it.

Trivia 5 The goal of this challenge is to familiarise the participant with a security analysis of a CPS. We provided another research paper [96] that presented a security analysis of a CPS using a formal model. In order to answer the challenge attacker should read the paper and understand it.

Trivia 6 The goal of this challenge is to familiarise the attacker with multi-point attacks on ICS. We provided a third research paper [2] that discussed multi-point attacks. A multi-point attack leverages more than one entry point, e.g, two or more communications links, to disturb the state of an ICS. In order to answer the challenge, the participant needed to read the paper and understand it.

4.4.4 Forensics Category

The forensics challenges focused on network capture files, in particular *pcap* files, that are easy to process using programs such as `wireshark` and `tcpdump`. The participant had to learn how to process and extract information from a file containing pre-recorded network traffic from an ICS. The target industrial protocol was EtherNet/IP. We now provide details on three of the four challenges.

Identify the ICS hosts. The goal of the first challenge is to perform an analysis of the ICS hosts inside a captured ICS network traffic. To achieve that goal, the attacker should search for the hosts inside the captured traffic, classify them based on their IP addresses, identify whether a host is inside the ICS network or not and enumerate them.

Finding the poisoning host. The goal of this challenge is to search for a host that has performed an ARP poisoning Main-in-the-Middle attack, inside a captured network traffic. Then, the attacker will identify the start point and end point of captured ARP poisoning attack inside the captured network traffic. As an example, the flag of this problem will be the start and end TCP sequence number in the form of `ascflag{A-B}`, where the A is the starting TCP sequence number and B is the ending TCP sequence number.

Understanding the CIP protocol structure. The goal of this challenge is to find a particular pattern inside the payload of CIP messages. In particular, the attacker has to recognize that a CIP payload contains encrypted data and then he has to decrypt it. So, the attacker can decrypt the ciphertext by performing a XOR it with a key included in the payload or performing a brute-force.

4.4.5 Results from the Online phase

Table 4.2 presents the final scores of each team, the number of captured flags, and an estimation of the time spent playing, computed as the difference between the last and the first flag submitted by a team. As we can observe from the table, two teams were

able to fully complete all tasks, with Team 6 being by far the most efficient. On average teams spent 25.67 hours to solve the challenges (53% of the maximum of 48 straight hours), with a standard deviation of 13.06 hours. The teams scored an average of 268.83 points (52.7% of the maximum of 510). We believe that both the time invested and the percentage of challenges solved shows a notable investment in the game, and provides evidence on the engagement generated by the gamification strategy. In addition, we note a correlation between the number of hours invested, and the points achieved. In fact, when the outlier (Team 6) is removed, there is a 0.97 Pearson correlation coefficient (PCC) between time spent and points achieved.

TABLE 4.2: S3 Online Results summary. Category names: MCPS=MiniCPS, T=Trivia, F=Forensics, P=PLC, M=Misc

	Flags per category							
Team	MCPS	T	F	P	M	Σ Flags	Score	**Time**
T1	2	6	4	0	1	13	250	30h
T2	5	6	4	3	2	20	510	44h
T3	0	4	2	0	1	7	86	27h
T4	4	4	2	0	0	10	161	28h
T5	0	4	2	0	1	7	66	21h
T6	5	6	4	3	2	20	510	4h

To conclude the online event, we believe that the gamification factor played an important role. Gamification helped us to combine different categories in an unified ICS security theme, to motivate the attacker teams to do their best to get the maximum points, and to implicitly train them for the upcoming live phase.

4.5 Live phase of S3

As discussed in Section 4.3, the goal of the online phase was to prepare teams for the live phase of S3. In this section we will present the SWaT Security Showdown live event, and the details about its setup, and scoring system. Afterwards, we will describe two of the academic detection mechanisms, ARGUS and HAMIDS, that were used during the event. We conclude the section providing a summary of the collected results.

4.5.1 Live phase Setup

The live phase of S3 was held at SUTD over the course of 2 days in July 2016. All six attacker teams that participated in the online phase were invited. Each team was assigned a three hour timeslot, in which it would have free access to SWaT to test and deploy a range of attacks, taking advantage of the knowledge gained during the online phase presented in Section 4.4. In addition, teams were able to visit the SWaT testbed for one working day, to perform passive inspection before the event to prepare themselves.

The main goal of the live phase was two-fold: firstly, the teams would be able to learn more about an actual ICS and its security (second and third goals from Section 4.3.3). Secondly, we would be able to test a number of (internally developed) detection systems that were deployed in SWaT to evaluate and compare their performances (fourth goal from Section 4.3.3).

4.5.2 Scoring and Attacker Profiles

We designed the scoring system for the live phase with the following goals:

- Incentivise more technically challenging attacks.
- De-incentivise re-use of same attack techniques.
- Provide challenges with different difficulty levels.
- Relate the attack techniques to realistic attacker models.
- Minimize damages to the participants and the actual system.

We now briefly summarize the scoring system we devised. In general, points were only be awarded if the attack result could be undone by the attacker (to minimize the risks of permanent damages).

Equation 4.1 defines how to score an attack attempt:

$$s = g * c * d * p \tag{4.1}$$

With s being the final score, g a value representing the base value of the goal, c a control modifier to value the level of control the attacker has, d a detection modifier, and p the attacker profile modifier. Most modifiers were in the range $[1, 2]$, while the base value for the targets was in the range $[100, 200]$.

We now describe in detail the four modifiers from Equation 4.1. Goals could be chosen from two sets: *physical process goals* and *sensor data goals*.

Physical Process Goals g. Control over actuators, and physical process (water treatment):

- 100 points: Motorised Valves (open/close/intermediate).
- 130 points: Water Pumps (on/off).
- 145 points: Pressure.
- 160 points: Tank fill level (true water amount, not sensor reading).
- 180 points: Chemical dosing.

Sensor Data Goals g. Control over sensor readings at different components:

- 100 points: Historian values.
- 130 points: HMI/SCADA values.
- 160 points: PLC values.

- 200 points: Remote I/O values.

Control modifiers c**.** The control modifier determined how precise control the attacker had. As guideline the modifier was 0.2 if the attacker could randomly (value and time) influences the process, up to 1.0 if the attacker could precisely influence the process or sensor value to a *target value* chosen by the judges.

Detection modifiers d**.** Not triggering a detection mechanism while the attack is executed would increase the detection modifier, using the following formula: $2-x/6$, with x the number of triggered detection mechanisms.

Attacker profile modifiers p**.** For each attack attempt, the attacking team had to inform the judges about the chosen attacker model before the attack is started. The overall idea is that a weaker attacker profile would yield a higher multiplier. The attacker profiles were based on [152] and the higher is the modifier the weaker is the attacker model. The SWaT Security Showdown live event used three attacker profiles: the cybercriminal, the insider, and the strong attacker.

The *cybercriminal* model had a factor of 2. The cybercriminal was assumed to have remote control over a machine in the ICS network, and was able to use own or standard tools such as `nmap`, and `ettercap`. The cybercriminal did not have access to ICS specific tools, such as Studio 5000 (IDE to configure SWaT's PLCs), or access to administrator accounts.

The *insider* attacker had a factor of 1.5. It represented a disgruntled employee with physical access and good knowledge of the system, but no prior attack experience, and only limited computer science skills. In particular, the attacker was not allowed to use tools such as `nmap` or `ettercap`, but had access to engineering tools (such as Studio 5000), and administrator accounts.

The *strong* attacker effectively combined both other attackers, resulting in the strongest available attacker model, and yielded a factor of 1.

Attackers could earn points for one or more attacks. If more than one attack was successfully performed, the highest scores from each goal was aggregated as final score. For example, if an attack on pumps was successful both using the strong attacker model, for a total score s of 130 points, and the cybercriminal attack model, for a total score s of 200 points, then only 200 points would be counted for that attack goal (attack a pump).

4.5.3 Detection mechanisms

As discussed in Section4.3, as part of the design of our approach we included academic and commercial detection mechanisms as means to incentivate the creativity of attackers: the less detection mechanisms triggered the more points obtained, as discussed in the previous subsection. Also, the experience was designed to serve as feedback to the designers of detection mechanisms when confronted with various human attackers and a wider range of attack possibilities. In the following we emphasise two academic detection mechanisms implemented at SUTD.

Distributed detection system

The distributed attack detection method presented in [3] was implemented in Water Treatment Testbed as one of the defense methods used in the S3 event. The method

is based on physical invariants derived from the CPS design. A "Process invariant," or simply invariant, is a mathematical relationship among "physical" and/or "chemical" properties of the process controlled by the PLCs in a CPS. Together at a given time instant, a suitable set of such properties constitute the observable state of SWaT. For example, in a water treatment plant, such a relationship includes the correlation between the level of water in a tank and the flow rate of incoming and outgoing water across this tank. The properties are measured using sensors during the operation of the CPS and captured by the PLCs at predetermined time instants. Two types of invariants were considered: state dependent (SD) and state agnostic (SA). While both types use states to define relationships that must hold, the SA invariants are independent of any state based guard while SD invariants are. An SD invariant is true when the CPS is in a given state; an SA invariant is always true.

The invariants serve as checkers of the system state. These are coded and the code placed inside each PLC used in attack detection. Note that the checker code is added to the control code that already exists in each PLC. The PLC executes the code in a cyclic manner. In each cycle, data from the sensors is obtained, control actions computed and applied when necessary, and the invariants checked against the state variables or otherwise. Distributing the attack detection code among various controllers adds to the scalability of the proposed method. During S3, the implementation was located inside the Programmable Logic Controllers (PLCs).

The HAMIDS framework

The HAMIDS (HierArchical Monitoring Intrusion Detection System) framework [149] was designed to detect network-based attacks on Industrial Control Systems. The framework leverages a set of distributed Intrusion Detection System (IDS) nodes, located at different layers (segments) of an ICS network. The role of those nodes is to extract detailed information about a network segment, combine the information in a central location, and post-process it for real-time security analysis and attack detection. Each node uses the Bro Intrusion Detection System (IDS) [138].

Figure 4.5 shows our deployment of the HAMIDS framework instance deployed in the SWaT. As we could see from Figure 4.5, each L0 (Layer 0) DLR segment has an additional Bro IDS node that is collecting data flowing from a PLC to the RIO device. An additional Bro IDS node is connected to a mirroring port of the L1 (Layer 1) industrial switch, and is collecting the traffic in the L1 star topology. Every Bro IDS node is sending data to the central HAMIDS host, by means of a secure channel (using SSH). Elasticsearch [56], a distributed, RESTful storage and search engine, is used to provide a scalable and reliable information recording and processing.

The HAMIDS detection mechanism is entirely isolated from the ICS network, and thus as part of the live event, the attackers were not able to have direct access to the detection system. So the attackers will have hard times trying to stop the detection mechanisms of the HAMIDS framework. On the other side, there are two ways to access data from a defender perspective: a Web interface and a SQL API.

The web interface is a user-friendly interface that can be used by less technical ICS operators and it is capable of listing all alarms generated by the central node, and eventually help in detecting an ongoing attack. The expert user can directly use the SQL API to query the central node, and obtain more detailed data about the observed packets.

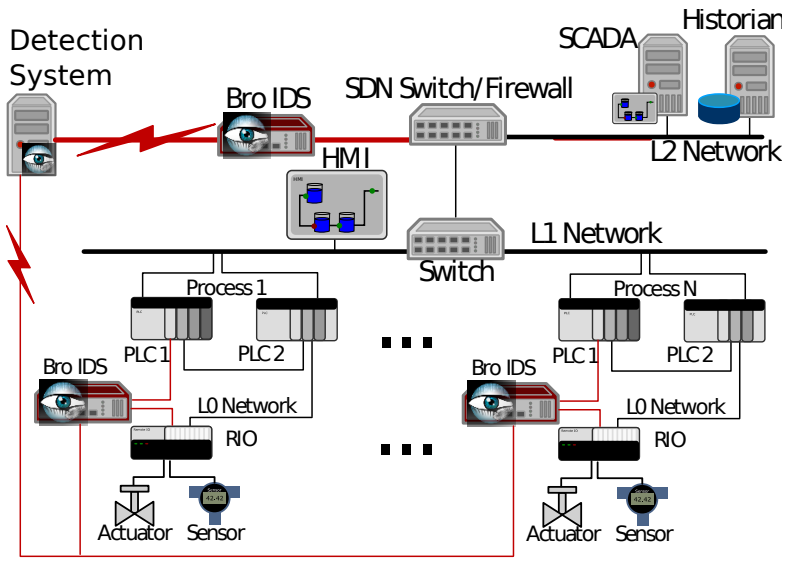


FIGURE 4.5: The HAMIDS framework instance: Bro IDS nodes are placed both at L0 and L1 network segments of the SWaT.

For the event, the framework was configured to present high-level information about the status of the detection system, suitable for non-expert defenders. Using the web user interface, the defender could read the generated alarms related to triggered alarms due to observed network traffic in the industrial control system. In addition, expert defenders could read the detailed information about the ICS process by using manual SQL queries to retrieve data for further analysis.

4.5.4 Results from the Live phase

Table 4.3 shows the final scores, the number of performed attacks and the cumulative detection rate d_{rate} of the live phase. The cumulative detection rate was computed as the average number of detection mechanisms triggered (considering only the two academic detection mechanisms discussed before) in all successful attacks by a given team.

TABLE 4.3: SWaT Security Showdown Live Results summary.

Team	Score	Successful Attacks	d_{rate}
T1	666	4	1
T2	458	2	1
T3	642	3	1
T4	104	1	1
T5	688	5	$\frac{6}{5}$
T6	477	3	$\frac{4}{3}$

In order to show more in depth insights of the live phase, we now provide details on several attacks that were conducted by the participants during the SWaT Security Showdown live event (see Table 4.4). We classify those attacks in two types, the "cyber" attacks were conducted over the network using either the cybercriminal, or the strong attacker model, while the "physical" attacks were conducted having direct access to the SWaT using either the insider, or the strong attacker model. We now describe each of those attacks.

TABLE 4.4: S3 Live Attacks and Detections summary: ○= undetected, ●= detected.

Attack	Type	Score	ARGUS	HAMIDS
SYN flooding (DoS) PLC	Cyber	396	○	●
DoS Layer 1 network	Cyber	104	○	●
Tank level sensor tampering	Physical	324	●	●
Chemical dosing pump manipulation	Physical	360	●	○

DoS by SYN flooding. The first attack was a cyber-attack, and the attacker used the insider attacker model. The attacker had access to the administrator account and associated tools. The attacker performed a SYN flooding attack on PLC1's EtherNet/IP server. SYN flooding is a Denial-of-Service (DoS) attack, where the attacker (the client) continuously try to establish a TCP connection, sending a SYN request to the EtherNet/IP server, the EtherNet/IP server then responds with an ACK packet, however the attacker never completes the TCP three-way-handshake and continues to send only SYN packets. As a result of this DoS attack, the HMI's is unable to obtain current state values to display, and would display 0 or * characters instead. Such effects would impede the supervision of ICS in real applications. However, the attack did not interrupt or harm the physical process itself. The HAMIDS detectors was able to detect the attack by observing the high number of SYN requests without follow-up. The ARGUS detector was not able to detect the attack, as the physical process was not impacted.

DoS. The second attack was a cyber-attack, the attacker used the cybercriminal attacker model. The attacker had access to the network and attack tools. The attacker performed an ARP poisoning Man-in-the-Middle attack, that redirected all traffic addressed to the HMI. The redirected traffic was then dropped and prevented from being received. The attack drove the HMI to an unusable state, and it took a while to restore the system state after the attack. We did not allow the attack to run long enough to affect the physical process. HAMIDS detected the attack due to the changes in network traffic (i.e. malicious ARP traffic, changed mapping between IP and MAC addresses in IP traffic). In contrast, ARGUS did not detect the attack, as the physical process continued to operate without impact.

Tank level sensor tampering. The third highlighted attack involved an on-site interaction with the system, the attacker used the strong attacker model. The attacker focused on one of the L0 segments, and he demonstrated control over the packets sent in the Ethernet ring. Indeed, the attacker was able to alter the L0 traffic in real time, and

manipulate the communication between the PLC and the RIO. ARGUS was able to detect the attack due to the sudden changes in reported sensor values (bad data detector). In addition, the HAMIDS framework detect the attack by observing the change in data reported from the PLC to the SCADA (and potentially, in L0 as well).

Chemical dosing pump manipulation. The fourth attack was a physical-attack, and the attacker used the insider attacker model. The attacker was able to alter the chemical dosing in the second stage (Pre-treatment) of the SWaT by interacting directly with the HMI interface, and overriding the commands sent by the PLC (that was set in manual mode). The attack would have resulted in an eventual degradation of the quality of the water, however we stopped the attack before that case occurred. ARGUS was able to detect the attack because the updated setpoints diverged from their hard-coded counterpart in the detection mechanism. The HAMIDS detection was unable to detect this scenario as the network traffic did not show unusual patterns or changes (as typical for attacks using the insider model).

4.6 Related work

In [121] Mink presents an empirical study that evaluates how exercises based on gamification and offensive security increase the motivation and the final knowledge of the participants. Our work tries to extend this message to ICS security, while Mink's paper focuses on traditional Information security.

DEFCON [42] is an annual hacking conference organized by information security enthusiasts. The DEFCON CTF is part of the main event, and it is one of the most well known, and competitive CTF contest worldwide. Like S3, it has a Jeopardy-style qualification phase, and an attack-defense final phase. However ICS security is not the main focus of DEFCON's CTF. Several other similar CTF competitions are listed in [45]. In [180] Vigna proposes to use gamified live exercises to teach network security. The motivations and philosophy of this work are similar to ours. However the focus of the paper is on IT network security (e. g. gain root privileges on a webserver or steal data from a SQL database) and not on OT network security (industrial network devices and protocols). Inspired by [180], in [38] authors of the iCTF event presented two novel, live, and large-scale security competitions. The first is called "treasure hunt" and it exercises network mapping and multi-step network attacks. The second is a "Botnet-inspired" competition and it involves client-side web security tasks such as Web browsers exploitation. Unlike the presented chapter, both competitions focus on traditional client-server IT network architectures and attack-only scenario.

The MIT/LL CTF [185] was an attack-defense CTF with a focus on web application security. The main goal of the event was to attract more people towards practical computer security exercises. The CTF takes inspiration from Webseclab [28], a web security teaching Virtual Machine that is packed with an interactive teaching web application, a sandboxed student development environment, and a set of useful programs. Both are interesting projects but they are not covering the ICS security domain, even though they share some of the presented goals. BIBIFI [154] is a cyber-security competition held mainly in academic environments that combines in the same contest: secure development (Build-it), attacks development (Break-it) and patch development (Fix-it).

This effort was targeted at improving secure software construction education, and thus the exercises proposed in this competition do not cover the ICS security domain.

4.7 Conclusions

In this work we discussed problems faced by security experts and ICS engineers in the context of ICS security education and research. In particular, security experts require access to real ICS infrastructures to learn about ICS (security) and practise applied attacks and defenses. In addition, ICS engineers require additional training focused on basic cyber-security concepts and offensive and defensive security techniques. We propose to use gamified security competitions such as online and live Capture-The-Flag to address and mitigate those problems. To demonstrate the feasibility of such events, we designed and implemented the *SWaT Security Showdown (S3)*, leveraging the Secure Water Treatment (SWaT) water treatment testbed.

To the best of our knowledge, the S3 event was the first security competition involving access to live and virtual ICS infrastructures (e. g. MiniCPS). The online phase consisted of a Jeopardy-style CTF that included novel challenges specifically designed for ICS security. For example, we gave to the attacker remote access to a real PLC programming environment (e. g. Studio 5000) and we asked them to understand a ladder logic program. Overall, six participating attacker teams submitted 77 correct flags in the online phase of the S3 event.

In the live phase (an attack-defense CTF), the participating teams performed 18 successful attacks in SWaT within a limited time frame. The timing was an important factor because it increased the level of realism of the competition. During the S3 live phase the teams demonstrated a wide range of different attack approaches, and adapted their attacks to challenges posed by the complexity of the real testbed. In addition we also evaluated several (novel) detection mechanisms including two internally-developed ones (e. g. ARGUS and HAMIDS). Most of the attacks were detected by at least one of our detection mechanisms.

In summary, S3 was an enriching experience for everybody, including us (the organizers). We hope that such event provides a foundation to enable others to run similar ICS security educational experiments in the near future.

Chapter 5

Conclusion about Cyber-Physical Systems Security

Cyber-Physical Systems (CPS) are systems managing physical processes using sensors, actuators, and interconnected controllers. Examples of CPS include water treatment and distribution plants. In the first part of this thesis we focused on securing these systems. We started by defining three intertwined problems: technologies and processes, multi-disciplinary communities, threat models and incentives. Technologies related to CPS require an understanding of a broad and complex set of topics such as control theory, network engineering, process engineering and information security. Different CPS professionals work in isolated communities with separate (and sometimes conflicting) regulations. CPS were not designed with security in mind but for safety.

We believe that to tackle these problems we need a free, modern, accessible and extendible toolkit to experiment with CPS. MiniCPS [10] [1] is the main result of our vision. MiniCPS is a free and open source toolkit to build lightweight and real-time simulations of CPS built on top of mininet. It is the first product that allows to emulate an CPS network in real-time while providing a simulated physical process and simulated or emulated control devices. Currently. MiniCPS supports EtherNet/IP and Modbus/TCP two popular industrial protocol.

We used MiniCPS to prototype the first virtual high-interaction honeypot for industrial control systems [7]. Honeypots are defense mechanism widely used in the IT domain to lure attackers into attacking fake systems. In our implementation we mimic a water distribution system. MiniCPS was used as an educational tool to implement novel CTF challenges for a cybersecurity competition targeted at CPS security professionals. We used real-time simulation to let the participant access and attack a virtualized water treatment testbed [13]. MiniCPS was used as an attack tools as the basis of CPSBot a framework to build botnets for CPS [8]. The real-time simulation capabilities can be used by an attacker to predict dynamically predict the best attack decisions.

MiniCPS was used by and/or exposed to the following institutions: University of Oxford, Georgia Tech, UT Dallas, University of Luxembourg, Rutgers University, University of Padua, Roma Tre University and the following industries: Pacific Northwest National Laboratory, ST Engineering[2].

[1] https://francozappa.github.io/project/minicps/

[2] See Chapter 10 for our conclusion about wireless systems security.

5.1 Lessons Learnt

During this PhD I've learnt many valuable lessons. Here I'm sharing some of them regarding cyber-physical system security:

- Security through obscurity is still a problem for CPS, especially in the case of ICS.
- Mathematical modeling of a physical process is not directly portable across different CPS (even of the same type).
- IT and OT security communities are diverse and it is hard to connect them.

5.2 Future Work

These are the main research directions that we would like to see in the future:

- Experimentation with real-time simulation of complex physical processes.
- Emulation of less popular CPU architectures, operating systems and firmwares.
- Dissemination of data sets collected from CPS testbeds and/or actual plants.
- Development and Sharing of physical and cyber models of CPS.

Part II

Wireless systems security

Chapter 6

Introduction to Wireless Systems Security

6.1 Problem Statement

Wireless communication technologies are providing key services to our society and they are evolving at unprecedented scale and speed. Nowadays, even a low-cost smartphone is equipped with multiple baseband technologies, e.g. cellular, Wi-Fi, Bluetooth, Global Positioning System (GPS), and Near-field communication (NFC) to provide a variety of wireless services. Novel wireless technologies are also developed for emerging applications such as (industrial) internet of things and unmanned aerial vehicle. Common wireless technologies are developed to have adequate throughput, range, and reliability. Specialized ones look at other factors such as real-time constraints and low power consumption. Despite the variety of requirements and use cases, wireless communications involving sensitive data must also provide security guarantees, such as confidentiality, integrity, and authentication.

There are two main research branches aiming to secure wireless communications: *cryptography* and *physical layer security*. Cryptography develops security mechanisms from the OSI link layer upwards by securing bits using cryptographic schemes, e.g., RSA and ECDH. Cryptographic solutions are widely adopted by wireless technology standards such as 802.11X for Wi-Fi, and Secure Simple Pairing and Secure Connections for Bluetooth. Physical layer security mechanisms are complementary to cryptography and they are based on the characteristics of the wireless channel and the physical layer parameters of the transceivers. For example, to combat eavesdropping, the defender might introduce artificial noise (i. e. friendly jamming) [181] or he might use spatial diversity techniques (i. e. beamforming) [4]. At the time of writing, physical layer solutions are not adopted by any commercial wireless standards (e. g. Bluetooth and Wi-Fi).

In this dissertation we deal with the following problems of wireless communication systems security:

1. *Effectiveness of physical layer features as defense mechanisms*
2. *Complexity and accessibility of wireless technologies*
3. *Security evaluations and hardening of wireless technologies*

6.1.1 Effectiveness of physical layer features as defense mechanisms

In the last decade, wireless network communication has grown tremendously, mainly due to the evolution of wireless standards such as 3G, 4G and 5G. These standards introduced several physical layer features to increase the throughput and reliably of the link. For example, recent amendments of the 802.11 Wireless local area network (WLAN) standard introduced Multiple-Input-Multiple-Output (MIMO), spatial diversity (CSD, TxBF, STBC), and spatial multiplexing (MU-TxBF) techniques [87]. MIMO allows a node to use multiple antennas to transmit or receive multiple wireless signals. Spatial diversity increases the reliably of a point-to-point link by taking advantage of multiple antennas to send or receive a single wireless signal. Spatial multiplexing can be used by a transmitter to reach multiple users at the same time taking advantage of multiple antennas.

It was theoretically demonstrated that some physical layer features might also be used to degrade the performance of different types of attackers [198]. For example, spatial diversity (beamforming) might degrade the signal to noise ratio (SNR) of a passive eavesdropper that is far from the main transmission lobe [4]. However, no prior work has been done to quantify the disadvantage in a practical scenario (e. g. an access point that is connecting multiple laptop to the Internet). Such an analysis would be useful because if some physical layer features that are already deployed can effectively be used to mitigate some attacks, then new defense mechanisms, complementary to cryptographic ones, can be integrated almost for free on commercial off the shelf (COTS) devices.

6.1.2 Complexity and accessibility of wireless technologies

Wireless technologies are based on complex specification and implemented using complex technologies that might be difficult to access. The protocol specification might be freely available, (e. g. Bluetooth), available upon subscription fees (e. g. 802.11 amendments), partially available or not available at all. The implementation of the protocol suffers the same issues of the specifications. Typically, a modern protocol is divided into components and these components are implemented on different devices (with different requirements). It might be doable to get access to the source code of OS drivers and kernels, e. g. Linux and bluez. Unfortunately, the source code for radio firmwares is not available and it has to be reversed engineered [158, 114].

As a result of these issues *security through obscurity is still a problem for wireless communication technologies*. It is indeed important to develop effective and low-cost techniques to analyze (proprietary) wireless protocols and to build low-cost platform to experiments with radio signals. Projects such as scapy [25], Frida [148], Ubertooth [135] and HackRF [134] are excellent examples going in that direction.

6.1.3 Security evaluations and hardening of wireless technologies

Wireless technologies, despite their underlying complexity and poor accessibility, are a very attractive target for attackers. A single vulnerability in the specification of a wireless protocol automatically affects *all* standard-compliant devices, regardless their implementation details such as underlying CPU architecture, operating system, and

firmware. It is important to perform security analysis and evaluations of standard wireless technologies such as Wi-Fi and Bluetooth. Proprietary wireless technologies from big companies such as Apple and Google are also an interesting target for attackers because they are shipped with all their devices such as smartphones, tablets, laptops and IoT gadgets.

Even tough a security evaluation might have an significant initial cost (reverse engineering code, protocols and crypto) the evaluation will provide huge benefits in the long term that can (and should) be reused across different analyses. The biggest return of a security evaluation is finding, exploding and fixing high-impact security vulnerabilities and this will be one of the focus on this part of the dissertation. We took inspiration from several pivotal papers in the domain of Bluetooth attacks [164], 802.11 attacks [27] and applied crypto attacks [93].

6.2 Our Vision, Research Directions and Questions

We believe that physical layer features can and should be used as a complementary defense mechanisms to cryptography. We would have to find the best solution that is capable of adding security guarantees without adding too much performance overhead. The best way to start this transition is by looking at already deployed physical layer features, such as MIMO and beamforming, and see whether or not they can help us in practice. The usage of such features is expected to grow with the advent of 5G technologies that will employ a massive number of directional antennas. In the case of 802.11 a comparison between its recent amendments would be useful to understand if some already implemented physical layer features can be effectively used as defense mechanisms to protect against some kind of attacks. Additionally, if this analysis is successful for 802.11 then It might be reused for other deployed technologies such as Bluetooth and ultra wideband schemes.

We believe that wireless communications technologies should be more open to security researcher. Security through obscurity has been proven ineffective multiple times already and it is a risky business especially when dealing with complex software and hardware systems. We would like to find and share effective techniques to analyze wireless systems including proprietary ones. In the case of proprietary wireless technologies such as Google Nearby Connections, a reverse-engineering phase and the reimplementation of the protocol would be useful to understand the internal details of the protocol, to provide guidelines to other researchers interested into analyzing other proprietary protocol and to help the protocol developers to find and fix security vulnerabilities.

Even if a wireless technology has a standard body behind with available specification it does not mean that it is secure. We would like to test the security posture of the most popular standard wireless technologies such as Bluetooth. We would have a look at the security architecture of the protocol and the provided security guarantees. Given the lack of a reference implementation for Bluetooth some extra effort might be necessary to test specific sub-systems of the protocol such as handshakes, key generation and distribution. The result of this analysis will be beneficial in any case, if we find some vulnerabilities we can fix them and if we don't find any of them we can confirm that the protocol is providing what it claims to offer.

Here is a list of research questions that we want to tackle:

- Are there deployed physical layer features that can be (re)used to provide security guarantees against different attacker models?
  - What about MIMO and spatial diversity?
  - Are these features implemented and used by 802.11 (WLAN) devices?
- How can we effectively analyze and evaluate wireless communication system?
  - What about proprietary technologies (from big corporations)?
  - Shall we use static or dynamic analysis techniques?
- What is the security posture of standard wireless technologies such as Bluetooth?
  - Are the specifications secure from a cryptographic point of view?
  - Are the products keeping up with the specification evolution?

6.3 Wireless Systems Security Contributions

The second part of the thesis makes contributions in the area of wireless systems security, with results published in refereed venues. The contributions about cyber-physical systems security are presented in Section 1.3.

- Chapter 7 discusses the impact of recent 802.11 features, such as Multiple-Input-Multiple-Output (MIMO) and beamforming, on the physical layer security of 802.11n and ac networks, using 802.11b (no MIMO) as a baseline for comparisons. We are interested in passive eavesdropping attacks where both the intended receiver and the attacker are using COTS devices. The fact that MIMO and beamforming are introducing a disadvantage for the passive eavesdropper is a well know theoretical result, and it is related to the concepts of spatial diversity and multiplexing. However, the impact of MIMO and beamforming on 802.11n and ac networks has not yet been discussed and experimentally evaluated. We present a theoretical discussion and a statistical analysis to predict the effect on the eavesdropper's Signal-to-Noise-Ratio (SNR) and Packet-Error-Rate (PER). We show that the PER in 802.11n increases up to 98% (compared to 802.11b) at a distance of 20 meters between the sender and the eavesdropper. To obtain a PER of 0.5 in 802.11n, the attacker's maximal distance is reduced by up to 129.5 m compared to 802.11b. In our experiments, we used the simplest MIMO beamforming setup leveraging COTS devices to reduce the variability of the results. The results validated the predicted feedings. We conclude that the MIMO and beamforming capabilities of modern 802.11 networks provide some soft countermeasure against passive eavesdropping using COTS devices. Chapter 7 appeared in Proceedings of the Cryptology and Network Security Conference (CANS) 2017 [9]

- Chapter 8 presents the first security analysis of the Google's Nearby Connections API, based on reverse-engineering of its Android implementation. The Nearby Connections API enables any Android (and Android Things) application to provide proximity- based services to its users, regardless of their network connectivity. We demonstrate several attacks grouped into two families: connection manipulation (CMA) and range extension attacks (REA). CMA-attacks allow an attacker to insert himself as a man-in-the-middle and manipulate connections (even

unrelated to the API), and to tamper with the victim's network interface and configuration. REA-attacks allow an attacker to tunnel any nearby connection to remote (non- nearby) locations, even between two honest devices. Our attacks are enabled by REarby, a toolkit we developed while reversing the implementation of the API. REarby includes a dynamic binary instrumenter, a packet dissector, and the implementations of custom Nearby Connections client and server. Chapter 8 appeared in Proceedings of the Network and Distributed System Security Symposium (NDSS) 2019 [11]

- Chapter 9 presents an attack on the encryption key negotiation protocol of Bluetooth BR/EDR (Bluetooth Classic). The attack allows a third party, without knowledge of any secret material (such as link and encryption keys), to make two (or more) victims agree on an encryption key with only 1 byte (8 bits) of entropy. Such low entropy enables the attacker to easily brute force the negotiated encryption keys, decrypt the eavesdropped ciphertext, and inject valid encrypted messages (in real-time). The attack is stealthy because the encryption key negotiation is transparent to the Bluetooth users. The attack is standard-compliant because all Bluetooth BR/EDR versions require to support encryption keys with entropy between 1 and 16 bytes and do not secure the key negotiation protocol. As a result, the attacker completely breaks Bluetooth BR/EDR security without being detected. We call our attack Key Negotiation Of Bluetooth (KNOB) attack. We describe how to perform the KNOB attack, and we implement it. We evaluate our implementation on more than 14 Bluetooth chips from popular manufacturers such as Intel, Broadcom, Apple, and Qualcomm. Our results demonstrate that all tested devices are vulnerable to the KNOB attack. We discuss countermeasures to fix the Bluetooth specification and its implementation. Chapter 9 appeared in Proceedings of the USENIX Security Symposium 2019 [12]

- Chapter 10 concludes the second part of this dissertation, summarizing our contributions, lessons learnt, and future works.

Chapter 7

Practical Evaluation of Passive COTS Eavesdropping in 802.11b/n/ac WLAN

Keywords: WLAN, 802.11, Eavesdropping, MIMO, Beamforming.

7.1 Introduction

In the last decade, wireless network communication has grown tremendously mainly due to standards such as UMTS (3G) and LTE (4G) for cellular networks and IEEE 802.11 (WLAN) for wireless networks. Cisco estimated that in 2017, 68% of all Internet traffic will be generated by wireless devices [39]. As a result, it can be expected that a majority of sensitive communication services, such as mobile banking and online payments will involve wireless networks. Indeed, it is paramount to secure the broadcast wireless channel against eavesdroppers to protect the confidentiality and integrity of the information.

In this work, we present a theoretical discussion, a numerical analysis (using path loss models), and a practical evaluation of passive eavesdropping attacks targeting several 802.11 (WLAN) networks. Recent 802.11n/ac amendments introduced interesting physical layer and link layer features such as Multiple-Input-Multiple-Output (MIMO), spatial diversity (e. g. CSD, TxBF, STBC) , spatial multiplexing (e. g. MU-TxBF), dual-band antennas[1] and frame aggregation [87]. It is believed that some of those features, that were developed mainly to increase the robustness and throughput of the channel might also *degrade* the performance of a passive eavesdropper. We would like to investigate this claim and experimentally measure whether this degradation happens or not in practice in a simple but yet realistic scenario (e. g. eavesdropping WLAN networks with COTS devices).

Several theoretical discussions have already been presented about passive and active eavesdropping in the wireless channel. The seminal work by Wyner [191] started the wiretap channel research track that has been extended to Gaussian [106], fading [76], and MIMO [130] channels. This set of papers studies asymptotic conditions that very rarely happen in practice. Recently, special attention was given to MIMO and beamforming as a defense mechanism against passive eavesdropping [145, 192, 139]. However, those works do not focus on 802.11 and they consider only a subset of the 802.11

[1] In this work we always use the word *antennas* rather than *antennae*.

features. There are also some alternative techniques already proposed against passive eavesdropping including multi-user cooperative diversity and the use of artificial noise [51, 198, 125]. However, those techniques are neither listed in any 802.11 standards nor implemented in any COTS device.

In this chapter, we investigate the disadvantages that a passive eavesdropper has to face when attacking the downlink of an 802.11n/ac (MIMO) network versus an 802.11b (SISO) network. We focus on 802.11 networks in infrastructure mode (e. g. an access point connecting several laptops to the Internet) that use Commercial-Of-The-Shelf (COTS) devices. In particular, we compare *three* of the most widely used 802.11 amendments: b, n, and ac. We look at the downlink (e. g. traffic from the access point to the terminals) because it is the link that supports most of the advanced features of 802.11n/ac (e. g. spatial diversity and spatial multiplexing). We use 802.11b as a baseline. Our attacker model choice is explained in detail in Section 7.3.1, and a brief discussion about a stronger attacker model is presented in Section 7.4.5.

In our theoretical discussion, we estimate lower and upper bounds for the expected Signal-to-Noise-Ratio (SNR) disadvantage of an eavesdropper in 802.11n and ac compared to 802.11b. We numerically derive the expected Packet-Error-Rate (PER) of the intended receiver and the eavesdropper with respect to their distances to the sender. Finally, we present an 802.11b/n/ac downlink empirical evaluation using COTS devices. After the experiments, we are able to confirm that in 802.11n/ac networks, the PER of the eavesdropper increases with respect to her distance to the sender, given a minimum distance between the attacker and the intended receiver.

We summarize our contributions as follows:

- We derive the theoretically expected eavesdropper's SNR disadvantage (in dB), for attacks using COTS radios, in 802.11b/n/ac downlinks.
- We discuss how the theoretical SNR disadvantage translates to practical constraints (e. g. reduced range, higher PER) for the attacker.
- We perform a series of experiments to validate that the expected disadvantage is experienced in practice and that its effects were correctly predicted.

The structure of this work is as follows: in Section 7.2 we provide the required wireless communications background. In Section 7.3, we present the system and attacker models, we compare passive eavesdropping 802.11b and 802.11n/ac downlinks, and we estimate the SNR and PER disadvantages for a passive eavesdropper in 802.11n/ac. In Section 7.4, we present our results from a series of eavesdropping experiments that validate our predicted impediments. We summarize related work in Section 7.5, and conclude the chapter in Section 7.6.

7.2 Background

We now provide a summary of the important concepts used in this work: the fading wireless channel, the 802.11b/n/ac amendments, and three wireless communication metrics (SNR, BER, and PER). For additional details, we refer to influential books such as [140, 65].

7.2.1 The Fading Wireless Channel

The progression of wireless communication systems evolved around two main metrics: *robustness* and *throughput*. Those metrics are severely influenced by channel fading. Fading can be described as a random process affecting the quality of the transmitted wireless signal, by means of attenuation and distortion over time and frequency. There are three additive phenomena contributing to fading: path loss, shadowing, and multipath.

Path loss is a large-scale fading event due to the propagation nature of the electromagnetic waves (that are carrying the useful signal). There are different path loss models according to the system parameters and the channel environment. For example, in the Free Space Path Loss (FSPL) model the transmitted power decays quadratically with the distance from the transmitter to the receiver. Shadowing is another large-scale fading event due to the presence of obstacles between the transmitter and the receiver. There are different ways to model shadowing such as using a log-normal random variable. Multipath is a small-scale fading phenomenon that takes into account constrictive and/or destructive interference at the receiver between direct, reflected and scattered electromagnetic waves.

There are two well-known fading models that take into account all three fading aspects: *Rayleigh fading* for non-line-of-sight (NLOS) environments, and *Rician fading* for line-of-sight (LOS) environments. In both cases, each channel coefficient h is modeled with a complex random number. Each channel coefficient is providing random attenuation (change in amplitude) and distortion (change in phase). In the Rayleigh fading model, the real and imaginary parts of h are modeled with independent identically-distributed (IID) Gaussian random variables with 0 mean and equal variances and the amplitude of h is Rayleigh distributed. In the (more generic) Rician fading model, the amplitude of h is Rice distributed.

7.2.2 IEEE 802.11 Standard (WLAN)

802.11 is a family of IEEE standards that regulates wireless local area networks [44]. The standards define the physical layer (PHY), and the link layer specifications. An example of physical layer specification is the modulation and coding scheme (MCS) table that lists the supported modulation types, spatial streams, coding rates, bandwidths and data rates of a given PHY. An example of link layer specification is the medium access control (MAC) protocol that governs how the nodes share the wireless medium.

Table 7.1 lists some relevant physical layer specifications for 802.11b, n, and ac [87]. 802.11b uses Single-Input-Single-Output (SISO) scheme with direct-sequence spread spectrum (DSSS) modulation techniques. In contrast, 802.11n and 802.11ac are Multiple-Input-Multiple-Output (MIMO) schemes, based on orthogonal frequency division multiplexing (OFDM) modulation techniques. Single user MIMO is supported by 802.11n, while 802.11ac supports multi-user MIMO. The major advantage in terms of throughput and robustness of the channel from b to n/ac is given by the usage of multiple radios and antennas that allows transmitting different independent symbol at the same time (spatial multiplexing) or the same symbol on multiple antennas at the same time (spatial diversity). In particular, 802.11n/ac support transmit-beamforming (TxBF) at the downlink for single user (n) and multiple users (ac). By using TxBF, an access point

TABLE 7.1: Relevant 802.11b/n/ac physical layer specifications. f_c is the carrier frequency, λ is the wavelength, s_{dr} is the theoretical maximum throughput of the channel, n_S is the number of maximum independent data streams, TxBF indicates support for single-user (SU) or multi-user (MU) transmit-beamforming, d_i and d_o are the expected ranges for indoor and outdoor communications.

	Technology	Mod	f_c	λ	s_{dr}	n_S	TxBF	d_i	d_o
b	SISO	DSSS	2.4	12.5	11	N/A	N/A	35	140
n	SU-MIMO	OFDM	2.4, 5	12.5, 6	135	4	SU	70	250
ac	MU-MIMO	OFDM	5	6	780	8	MU	35	N/A

can optimize the transmission of the symbols to a device located in a particular region of space, given an estimate of the condition of the downlink channel. For a more detailed comparison among the three 802.11 amendments please refer to [84, 132].

7.2.3 Wireless Communications Metrics

Here we present the three wireless communication metrics used in our work:

- The *Signal-to-Noise-Ratio (SNR)* is the ratio between the power of the useful signal denoted with P, and the noise power σ^2. It is typically expressed in decibel dB, and it convertible from logarithmic to linear scale using: $10\log_{10}\mathrm{SNR} = \mathrm{SNR}_{\mathrm{dB}}$.
- The *Bit-Error-Rate (BER)* is the expected probability of error while decoding 1-bit at the receiver. The BER is not an exact quantity. It can be modeled and estimated according to different factors such as the modulation/coding schemes, the fading model and the number of antennas. Typically, 10^{-6} is considered a reasonable BER value, i. e. 1-bit error per Mbit.
- The *Packet-Error-Rate (PER)* is directly proportional to the BER, and it is computed as: $\mathrm{PER} = 1 - (1 - \mathrm{BER})^N$, where N is the average packet size in bits. In this work, we assume that one or more bit errors in a packet will lead to an incorrect link layer checksum. Packets with an incorrect checksum are not acknowledged by the (legitimate) receiver, and retransmitted by the sender.

7.3 Passive 802.11 Downlink Eavesdropping

We start this section introducing the system and attacker models. Then we present a theoretical discussion and a numerical analysis (based on 802.11 path loss models) to estimate the SNR and PER disadvantages of a passive eavesdropper in an 802.11n/ac (MISO) downlink, compared to an 802.11b (SISO) downlink.

7.3.1 System and Attacker Model

Our system model focuses on the *downlink* of indoor 802.11b/n/ac networks in infrastructure mode (e. g. access point that communicates with several wireless terminals), using Commercial-Of-The-Shelf (COTS) devices. The access point is equipped with

multiple antennas. The intended receiver and the attacker are equipped either with a single or multiple antennas according to the scenario. We are looking at the ratio of packets that the attacker successfully eavesdrop on the physical layer and we are agnostic to any encryption scheme used at the link layer or above. Attacks on those schemes are possible, but out of the scope of this work [27, 151].

The attacker is assumed to be *equipotent* to the intended receiver in terms of hardware and software capabilities. In particular, both use COTS devices, with a similar chipset, driver, feature set, and maximum throughput. With COTS devices we refer to wireless radios either built into laptops, smartphones, access point or USB dongles. We do not consider an attacker equipped with a software-defined-radio (SDR) or similar devices. We focus on a *passive* eavesdropper who wants to capture the downlink packets in real-time using her wireless card in monitor mode. We are not considering an attacker who is recording and post-processing the traffic offline. We assume an attacker that is static and we evaluate her eavesdropping performance at different distances from the sender. If the sender is using beamforming, we assume that the attacker is outside the beamforming region.

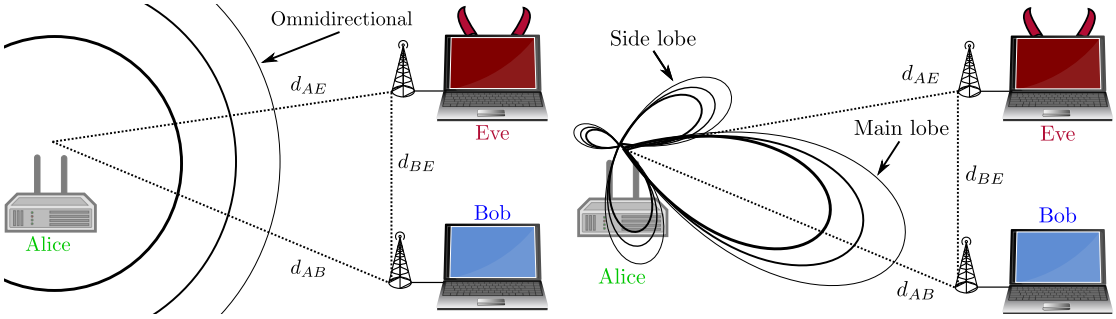
The effectiveness of the attacker is assessed from the Signal-to-Noise-Ratio (SNR) and the Packet-Error-Rate (PER) at her receiver. We chose PER as metric because we are mainly interested in the relative performance of eavesdropping on 802.11b vs. 802.11n/ac. As our passive attacker is unable to request retransmissions, the only chance to recover from bit errors would be to find the offending bit(s) and correct it using a checksum (possibly by brute force). We note that such corrections are expected to have significant cost for increasing number of flipped bits, and that the number of flipped bits is expected to quickly increase with distance. We plan to further investigate this in future work.

Without loss of generality and to simplify our discussion, we are considering an attacker focused on eavesdropping the downlink channel of one pair of transmitter and intended receiver. We understand that our attacker model is relatively weak (e. g. a single attacker, no SDR), however, given the lack of related experimental work and the number of involved moving parts, we decided to start with a simple scenario that is easy to evaluate (e. g. worst-case scenario for the passive eavesdropper). We look forward to investigate more complex attacker models in future work.

Finally, we present the notation used in this chapter. The access point is referred as Alice (the transmitter), the victim as Bob (the intended receiver), and the attacker as Eve (passive eavesdropper). We will use A, B, and E subscripts to identify quantities related to Alice, Bob, and Eve respectively. We use x to denote Alice's transmitted symbol, h for complex channel coefficients, and n for the noise at a specific receiver. The relative distances between Alice, Bob, and Eve are written as: d_{AB}, d_{BE}, d_{AE}. Alice is equipped with L antennas and L radios.

7.3.2 SISO and MISO Channels Eavesdropping

In this section, we analyze and compare two different eavesdropping scenario: 802.11b SISO downlink and 802.11n/ac MISO downlink. Then we derive two essential conclusions about passive eavesdropping in SISO vs. MIMO 802.11 downlinks.



(A) Omnidirectional radiation ($L = 1$). Eve's success depends on d_{AE}.

(B) Transmit-beamforming ($L > 1$). Eve's success depends also on d_{BE} and L.

FIGURE 7.1: 802.11b SISO (left) vs. 802.11 n/ac MISO (right) passive eavesdropping. Bob and Eve have one antenna. Dashed lines represent distances. Black circles and lobes represent omnidirectional and directional transmission ranges. Circles and lobes decreasing thickness represent the transmission power decay with respect to distance from the transmitter. Both channels are affected by random noise and fading.

802.11b SISO downlink. Figure 7.1a shows Eve trying to intercept the communication from Alice to Bob in an 802.11b SISO network. We can represent the signals received by Eve and Bob as:

$$y_E = x \cdot h_E + n_E \tag{7.1}$$

$$y_B = x \cdot h_B + n_B \tag{7.2}$$

Intuitively, it is possible to represent Alice's two-dimensional transmission coverage with concentric circles. In free space, the greater is the distance from the transmitter the higher is the transmitted power decay. While one might assume that every receiver inside these circles will be "in range" and receive all transmissions by Alice, this is not the case in practice. If circles are shown around transmitters, their radius commonly refers to a distance in which the average received signal strength is above a certain threshold. However, due to random deep fading (mostly due to multipath), the instantaneous received power will constantly vary. In other words, it is possible to "miss transmissions" while being in the outer circle, or even receive transmissions just outside the outer circle. In this case, Eve's success rate depends on her distance to Alice (d_{AE}) regardless of her distance to Bob (d_{BE}), and random channel characteristics. The SISO wireless channel is providing some sort of resiliency against eavesdropping that an attacker can compensate with other means (eg: increase receiver sensitivity, use directional antenna).

802.11n/ac MISO downlink. Figure 7.1b shows Eve attempting to intercept the communication from Alice to Bob in an 802.11n/ac MISO network. Alice is equipped with L antennas and uses transmit-beamforming. In this scenario, beamforming has been theoretically proven to provide resiliency against passive eavesdropping [83]. The received signals by Eve and Bob are as follows:

$$y_E = x \cdot g_E + n_E \tag{7.3}$$
$$y_B = x \cdot g_B + n_B \tag{7.4}$$

We can derive two benefits in terms of eavesdropping resiliency, one from g_B, and one from g_E. $\|g_B\|^2$ is defined as the *beamforming gain* and it is modeled by a Chi-squared random variable, with parameter $2L$ (being the sum of squared IID standard Gaussian random variables). Indeed, if $L = 2$ (Alice is using two antennas), then Bob's received signal will be the sum of two signals with independent fading paths. The correspondent beamforming gain is computed as:

$$\|g_B\|^2 = \|h_{B1}\|^2 + \|h_{B2}\|^2 \tag{7.5}$$

and this quantity is certainly greater (or equal) to $\|h_{B1}\|^2$ and $\|h_{B2}\|^2$. The net result is a better SNR at Bob's receiver with respect to the SISO case.

The second benefit arising from transmit-beamforming is encapsulated by g_E. Eve's ability to eavesdrop depends on two more factors with respect to the SISO case. Firstly, her distance from Bob (d_{BE}), and secondly the number of antennas used by Alice (L). This is a consequence of transmit-beamforming employed by Alice (the beamformer) towards Bob (the beamformee). Figure 7.1b shows Alice beamforming in the direction of Bob (e. g. inside the main lobe) while Eve is outside the main and the side lobes. This results in a smaller SNR at her receiver compared to the one of Figure 7.1a (given the same relative distances). Even if we decrease the distance between Eve and Alice, the disadvantage will still hold until Eve is outside the beamforming region. Furthermore, Eve's SNR will be inversely proportional to L because the more antennas are used by Alice to beamform, the more Alice can focus the beam towards a narrower but longer region in space [177].

7.3.3 Eavesdropper's Theoretical SNR Disadvantage in 802.11n/ac

In the previous section we argued that MISO beamforming from Alice to Bob will degrade Eve's eavesdropping performance according to d_{AE}, d_{BE}, and L. In this section, we will quantify the expected disadvantage of Eve in an 802.11n/ac network compared to an 802.11b network. We will estimate upper and lower bounds for the SNR at Eve's receiver with respect to L. We will provide numerical results for $L = 4$ to match the experimental setup of Section 7.4.1. We note that the bounds we are providing are not supposed to be *strict*—the actual SNR disadvantage will depend on many factors. Nevertheless, we compute the bounds based on the modeling assumptions to provide an intuition about the theoretically expected disadvantage.

Upper Bound. We start comparing high-level wireless channel characteristics of SISO and MISO channels. Table 7.2 lists the closed-form expressions for the SNR and the BER of SISO and MISO networks using BPSK modulation scheme. In general, we note that the number of antennas deployed by Alice (L) is playing a central role. If we fix the expected BER to 10^{-6}, then we can compute the minimum SNR for the SISO (57 dB) and the MISO case with $L = 4$ (16 dB). There is a notable difference in SNR of 41 dB

TABLE 7.2: SNR and BER of 802.11b (SISO) and 802.11n/ac (MISO transmit-beamforming with L antenna) using BPSK modulation scheme.

Metric	**SISO**	**MISO Beamforming**
SNR	$\|h\|^2 \frac{P}{\sigma^2}$	$\|g\|^2 \frac{P}{\sigma^2}$
BER	$\frac{1}{2}(1-\lambda)$	$\left(\frac{1-\lambda}{2}\right)^L \sum_{i=0}^{L-1} \binom{L+i-1}{i} \left(\frac{1+\lambda}{2}\right)^i \quad \lambda = \sqrt{\frac{\text{SNR}}{2+\text{SNR}}}$
DO	1	L

between the SISO and the MISO cases. We use 41 dB as an upper bound for the SNR disadvantage of Eve with respect to Bob.

Lower Bound. For the lower bound of Eve's SNR disadvantage, we use a standard formula to compute the beamforming gain in a MISO channel where Alice is using Cyclic Delay Diversity (CDD) with L antennas [116]. In this case, the beamforming gain in dB can be computed as follows:

$$\|g\|^2 = 10\log_{10}(L) \quad dB \tag{7.6}$$

Assuming a COTS access point with 4 antennas and a single receiving antenna, Bob's beamforming gain is 6 dB. As Eve's COTS radio will not benefit from the beamforming gain (being outside the main lobe) Eve's SNR disadvantage lower bound is thus 6 dB with respect to Bob.

Summary. We estimate that an 802.11n/ac downlink that is using transmit-beamforming with four antennas provides an reduction in the SNR of a passive eavesdropper (outside the main lobe, using a COTS receiver) that is bounded *between 6 dB and 41 dB*. The reduction in SNR at Eve's receiver depends on a deterministic and measurable factors: d_{AE} (distance between Alice and Eve) and L (number of antennas used by the Alice). We note that Eve's SNR variation depends also on channel (Rayleigh) fading, however this factor is not considered in our discussion because it randomly affects both Bob and Eve, providing no deterministic disadvantage to Eve. Given this theoretically expected disadvantage, the question now is: "*How does the eavesdropper SNR disadvantage translate to practical constraints on 802.11 passive eavesdroppers?*"

7.3.4 Numerical Path Loss Analysis

In this section, we present a numerical analysis using three indoor path loss models for 802.11 networks. The models includes both the 2.4 and 5 GHz bands and they are taken from [140]. We now describe their relevant parameters. In particular, d_{BP} is defined as the breakpoint distance between the transmitter and the receiver and it determines the cutoff span between LOS and NLOS channel condition. σ_{SF} represents the standard deviation in dB of the log-normal random variable that models the shadowing term of the path loss. s_{PL} represents the path loss slope before and after d_{BP}. Comma-separated values in the following list indicate values before and after the breakpoint distance:

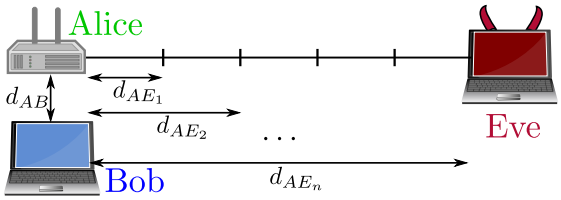


FIGURE 7.2: Setup used for our numerical analysis and for the experiments: Bob is at a fixed distance away from Alice, Alice is sending 802.11 traffic and Eve is passively eavesdropping from different (stationary) distances on a line perpendicular to Bob.

- **Model B**: Residential (e. g. intra-room, room-to-room).
  - $d_{BP} = 5$ m
  - $\sigma_{SF} = 3, 4$ dB
  - $s_{PL} = 2, 3.5$
- **Model D**: Office (e. g. large conference room, sea of cubes).
  - $d_{BP} = 10$ m
  - $\sigma_{SF} = 3, 5$ dB
  - $s_{PL} = 2, 3.5$
- **Model E**: Large office (e. g. multi-storey building).
  - $d_{BP} = 20$ m
  - $\sigma_{SF} = 3, 6$ dB
  - $s_{PL} = 2, 3.5$

Figure 7.2 shows the setup used for our numerical analysis and for the experiments. Bob is placed at a fixed distance away from Alice, Eve is placed at different (stationary) distances d_i from Alice, and Alice is constantly sending traffic to Bob. In a two-dimensional plane, Bob and Eve distance vectors are perpendicular to avoid Eve being in the main lobe when Alice is using transmit-beamforming. We note that in an indoor environment multipath is playing a major role than visual of RF line-of-sight conditions that is why we decided to keep altitude and angle constant and vary only the distance between Alice and Eve [41].

The path loss model function L_P is constructed considering the sum of a free-space loss component (L_{FS}), a shadowing log-normal component due to obstacles (S_F), and a post breakpoint distance component. All terms vary according to the distance d between the transmitter and the receiver. We used the following equations from [140]:

$$L_P(d) = \begin{cases} L_{FS}(d) + S_F(d) & \text{if } d \leq d_{BP} \\ L_{FS}(d_{BP}) + S_F(d) + 35\log_{10}\left(\frac{d}{d_{BP}}\right) & \text{otherwise} \end{cases} \tag{7.7}$$

$$L_{FS}(d) = 20\log_{10}(d) + 20\log_{10}(f) - 147.5 \tag{7.8}$$

$$S_F(d) = \frac{1}{\sqrt{2\pi}\sigma_{SF}} \exp\left(-\frac{d^2}{2\sigma_{SF}^2}\right) \tag{7.9}$$

Figure 7.3 and Figure 7.4 shows the predicted BER and PER for model B (Residential) vs. distance between the transmitter and the receiver. Solid lines represent results for 2.4 GHz and dash-dotted lines represent results for 5.0 GHz. Red lines represent Eve's expected BER and PER. The other lines represent Bob's expected BER and PER when Alice is using transmit beamforming with two (green lines) and four (blue lines)

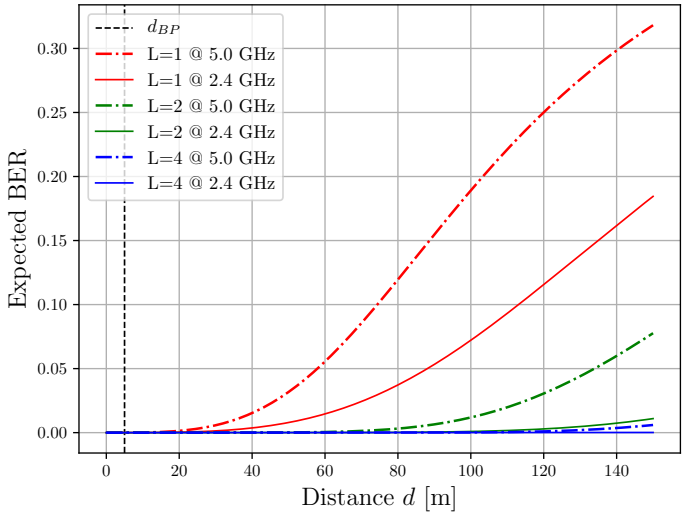


FIGURE 7.3: 802.11n Model B (Residential) expected BER estimation using BPSK. Red lines represent Eve. Green and Blue lines represent Bob when L=2 and L=4.

antennas. If we focus on the solid lines of Figure 7.4, then we note that a distance between Alice and Eve d_{AE} of 12.5 m is sufficient to drop Eve's expected PER from 0 to 0.5 (50% chance of decoding). Furthermore a d_{AE} of 20 m is sufficient to increase Eve's PER to 0.98 (0.2% chance of decoding). On the other hand, a d_{AB} of 142 m is required to experience a PER of 0.5 at Bob's receiver when Alice is using four antennas (L=4).

Figure 7.5 shows the result of our BER and PER analysis using model D. Figure 7.6 shows the result of our BER and PER analysis using model E. Figure 7.7 shows expected BER and PER for a free-space path-loss model.

7.3.5 Eavesdropping Analysis Summary

In this section, we argued that in 802.11n/ac downlink a passive eavesdropper (Eve) using a COTS radio will have a disadvantage in terms of SNR compared to an eavesdropper in an 802.11b downlink. This disadvantage is due to different features provided by recent 802.11n/ac such as MIMO, and spatial diversity. This disadvantage can be expressed in an SNR decrease at the eavesdropper receiver of 6-41 dB (depending on the chosen scenario). We also express this disadvantage in terms of the distance that the eavesdropper has to be closer to the sender to achieve the same PER as a legitimate receiver, which can reach up to 129.5 m. In contrast, there is no such distance disadvantage for the eavesdropper in 802.11b. Furthermore, we can express the disadvantage in terms of PER at the eavesdropper receiver compared to her distance from the transmitter (d_{AE}). For example, if d_{AE} is 12.5 m, then the PER of Eve is increased to 50%, and if d_{AE} is 20 m, then the PER of Eve is increased to 98%.

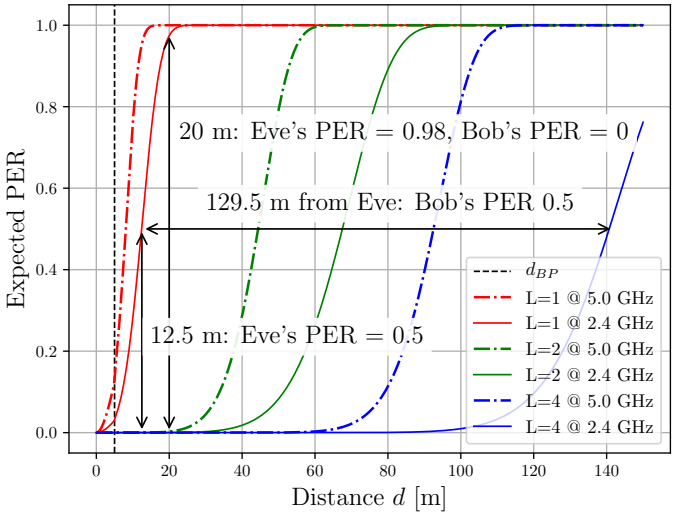


FIGURE 7.4: 802.11n Model B (Residential) expected PER estimation using BPSK. Red lines represent Eve. Green and Blue lines represent Bob when L=2 and L=4.

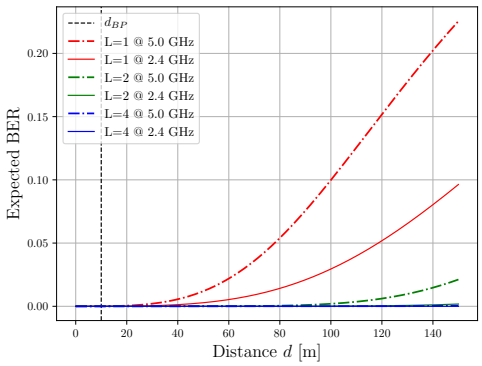
7.4 Experimental Validation

In this section, we present an experimental evaluation of COTS passive eavesdropping in 802.11b/n/ac downlink networks. The presented results are in line with the theoretical estimations from Section 7.3.

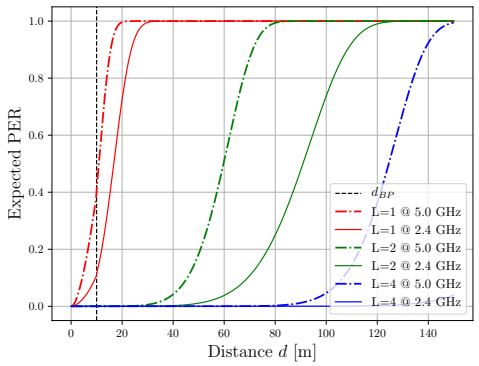
7.4.1 Experimental Methodology

We focus our experiments on SNR and PER measurements at Eve's receiver using the setup presented in Figure 7.2. We keep a ninety-degree angle between Bob and Eve to ensure that when beamforming is used Eve is outside the beamforming region. We vary the distance from Bob to Eve (d_{BE}) while keeping the distance from Alice to Bob (d_{AB}) constant. Table 7.3 lists the parameters that we fix for our experiments with a short description. As stated in Section 7.3.1 we are not using link-layer encryption (which does not influence our measurements). Figure 7.8 shows the layout of the indoor office environment where we conducted the experiments.

Our setup consists of an open access point (Alice) and a laptop (Bob) associated to it. The access point is a Linksys WRT3200 ACM device, equipped with four antennas and supporting 802.11a/b/g/n/ac. We installed the OpenWrt [171] operating system on the access point to have more configuration options at our disposal. For the 802.11b/n experiments (at 2.4 GHz), Bob's laptop runs Ubuntu 16.04 and has a TP-Link TL-WN722N wireless adapter. The adapter has a single antenna and supports 802.11b/g/n. Eve's laptop runs Ubuntu 16.04, and it uses the same TP-Link TL-WN722N wireless adapter. Eve's adapter is not associated with the access point and it tries to record the traffic from Alice and Bob, in monitor mode using `tcpdump`. Eve

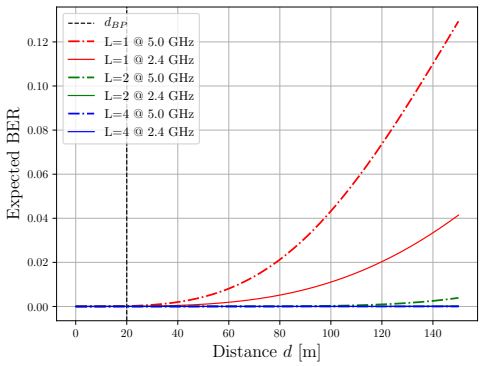


(A) Expected BER vs. Distance.

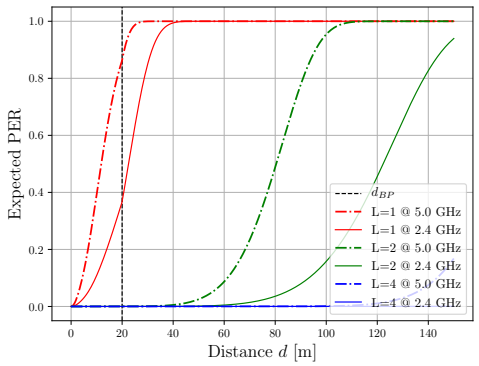


(B) Expected PER vs. Distance.

FIGURE 7.5: 802.11n Model D (office) BER/PER using BPSK. Red lines represent Eve. Green and Blue lines represent Bob when L=2 and L=4.



(A) Expected BER vs. Distance.

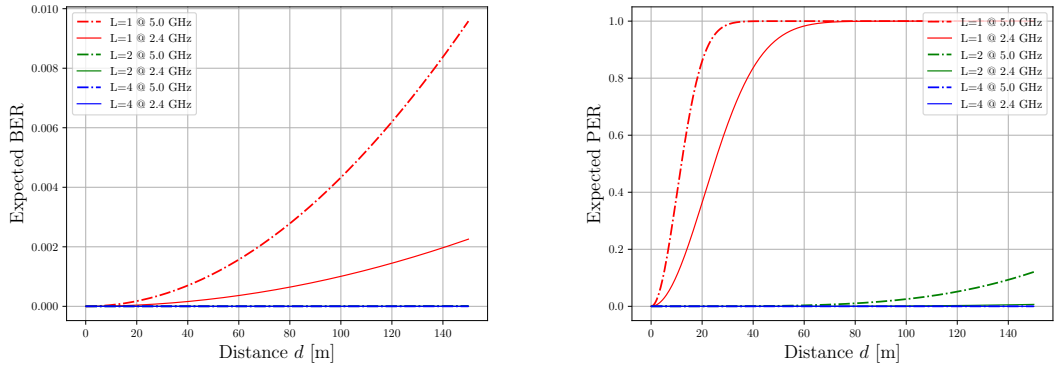


(B) Expected PER vs. Distance.

FIGURE 7.6: 802.11n Model E (Large office) BER/PER using BPSK. Red lines represent Eve. Green and Blue lines represent Bob when L=2 and L=4.

TABLE 7.3: Parameters used for the experiments.

Parameter	**Value**	**Description**
P_A [dBm]	23	Alice's transmitted power
N_0 [dBm]	-91	Mean noise power at the receivers
$Ch_{b/n/ac}$	11, 11, 36	Channels used for 802.11 b/n/ac
d_{AB} [m]	2	Fixed distance from Alice to Bob
$\vec{d}_{AE}$ [m]	$[2.5, 5.0, \ldots, 20]$	Eight distances from Alice to Eve



(A) Expected BER vs. Distance.

(B) Expected PER vs. Distance.

FIGURE 7.7: Free Space Path Loss (LOS) BER/PER using BPSK. Red lines represent Eve. Green and Blue lines represent Bob when L=2 and L=4.

listens to the same channel used by Alice and Bob (channel 11 for b and n, channel 36 for ac).

For the 802.11ac experiments (at 5 GHz), Bob's laptop runs Ubuntu 16.04 and uses an Asus USB-AC68 wireless adapter. The adapter has a 3x4:3 antenna configuration and supports 802.11a/b/g/n/ac. Eve's laptop is a MacBook Pro with an inbuilt adapter with 3x3:3 configuration compatible with 802.11a/b/g/n/ac. We use a different adapter for Eve because the Asus adapter could not be put into monitor mode due to some issues with its driver. The other parameters remain the same as in the 802.11b/n experiments.

For all the experiments, we vary Eve's distance from Bob and we obtain pcap traces of the packets transferred from Alice to Bob. The distance between Alice and Bob (d_{AB}) is fixed at 2 m. We used `iperf` to generate UDP downlink traffic. We decided to use UDP to avoid retransmissions at the transport layer. The PER is computed based on the number of received UDP packets with a valid UDP checksums. We acknowledge that this approach slightly underestimates the actual PER, as packets with a valid UDP checksum but incorrect link-layer checksum (FCS) might be included in this calculation.

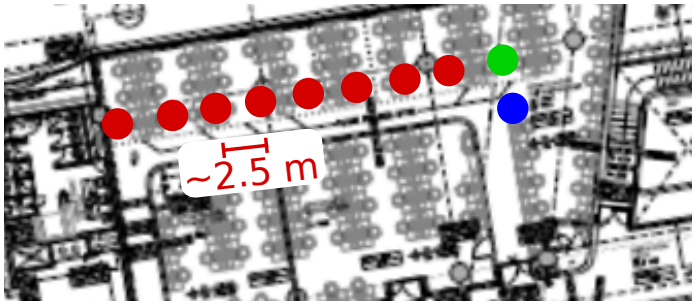


FIGURE 7.8: The layout of the indoor office environment where we conducted the experiments. The green and blue dots indicate the location of Alice and Bob. The red dots indicate the positions of Eve.

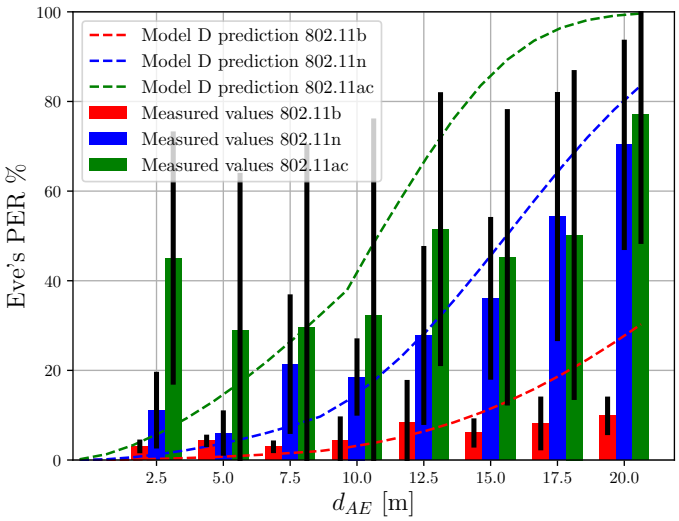


FIGURE 7.9: Eve's measured PER (bars) vs. Model D predicted PER (dashed lines).

The transmission power of Alice is set to 23 dBm. From the experiments, we are able to obtain the traces from Eve at d_{AE} between 2.5 m and 20 m, using increments of 2.5 m. We do not change the orientation of Eve with respect to Alice in our tests to better compare the results. All the devices have the same fixed elevation, without a visual line-of-sight path between them. The information about the recorded traffic is obtained from the 802.11 PHY radiotap headers. In the subsequent section we will compare the experimental results with our estimations from the path loss model D (Office). Figure 7.5 shows the predicted BER and PER curves at Eve's receiver (red curves), and at Bob's receiver when Alice is using transmit-beamforming with two (green curves) and four antennas (blue curves).

7.4.2 Comparison between 802.11b/n/ac Networks

For the comparison between 802.11b/n/ac networks, we set a 2.4 GHz band for 802.11b/n and a 5 GHz band for 802.11ac. To extract the results we capture packets both from Eve and Alice. We measured two parameters—the PER of the passive eavesdropper, and her SNR. We compute Eve's PER by comparing her pcap traces with the ones from Alice. We compute the SNR by dividing the extracted signal strength values by the average channel noise. We computed the average channel noise using noise measurements from the access point, and it resulted in -91 dBm. We repeat the same experiments with the same distances 30 times and we average the results to obtain mean SNR and PER values, and related errors (standard deviations).

Figure 7.9 shows Eve's PER measurements and estimated values for d_{AE} varying from 0 m to 20 m. The red/blue/green bars indicate the experimental results for 802.11b/n/ac, respectively. The dotted lines indicate the predicted estimates (from model D). It can be observed that Eve's PER is almost always *increasing* from b to n and from n to ac. In particular, the PER starts to increase significantly when d_{AE} is greater

TABLE 7.4: Results from 802.11n and 802.11ac experiments. d_{AE} is the distance from Alice to Eve in meters. n_r is the total number of runs. μ_p is the average number of UDP packets sent by Alice per run. μ_{PER} and σ_{PER} are the Eve's PER means and standard deviations measured in our experiments for 802.11n (n subscript) and 802.11ac (ac subscript).

$d_{AE}[m]$	n_r	μ_p	μ_{PER_n}	σ_{PER_n}	$\mu_{PER_{ac}}$	$\sigma_{PER_{ac}}$
2.5	30	894.00	11.13	8.56	45.07	28.25
5.0	30	894.00	6.02	5.06	28.94	35.13
7.5	30	894.00	21.39	15.57	29.64	40.86
10.0	30	894.00	18.52	8.63	32.33	43.88
12.5	30	894.00	27.79	19.97	51.52	30.55
15.0	30	894.00	36.08	18.16	45.23	33.07
17.5	30	894.00	54.33	27.79	50.20	36.80
20.0	30	894.00	70.32	23.46	77.01	28.80

than 15 meters. While such (relatively small-scale) experiments will hardly produce the exact same results as our theoretical analysis, we observe that the increase in PER that was predicted by us, for even relatively short distances of around 20 m, can be observed in practice. In particular, our D model predicted a PER for Eve in an 802.11n downlink of around 78% when $d_{AE} = 20$ m, and in our experiments the average PER was around 70%. For convenience, we tabulate in Table 7.4 the numerical results of Figure 7.9.

Figure 7.10 shows Eve's mean SNR varying her distance (d_{AE}) from Alice for 802.11b (red bars), 802.11n (blue bars) and 802.11ac (green bars). It can be observed that Eve's SNR in 802.11n/ac is always *smaller* than in 802.11b—an effect that we assumed to be caused by advanced 802.11n/ac physical and link layer features (such as TxBF).

7.4.3 Bob vs. Eve in 802.11n/ac

We conducted a second set of experiments targeting Bob in order to compare his SNR and PER with respect to Eve's SNR and PER in 802.11n/ac networks. In this case, we increased Bob's distance from Alice. As in the previous experiments, we start from 2.5 m and we end at 20 m, with increments of 2.5 m. Bob is placed at the same location that Eve was placed in the previous case. In this scenario, we are expecting that Bob would benefit from 802.11n/ac features. We are not showing the plot for Bob's PER compared to the one Eve experienced in Figure 7.9. This is because we observed that Bob's PER is very low (less than 1%), and yet not comparable with Eve's PER. This confirms our assumption that the intended receiver experiences significantly *lower* PER than a passive eavesdropper in 802.11n/ac networks.

Interestingly, as we can see from Figure 7.11a, the mean SNR of Bob and Eve at various distances are relatively close. In particular, Bob's SNR in 802.11n is always higher than Eve's SNR (as expected). However in the 802.11ac case, we measure a

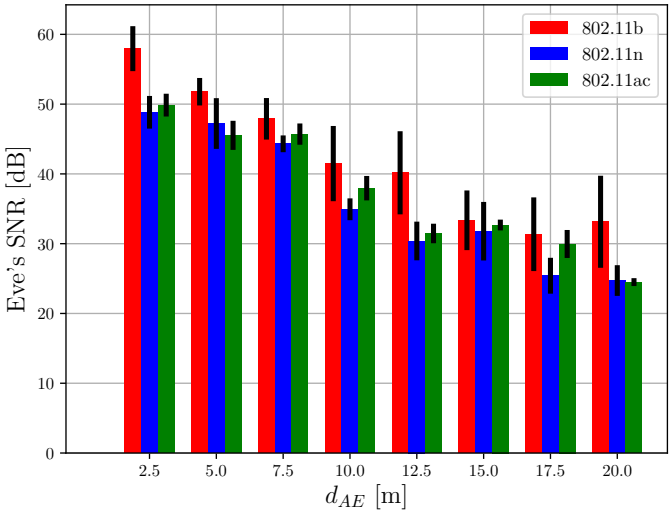


FIGURE 7.10: Eve's measured SNR with respect to d_{AE}.

higher SNR for Eve than Bob. We assume that this is an artifact resulting from the fact that Eve's SNR is reported only for successfully received packets.

7.4.4 Eve's PER and PER Thresholds

We note that even a small decrease in PER could affect a passive eavesdropper depending on the type of exchanged traffic. That is why we decided to analyze Eve's PER compared to different PER thresholds and distances d_{AE}. Table 7.5 shows the results of our analysis for 802.11b/n/ac. For example, if we fix the threshold to 15%, then Eve's PER in 802.11ac is above the threshold in at least 33% of all cases. The same holds for 802.11n except for the 5m measurement. With regards to 802.11b, fixing the same 15% threshold, we note that Eve's PER does not exceed the threshold in more than 16% of all cases. This is another way to confirm our predictions about 802.11n/ac passive eavesdropping.

7.4.5 Eve with Two COTS Radios in 802.11n

We argued earlier that attackers with COTS radios will not be able to benefit from advanced 802.11n/ac physical layer and link layer features, and discussed an attacker with a single COTS radio. We now discuss a passive eavesdropper with multiple COTS radios in an 802.11n downlink. The attacker aggregates the eavesdropped packets to reduce the number of packets lost (e. g. due to deep fading). In Figure 7.11b, we show the PER for an attacker with *two* COTS radios. The radios are placed at a distance of 50cm from each other (to avoid mutual coupling). Note that we used a different data set from the previous experimental section, and we repeated this experiment 30 times. It can be observed that such a scheme *reduces* the number of lost packets for the attacker (as expected). However, the PER in the aggregated case is still higher than the 802.11b one, especially at distances greater than 5m. For a threshold PER of 15%, the PER for

TABLE 7.5: Eve's PER vs. PER Thresholds vs. Distances. Columns represent different distances from Eve to Alice (d_{AE}). Rows represent different PER thresholds. Comma-separated values represent the rounded-down percentage of experimental runs where Eve's PER was above the threshold for 802.11b, n, and ac.

	5.0 [m]	**7.5 [m]**	**10.0 [m]**	**12.5 [m]**	**15.0 [m]**	**17.5 [m]**
5%	33, 36, 50	10, 100, 33	20, 100, 33	36, 100, 90	43, 100, 80	60, 100, 96
10%	0, 26, 40	0, 73, 33	6, 83, 33	30, 90, 83	16, 96, 70	30, 100, 70
15%	0, 3, 36	0, 56, 33	6, 53, 33	16, 66, 76	0, 90, 63	13, 100, 60
20%	0, 0, 33	0, 43, 33	3, 36, 33	13, 53, 56	0, 76, 56	6, 96, 53
25%	0, 0, 33	0, 30, 33	3, 26, 33	10, 40, 53	0, 66, 56	0, 83, 53
30%	0, 0, 33	0, 20, 33	0, 13, 33	6, 30, 50	0, 60, 43	0, 73, 53
35%	0, 0, 30	0, 13, 30	0, 3, 33	3, 30, 43	0, 56, 43	0, 63, 50
40%	0, 0, 30	0, 10, 30	0, 0, 33	0, 23, 43	0, 40, 43	0, 53, 46
45%	0, 0, 26	0, 10, 30	0, 0, 33	0, 16, 43	0, 26, 43	0, 46, 46
50%	0, 0, 23	0, 6, 26	0, 0, 33	0, 16, 33	0, 16, 36	0, 43, 46

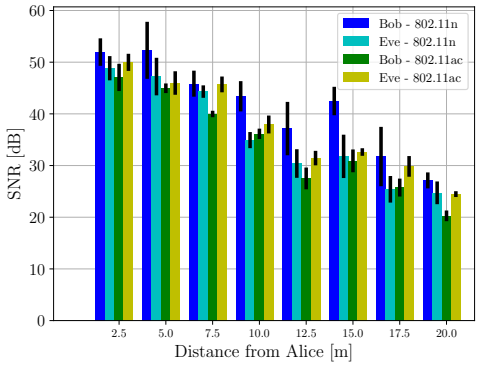
the aggregated case is higher than the threshold in about 23% of the runs, compared to 6% for 802.11b.

7.4.6 Summary of 802.11b/n/ac Experiments

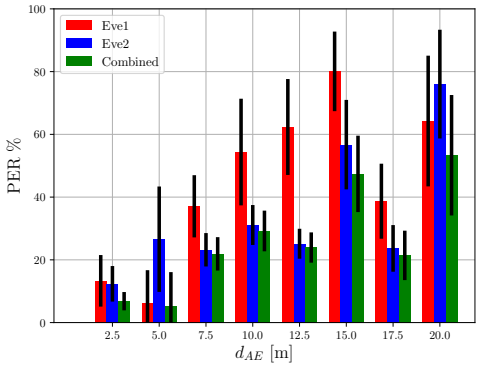
Overall, we were able to experimentally confirm our main findings: a) there is a significant increase of the PER of a passive eavesdropper attacking 802.11n/ac networks compared to 802.11b ones. In our experiments, the difference was approximately 60% increased PER for 802.11n and 70% increased PER for 802.11ac at 20 m distance. In addition, the PER rises from around 12.5 m onward, similar to our predictions based on the theoretical analysis. We also confirmed that the PER experienced by the attacker is related to the non-cooperating Alice. In particular, legitimate receivers at the same locations were able to receive traffic with close to zero PER.

7.5 Related Work

There are several empirical studies for 802.11 networks. Most of them focus on specific link layer [122] or physical layer [161] features. There are also more generic empirical studies, for example about enterprise WLAN [36], intrusion detection [102], denial of service [21] co-existence [80] and signal manipulation [143] Anyway, those studies neither focuses on wireless security nor compares end experimentally evaluate eavesdropping in various 802.11 networks.



(A) 802.11n and 802.11ac SNR comparison between Bob and Eve at different distances from Alice.



(B) 802.11n PER of Eve using two COTS radios. The green bars represent combined PER.

FIGURE 7.11: Experimental results from Section 7.4.3 (a) and Section 7.4.5 (b).

An interesting aspect of eavesdropping is to study how to optimally place a set of antennas in a multiple users scenario to obtain the maximum amount of private information. In [183] Wang et al compare co-located vs. distributed eavesdropping schemes performance with respect to Eve's number of antennas and the presence of a guard zone. The de-facto standard countermeasure against eavesdropping (complementary to physical layer security) is cryptography. Several studies were done to secure [14] and break [27, 151] cryptographic systems used by 802.11 such as WEP and WPA.

7.6 Conclusions

In this work, we investigated the impact of novel 802.11n/ac features over a passive eavesdropper using COTS devices. We focused on downlink networks in infrastructure mode. We performed a theoretical discussion, a numerical simulation and several experiments comparing the Signal-to-Noise-Ratio and Packet-Error-Rates of the eavesdroppers in 802.11b/n/ac. We showed that theoretically the eavesdropper's effective SNR is decreased by 6-41 dB in 802.11n/ac networks with four antennas ($L = 4$), which translates to a Packet-Error-Rate increase of up to 98% at a distance of 20 m between sender and eavesdropper. To obtain same Packet-Error-Rates as in a legitimate receiver, the attacker's maximal distance has to be reduced by 129.5 m in the case of 802.11n. In our practical experiments, we showed that the predicted effects occur in practice (although we were not able to exactly reproduce the theoretic predictions). Eve's PER for n was at least 20% higher than for b, and more than 30% for ac (with increasing impact over distances greater than 10m).

We conclude that the evolution of the 802.11 standard actually introduced several physical and link layer features, such as MIMO and spatial diversity, that might degrade the performance of a passive eavesdropper. If properly exploited those features could be used as a part of a defense-in-depth strategy as a complement to well-known

eavesdropping defense mechanism. Nevertheless, we understand that further investigations are necessary to characterize the benefits against stronger attacker models and in more complex scenarios. We leave those discussions to future work.

Chapter 8

Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android

Keywords: Google, Android, WiFi, Bluetooth, RE, Attacks.

8.1 Introduction

Google's Nearby Connections API enables Android (and Android Things) developers to offer proximity-based services in their applications. A proximity-based service allows users of the same application to share (sensitive) data only if they are "nearby", e. g. within Bluetooth radio range. The API uses Bluetooth BR/EDR, Bluetooth LE and Wi-Fi and it claims to automatically use the best features of each depending on the type of communication required. For example, it uses Bluetooth for short-range low-latency communications and Wi-Fi for medium-range high-bandwidth ones. The API provides two different connection strategies (`P2P_STAR` and `P2P_CLUSTER`), that allows clients (discoverers) and servers (advertisers) to be connected using different topologies.

The Nearby Connections API is implemented as part of Google Play Services. Google Play Services is a proprietary, closed-source and obfuscated library that allows Google to provide the same services to any Android and Android Things application, regardless of the version of the operating systems. The API is compatible with any Android device, version 4.0 or greater, and it is updated by Google without user interaction [5]. An attacker who can exploit this API can target (at least) any application using Nearby Connections in any Android mobile and IoT device. This implies a large attacker surface and represents a critical threat with severe consequences such as data loss, automatic spread of malware, and distributed denial of service.

The design specifications and implementation details of the Nearby Connections API are not publicly available. The only public source of information about the library is a few blog posts detailing sporadic security guarantees [72, 69]. According to these sources, the API uses encryption by default, but it does not mandate user authentication. The API automatically manages and uses multiple physical layers and this does not sound trivial. The API uses a custom application layer connection mechanism. A device can simultaneously be a client and a server, and can connect to different applications at the same time. A Nearby Connections application is uniquely identified

by a string named `serviceId` and clients and servers with different `serviceId` (or connection strategies) will not be able to connect.

Proximity-based services similar to the Nearby Connections API have been investigated in the context of wireless sensor networks for a long time, together with related security challenges [99, 141, 97]. In particular, eavesdropping and wormhole attacks are often discussed, which would allow the attacker to read or manipulate the traffic exchanged by nearby devices. Typically, such attacks are considered on link or network layer, i.e., on the routing or path finding protocols. As Nearby Connections is providing an application-layer service, the setting is different to most established work in the field, although similar security challenges does apply. In addition to attacks on routing, authentication of (mobile) users is known to be challenging, together with key exchange to establish a secure channel [35, 110].

In this work, we assess the security of the Nearby Connections API. We analyze the API by reverse-engineering its closed-source and obfuscated implementation. To perform this task we use advanced techniques such as dynamic binary instrumentation and manipulations of raw packets. We develop compatible implementations of Nearby Connections client and server. Based on the knowledge gained, we identify and implement several attacks. The impact of these attacks ranges from intercepting and decrypting application-layer data (using man-in-the-middle or impersonation), to forcing the establishment of TCP connections between the victim and arbitrary (non-nearby) devices. In addition, the attacker is able to introduce system wide default network routes on the victims' devices. As a result, the attacker is able to redirect a victim to an access point under his control and gains access to all the Wi-Fi traffic of the victim, including the traffic generated by applications that are not using the Nearby Connections API.

We summarize our main contributions as follows:

- We reverse engineer and perform the first security analysis of the closed-source and obfuscated Nearby Connections API.
- We identify and perform several attacks grouped into two families: connection manipulation and range extension attacks. The attacks can be performed by very weak adversaries and have severe consequences such as remote connection manipulation and data loss.
- We design and implement REarby, a toolkit that enables reverse engineering and attacking the Nearby Connections API. We released parts of the toolkit as open source in our proof of concept code[1].

Our work is organized as follows: in Section 8.2, we introduce the Nearby Connections API. We present a security analysis of the API based on our reverse engineering in Section 8.3. In Section 8.4, we describe the connection manipulation and range extension attacks. The implementation details of REarby are discussed in Section 8.5. We present the related work in Section 8.6. We conclude the chapter in Section 8.7.

[1]Repository at https://github.com/francozappa/rearby.

8.2 Background

8.2.1 Introduction to the Nearby Connections API

The Nearby Connections API is used to develop *proximity-based* applications. These applications provide services to users within radio range (approx 100 m) [71] and consider these users as "nearby". Typical use cases of the API are: file sharing, gaming, and streaming of content. The API enables proximity-based services using a combination of three wireless technologies: Bluetooth BR/EDR, Bluetooth LE and Wi-Fi. Bluetooth BR/EDR stands for basic rate and extended data rate and it is typically used by high-end mobile devices. Bluetooth LE stands for low energy and it is typically used by low-end and high-end mobile devices [162]. In the rest of the chapter we indicate Bluetooth BR/EDR with Bluetooth and Bluetooth LE with LE. For more information about the differences between the two refer to [162].

The Nearby Connections API is available on Android and Android Things. Android Things is a new operating system based on Android developed by Google that targets IoT devices. In this work, we focus on the Android implementation of the API. The latest major release of the API was introduced in June 2017 in Global Positioning System (GPS) version 11.0 [72]. The GPS library is a core proprietary product of Google, and relies partly on the *security through obscurity* model. The main functional benefit and potential security weakness of the GPS library is that it allows its services (including Nearby Connections) to be usable by any application, across different Android versions from 4.0 onwards. The updates of the GPS library are pushed by Google and do not require user interaction to complete [5].

In a nearby connection there are two types of actor: the *discoverer* and the *advertiser*. The former acts as a client, while the latter acts as a server. Figure 8.1 describes the actions performed by these actors while using the Nearby Connections API. The client attempts to discover a service identified by a `serviceId`. The server announces the service (`serviceId`) along with a name (`ncname`), using one of two different strategies described in a moment. Two actors are allowed to connect if they use the same strategy and `serviceId`. A single device can discover and advertise different services at the same time. Each `serviceId` is meant to uniquely identify an application. Google suggests to set it equal to the package name of the application [68].

If the client discovers the server then he sends a connection request to the server and the actors can optionally authenticate themselves. The actors mutually accept to connect and then the connection is established. The connection is *always* requested, initiated and established over Bluetooth. Once the connection is established, the actors optionally switch to a different physical layer, e. g. to Wi-Fi, and then they start exchanging (encrypted) data payloads. The API provides three types of payloads: `BYTE`, `FILE`, and `STREAM`. The first type is used to transmit chunks of bytes, the second files and the third streams of data. Each payload type has a related proximity-based service associated, e. g. use `FILE` payloads in a file-sharing application. Each actor addresses a payload to a receiver with a unique four-digit string called `endpointId`. The nearby connection is closed whenever one of the two actor disconnects.

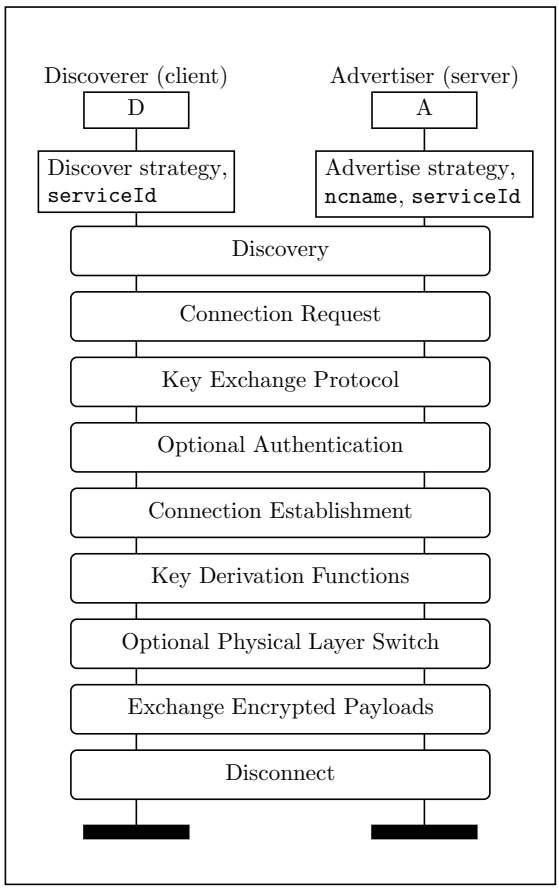


FIGURE 8.1: The Nearby Connections API has two types of actors: the Discoverer (client) and the Advertiser (server). It uses application layer encryption and optional user authentication.

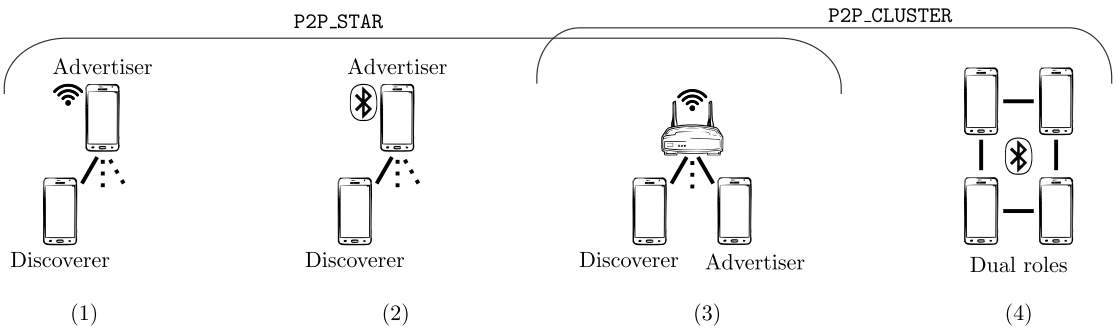


FIGURE 8.2: Nearby Connections connection strategies using Bluetooth and Wi-Fi. On the left, three `P2P_STAR` topologies; on the right, two `P2P_CLUSTER` topologies. In each case, all actors use the same `serviceId`.

8.2.2 Nearby Connections Strategies

The nearby connection strategy dictates the connection topology and the physical layer switch. At the time of writing, the Nearby Connections API provides two strategies: `P2P_STAR` and `P2P_CLUSTER`. There is also a third one called `P2P_POINT_TO_POINT` that it is still not public. In Figure 8.2 we show three examples of `P2P_STAR` links and two `P2P_CLUSTER` links. These links are established after the actors already completed all the phases from Figure 8.1 up to the optional physical layer switch. The `P2P_STAR` strategy has three types of links: (1) the advertiser acts as a soft access point and the discoverer connects to it; (2) the discoverer is the master and the advertiser is the slave of a Bluetooth network; and (3) both actors are connected to the same access point and they exchange payloads through it. There are only two options for the `P2P_CLUSTER` strategy: (3) is the same as for the `P2P_STAR` strategy and (4) several actors can connect to each other using Bluetooth in a mesh-like network.

Hence, `P2P_CLUSTER` allows the connection of multiple discoverers and advertisers while `P2P_STAR` allows only one advertiser to be connected with multiple discoverers. Google recommends to use `P2P_STAR` for higher throughput and `P2P_CLUSTER` for more flexible network topologies. In any case, the actors can exchange payloads without being connected to the Internet and two discoverers always communicate through an advertiser (even when using the `P2P_CLUSTER` strategy).

8.3 Reversing and Analyzing Nearby Connections

In this section we describe our understanding of the Nearby Connections API after reverse engineering its implementation on Android. Our main goal is to perform a security assessment of the API because of its wide attack surface and complex interactions between wireless technologies. Unfortunately, the implementation of the API is proprietary, closed-source, and obfuscated. To overcome these obstacles we developed REarby a toolkit to reverse engineer the API, its implementation is presented Section 8.5. For the remainder of this work, we refer to the target of our analysis as "the library". We note that, without access to the source code or specification of the library, we cannot claim that our findings are always complete.

The analysis of the library allowed us to identify and perform several attacks that we present in Section 8.4. In particular, we use details of the authentication and interactions between advertisers and discoverers (Section 8.3.1) to perform attacks in which we impersonate them and we manipulate Nearby Connections traffic. Details of the Nearby Connections keep-alive mechanism (Section 8.3.5) are required for range extension attacks. The physical layer switch (Section 8.3.6) is exploited to manipulate system-wide routing tables of victims.

8.3.1 Discovery and Connection Request

A nearby connection is always requested using Bluetooth, regardless of the nearby connection parameters. Discovering and advertising are done in a deterministic and predictable way without using encryption. An attacker who posses a clone of the library can pretend to be any advertiser and discoverer of any application, and he can use any Bluetooth compatible device to request a connection. The Bluetooth connection

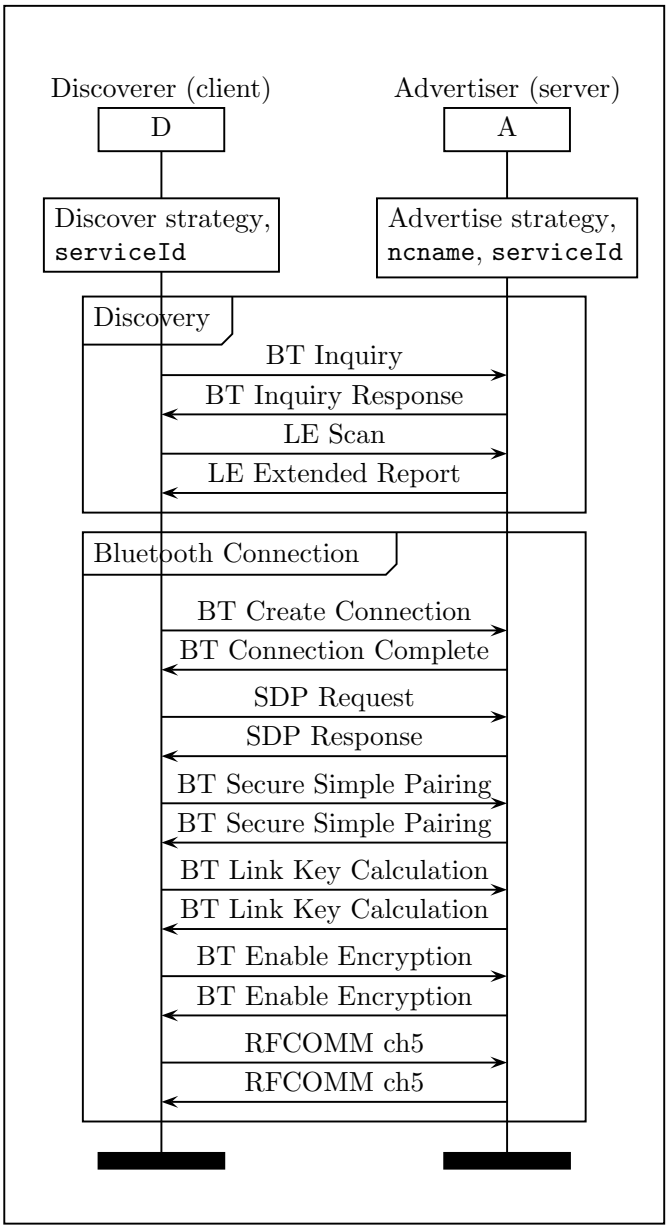


FIGURE 8.3: Nearby Connections Request. BT is Bluetooth BR/EDR. Secure Simple Pairing provides link-layer encryption.

uses Secure Simple Pairing (SSP) with a link key that is not authenticated and not persistent. Indeed, an attacker could insert himself as a man in the middle in the Bluetooth link and he can force the re-establishment of the link key at any time.

An example of a nearby connection request is shown in Figure 8.3. The advertiser (server) advertises a strategy, a `serviceId` and a `ncname`. The discoverer (client) discovers a strategy and a `serviceId`. To find each other, both have to look for the same `serviceId`, and use the same strategy. The server to be discovered changes its Bluetooth name (btname) and sets custom LE extended report. The btname is changed to a string that depends on the strategy, the `endpointId`, the `serviceId`, and the `ncname`. btname is computed as follows:

$$\text{btname} = \text{unpad}(\text{b64encode}(\text{strategy\_code} \parallel \texttt{endpointId} \parallel \text{SHA256}(\texttt{serviceId})[:3] \parallel \text{separator} \parallel \texttt{ncname}))$$

The strategy_code is `0x21` for `P2P_STAR` and `0x22` for `P2P_CLUSTER`. The SHA256 of `serviceId`[:3] are the first three bytes of the SHA256 digest of the `serviceId`. The unpad function is used to remove the padding characters (=) from the base64 encoded string. For example, a server with strategy `P2P_CLUSTER`, `serviceId` = `sid`, `endpointId` = `aXCV` and `ncname` = `name` advertises with the following btname: `IjR1ZEE0s2QAAAAAAAAABG5hbWU`. The length of btname depends on `ncname`, in our experiments we discovered that the maximum length of the name is 131 bytes. The btname is easy to spot (by an attacker) because it always starts with `I` and contains the `AAAAAAAAA` separator. On the LE side, the same parameters are used in a similar way to set the LE extended report. Some devices (such as the Nexus 5) do not support LE extended reports and only use Bluetooth while advertising. The client (while discovering) sends Bluetooth inquiries and enables LE scanning. The server sends back Bluetooth inquiry responses containing btname and LE extended reports. The client discovers the server (endpoint) through these responses and establishes a Bluetooth connection with it.

After the Bluetooth connection is established, the client sends a service discovery protocol (SDP) request using a custom `uuid`. SDP is a protocol used to discover Bluetooth services. The information about each service is obtained by sending a SDP request containing its correspondent universally unique identifier (`uuid`) [162]. The nearby connection custom `uuid` is computed from the MD5 digest of the `serviceId` and some extra string manipulations. For example, if `serviceId` = `sid` then the `uuid` is `b8c1a306-9167-347e-b503-f0daba6c5723`. The client receives an SDP response containing the following fields: `name` = `serviceId`, `host` = Bluetooth address of the server and RFCOMM `port` = 5. RFCOMM is a serial cable emulation transport protocol based on ETSI TS 07.10 providing similar guarantees of TCP [162].

The client uses Secure Simple Pairing (SSP), with optional Secure Connections, to share a secret with the server. The Secure Connections mode is enabled if both radio chips support it. The client and the server compute the same link key, and they agree on enabling link-layer encryption. Finally, the client establishes a link-layer encrypted RFCOMM connection with the server, always on port 5.

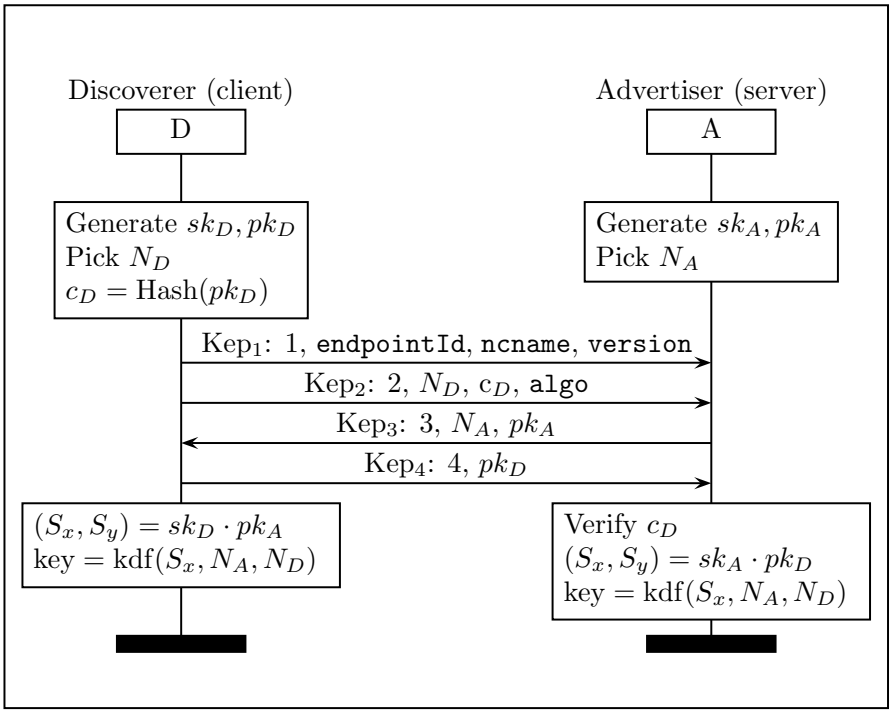


FIGURE 8.4: Nearby Connections Key Exchange Protocol (KEP) based on ECDH (secp256r1). `algo` is always `AES_256_CBC-HMAC_SHA256`.

8.3.2 Key Exchange Protocol (KEP)

We found that the library uses a custom key exchange protocol (KEP) based on elliptic-curve Diffie Hellman (ECDH) on the secp256r1 (NIST P-256) curve. ECDH is a good choice for mobile embedded system because it is faster and requires shorter keys than finite field Diffie-Hellman. The secp256r1 curve is recommended by NIST [18], however some crypto experts have questioned the security of this curve [22]. The key exchange protocol consists of four packets that we refer to as: Kep_1, Kep_2, Kep_3, and Kep_4. Table 8.1 lists their most relevant fields. This protocol provides several security guarantees, e. g. fresh shared secrets, negotiation of strong crypto primitives for confidentiality and integrity and usage of sequence numbers and sanity checks of the elliptic curve points. However, we identified a number of issues, e. g. lack of a standard key derivation function (such as HKDF), weird usage of nonces and commitments, and transferring of useless key material (y coordinate of the public keys).

The key exchange protocol is shown in Figure 8.4. The client generates a key pair sk_D, pk_D and the server generates a key pair sk_A, pk_A. sk is the private (secret) key, and pk is the public key. Each public key is a point on the secp256r1 curve. The client and the server generate two 32 byte random nonces N_D and N_A. The client builds Kep_1 and Kep_4. The relevant fields of Kep_1 are: 1 (sequence number), `endpointId`, `ncname` and what we believe is the Nearby Connections version number (`version`). In our experiments we observed protocol version 0x2 and 0x4. The relevant field of Kep_4 are: 4 (sequence number) and pk_D (the client's public key).

TABLE 8.1: Main Fields of the Key Exchange Protocol packets. (G_x, G_y) is the generator point for the ECDH curve.

Packet	Field	Description	Default value(s)
Kep_1	sn	Sequence number	1
	`endpointId`	Discoverer id	None
	`ncname`	Discoverer name	None
	`version`	Protocol version	0x02, 0x04
Kep_2	sn	Sequence number	2
	N_D	Nonce	Random
	c_D	Commitment	SHA512(Kep_4[4:])
	`algo`	Negotiated ciphers	`AES_256_CBC-HMAC_SHA256`
Kep_3	sn	Sequence number	3
	N_A	Nonce	Random
	x_A	x-coord of pk_A	x-coord from $(G_x, G_y) \cdot sk_A$
	y_A	y-coord of pk_A	y-coord from $(G_x, G_y) \cdot sk_A$
Kep_4	sn	Sequence number	4
	x_D	x-coord of pk_D	x-coord from $(G_x, G_y) \cdot sk_D$
	y_D	y-coord of pk_D	y-coord from $(G_x, G_y) \cdot sk_D$

The client computes an hash of pk_D (based on SHA512) that should be a commitment on his public key. We define the commitment as c_D. Then, the client builds Kep_2 that has the following relevant fields: 2 (sequence number), N_D, c_D and a string that we define as `algo`. The value of `algo` is fixed to `AES_256_CBC-HMAC_SHA256`. This means that the nearby connection application layer uses encryption and message authentication codes, and that strong crypto primitives are used: AES256 in CBC mode, and HMAC with SHA256. The presence of `algo` could indicate that the developers are planning to introduce cipher negotiation as a feature, otherwise there seems little point in exchanging this information. The server builds Kep_3 that has the following relevant fields: 3 (sequence number), N_A, and pk_A (the server's public key). Note that Kep_4 is build before Kep_2 because the latter contains c_D (commitment) that is computed over the former.

After that, the network traffic takes place (over link layer encrypted RFCOMM). The client sends Kep_1 and Kep_2 to the server, the server answers with Kep_3, and the client sends Kep_4. In our experiments these packets are always exchanged in this order. Sometimes, Kep_1 is split and transmitted using two sequential RFCOMM packets. The server verifies the client's commitment using c_D and Kep_4. Afterwards, both nodes compute the (same) secret point in the curve, defined as (S_x, S_y), by multiplying their own private key with the public key of the other. The x coordinate of the secret point is used as key (shared secret). We refer to this as S_x. The library does not use any (recommended) ECDH key derivation functions such as HKDF or NIST-800-56-Concatenation-KDF [18]. The details about how we managed to discover it are presented in Section 8.5. After the client and the server completed the KEP, the nearby connection is initiated (but not yet established).

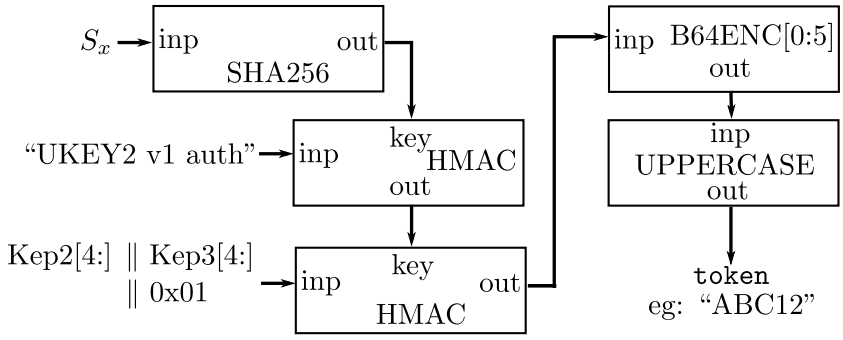


FIGURE 8.5: Computation of the authentication token.

8.3.3 Optional Authentication and Connection Establishment

Before establishing a nearby connection, the client and the server can *optionally* authenticate each other. The Nearby Connections authentication is based on a five-digit token, containing uppercase alphabetic characters, numbers, and special characters from the base64 alphabet (search space of size 38^5). The token is generated by the library, it depends on S_x (the shared key), and it is computed by the client and the server even if it is not used. It is up to the Nearby Connections application developer to decide whether and how to utilize it. At the time of writing, the nearby connection sample code from Google *ignores* the authentication tokens [75]. In a proper application, the users would have to visually authenticate each other, e. g. they must confirm that they are seeing the same token on both screens.

The reversed procedure to generate the authentication `token` consists of five sequential steps. These steps are described with the help of Figure 8.5. First, a SHA256 hash of S_x is computed. Second, this hash is used as the key for a SHA256 HMAC, from now on HMAC, of the UKEY2 v1 auth string. This string reveals that the generation procedure of the token is versioned (v1), and it is labeled as auth. Third, the output of the first HMAC is used as a key in a second HMAC. The input of the second HMAC is the concatenation ($\|$) of a subset of Kep_2, a subset of Kep_3 and the integer 0x01. This means that the entropy used in the computation of `token` is fresh and comes both from the client (Kep_2) and the server (Kep_3). This choice provides security guarantees such as protection against replay attacks. In the fourth step, the output of the second HMAC is base64 encoded and truncated to its first five characters (bytes). Finally, `token` is generated by converting these five characters to uppercase. An example of `token` is ABC12.

In the connection establishment both devices have to accept the connection, and it does not matter who accepts it first. In our experiments, we observed that the devices always use *the same* payloads:

- `0x0000000808011204080532000` to keep the pre-connection alive
- `0x0000000a0801120608021a020800` to accept a connection
- `0x0000000b0801120708021a0308c43e` to reject a connection

As in the connection request phase, Wi-Fi and Bluetooth LE are not used to establish a nearby connection. We discovered that devices with the same `ncname` are allowed

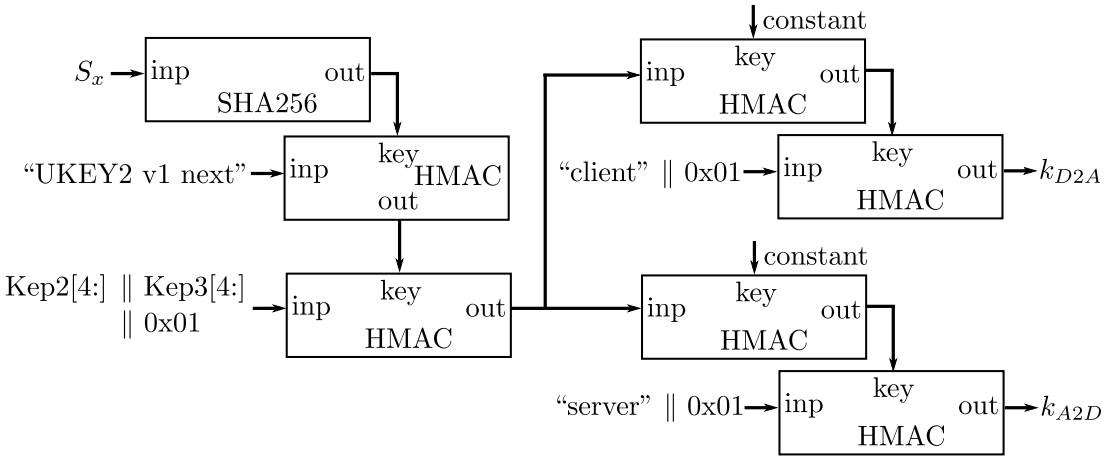


FIGURE 8.6: Computation of k_{D2A} and k_{A2D}.

to connect, this means that in a nearby connection only the `endpointId` uniquely identifies a node. While testing the connection establishment phase we discovered interesting things about the `serviceId`. Any advertiser leaks its `serviceId` if queried with a generic SDP request, e. g. using `sdptool browse adv_btaddress`. Moreover, two devices from different applications can use the same `serviceId` to establish a connection. This means that it is possible to predict the `serviceId` of any application. We also discovered that the library is still using an undocumented `serviceId` named `__LEGACY_SERVICE_ID__`.

8.3.4 Key Derivation Functions (KDFs)

We were able to reverse the key derivation functions responsible for the creation of the session, encryption, and message authentication code keys. When a nearby connection is established then the client and server compute two symmetric session keys that we define as k_{D2A} and k_{A2D}. The former is used to secure communications from the client to the server, and the latter from the server to the client. The reversed computation of these keys is shown in Figure 8.6. The description of the steps is similar to the one presented in Section 8.3.3: it starts with SHA256 hashing and then it uses a chain of HMACs. We omit the detailed description of the steps, as it is similar to the one of Figure 8.5.

However, it is important to note four points from Figure 8.6. First, from the UKEY2 v1 next string we deduce that the library uses the same version number of the auth phase (see Figure 8.5), and it labels this phase as next. Second, the only thing that differentiates k_{D2A} from k_{A2D} is their last HMAC step: the former uses client as part of the input and the latter uses server. Third, k_{D2A} and k_{A2D} depend on Kep_2, Kep_3 and S_x, indeed they enjoy the same security benefits of `token`. Finally, the library uses one session key for each direction of communication, this is a good practice in protocol design, e. g. it prevents reflection attacks.

The session keys are used to derive the encryption and the message authentication code (MAC) keys. In other words, k_{D2A} and k_{A2D} generate respectively the keys to encrypt, decrypt, sign and verify packets from the client to the server and from the

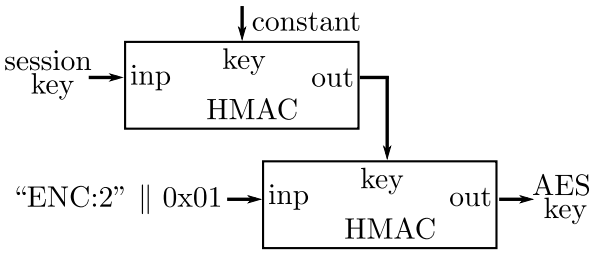


FIGURE 8.7: Computation of the AES (symmetric) key.

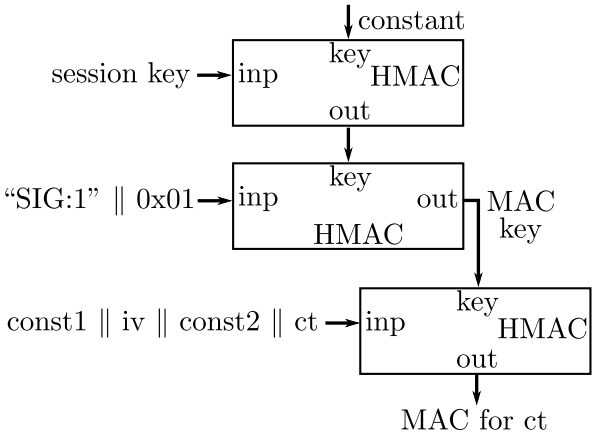


FIGURE 8.8: Computation of the MAC key and the MAC.

server to the client. The generation of the AES key from a session key is a two step process (involving HMAC), which is shown in Figure 8.7. The string ENC:2 indicates that this process is computing an encryption key, but it is not clear yet what does 2 refer to.

The computation of the MAC key is similar to the AES key computation and it is shown in Figure 8.8. In this case, we find the SIG:1 string. Again, the string indicates that a signing key is computed, but it is not clear what does the 1 refers to. Figure 8.8 describes also how the library does compute a MAC. Nearby Connections uses *encrypt-then-mac with HMAC using SHA256*. The MAC is computed using the MAC key and as input a subset of a payload containing the ciphertext (ct), an initialization vector (iv) and some constant fields.

Using dynamic binary instrumentation (discussed in Section 8.5), we see that the library *for each application layer packet re-derives the same AES and MAC keys from the session keys*. In particular, every time a node wants to transmit a packet, the library performs the following: it (re)computes the AES key, encrypts the payload, (re)computes the MAC key, signs the payload and then builds the packet. Similarly, when it is time to receive a packet, the library (re)computes the MAC key, verifies the MAC, (re)computes the AES key and decrypts the packet's payload. In our opinion, these repeated computations are not very efficient. It is possible that the library's developers intend to use a key evolution mechanism where the AES and MAC keys could change over time according to some logic. However, in our experiments the library recomputes always the same keys. Another reason to use these procedures may relate to key storage in

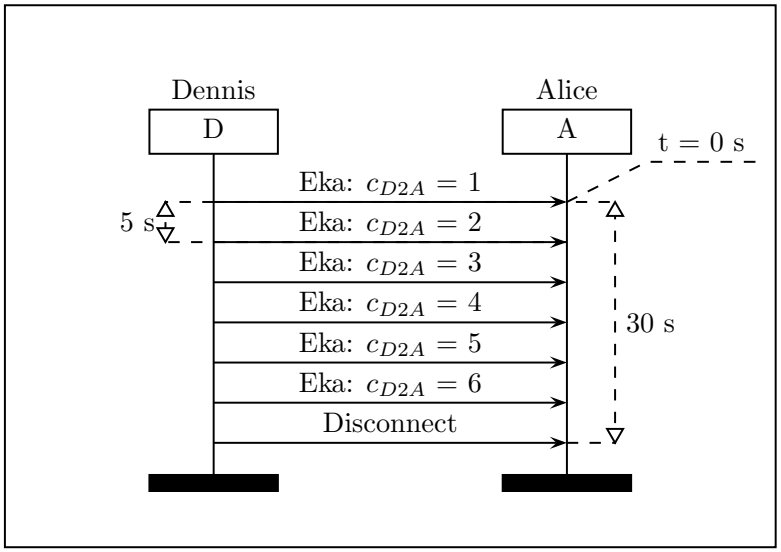


FIGURE 8.9: Nearby Connections Encrypted Keep-Alive. The period is 5 seconds, the timeout is 30 seconds.

memory. It is true that—if you recompute a key and discard it when you do not need it—the key stays in memory for a shorter time. But the library needs the session keys k_{D2A} and k_{A2D} to be able to compute the AES and the MAC keys and these are stored in memory in any case.

8.3.5 Exchange Encrypted Payloads

Once a nearby connection is established and the session keys are derived, the protocol can be considered symmetric, e. g. it does not matter who is the discoverer (client) and the advertiser (server). To emphasize this, we rename the discoverer to Dennis and the advertiser to Alice. Dennis uses k_{D2A} and Alice uses k_{A2D} and their communications are encrypted (AES256 in CBC) and authenticated (HMAC with SHA256) at the application layer, and encrypted at the link-layer (SSP).

Dennis and Alice keep the connection alive by using the encrypted keep-alive protocol (EKA) shown in Figure 8.9. From our experiments we discovered that *a nearby connection has a connection timeout of 30 seconds*. In Section 8.4, we discuss how that can be exploited to extend the range of our attacks. The EKA protocol uses a custom type of packet that we define as `Eka`. These packets have a constant header and contains a directional counter.

As we can see from Figure 8.9, Dennis initializes a directional counter, that we define as c_{D2A}, to 1. c_{D2A} counts the number of packets sent from Dennis to Alice. Dennis builds an `Eka` packet and send it to Alice. Alice maintains her local c_{D2A} counter and checks that her local values match with the ones that she gets in the packets. Dennis sends an `Eka` packet every 5 seconds incrementing each time c_{D2A}. Alice may answer with either an `Eka` packet (that counts the packets in the other direction) or a packet containing a nearby connection payload. Dennis closes the nearby connection after sending six sequential unanswered `Eka` packets. This means that the EKA timeout is

Algorithm 1 Nearby Connections Physical Layer Switch.

Require: D = discoverer, A = advertiser
Ensure: Bluetooth uses RFCOMM, Wi-Fi uses TCP, no secrets shared between Wi-Fi and Bluetooth
  if A is connected to an hotspot **then**
    A tells D how to switch to a shared WLAN
    D contacts A over TCP
  else if strategy is `P2P_STAR` **then**
    if D and A support Wi-Fi Direct **then**
      A tells D how to switch to Wi-Fi Direct
    else
      A tells D how to switch to hostapd
    end if
    D connects to A's soft AP
    D contacts A over TCP
  else
    A and D continue to use Bluetooth
  end if

30 seconds. The same encrypted keep alive protocol happens asynchronously from Alice to Dennis using c_{A2D} to count the packets sent from Alice to Dennis and Dennis maintains its local c_{A2D} counter.

While the nearby connection is alive, Dennis and Alice are able to exchange payloads. There are three types of payloads: `BYTE`, `FILE`, and `STREAM`. The payloads are sent either using Bluetooth (over RFCOMM) or Wi-Fi (over TCP) and they are encoded using custom application layer packets. Each payload generates at least two packets and each one contains the appropriate directional counter value (either c_{D2A} or c_{A2D}). The packets are sent sequentially without application layer acknowledgments. A node can send and receive payloads asynchronously. A payload packet contributes to keep the connection alive in the EKA protocol.

8.3.6 Optional Physical Layer Switch

Once a nearby connection is established, the client and the server might switch from Bluetooth to Wi-Fi. From our experiments, we see that the physical layer switch can be *predicted and manipulated*. We misuse this mechanism to perform a connection manipulation attacks (CMA) presented in Section 8.4. The switch always happens from Bluetooth to one of the three Wi-Fi modes supported by the library. We define these modes as: shared WLAN, hostapd, and Wi-Fi Direct. In the shared WLAN case the devices use a common access point: see (3) and (4) in Figure 8.2. In the hostapd and Wi-Fi Direct cases the advertiser acts as a soft AP, see (1) in Figure 8.2. The nearby connection documentation tells us that the library uses a real-time heuristic to determine when and how to switch.

Algorithm 1 describes the result of our reversing of the physical layer switch. The advertiser is always in charge of the switch and there is no negotiation with the discoverer. In order to bind the Wi-Fi and the Bluetooth connections we would expect

the library sharing secret material between Bluetooth (that is encrypted at the link and application layers) and Wi-Fi. If this material exists it should be exchanged before switching to Wi-Fi. However, in our experiments we observed that this is not the case because *the devices after the switch only use application layer encryption over TCP*. Hence, the shared WLAN link is cryptographically weaker than the Bluetooth link because it uses only one layer of encryption. We believe that the automatic Wi-Fi switch is enforced by `autoUpgradeBandwidth=true` (a private parameter of the library that we reversed).

From Algorithm 1 we note that the shared WLAN mode has the highest priority regardless of the nearby connection strategy. If shared WLAN is used we expect to see the exchange of network parameters over the Bluetooth link before the Wi-Fi switch. In our experiments we observed such exchange (and show how to leverage this as an attacker in Section 8.4). Wi-Fi Direct and hostapd are used only if the strategy is `P2P_STAR`. Both modes allow the advertiser to act as a soft access point without an Internet connection. The discoverer should be able to find its `essid` and connect to the it. When any of these modes is in use we expect to see an exchange of information about the soft AP over Bluetooth before the Wi-Fi switch. In our experiments we observed this information, and we leverage this mechanism in our attacks. Wi-Fi Direct uses a constant `essid` (22 Bytes, always starts with `DIRECT-`) and a WPA2 password (8 Bytes). hostapd uses a randomized base64-encoded `essid` (28 Bytes) and a WPA password (12 Bytes). In both cases, the advertiser sends the `essid` and the password to the discoverer. When the connection is terminated, the library does not restore the original hotspot configuration of the device.

The capability of the Nearby Connections library to switch from Bluetooth to Wi-Fi has side effects that are valuable for an attacker. In Section 8.4, we show how to abuse this capability to to switch on the Bluetoothand Wi-Fi (hotspot) antenna of the victims without user interaction. Another side effect of the switch is that the library can be misused to interrupt any active Wi-Fi connection of any node by forcing a physical layer switch to either hostapd or Wi-Fi Direct. In both cases the victim loses Internet connectivity.

8.4 Attacking Nearby Connections

In this section we present the attacks that we on the Nearby Connections library, based on our reverse engineering presented in Section 8.3. To perform the attacks we developed several tools that are presented in Section 8.5. We classify our attacks in two families: connection manipulation attack (CMA), and range extension attack (REA). The attack families are orthogonal, and can be combined. *We remark that our attacks manipulate application layer packets to reach some goal and indeed are effective regardless the specific Android application that is using Nearby Connections as a service.* In general, if an attack is effective regardless whether the attacker is the discoverer or the advertiser, we indicate the victim and the attacker as *nodes*.

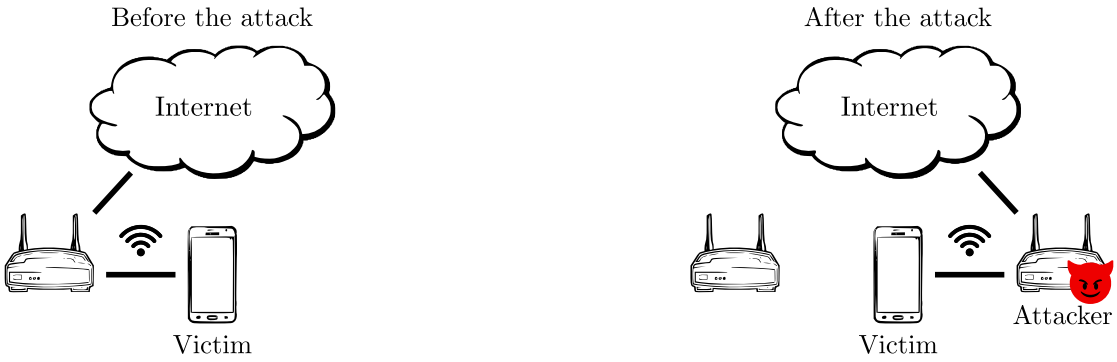


FIGURE 8.10: Soft AP manipulation attack. On the left, before the attack the victim is connected to the Internet through a benign AP. On the right, after the attack the adversary has forced the victim to connect to a malicious access point, and inserted a new default route in the victim's routing table. The adversary is able to intercept and manipulate any Internet-bound traffic sent by *any* application on the victim's phone that is using Internet.

8.4.1 Threat Model

Our attack scenario includes the victims (discoverer and advertiser) and an attacker who posses at least one device in range with the victims. The legitimate discoverer and the advertiser establish nearby connections as described in Section 8.2 and Section 8.3. The attacker has two main goals. He wants to *tamper with nearby connections nodes from remote locations*, e. g. establish a nearby connection between two countries. This violates the basic assumption that only devices within radio range can establish nearby connections. In addition, he wants to *manipulate these connections in arbitrary ways*, e. g. install himself as man-in-the-middle, take over existing connection, manipulate the Bluetooth and Wi-Fi radio of the victims, and break or weaken the security mechanisms of the nearby connection library.

The attacker has the same knowledge of the library that we describe in Section 8.3. He is capable of using custom advertiser and discoverer and to craft raw packets conforming to the nearby connection protocol using tools similar to the ones developed (REarby). An extensive discussion of the tools is presented in Section 8.5. The attacker can create his own Wi-Fi access point, and jam the wireless links. He does not have access to the victims devices e. g. he is not able to install rootkits or malicious applications. The applications and libraries used by the victims are considered safe. The attacker does not require advanced and potentially expensive instrumentation such as software-defined radio, directional antennas, rooted devices and commercial wireless sniffers.

8.4.2 Connection Manipulation Attacks

In principle, a node should establish nearby connections only with trusted nodes. However, the library presents several authentication issues that allows an attacker to manipulate a connection. In particular, the library does do not perform any of the following: authenticate the Bluetooth link key, bind the Bluetooth and the Wi-Fi physical layers,

mandate user authentication, and authenticate the application that is requesting the nearby connection service. The documentation suggests that "encryption without authentication is essentially meaningless" [70] and we argue that this is the case here.

Advertiser and discoverer impersonation The library uses only the strategy and the `serviceId` to uniquely identify a nearby connection, and both are predicable. We can learn the `serviceId` of an application by using a Bluetooth SDP request and we can guess the strategy (only 2 options). As a result, we can impersonate both an advertiser advertising a proper `serviceId` and a discoverer trying to connect to a legitimate advertiser. This capability is a stepping stone to perform more elaborate attacks that we present in this section.

Application layer MitMs The lack of proper authentication of the library allows us (the attacker) to man-in-the-middle two victims at the application layer. We accomplish this attack using a malicious advertiser and a malicious discoverer at the same time. The malicious advertiser gets discovered by the first victim discoverer, and the malicious discoverer connects to the second victim advertiser (the two malicious devices forward traffic between each other). Then, we can complete two parallel Bluetooth pairings with the victims, we perform the application layer phases of Figure 8.1, and we compute the shared secrets and the keys (session, encryption and mac) for each victim. As a result we are able to decrypt, observe, manipulate and encrypt the application layer packets.

If the advertiser requests a physical layer switch we launch a parallel man-in-the-middle attack on the Wi-Fi link. We know the credentials of the Wi-Fi network because they are transmitted by the victim advertiser on the Bluetooth link from which we are eavesdropping. The Wi-Fi MitM is accomplished using a simple ARP spoofing attack. Wi-Fiis not encrypted at the ink layer, thus we can continue to observe and manipulate the application layer traffic also in the Wi-Fi link. We note that the MitM works even between a victim discoverer and a victim advertiser using different Android applications (different `serviceId`).

Shared WLAN manipulation The attacker can also manipulate the physical layer switch phase, regardless the nearby connection strategy. As shown in Algorithm 1, the advertiser dictates when and how to switch (from Bluetooth to Wi-Fi) and the discoverer "blindly" follows him. For example, if the advertiser is connected to an hotspot it will tell the discoverer its IP address and TCP port (shared WLAN Wi-Fi mode). However, we are able to redirect a discoverer to an arbitrary IP and TCP port by intercepting and crafting the legitimate physical layer switch packet sent by the advertiser before the switch. The details about how we craft this and other types of packets are presented in Section 8.5.6. As result of this attack, the victim activates her Wi-Fi interface, associates to a legitimate AP and establishes a TCP session with a target determined by the attacker. Note that, this attack work regardless the connection strategy because the shared WLAN mode is picked as first choice in both, and the IP spoofed by the attacker can be outside of the local area network of the victim.

Soft AP manipulation If the target nearby connection uses the `P2P_STAR` strategy, the advertiser might act as a soft access point without an Internet connection, and provide the AP credentials to the discoverers. We take advantage of this feature *to redirect discoverers to an access point controlled by the attacker that is connected to the Internet*. Figure 8.10 shows the victim connection status before (left side) and after the attack. Initially, the victim (discoverer) is connected to a legitimate AP. The victim connects to an advertiser using the `P2P_STAR` strategy. The attacker either is the advertiser or performs the application layer MitM attack presented earlier. The attacker also controls a rogue access point.

Once the nearby connection is established the attacker manipulates either hostapd or Wi-Fi Direct switch packets to redirect the victim to the rogue access point i. e. he provides the victim valid essid and password. Then, the victim associates to the malicious AP (see Figure 8.10), and configures her Wi-Fi interface with values supplied by the attacker over DHCP. That enables the attacker to install a new default route (along with suitable IP configurations), which redirects all the victim's traffic to the rogue AP. Indeed, the attacker is able to monitor and tamper with *all the Wi-Fi traffic* coming from the victim. The traffic includes the packets generated by other applications (not using the library) requiring Internet access such as email clients, web browsers, and cloud services. We understand that most of such traffic is secured with TLS, but we still believe that this attack is novel and it has serious consequences. In particular, the attacker can target a single application using Nearby Connections to get access to all Wi-Fi network traffic of the victim. The attacker then can perform traffic analysis even on encrypted packets [170].

DoS Internet connections The library does not care about the connection status of the devices before using hostapd and Wi-Fi Direct. We leverage this fact to launch a denial of service (DoS) attack on several discoverers at the same time. The attack assumes that one ore multiple discoverers are connected to a legitimate Wi-Fi network and they want to use Nearby Connections. The attacker uses a malicious advertiser to connect to the victims and tell them to use either hostapd or Wi-Fi Directand to connect to a non-existent AP. The victims try to connect to the AP, and as a result they lose their Internet connectivity. This issue indirectly affects all the applications running on the victim's device that need an Internet connection [73].

Alter network configurations and radios The library does not backup and restore the original wireless network configuration. In particular, the original soft AP (Wi-Fi hotspot) configuration is overwritten by the library and the newly created network is appended to the list of known ones. This allows an attacker to append and overwrite arbitrary essid-password pairs in the network configuration files of the victim. The attack technique is the same presented in the DoS attack paragraph.

The library is able to switch on the Bluetooth and Wi-Fi antenna of device that is using it, and it does not switch them off after a disconnection. We use this feature manipulate the antennas of a victim device, regardless of the nearby connection strategy. We are able to switch on the Bluetooth antenna of the victim by establishing a nearby

connectionand then disconnecting. We can switch on the Wi-Fi antenna of any discoverer by using our custom advertiser to connect with the victim and by telling the him to switch to Wi-Fi and to disconnect.

8.4.3 Range Extension Attacks (REA)

The Nearby Connections API is supposed to be used by devices that are effectively nearby. The documentation suggests that they have to be within radio range (approx 100 m) [71]. However, at the time of writing, *the library does not enforce strict time requirements between connected nodes and does not check the geo-location of the nodes.* The library uses an encrypted keep alive protocol with a generous timeout of 30 seconds, which is more than enough to forward traffic across continents without aborting the nearby connection [62, 115]. This allows an attacker to violate the fundamental assumption that nearby-connected devices are in proximity by extending the range of any nearby connection.

The attacker can extend the range of any attack presented in Section 8.4.2. In our experiment we implement a wormhole-attack to extend the range of the application layer MitM attack. We are able to let two non-nearby victims talk between each other, i. e. the two victims might be advertising and discovering nearby connection in different continents. The attacker uses two devices, each one in range with a victim, to perform the MitM attack and then forwards the traffic over the Internet creating a wormhole. The attacker has 30 seconds to forward the packets in each direction (to keep the connection alive) and he could even answer to the keep alive requests himself, effectively allowing arbitrary delays. This attack technique takes inspiration from [86, 175, 62]. Range extension is not advisable because a victim perceives a false sense of security given by the fact that a nearby connection is supposed to be within radio range and it is expected not to use Internet. In other words, a victim might better trust a proximity-based secure service than a secure cloud service.

8.5 REarby Toolkit Implementation

To implement the attacks presented in Section 8.4, we require several capabilities based on our analysis of the Nearby Connections library presented in Section 8.3. These capabilities includes: establishment of wireless connections using Bluetooth and Wi-Fi, ad-hoc usage of cryptographic primitives, creation and manipulation manipulation of raw network packets, and usage of custom (security) protocols. For these purposes, we develop a set of tools and we group them in a project that we call REarby.

REarby includes custom discoverer and advertiser capable to perform all the nearby connection phases from Figure 8.1. It includes a dynamic binary instrumenter to analyze the library at runtime and a packet dissector usable to decode and manipulate application layer packets. REarbyalso contains a custom Android application that we developed for testing. Out toolkit makes use of three programming languages: Python, Java, and JavaScript and contains approximately 2000 lines of code. It requires a minimal setup, e. g. a laptop running Linux. In the rest of this section we explain how we implement all the phases of a nearby connection.

8.5.1 Discovery and Connection Request

We manage all the Bluetooth operations with the `bluetooth` Python module. The discovery phase begins using the `discover_bt` function. This function returns the Bluetooth's name (btname) of all the discoverable devices that are in range. The custom discoverer detects the presence of any advertiser by looking at btnames. It extracts the strategy, `endpointId` and `ncname` using in reverse the btname formula of Section 8.3.1. The custom advertiser starts an RFCOMM server on port 5 using `BluetoothSocket(RFCOMM)`. Then the custom advertiser computes the custom `uuid` based on the `serviceId` and it starts an SDP server advertising the `uuid` using the `serviceId` as the name of the SDP service. The custom advertiser waits for the discoverers and manages each of them with separate sockets. All the Bluetooth connections are encrypted at the link layer using the shared secret computed from the secure simple pairing (SSP).

8.5.2 Key Exchange Protocol (KEP)

To initiate a connection the custom discoverer computes the custom `uuid` (based on the `serviceId`) and performs an SDP request using that `uuid`. The response contains the `serviceId` and the Bluetooth address of the advertiser. The custom discoverer uses an RFCOMM socket to connect the advertiser on port 5. The `bluetooth`'s Python module manages the low level details of the RFCOMM socket.

Once connected, the custom advertiser and the custom discoverer respectively complete the Nearby Connections key exchange protocol as in Figure 8.4. To be able to send meaningful KEP packets we perform several steps. We develop a custom Android application based on [74] and we use it to generate samples of KEP packets. We capture the unencrypted packets using the HCI snoop log functionality of Android. HCI is the host controller interface protocol spoken by the Bluetooth host (the OS) and the Bluetooth controller (the radio chip) [162]. We analyze the packets using scapy [25], a network analysis tool. We discover that the plaintext is serialized using custom serialization mixing variable and fixed length fields. This is confirmed by the fact that the library uses the `Serializable` Java class.

Listing 1 shows the `Kep3` Scapy dissection class that we use to decode the serialized data in the Kep_3 packets. Kep_3 is sent from the advertiser to the discoverer and it contains four relevant fields: the sequence number (`sn`), a nonce (`nA`), the coordinates of the public key of the advertiser (`xA`, `yA`). These values are serialized using variable length fields. A variable length field has a leading Byte containing the length of the field in Bytes concatenated with the actual Bytes containing the value of the field. For example, `nA_len` indicates that `nA` is a 32 Byte nonce and its value is referenced by `nA`. The same holds for `xA` and `yA`. We use similar classes to decode Kep_1 Kep_2, and Kep_4. From the decoded KEP packets we realize that the library uses ECDH on the secp256r1 curve, by testing the public keys contained in Kep_3 and Kep_4 against standard elliptic curves.

The next task is to correctly compute the shared secret (S_x). To understand how to compute it we use dynamic binary instrumentation (DBI), a technique that allows to monitor a target application (process) in real-time by attaching a monitoring process to

Listing 1 `Kep3` scapy dissection class for Kep_3.

```
1  class Kep3(Packet):
2      name = "Kep3: adv -> dsc"
3      fields_desc = [
4          IntField("len1", None),
5          XByteField("sep1", 0x08),
6          ByteField("sn", 3),
7          XByteField("NC_SEP", NC_SEP),
8          ByteField("len2", None),
9          StrFixedLenField("NC_HEAD2", NC_HEAD2, length=3),
10         BitFieldLenField("nA_len", None, size=8, length_of="nA"),
11         StrLenField("nA", "", length_from=lambda pkt:pkt.nA_len),
12         StrFixedLenField("NC_KEP3_HEAD", NC_KEP3_HEAD, length=3),
13         ByteField("len3", None),
14         StrFixedLenField("NC_HEAD2", NC_HEAD2, length=3),
15         ByteField("len4", None),
16         XByteField("NEWLINE", NEWLINE),
17         BitFieldLenField("xA_len", None, size=8, length_of="xA"),
18         StrLenField("xA", "", length_from=lambda pkt:pkt.xA_len),
19         XByteField("NC_SEP", NC_SEP),
20         BitFieldLenField("yA_len", None, size=8, length_of="yA"),
21         StrLenField("yA", "", length_from=lambda pkt:pkt.yA_len),
```

it. To implement our DBI we use Frida [148] a reverse-engineering toolkit that has native compatibility with Android. After observing our Android application generating ECDH shared secrets we find out that the cryptographic operations are managed by a separate Android process called `com.google.android.gms.nearby.connection` that we indicate with `ncproc`.

Using Frida we list all the Java classes and shared libraries of `ncproc` and isolate all the security related classes, methods and functions. `OpenSSLECDHKeyAgreement` reveals that the library is using only the x coordinate of S_x as the key, i. e. it is not using a standard key derivation function. These information enables us to initiate a nearby connection by implementing the Nearby Connections KEP protocol from Figure 8.4. We use the Python `cryptography` module to implement all the cryptographic procedures. Note that, our setup allows us to tamper with and fuzz every field of the KEP packets such as the public keys, `algo`, `version`, `endpointId` and the nonces.

8.5.3 Optional Authentication and Connection Establishment

After the nearby connection is initiated we compute `token` to perform the optional user authentication phase. Listing 2 shows how we monitor and overload the `encodeToString` method of the `android.util.Base64` class [49] using Frida. This method is called in the last step of the `token` computation from Figure 8.5 and it takes an array of bytes as input and returns its base64 encoding representation as a string. The most important lines of code are from line 6 to 16. In line 6, we use `B64ENC_COUNT` to count how many times this method is called at runtime. In line 8 we use `print_backtrace` to (recursively) see who called the method using which parameters. In line 10-11 we save the original input of the method as a string in `inp_str`, and we print it on our console. The original method is called in line 13 with its original

Listing 2 Frida (JavaScript API) function to overload `android.util.Base64.encodeToString`.

```
1  // input is a byte[], return value is a String
2  function Base64_encodeToString() {
3    Java.perform(function () {
4      var target = Java.use("android.util.Base64");
5          target.encodeToString.overload('[B', 'int').implementation =
             ↪ function(inp, flags) {
6        B64ENC_COUNT += 1
7
8        print_backtrace();
9
10       var inp_str = JSON.stringify(inp)
11       console.warn("B64ENC " + B64ENC_COUNT + " inp: " + inp_str)
12
13       var retval =  this.encodeToString(inp, flags);
14       console.warn("B64ENC " + B64ENC_COUNT + " out: " + retval)
15
16       return retval
17     };
18   });
19 }
```

inputs (`inp`, `flags`) and its return value is saved into `retval`. In line 14-16, we print the return value and return the control to the `ncproc` process.

We use functions similar to Listing 2 to reconstruct the computation of `token` from Figure 8.5. We implement its computation in our custom advertiser and discoverer using standard Python modules. To establish the connection we implement the pre-connection keep alive, the connection acceptance and rejection reusing the constant payloads mentioned in Section 8.3.3.

8.5.4 Key Derivation Functions

Our dynamic binary instrumentation setup is quite powerful. It allows us to observe, tamper with, and reimplement every method call used by *any* Android process, such as `ncproc`. All of this without having access to the implementation of the Nearby Connections library. Using our DBI we reconstruct all the key derivation functions presented in Section 8.3.4, and we implement them using standard Python modules. This enables our custom discoverer and advertiser to establish a nearby connection and compute the correct session (Figure 8.6), encryption (Figure 8.7), and MAC keys (Figure 8.8).

8.5.5 Encrypted Keep-Alive and Payloads

An established nearby connection is kept alive using the encrypted keep-alive (EKA) protocol from Figure 8.9. The EKA protocol requires the knowledge of the cryptographic stack used to encrypt-then-mac the application layer packets and the capability to build meaningful application layer packets. For example, we have to maintain the

Listing 3 Backtrace of the `ncproc` crypto stack. Long lines are truncated with three dots (...). The library is using javax.crypto that calls conscrypt that calls BoringSSL.

```
1  at com.google.android.gms.org.conscrypt...
2  at com.google.android.gms.org.conscrypt...
3  at com.google.android.gms.org.conscrypt...
4  at com.google.android.gms.org.conscrypt...
5  at com.google.android.gms.org.conscrypt...
6  at javax.crypto.Cipher.doFinal(Cipher.java:1502)
7  at blah.a(:com.google.android.gms...
8  at blam.a(:com.google.android.gms...
9  at blam.a(:com.google.android.gms...
10 at bkyj.a(:com.google.android.gms...
11 at bkyg.b(:com.google.android.gms...
12 at acxt.c(:com.google.android.gms...
13 at acyg.a(:com.google.android.gms...
14 at acyd.run(:com.google.android.gms...
15 at pmz.run(:com.google.android.gms...
16 at java.util.concurrent.ThreadPoolExecutor...
17 at java.util.concurrent.ThreadPoolExecutor...
18 at ptb.run(:com.google.android.gms...
19 at java.lang.Thread.run(Thread.java:818)
```

directional counters (c_{A2D}, c_{D2A}) as explained in Section 8.3.5. Implementing the EKA protocol is a key requirement to perform the attacks presented in Section 8.4.

We reverse the cryptographic stack of the library by looking at the backtrace of its lowest level cryptographic methods e. g. `NativeCrypto_EVP_CipherFinal_ex()`. Listing 3 shows the backtrace with its *nineteen (19)* stack frames (truncated lines are terminated with three dots). Starting from the top we see that the low level crypto functions are managed by conscrypt (line 1-5). Conscrypt is an open-source Java security providers developed by Google [67]. Conscrypt in turns uses BoringSSL [66], Google's open-source fork of OpenSSL. Note that BoringSSL is native code. Going down the backtrace we see the use of javax.crypto module (line 6). This module provides high level interfaces for Java cryptographic operations. The next 9 stack frames (line 7-15) are created by the Nearby Connections library and the names of the classes and the methods are obfuscated, most probably using ProGuard [79]. The lowest part of the backtrace contains calls to thread methods.

Overall, the cryptographic stack of the library uses standard implementations that we are able to replicate using the Python `cryptography` module. Using our crypto stack we create valid ciphertext using the symmetric key computed as in Figure 8.7 and valid message authentication code using the key as in Figure 8.8. We use scapy to build properly formatted application layer packets that include the length fields, the ciphertext, the mac and the appropriate directional counter (either c_{A2D} or c_{D2A}).

8.5.6 Optional Physical Layer Switch

The correct implementation of the physical layer switch is paramount to perform the attack presented in Section 8.4. There are two packets of interests that the advertiser has to send to the discoverer to tell him when and how to switch from Bluetooth to Wi-Fi. We use two scapy dissection classes, that we call `WL` and `HA`, to manage these

packets. Their relevant fields are shown in Table 8.2. `WL` is used with the shared WLAN mode, our custom advertiser (the attacker) sends it to the discoverer to redirect him to arbitrary `ip` and `tcp_port`. This packet is usually sent just after the start of the EKA protocol. `HA` is used with the Wi-Fi Direct and the hostapd. By spoofing the `essid` and `password` fields in of this packet our custom advertiser redirects the victim (discoverer) to an arbitrary soft AP. Usually, this packet is sent by the advertiser after three `Eka` packets.

8.5.7 Summary

Our REarby toolkit allows to analyze and attack the Nearby Connections library. It includes several components such a custom discoverer and advertiser, dynamic binary instrumentation based on Frida, and packet dissector based on scapy. Our custom discoverer and advertiser are able to discover, advertise, request, initiate, authenticate, accept/reject, establish a connection and tell when and how to switch to a different physical layer. They maintain the connection alive by speaking the EKA protocol. They send `BYTE` and `FILE` payloads. They allow the attacker to specify the strategy, `serviceId`, `ncname` and `endpointId` and to modify and fuzz any dissected packets.

Table 8.2 summarizes the most important scapy dissection classes that are in use. Table 8.3 lists the Java classes and methods that we monitored while reversing `ncproc`. Table 8.4 lists the device that we use for our analysis and attacks.

TABLE 8.2: scapy dissection classes used to reverse the Nearby Connections library. The count field is the application layer directional counter.

ClassName	Relevant Fields	Usage	Direction
`Kep1`	sn, eid, ncname, version	ECDH	D $\longrightarrow$ A
`Kep2`	sn, N_D, c_D, algo	ECDH	D $\longrightarrow$ A
`Kep3`	sn, N_A, xA, yA	ECDH	D $\longleftarrow$ A
`Kep4`	sn, xD, yD	ECDH	D $\longrightarrow$ A
`Eka`	iv, ct, mac	App Layer	D $\longleftrightarrow$ A
`KA`	count	App Layer	D $\longleftrightarrow$ A
`Pay`	iv, ct, mac	App Layer	D $\longleftrightarrow$ A
`Pt`	pid, ptype, pay_len, pay, count	App Layer	D $\longleftrightarrow$ A
`Pt2`	pid, ptype, pay_len, pt_len, count	App Layer	D $\longleftrightarrow$ A
`WL`	ip, tcp_port, count	Wi-Fi	D $\longleftarrow$ A
`HA`	essid, password, count	Wi-Fi	D $\longleftarrow$ A
`Error`	emsg	Misc	D $\longleftrightarrow$ A

8.6 Related Work

We are not aware of related work on the Nearby Connections API (for Android devices, or others). In general, a large amount of academic work has investigated security issues in wireless standards, such as cryptographic weaknesses in early versions of Bluetooth [94], and practical sniffing attacks on Bluetooth [164]. Similarly, cryptographic

TABLE 8.3: Security related classes and methods used by `ncproc`.

ClassName	MethodName	Usage
OpenSSLCipher	engineDoFinal()	AES256 in CBC
	engineInit()	HMAC with SHA256
	engineUpdate()	HMAC with SHA256
	engineDoFinal()	HMAC with SHA256
OpenSSLMessageDigestJDK	engineUpdate()	SHA1, SHA2
	engineDigest()	SHA1, SHA2
OpenSSLECDHKeyAgreement	engineInit()	ECDH
	engineDoPhase()	ECDH
	engineGenerateSecret()	ECDH
NativeCrypto	RAND_bytes()	RNG
Base64	encodeToString()	Encode base64
	decode()	Decode base64

TABLE 8.4: Devices used in our Nearby Connections experiments and attacks. GPS is Global Positioning System. SSP stands for Secure Simple Pairing and SC for Secure Connections.

Name	Library	Bluetooth	Soft AP
Android 6.0.1			
Motorola G3 (x2)	GPS 12.8.74	4.1 SSP with SC	Wi-Fi Direct, hostapd
Nexus 5 (x2)	GPS 12.8.74	4.1 SSP	hostapd
Linux 4.4			
Thinkpad x1	Bluez 5.49-4	4.2 SSP	hostapd, airbase-ng

attacks have been found on the confidentiality of Bluetooth LE [156] and early versions of IEEE 802.11/Wi-Fi [27], and later improvements such as WPA [172]. Lately, additional key reinstallation attacks were discovered for WPA2 [178], which compromise key freshness. The impact of physical layer features (such as MIMO and beamforming in recent standard amendments) on practical eavesdropping on 802.11 was discussed in [9]. In addition to analysis of standards, libraries that implement the standards have also been investigated. For example, a number of vulnerabilities in Apple and Android devices' Bluetooth stack were identified in 2017, dubbed "BlueBorne" [15].

In this chapter our focus is not on attacking any design or implementation aspect of Bluetooth and Wi-Fi standards. Instead, we point out that introduction of libraries such as Nearby Connections can lead to unexpected side effects for traffic of all applications on the victim, e.g., by disconnecting hosts, redirecting traffic via and attacker, and can lead to effects such as resource exhaustion (due to attackers manipulating radio states). Usage of third-party libraries has already been studied for the Android ecosystem, e.g., in [16]. However, our work investigates a library developed directly by Google through its Google Play Services service, pushed to end users. The misuse of cryptographic primitives on Android is also well-known [55]. While in the case of

Nearby Connections, the developer (or user) is not responsible for choosing the cryptographic primitives, he (or she) has to trust the library to be securely designed.

In [127], the authors investigated security issues arising from interactions of malicious apps with paired mobile devices, e.g., via Bluetooth, essentially related to lack of access controls. We argue that Nearby Connections poses an even bigger threat, as it is ubiquitously supported and optimized to require no user interactions.

A rich set of literature about attacks on routing schemes in the context of wireless sensor networks is available [86]. We argue that in this work, the system attacked is not a (multi-hop) routing scheme, as it is intended for direct communication between nearby hosts. Practical wormhole attacks that extend communication range from nearby hosts to remote hosts have been demonstrated in the context of car keys [62] and NFC communication [115].

8.7 Conclusion

In this work, we present the first security analysis of the proprietary, closed-source and obfuscated Nearby Connections API by Google. The API is installed and available to applications on any Android device from version 4.0 onwards. It is also available on Android Things, an OS developed by Google for IoT devices. The API connects nearby devices using multiple physical layers. In order to perform our analysis, we studied its public API, and reverse engineered its implementation on Android. We found and implemented several attacks (classified as connection manipulation and range extension attacks).

For example, in the Soft AP manipulation attack, we trick the victim (discoverer) to disconnect from its currently associated access point, and connect to an access point controlled by the attacker. Consequently, the attacker is able to push a default route in the victim's network configuration (via DHCP) and to redirect all his Wi-Fi traffic (not only from the Nearby Connections application) to him. This is a novel attack, in which a vulnerability in the Nearby Connections API, is impacting all the network communication of the victim.

Our implementation of the attacks is based on REarby, a toolkit we developed to reverse engineer and analyze the Nearby Connections API. REarby includes custom discoverer and advertiser capable of performing the nearby connection phases. It also includes a dynamic binary instrumenter, a packet dissector and a custom Android application. Our findings were acknowledged by Google in a responsible disclosure process, and we released a proof of concept of the Soft AP attack as open source at `https://github.com/francozappa/rearby`. Our findings show that in the current state, Google's Nearby Connections API is not only open to attack, but actively posing a threat to all Android applications using it (the library is automatically installed and updated without user interaction) and even to Android applications that do not use it.

Chapter 9

The KNOB is broken: Exploiting Low Entropy in the Encryption Key Negotiation of Bluetooth BR/EDR

Keywords: Bluetooth, Encryption, Entropy, Attack, Firmware.

9.1 Introduction

Bluetooth BR/EDR (referred for the rest of this chapter as Bluetooth), is a short-range wireless technology widely used by many products such as mobile devices, laptops, IoT and industrial devices. Bluetooth provides security mechanisms to achieve authentication, confidentiality and data integrity at the link layer [162, p. 1646].

The security and privacy of Bluetooth has been attacked and fixed several times, going all the way back to Bluetooth v1.0. [94, 190]. Several successful attacks on the (secure simple) pairing phase [160, 81, 24] have resulted in substantial revisions of the standard. Attacks on Android, iOS, Windows and Linux implementations of Bluetooth were also discussed in [15]. However, little attention has been given to the security of the *encryption key negotiation protocol*, e. g. the Bluetooth security overview in the latest Bluetooth core specification (v5.0) does not mention it [162, p. 240].

The encryption key negotiation protocol is used by two Bluetooth devices to agree on the entropy of the link layer encryption key. Entropy negotiation was introduced in the specification of Bluetooth to cope with international encryption regulations and to facilitate security upgrades [162, p. 1650]. To the best of our knowledge, all versions of the Bluetooth standard (including the latest v5.0 [162]) *require* to use entropy values between 1 and 16 bytes. The specification of Bluetooth states this requirement as follows: "For the encryption algorithm, the key size may vary between 1 and 16 octets (8 - 128 bits)" [162, p. 1650]. Our interpretation of this requirement is that any device to be standard-compliant has to support encryption keys with entropy varying from one to sixteen bytes. The attack that we present in this work confirms our interpretation.

The encryption key negotiation protocol is conducted between two parties as follows: the initiator proposes an entropy value N that is an integer between 1 and 16, the other party either accepts it or proposes a lower value or aborts the protocol. If the other party proposes a lower value, e. g. $N - 1$, then the initiator either accepts it or proposes a lower value or it aborts the protocol. At the end of a successful negotiation the two parties have agreed on the entropy value of the Bluetooth encryption

key. The entropy negotiation is performed over the Link Manager Protocol (LMP), it is not encrypted and not authenticated, and it is transparent to the Bluetooth users because LMP packets are managed by the firmware of the Bluetooth chips and they are not propagated to higher layers [162, p. 508].

In this chapter we describe, implement and evaluate an attack capable of making two (or more) victims using a Bluetooth encryption key with 1 byte of entropy without noticing it. The attacker then can easily brute force the encryption key, eavesdrop and decrypt the ciphertext and inject valid ciphertext without affecting the status of the target Bluetooth piconet. In other words, *the attacker completely breaks Bluetooth BR/EDR security without being detected*. We call this attack the **Key Negotiation Of Bluetooth (KNOB)** attack.

The KNOB attack can be conducted remotely or by maliciously modifying few bytes in one of the victim's Bluetooth firmware. Being a standard-compliant attack it is expected to be effective on any firmware implementing the Bluetooth specification, regardless of the Bluetooth version. The attacker is not required to posses any (pre-shared) secret material and he does not have to observe the pairing process of the victims. The attack is effective even when the victims use the strongest security mode of Bluetooth (Secure Connections). The attack is stealthy because the application using Bluetooth and even the operating systems of the victims cannot access or control the encryption key negotiation protocol (see Section 9.3.2 for the details).

After explaining the attack in detail, we implement it leveraging our development of several Bluetooth security procedures to generate valid link and encryption keys, and the InternalBlue toolkit [114]. Our implementation allows a man-in-the-middle attacker to intercept, manipulate, and drop LMP packets in real-time and to brute force low-entropy encryption keys, without knowing any (pre-shared) secret. We have disclosed our findings about the KNOB attack with CERT and the Bluetooth SIG, and following that, we plan to release our tools as open-source at `https://github.com/francozappa/knob`. This will enable other Bluetooth researchers to take advantage of our work.

We summarize our main contributions as follows:

- We develop an attack on the encryption key negotiation protocol of Bluetooth BR/EDR that allows to let two unaware victims negotiate a link-layer encryption key with 1 byte of entropy. The attacker then is able to brute force the low entropy key, decrypt all traffic and inject arbitrary ciphertext. The attacker does not have to know any secret material and he can target multiple nodes and piconets at the same time.

- We demonstrate the practical feasibility of the attack by implementing it. Our implementation involves a man-in-the-middle attacker capable of manipulating the encryption key negotiation protocol, brute forcing the key and decrypting the traffic exchanged by two (or more) unaware victims.

- All standard-compliant devices should be vulnerable to our attack, including the ones using the strongest Bluetooth security mode. In order to demonstrate that this problem has not somehow been fixed in practice, we test more than 14 different Bluetooth chips and find all of them to be vulnerable.

- We discuss what changes should be made, both to the Bluetooth standard and its implementation, in order to counter this attack.

Our work is organized as follows: in Section 9.2 we introduce the Bluetooth BR/EDR stack. In Section 9.3 we present the Key Negotiation Of Bluetooth (KNOB) attack. An implementation of the attack is discussed in Section 9.4. We evaluate the impact of our attack in Section 9.5 and we discuss the attack and our proposed countermeasures in Section 9.6. We present the related work in Section 9.7. We conclude the chapter in Section 9.8.

9.2 Background

9.2.1 Bluetooth Basic Rate/Extended Data Rate

Bluetooth Basic Rate/Extended Data Rate (BR/EDR), also known as Bluetooth Classic, is a widely used wireless technology for low-power short-range communications maintained by the Bluetooth Special Interest Group(SIG) [162]. Its physical layer uses the same 2.4 GHz frequency spectrum of WiFi and (adaptive) frequency hopping to mitigate RF interference. A Bluetooth network is called a piconet and it uses a master-slave medium access protocol. There is always one master device per piconet at a time. The devices are synchronized by maintaining a reference clock signal, defined as CLK. Each device has a Bluetooth address (BTADD) consisting of a sequence of six bytes. From left to right, the first two bytes are defined as non-significant address part (NAP), the third byte as upper address part (UAP) and the last three bytes as lower address part (LAP).

To establish a secure Bluetooth connection two devices first have to pair. This procedure results in the establishment of a long-term shared secret defined as *link key*, indicated with K_L. There are four types of link key: initialization, unit, combination and master. A initialization key is always generated for each new pairing procedure. A unit key is generated from a device and utilized to pair with every other device, and its usage is not recommended because it is insecure. A combination key is generated using Elliptic Curve Diffie Hellman (ECDH) on the P-256 elliptic curve. This procedure is defined as Secure Simple Pairing (SSP) and it provides optional authentication of the link key. Combination keys are the most secure and widely used. A master key is generated only for broadcast encryption and it has limited usage. The master key is temporary, while the others are semi-permanent. A semi-permanent key can persist until a new link key is requested (link key is bonded) or it can change within the same session (link key is not bonded). In this work we deal with combination link keys generated using authenticated SSP.

The specification of Bluetooth defines custom security procedures to achieve confidentiality, integrity and authentication. In the specification their names are prefixed with the letter E. In particular, a combination link key K_L is mutually authenticated by the E_1 procedure. This procedure uses a public nonce (AU_RAND) and the slave's Bluetooth address (BTADD_S) to generate two values: the Signed Response (SRES) and the Authenticated Ciphering Offset (ACO). SRES is used over the air to verify that two devices actually own the same K_L.

The symmetric encryption key K_C is generated using the E_3 procedure. When the link key is a combination key E_3 uses ACO (computed by E_1) as its Ciphering Offset Number (COF) parameter, together with K_L and a public nonce (EN_RAND). E_1 and E_3 use a custom hash function defined in the specification of Bluetooth with ₹ The hash function is based on SAFER+, a block cipher that was submitted as an AES candidate in 1998 [117].

Once the encryption key K_C is generated there are two possible ways to encrypt the link-layer traffic. If both devices support Secure Connections, then encryption is performed using a modified version of AES CCM. AES CCM is an authenticate-then encrypt cipher that combines Counter mode with CBC-MAC and it is defined in the IETF RFC 3610 [88]. As a side note, the specification of Bluetooth defines a message authentication codes (MAC) with the term message integrity check (MIC). If Secure Connections is not supported then the devices use the E_0 stream cipher for encryption. The cipher is derived from the Massey-Rueppel algorithm and it is described in the specification of Bluetooth [162, p. 1662]. E_0 requires synchronization between the master and the slaves of the piconet, this is achieved using the Bluetooth's clock value (CLK).

Modern implementations of Bluetooth provides the Host Controller Interface (HCI). This interface allows to separate the Bluetooth stack into two components: the host and the controller. Each component has specific responsibilities, i. e. the controller manages low-level radio and baseband operations and the host manages high-level application layer profiles. Typically, the host is implemented in the operating system and the controller in the firmware of the Bluetooth chip. For example BlueZ and Bluedroid implement the HCI host on Linux and Android, and the firmware of a Qualcomm or Broadcom Bluetooth chip implements the HCI controller. The host and the controller communicate using the Host Controller Interface (HCI) protocol. This protocol is based on commands and events, i. e. the host sends (acknowledged) commands to the controller, and the controller uses events to notify the host.

The Link Manager Protocol (LMP) is used over the air by two controllers to perform link set-up and control for Bluetooth BR/EDR. LMP is neither encrypted nor authenticated. The LMP packets do not propagate to higher protocol layers, hence, the hosts (OSes) are not aware about the LMP packets exchanged between the Bluetooth controllers.

9.3 Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR

In this section we describe the Key Negotiation Of Bluetooth (KNOB) attack. The attack allows Charlie (the attacker) to reduce the entropy of the encryption key of any Bluetooth BR/EDR (referred as Bluetooth) connection to 1 byte, without being detected by the victims (Alice and Bob). The attacker can brute force the encryption key without having to know any (pre-shared) secret material and without having to observe the Secure Simple Pairing protocol. As a result, the attacker can eavesdrop and decrypt all the traffic and inject arbitrary packets in the target Bluetooth network (piconet). The attack works regardless the usage of Secure Connections (the strongest security mode

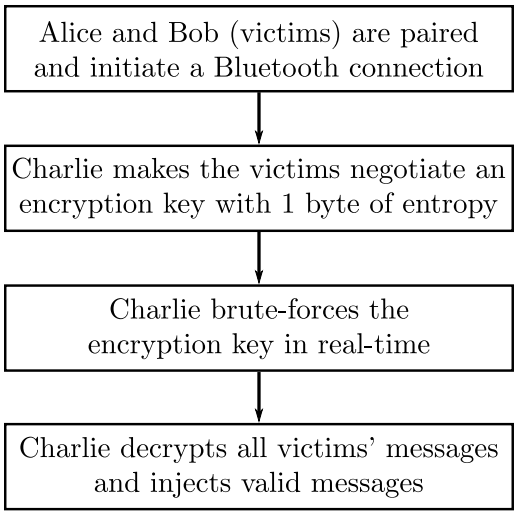


FIGURE 9.1: High level stages of a KNOB attack.

of Bluetooth). The KNOB attack high level stages are shown in Figure 9.1 and they are described in detail in the rest of this section.

9.3.1 System and Attacker Model

We assume a system composed of two or more legitimate devices that communicate using Bluetooth (as described in Section 9.2). One device is the master and the others are slaves. Without loss of generality, we focus on a piconet with one master and one slave (Alice and Bob). We indicate their Bluetooth addresses with BTADD_M and BTADD_S, and the Bluetooth clock with CLK. The clock is used for synchronization and it does not provide any security guarantee. The victims are capable of using Secure Simple Pairing and Secure Connections. This combination enables the highest security level of Bluetooth and should protect against eavesdropping and active man in the middle attacks. For example, if both devices have a display their users have to confirm that they see the same numeric sequence to mutually authenticate.

The attacker (Charlie) wants to decrypt all messages exchanged between Alice and Bob and inject valid encrypted messages, without being detected. The attacker has no access to any (pre-shared) secret material. i. e. the link key K_L and the encryption key K_C. Charlie can observe the public nonces (EN_RAND and AU_RAND), the Bluetooth clock and the packets exchanged between Alice and Bob.

We define two attacker models: a *remote* attacker and a *firmware* attacker. A remote attacker controls a device that is in Bluetooth range with Alice and Bob. He is able to passively capture encrypted messages, actively manipulate unencrypted communication, and to drop packets using techniques such as network man-in-the-middle and manipulation of physical-layer signals [187, 143]. The firmware attacker is able to compromise the firmware of the Bluetooth chip of a single victim using techniques such as backdoors [43], supply-chain implants [77], and rogue chip manufacturers [150]. The firmware attacker requires no access to the Bluetooth host (OS) and applications used by the victims.

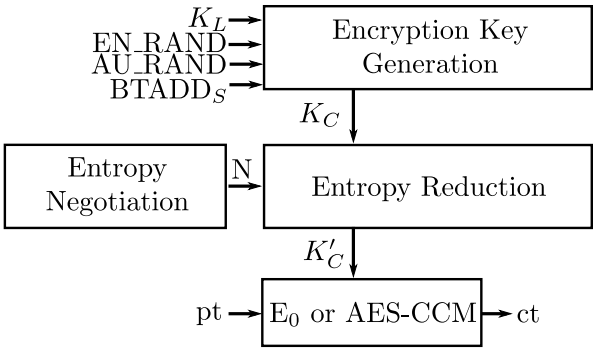


FIGURE 9.2: Generation and usage of the Bluetooth link layer encryption key (K'_C). Firstly, K_C is generated from K_L and other public parameters. K_C has 16 bytes of entropy, and it is not directly used as the encryption key. K'_C, the actual encryption key, is computed by reducing the entropy of K_C to N bytes. N is an integer between 1 and 16 and it is the result of the encryption key negotiation protocol. The N byte entropy K'_C is then used for link layer encryption by either the E_0 or the AES-CCM cipher.

9.3.2 Negotiate a Low Entropy Encryption Key

Every time a Bluetooth connection requires link-layer encryption, Alice and Bob compute an encryption key K_C based on K_L, BTADD$_S$, AU_RAND, and EN_RAND (see top part of Figure 9.2). K_L is the link key established during secure simple pairing and the others parameters are public. Assuming ideal random number generation, the entropy of K_C is always 16 bytes.

K_C is not directly used as the encryption key for the current session. The actual encryption key, indicated with K'_C, is computed by reducing the entropy of K_C to N bytes. N is the outcome of the Bluetooth *encryption key negotiation protocol* (Entropy Negotiation in Figure 9.2). The protocol is part of the Bluetooth specification since version v1.0, and it was introduced to cope with international encryption regulations and to facilitate security upgrades [162, p. 1650]. The specification of the Bluetooth encryption key negotiation protocol contains three significant problems:

1. It allows to negotiate entropy values as low as 1 byte, regardless the Bluetooth security level.
2. It is neither encrypted nor authenticated.
3. It is implemented in the Bluetooth controller (firmware) and it is transparent to the Bluetooth host (OS) and to the user of a Bluetooth application.

Hence, an attacker (Charlie) can convince any two victims (Alice and Bob) to negotiate N equal to 1, the lowest possible, yet standard-compliant, entropy value. As a result the victims compute and use a Bluetooth encryption key (K'_C) with one byte of entropy. The victims (and their OSes) are not aware about the entropy reduction of K'_C because the negotiation happens between the victims' Bluetooth controller (firmware) and the packets do not propagate to the victims' Bluetooth host (OS).

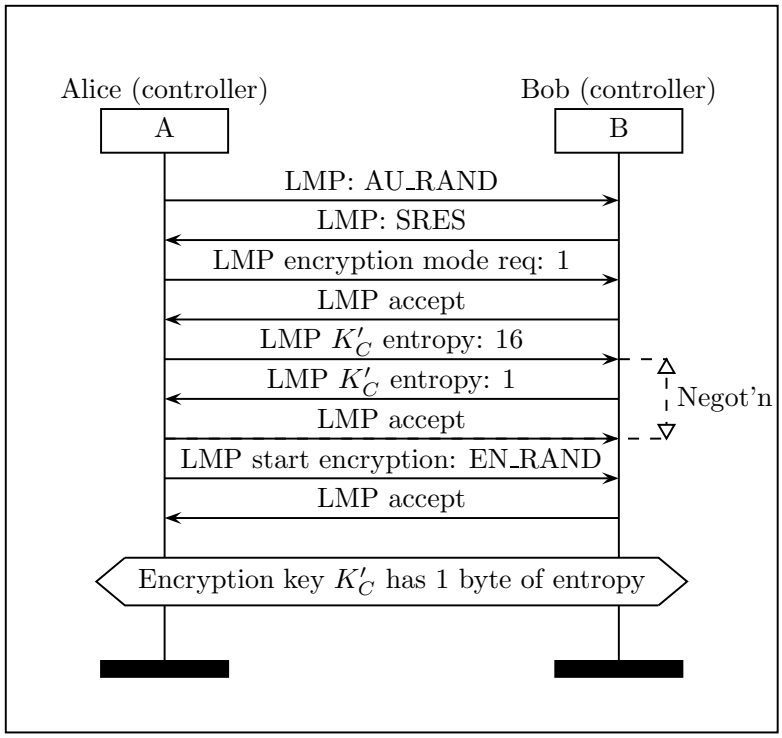


FIGURE 9.3: Alice and Bob negotiate 1 byte of entropy for the encryption key (K'_C). The protocol is run by Alice and Bob controllers (implemented in their Bluetooth chip) over the air using LMP.

To understand how an attacker can set N equal to 1 (or to any other standard-compliant value), we have to look at the details of the encryption key negotiation protocol. The protocol is run between the Bluetooth chip of Alice and Bob. In the following, we provide an example where Alice (the master) proposes 16 bytes of entropy, and Bob (the slave) is only able to support 1 byte of entropy (see Figure 9.3). The standard enables to set the minimum and maximum entropy values by setting two parameters defined as L_{min} and L_{max}. These values can be set and read only by the Bluetooth chip (firmware). Indeed, our scenario describes a situation where Alice's Bluetooth firmware declares $L_{max} = 16$ and $L_{min} = 1$, and Bob's Bluetooth firmware declares $L_{max} = L_{min} = 1$.

The encryption key negotiation protocol is carried over the Link Manager Protocol (LMP). The first two messages in Figure 9.3 allow Alice to authenticate that Bob possesses the correct K_L. Then, with the next two messages, Alice requests to initiate Bluetooth link layer encryption and Bob accepts. Now, the negotiation of N takes place (Negot'n in Figure 9.3). Alice proposes 16 bytes of entropy. Bob can either propose a smaller value or accept the proposed one or abort the negotiation. In our example, Bob proposes 1 byte of entropy because it is the only value that he supports and Alice accepts it. Then, Alice requests to activate link-layer encryption and Bob accepts. Finally, Alice and Bob compute the same encryption key (K'_C) that has 1 byte of entropy. Note that, the Bluetooth hosts of Alice and Bob do not have access to K_C and K'_C, they are only informed about the outcome of the negotiation. The key negotiation procedure

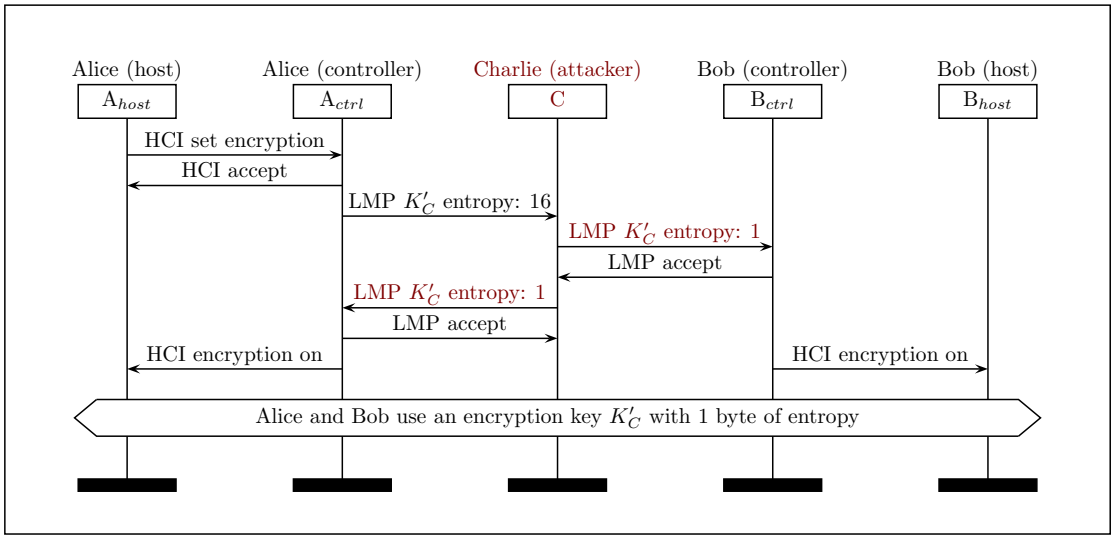


FIGURE 9.4: The KNOB attack sets the entropy of the encryption key (K'_C) to 1 byte. Alice requests Bob to activate encryption and starts the encryption key negotiation protocol. The attacker (Charlie) changes the entropy suggested by Alice from 16 to 1 byte. Bob accepts Alice's proposal and Charlie changes Bob's acceptance to a proposal of 1 byte. Alice, who originally proposed 16 bytes of entropy and she is asked to use 1 byte accepts the (standard-compliant) proposal. Charlie drops Alice's acceptance message because Bob already accepted Alice's proposal (modified by Charlie). Charlie does not know any pre-shared secret and does not observe SSP.

can also be initiated by the Bob (the slave), resulting in the same outcome.

Figure 9.4 describes how the attacker (Charlie) manages to let Alice and Bob agree on a K'_C with 1 byte of entropy when both Alice and Bob declare $L_{max} = 16$ and $L_{min} = 1$. In this Figure we also show the local interactions between hosts and controllers to emphasize that at the end of the negotiation the hosts are not informed about N and K'_C.

The attack is performed as follows: Alice's Bluetooth host requests to activate (set) encryption. Alice's Bluetooth controller accepts the local requests and starts the encryption key negotiation procedure with Bob's Bluetooth controller over the air. The attacker intercepts Alice's proposed key entropy and substitutes 16 with 1. This simple substitution works because LMP is neither encrypted nor integrity protected. Bob's controller accepts 1 byte. The attacker intercepts Bob's acceptance message and change it to an entropy proposal of 1 byte. Alice thinks that Bob does not support 16 bytes of entropy and accepts 1 byte. The attacker intercepts Alice' acceptance message and drops it. Finally, the controllers of Alice and Bob compute the same K'_C with one byte of entropy and notify their respective hosts that link-layer encryption is on.

It is reasonable to think that the victim could prevent or detect this attack using a proper value for L_{min}. However, the standard does not state how to explicitly take advantage of it, e. g. deprecate L_{min} values that are too low. The standard states the following: "The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested

key size." [162, p. 1663]. This statement is ambiguous because it is not clear what the definition of "application" is in that sentence. As we show in Section 9.5, this ambiguity results in no-one being responsible for terminating connections with low entropy keys in practice. In particular, the entity who decides whether to accept or reject the entropy proposal is the firmware of the Bluetooth chip by setting L_{min} and L_{max} and participating in the entropy negotiation protocol. The "application" (intended as the Bluetooth application running on the OS using the firmware as a service) cannot check and set L_{min} and L_{max}, and it is not directly involved in the entropy acceptance/rejection choice (that is performed by the firmware). The application can interact with the firmware using the HCI protocol. In particular, it can use the HCI Read Encryption Key Size request, to check the amount of negotiated entropy *after* the Bluetooth connection is established and theoretically abort the connection. This check is neither required nor recommended by the standard as part of the key negotiation protocol.

The low entropy negotiation presented in Figure 9.4 can be performed by both attacker models presented in Section 9.3.1. The remote attacker has the capabilities of dropping and injecting valid plaintext (the encryption key negotiation protocol is neither encrypted nor authenticated). The firmware attacker can modify few bytes in the Bluetooth firmware of a victim to always negotiate 1 byte of entropy. Furthermore, the negotiation is effective regardless of who initiates the protocol and the roles (master or slave) of the victims in the piconet.

9.3.3 Brute forcing the Encryption Key

Bluetooth has two link layer encryption schemes one is based on the E_0 cipher (legacy) and the other on the AES-CCM cipher (Secure Connections). Our KNOB attack works in both cases. If the negotiated entropy for the encryption key (K'_C) is 1 byte, then the attacker can trivially brute force it trying (in parallel) the 256 K'_C's candidates against one or more cipher texts. The attacker does not have to know what type of application layer traffic is exchanged, because a valid plaintext contains well known Bluetooth fields, such as L2CAP and RFCOMM headers, that the attacker can use as oracles.

We now describe how to compute all 1 byte entropy keys when E_0 and AES-CCM are in use. Each encryption mode involves a specific entropy reduction procedure that takes N and K_C as inputs and produces K'_C as output (Entropy Reduction in Figure 9.2). The specification of Bluetooth calls this procedure Encryption Key Size Reduction [162].

$$K'_C = g_2^{(N)} \otimes \left(K_C \bmod g_1^{(N)} \right) \tag{E_s}$$

In case of E_0, K'_C is computed using Equation (E_s), where N is an integer between 1 and 16 resulted from the encryption key negotiation protocol (see Section 9.3.2). $g_1^{(N)}$is a polynomial of degree $8N$ used to reduce the entropy of K_C to N bytes. The result of the reduction is encoded with a block code $g_2^{(N)}$, a polynomial of degree less or equal to $128-8N$. The values of these polynomials depend on N and they are tabulated in [162, p. 1668]. If $N = 1$, then we can compute the 256 candidate K'_C by multiplying all the possible 1 byte reductions $K_C \bmod g_1^{(1)}$ (the set `0x00...0xff`) with $g_2^{(1)}$ (that equals to `0x00e275a0abd218d4cf928b9bbf6cb08f`).

In case of AES-CCM the entropy reduction procedure is simpler than the one of E_0. In particular, the $16 - N$ least significant bytes of K_C are set to zero. For example, when $N = 1$ the 256 K'_C candidates for AES-CCM are the set `0x00...0xff`.

In the implementation of our KNOB attack brute force logic, we pre-compute the 512 keys with 1 byte of entropy and we store them in a look-up table to speed-up comparisons. Table 9.1 shows the first twenty K'_C with 1 byte of entropy for E_0 and AES-CCM. More details about the brute force implementation are discussed in Section 9.4.

E_0 K'_C in hex, MSB on the left	AES-CCM K'_C in hex, MSB on the left
0x00000000000000000000000000000000	0x00000000000000000000000000000000
0x00e275a0abd218d4cf928b9bbf6cb08f	0x01000000000000000000000000000000
0x01c4eb4157a431a99f2517377ed9611e	0x02000000000000000000000000000000
0x01269ee1fc76297d50b79cacc1b5d191	0x03000000000000000000000000000000
0x0389d682af4863533e4a2e6efdb2c23c	0x04000000000000000000000000000000
0x036ba322049a7b87f1d8a5f542de72b3	0x05000000000000000000000000000000
0x024d3dc3f8ec52faa16f3959836ba322	0x06000000000000000000000000000000
0x02af4863533e4a2e6efdb2c23c0713ad	0x07000000000000000000000000000000
0x0713ad055e90c6a67c945cddfb658478	0x08000000000000000000000000000000
0x07f1d8a5f542de72b306d746440934f7	0x09000000000000000000000000000000
0x06d746440934f70fe3b14bea85bce566	0x0a000000000000000000000000000000
0x063533e4a2e6efdb2c23c0713ad055e9	0x0b000000000000000000000000000000
0x049a7b87f1d8a5f542de72b306d74644	0x0c000000000000000000000000000000
0x04780e275a0abd218d4cf928b9bbf6cb	0x0d000000000000000000000000000000
0x055e90c6a67c945cddfb6584780e275a	0x0e000000000000000000000000000000
0x05bce5660dae8c881269ee1fc76297d5	0x0f000000000000000000000000000000
0x0e275a0abd218d4cf928b9bbf6cb08f0	0x10000000000000000000000000000000
0x0ec52faa16f3959836ba322049a7b87f	0x11000000000000000000000000000000
0x0fe3b14bea85bce5660dae8c881269ee	0x12000000000000000000000000000000
0x0f01c4eb4157a431a99f2517377ed961	0x13000000000000000000000000000000

TABLE 9.1: List of twenty K'_C used by E_0 (left column) and AES-CCM (right column) when $N = 1$ (key space is 256).

9.3.4 KNOB Attack Implications

The Key Negotiation Of Bluetooth (KNOB) attack exploits a vulnerability at the architectural level of Bluetooth. The vulnerable encryption key negotiation protocol endangers potentially all standard compliant Bluetooth devices, regardless their Bluetooth version number and implementation details. We believe that the encryption key negotiation protocol has to be fixed as soon as possible.

In particular the KNOB attack has serious implications related to its *effectiveness*, *stealthiness*, and *cost*. The attack is effective because it exploits a weakness in the specification of Bluetooth. The Bluetooth security mode does not matter, i. e. the attack works even with Secure Connections. The implementation details do not matter, e. g. whether Bluetooth is implemented in hardware or in software. The time constraints imposed by the Bluetooth protocols do not matter because the attacker can eavesdrop the traffic and brute force the low-entropy key offline. The type of connection does not

matter, e. g. the attack works with long-lived and short-lived connections. In a long-lived connection, e. g. victims are a laptop and a Bluetooth keyboard, the attacker has to negotiate and brute force a single low-entropy K'_C. In a short-lived connection, e. g. victims are two devices transferring files over Bluetooth, the attacker has to negotiate and brute force multiple low-entropy K'_C over time re-using the same attack technique without incurring in significant runtime and computational overheads.

The attack is stealthy because only the Bluetooth controllers (implemented in the victims' Bluetooth chip) are aware of N and K'_C. By design, the controllers are not notifying the Bluetooth hosts (implemented in the OSes) about N, but only about the outcome of the entropy negotiation. The users and the Bluetooth application developers are unaware of this problem because they use Bluetooth link-layer encryption as a trusted service.

The attack is cheap because it does not require a strong attacker model and expensive resources to be conducted. We expect that a remote attacker with commercial-off-the-shelf devices such as a software defined radio, GNU Radio and a laptop can conduct the attack.

9.3.5 KNOB Attack Root Causes

The root causes of the KNOB attack are shared between the specification and the implementation of Bluetooth BR/EDR confidentially mechanisms. On one side the specification is defining a vulnerable encryption key negotiation protocol that allows devices to negotiate low entropy values. On the implementation side (see Section 9.5), the Bluetooth applications that we tested are failing to check the negotiated entropy in practice. This is understandable because they are implementing a specification that is not mandating or explicitly recommending an entropy check.

We do not see any reason to include the encryption key negotiation protocol in the specification of Bluetooth. From our experiments (presented in Section 9.5) we observe that if two devices are not attacked they always use it in the same way (a device proposes 16 bytes of entropy and the other accepts). Furthermore, the entropy reduction does not improve runtime performances because the size of the encryption key is fixed to 16 bytes even when its entropy is reduced.

9.4 Implementation

We now discuss how we implemented the KNOB attack using a reference attack scenario. In particular, we explain how we manipulate the key negotiation protocol, brute force the encryption key (K'_C) using eavesdropped traffic, and validate K'_C by computing it from K_L as a legitimate device (as in Figure 9.2). In our attack scenario, the attacker is able to decrypt the content of a link-layer encrypted file sent from a Nexus 5 to a Motorola G3 using the Bluetooth OBject EXchange (OBEX) profile. A Bluetooth profile is the equivalent of an application layer protocol in the TCP/IP stack.

		Bluetooth			
Phone	Android	Version	MAC	SC	Chip
Nexus 5	6.0.1	4.1	48:59:29:01:AD:6F	No	BCM4339
Motorola G3	6.0.1	4.1	24:DA:9B:66:9F:83	Yes	Snapdragon 410

TABLE 9.2: Relevant technical specifications of Nexus 5 and Motorola G3 devices used to describe our attack implementation. The SC column indicates if a device supports Secure Connections.

Our implementation required significant efforts mainly due to the lack of low-cost Bluetooth protocol analyzers and software libraries implementing the custom Bluetooth security primitives (such as the modified SAFER+ block cipher). Using our implementation we conducted successful KNOB attacks on more than 14 different Bluetooth chips, the attacks are evaluated in Section 9.5.

9.4.1 Attack Scenario

To describe our implementation we use an attack scenario with two victims a Nexus 5 and a Motorola G3, Table 9.2 lists their relevant specifications. The Nexus 5 is used also as a man-in-the-middle attacker by adding extra code to its Bluetooth firmware. This setup allows us to *simulate* a remote man-in-the-middle attacker (more details in Section 9.4.2). To perform eavesdropping, we use an Ubertooth One [135] with firmware version 2017-03-R2 (API:1.02). To the best of our knowledge, Ubertooth One does not capture all Bluetooth BR/EDR packets, but it is the only open-source, low-cost, and practical eavesdropping solution for Bluetooth that we know about. To brute force K'_C and decrypt the ciphertext we use a ThinkPad X1 laptop running a Linux based OS.

The victims use the following security procedures: Secure Simple Pairing to generate K_L (the link key) and authenticate the users, the entropy reduction function from Equation (E_s), and E_0 legacy encryption. The victims use legacy encryption because the Nexus 5 does not support Secure Connections. Nevertheless, the KNOB attack works also with Secure Connections.

Every E_0-encrypted packet that contains data is transmitted and received as in Figure 9.5. A cyclic redundancy checksum (CRC) is computed and appended to the payload (Pay_{Tx}). The resulting bytes (P_{Tx}) are encrypted with E_0 using K'_C. The ciphertext is whitened, encoded, and transmitted over the air. On the receiver side the following steps are applied in sequence: decoding, de-whitening, decryption, and CRC check. The encryption and decryption procedures are the same because E_0 is a stream cipher, i. e. the same keystream is XORed with the plaintext and the ciphertext. Whitening and encoding procedures do not add any security guarantee and the Ubertooth One is capable of performing both procedures.

9.4.2 Manipulation of the Entropy Negotiation

We implement the manipulation of the encryption key negotiation protocol (presented in Section 9.3.2) by extending the functionalities of InternalBlue [114] and using it to

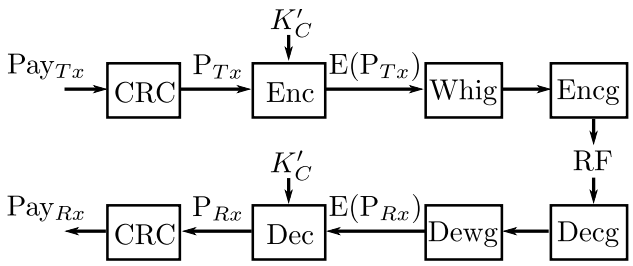


FIGURE 9.5: Transmission and reception of an E_0 encrypted payload. The concatenation of the payload and its CRC (P_{Tx}) is encrypted, whitened, encoded and then transmitted. On the receiver side the steps are applied in the opposite order. RF is the radio frequency wireless channel.

patch the Bluetooth chip firmware of the Nexus 5. Our InternalBlue modifications allow to manipulate all incoming LMP messages *before* they are processed by the entropy negotiation logic, and all outgoing LMP messages *after* they've been processed by the entropy negotiation logic. The entropy negotiation logic is the code in the Nexus 5 Bluetooth firmware that manages the encryption key negotiation protocol, and we do not modify it. As a result, we can use a Nexus 5 (or any other device supported by InternalBlue) as a victim and a remote KNOB attacker without having to deal with the practical issues related with wireless attacks over-the-air.

InternalBlue is an open-source toolkit capable of interfacing with the firmware of the BCM4339 Bluetooth chip in Nexus 5 phones. To use it, one has to root the target Nexus 5 and compile and install the Android Bluetooth stack with debugging features enabled. InternalBlue allows to patch the firmware in real-time (e. g. start LMP monitoring) and read the ROM and the RAM of firmware at runtime. InternalBlue provides a way to hook and execute arbitrary code in the Bluetooth firmware. At the time of writing, InternalBlue is not capable of hooking directly the key negotiation logic. However, we managed to extend it to enable two victims (one is always the Nexus 5) to negotiate one (or more) byte of entropy.

Our manipulation of the entropy negotiation works regardless the role of the Nexus 5 in the piconet and it does not require to capture any information about the Secure Simple Pairing process. Assuming that the victims are already paired, we test if two victims are vulnerable to the KNOB attack as follows:

1. We connect over USB the Nexus 5 with the X1 laptop, we run our version of InternalBlue, and we activate LMP and HCI monitoring.

2. We connect and start the Ubertooth One capture over the air focusing only on the Nexus 5 piconet (using UAP and LAP flags).

3. We request a connection from the Nexus 5 to the victim (or vice versa) to trigger the encryption key negotiation protocol over LMP.

4. Our InternalBlue patch changes the LMP packets as Charlie does in Figure 9.4.

5. If the victims successfully complete the protocol, then they are vulnerable to the KNOB attack and we can decrypt the ciphertext captured with the Ubertooth One.

We now describe how we extended InternalBlue to perform the fourth step of the list. In this context, the most important file of InternalBlue is `internalblue/fw_5.py`. This file contains all the information about the BCM4339 firmware, and it provides two hooks into the firmware, defined by Mantz (the main author of InternalBlue) as `LMP_send_packet` and `LMP_dispatcher`. The former hook allows to execute code every time an LMP packet is about to be sent and the latter whenever an LMP packet is received. The hooks are intended for LMP monitoring, and we upgraded them to be used also for LMP manipulation.

Listing 4 shows three ARM assembly code blocks that we added to `fw_5.py` to let the Nexus 5 and the Motorola G3 negotiate 1 byte of entropy. In this case the Nexus 5 is the master and it initiates the encryption key negotiation protocol. The first block translates to: if the Nexus 5 is sending an LMP K'_C entropy proposal then change it to 1 byte. This block is executed when the Nexus 5 starts an encryption key negotiation protocol. The code allows to propose any entropy value by `mov`ing a different constant into `r2` in line 5.

The second block from Listing 4 translates to: if the Nexus 5 is receiving an LMP accept (entropy proposal), then change it to an LMP K'_C entropy proposal of 1 byte. This code is used to let the Nexus 5 firmware believe that the other victim proposed 1 byte, while she already accepted 1 byte (assuming that she is vulnerable). The third blocks translates to: if the Nexus 5 is sending an LMP accept (entropy proposal), then change it to an LMP preferred rate. This allows to obtain the same result of dropping an LMP accept packet because the LMP preferred rate packet does not affect the state of the encryption key negotiation protocols. We developed and used similar patches to cover the other attack cases: Nexus 5 is the master and does not initiate the connection, Nexus 5 is the slave and initiates the connection and Nexus 5 is the slave and does not initiate the connection.

9.4.3 Brute Forcing the Encryption Key

Once the attacker is able to reduce the entropy of the encryption key (K'_C) to 1 byte, he has to brute force the key value (key space is 256). In this section we explain how we brute forced and validated a E_0 encryption key with 1 byte of entropy. The key was used in one of our KNOB attacks to decrypt the content of a file transferred over a link layer encrypted Bluetooth connection.

The details about the E_0 encryption scheme are presented in Figure 9.6, we describe them backwards starting from the E_0 cipher. E_0 takes three inputs: BTADD_M, CLK26-1 and K'_C. CLK26-1 are the 26 bits of CLK in the interval CLK[25:1] (assuming that CLK stores its least significant bit at CLK[0]). The BTADD_M is the Bluetooth address of the master and it is a public parameter. We did not have to implement the E_0 cipher because we found an open-source implementation [48] which we verified against the specification of Bluetooth. To provide valid K'_C candidates to E_0 we had to implement the E_s entropy reduction procedure. This procedure takes an input with 16 bytes of entropy (K_C) and computes an output with N bytes of entropy (K'_C). E_s involves

Listing 4 We add three ARM assembly code blocks to `internalblue/fw_5.py` to negotiate K'_C with 1 byte of entropy. In this case the Nexus 5 is the master and it initiates the encryption key negotiation protocol.

```
# Send LMP Kc' entropy 1 rather than 16
ldrb  r2, [r1]
cmp   r2, #0x20
bne   skip_sent_ksr
mov   r2, #0x01
strb  r2, [r1, #1]
skip_sent_ksr:

# Recv LMP Kc' entropy 1 rather than LMP accept
ldrb   r2, [r1]
cmp    r2, #0x06
bne    skip_recv_aksr
ldrb   r2, [r1, #1]
cmp    r2, #0x10
bne    skip_recv_aksr
mov    r2, #0x20
strb   r2, [r1]
mov    r2, #0x01
strb   r2, [r1, #1]
skip_recv_aksr:

# Send LMP_preferred rate rather than LMP accept
# Simulate an attacker dropping LMP accept
ldrb  r2, [r1]
cmp   r2, #0x06
bne   skip_send_aksr
ldrb  r2, [r1, #1]
cmp   r2, #0x10
bne   skip_send_aksr
mov   r2, #0x48
strb  r2, [r1]
mov   r2, #0x70
strb  r2, [r1, #1]
skip_send_aksr:
```

modular arithmetic over polynomials in Galois fields and we use the BitVector [95] Python module to perform such computations.

Our Python brute force script takes a ciphertext (captured over the air using Ubertooth One) and tries to decrypt it by using the E_0 cipher with all possible values of K'_C. We validate our script by decrypting the content of a file sent from the Nexus 5 to the Motorola G3 using the OBEX Bluetooth profile after the negotiation of 1 byte of entropy. The content of the file (in ASCII) is `aaaabbbbccccdddd`. We discuss several brute forcing practical issues in Section 9.6.3.

Once we found the matching plaintext we wanted to verify that the brute forced key was effectively the one in use by the victims. To do that we had to implement E_1 and E_3, the former is used to compute the Ciphering Offset Number (COF), the latter to compute K_C (see Figure 9.6). Both procedures use a custom hash function defined in the specification of Bluetooth with $\star$. We write E_1 and E_3 equations and label them

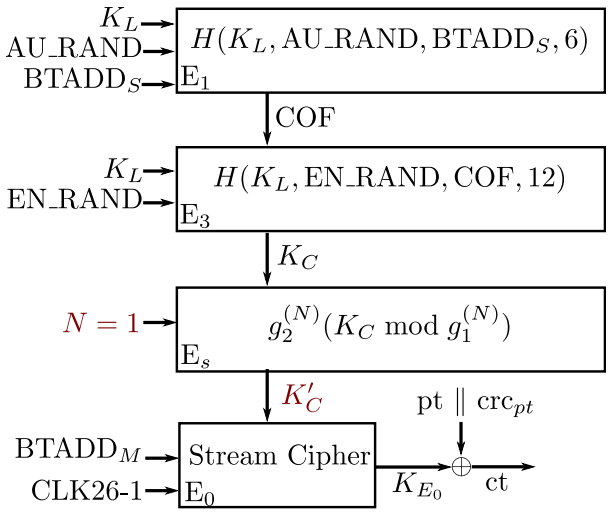


FIGURE 9.6: Implementation of the KNOB attack on the E_0 cipher. The attacker makes the victims agree on a K'_C with one byte of entropy ($N = 1$) and then brute force K'_C, without knowing K_L and K_C.

with their respective names as follows:

$$SRES\|ACO = H(K_L, \text{AU\_RAND}, \text{BTADD}_\text{S}, 6) \quad (E_1)$$
$$K_C = H(K_L, \text{EN\_RAND}, \text{COF}, 12) \quad (E_3)$$

Figure 9.7 shows how E_3 uses the ˝ hash function, ˝ internally uses SAFER+, a block cipher that was submitted as an AES candidate in 1998 [117]. SAFER+ is used with 128 bit block size (8 rounds), in ECB mode, and only for encryption. SAFER+' (SAFER+ prime) is a modified version of SAFER+ such that the input of the first round is added to the input of the third round. This modification was introduced in the specification of Bluetooth to avoid SAFER+' being used for encryption [162, p. 1677].

We implemented in Python both SAFER+ and SAFER+' including the round computations and the key scheduling algorithm. We tested the two against the specification of Bluetooth (where they are indicated with A_r and A_r' [162, p. 1676]). We also implemented the E and O blocks from Figure 9.7. The E block is an extension block that transforms the 12 byte COF into a 16 byte sequence using modular arithmetic. The same block is applied to the 6 byte BTADD_S in E_1. The O block is offsetting K_L using algebraic (modular) operations and the largest primes below 257 for which 10 is a primitive root. We implement the E and O blocks in Python and we tested them against the specification of Bluetooth. Then, we were able to implement ˝ and to use it to implement and test E_3 and E_1.

We validate the brute forced K'_C by using the necessary parameters from Figure 9.6 to compute K'_C from K_L. We captured the parameters using the Bluetooth logging capabilities offered by Android. Table 9.3 shows an example of actual public and private values used during one of our KNOB attacks. We plan to release our code implementing E_s, E_1 and E_3 as open-source to help researchers interested in Bluetooth's security, after we complete the responsible disclosure of our findings[1].

[1]See https://github.com/francozappa/knob

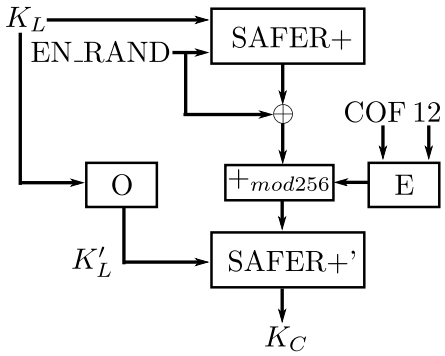


FIGURE 9.7: Bluetooth defines H a custom hash function based on SAFER+. H is used to compute K_C from K_L, EN_RAND, and COF (see Equation E_3).

9.4.4 Implementation for Secure Connections

The specification of Bluetooth allows to perform the KNOB attack even when the victims are using Secure Connections. We already implemented the entropy reduction function of the brute force script over AES–CCM. However, at the time of writing, InternalBlue is not capable of patching the firmware of a Bluetooth chip that supports Secure Connections, indeed we are not able to implement the low entropy negotiation part of the attack using InternalBlue.

9.5 Evaluation

Our implementation of the KNOB attack (presented in Section 9.4) allows to test if any device accepts an encryption key with 1 byte of entropy ($N = L_{min} = 1$). We focus our discussion on the attack best case (1 byte of entropy) while arguably any entropy value lower than 14 bytes could be considered not secure for symmetric encryption [19].

After successfully conducting the KNOB attack on a Nexus 5 and a Motorola G3 we conducted other KNOB attacks on more than 14 unique Bluetooth chips (by attacking 21 different devices). Each attack is easy to reproduce and testing if a device is vulnerable is a matter of seconds.

Based on our experiments, we concluded that there are no differences between the specification and the implementation of both the Bluetooth controller (implemented in the firmware) and the Bluetooth host (implemented in the OS and usable as an interface by a Bluetooth application). In the former case the specification is not enforcing any minimum L_{min} and it is not protecting the entropy negotiation protocol. The firmware's implementers (to provide standard-compliant products) are allowing the negotiation of 1 byte of entropy with an insecure protocol. The only exception is the Apple W1 chip where an attacker can only reduce the entropy to 7 bytes. In the latter case, the Bluetooth specification is providing an HCI Read Encryption size API but it is not mandating or recommending its usage, e. g. a mandatory check at the end of the LMP entropy negotiation. The host's implementers are providing this API and the applications that we tested are not using it.

Name	Value
Public	
BTADD_M	`0xccfa0070dcb6`
BTADD_S	`0x829f669bda24`
AU_RAND	`0x722e6ecd32ed43b7f3cdbdc2100ff6e0`
EN_RAND	`0xd72fb4217dcdc3145056ba488bea9076`
SRES	`0xb0a3f41f`
N	`0x1`
Secret	
K_L	`0xd5f20744c05d08601d28fa1dd79cdc27`
COF=ACO	`0x1ce4f9426dc2bc110472d68e`
K_C	`0xa3fccef22ad2232c7acb01e9b9ed6727`
K'_C	`0x7fffffffffffffffffffffffffffffff`

TABLE 9.3: Public and secret values (in hexadecimal representation) collected during a KNOB attack involving authenticated SSP and E_0 encryption. The encryption key (K'_C) has 1 byte of entropy.

9.5.1 Evaluation Setup

To perform our evaluation we collected as many devices as possible containing different Bluetooth chips. At the time of writing, we were able to test chips from Broadcom, Qualcomm, Apple, Intel, and Chicony manufacturers. For each chip we conducted the KNOB attack following the same five steps presented in Section 9.4.2. As explained earlier, the Nexus 5 is used as a (remote) attacker and a victim. For each test we recorded the manipulated encryption key negotiation protocol over LMP in a pcapng file and we manually verified the protocol's outcome with Wireshark.

Our evaluation setup is not hard to reproduce and easy to extend because it does not require expensive hardware and uses open-source software. We would like to see other researchers evaluating more Bluetooth chips and devices that currently we do not posses, e. g. Apple Watches.

9.5.2 Evaluation Results

Table 9.4 shows our evaluation results. Overall, we tested more than 14 Bluetooth chips and 21 devices. The first column contains the Bluetooth chip names. We fill the entries of this column with Unknown when we are not able to find information about the chip manufacturer and/or model number. The second column lists the devices that we tested grouped by chip, e. g. the Snapdragon 835 is used both by the Pixel 2 and the OnePlus 5. The third column contains a ✓ if the Bluetooth chip accepts 1 byte of entropy and a * if it accepts at least 7 bytes. The table's rows are grouped by Bluetooth version in four blocks: version 5.0, version 4.2, version 4,1 and version lower or equal than 4.0.

From the third column of Table 9.4 we see that all the chips accept 1 byte of entropy (✓) except the Apple W1 chip (*) that requires at least 7 bytes of entropy. Apple W1 and its successors are used in devices such as AirPods, and Apple Watches. Seven bytes of

Bluetooth chip	Device(s)	Vulnerable?
Bluetooth Version 5.0		
Snapdragon 845	Galaxy S9	✓
Snapdragon 835	Pixel 2, OnePlus 5	✓
Apple/USI 339S00428	MacBookPro 2018	✓
Apple A1865	iPhone X	✓
Bluetooth Version 4.2		
Intel 8265	ThinkPad X1 6th	✓
Intel 7265	ThinkPad X1 3rd	✓
Unknown	Sennheiser PXC 550	✓
Apple/USI 339S00045	iPad Pro 2	✓
BCM43438	RPi 3B, RPi 3B+	✓
BCM43602	iMac MMQA2LL/A	✓
Bluetooth Version 4.1		
BCM4339 (CYW4339)	Nexus 5, iPhone 6	✓
Snapdragon 410	Motorola G3	✓
Bluetooth Version $\leq$ 4.0		
Snapdragon 800	LG G2	✓
Intel Centrino 6205	ThinkPad X230	✓
Chicony Unknown	ThinkPad KT-1255	✓
Broadcom Unknown	ThinkPad 41U5008	✓
Broadcom Unknown	Anker A7721	✓
Apple W1	AirPods	*

TABLE 9.4: List of Bluetooth chips and devices tested against the KNOB attack. ✓indicates that a chip accepts one byte of entropy. * indicates that a chip accepts at least seven bytes of entropy. We note that, all chips and devices implementing any specification of Bluetooth are expected to be vulnerable to the KNOB attack because the entropy reduction feature is standard-compliant.

entropy are better than one, but not enough to prevent brute force attacks. For example, the Data Encryption Standard (DES) uses the same amount of entropy and DES keys were brute forced multiple times with increasing efficacy [103].

Table 9.4 also demonstrates that the vulnerability spans across different Bluetooth versions including the latest ones such as 5.0 and 4.2. This fact confirms that the KNOB attack is a significant threat for all Bluetooth users and we believe that the specification of Bluetooth has to be fixed as soon as possible.

9.6 Discussion

9.6.1 Attacking Other Bluetooth Profiles

Cable replacement wireless technologies such as Bluetooth are widely used for all sorts of applications including desktop, mobile, IoT, industrial and medical devices. Bluetooth defines its set of application layer services as profiles. In Section 9.4 we describe an attack on the OBject EXchange (OBEX) Bluetooth profile, where the attacker breaks Bluetooth security by decrypting the content of an encrypted file without having access to any (pre-shared) secret. Here we describe three KNOB attacks targeting other popular Bluetooth profiles. As in the OBEX case, the attacks have serious implications in terms of security and privacy of the victims. To the best of our knowledge, all the profiles that we discuss in this section rely only on the link-layer for their security guarantees and they are widely used across different vendors. Our list of attacks is not exhaustive and an attacker might exploit the vulnerable encryption key negotiation protocol of Bluetooth in other creative ways.

HID profile The attacker could perform a remote keylogging attack on any device that uses the Human Interface Device (HID) profile. This profile is used by input-output devices such as keyboards, mice and joysticks. As a result, the attacker can sniff sensitive information including passwords, credit card numbers, and emails regardless if these information are then encrypted on the (wired or wireless) Ethernet link.

Bluetooth tethering The attacker could mount a remote man-in-the-middle attack when the victim uses Bluetooth for tethering. Tethering is used by a device, acting as an hotspot, to share Internet connectivity with other devices in range. Bluetooth transports Ethernet over the Bluetooth Network Encapsulation Protocol (BNEP) [26]. This protocol encapsulates Ethernet frames and transports them over (link-layer encrypted) L2CAP. As a result, the attacker can sniff all Internet traffic of the victims using a Bluetooth hotspot.

A2DP profile The attacker could record and inject audio signals when the victim uses the Advanced Audio Distribution Profile (A2DP) profile. As a result, the attacker is able to record phone and Voice over IP (VoIP) calls even if the call is encrypted (e. g. 4G and Skype). The attacker can also tamper with voice commands sent to a personal assistant, e. g. Siri and Google Assistant. Recent mobile devices, such as smartphone and tablets, are particularly vulnerable to this threat because Bluetooth is a convenient solution to the lack of an analog audio connector (audio jack).

9.6.2 Attacking Multiple Nodes and Piconets

In our word we describe the implementation of KNOB attacks targeting two victims. If a Bluetooth piconet contains more than two devices, then (in the worst case for the attacker) each master-slave pair uses a dedicated set of keys. In this scenario the KNOB attack still works because it can be parallelized with minimal effort. For example, the attacker may run the same attack script on different computing units, such as processes or machines, and let each computing unit target a master-slave pair. Each parallel

instance of the attack negotiates an encryption key with one byte of entropy, captures the exchanged ciphertext, and brute forces the encryption key. For example, an attacker is able to decrypt all the traffic from a victim using multiple Bluetooth I/O devices to interact with his device e. g. a laptop connected with a keyboard, a mouse and an headset.

The KNOB attack is effective even if the attacker wants to target multiple piconets (Bluetooth networks) at the same time. In this case the attacker has to follow and use a different Bluetooth clock (CLK) value for each piconet to compute the correct encryption key. This is not a problem because the attacker can use parallel KNOB attack instances, where each instance follows a pair of devices in a target piconet.

9.6.3 Practical Implementation Issues

We spent considerable time to fine tune our brute force script. One main reason is that Ubertooth One, used to sniff Bluetooth BR/EDR packets over the air, does not reliably capture all packets and clock values (CLK). This is true even if we limit our capture to a specific piconet by setting the UAP and LAP parameters. As a result, we had to include extra logic in our brute force script to iterate over different CLK values and E_0 keystream offsets. Our basic brute force logic only iterates over the encryption key space (256 iterations). The extra logic can be removed if we get access to a commercial-grade Bluetooth protocol analyzer such as Ellisys [57] or similar. Unfortunately, these devices are very expensive.

We implemented our attack by simulating a remote attacker using InternalBlue. Alternatively, we could have conducted the attacks over the air using signal manipulation [143] and (reactive) jamming [187]. However, the InternalBlue setup is simpler, more reliable, cheaper, and easier to reproduce than the over-the-air setup and it affects the victims in the same way as a remote attacker.

9.6.4 Countermeasures

In this section we propose several countermeasures to the KNOB attack. We divide them into two classes: legacy compliant and non legacy compliant. The former type of countermeasure does not require a change to the specification of Bluetooth while the latter does. We already proposed these countermeasures to the Bluetooth SIG and CERT during our responsible disclosure.

Legacy compliant. Our first proposed legacy compliant countermeasure is to require a minimum and maximum amount of negotiable entropy that cannot be easily brute forced, e. g. require 16 bytes of entropy. This means fixing L_{min} and L_{max} in the Bluetooth controller (firmware) and results in the negotiation of proper encryption keys. Another possible countermeasure is to automatically have the Bluetooth host (OS) check the amount of negotiated entropy each time link layer encryption is activated and abort the connection if the entropy does not meet a minimum requirement. The entropy value can be obtained by the host using the HCI Read Encryption Key Size Command. This solution requires to modify the Bluetooth host and it might be suboptimal because it acts on a connection that is already established (and possibly in use), not as part of the entropy negotiation protocol. A third solution is to distrust the link

layer and provide the security guarantees at the application layer. Some vendors have done so by adding a custom application layer security mechanism on top of Bluetooth (which, in case of Google Nearby Connections, was also found to be vulnerable [11]).

Non legacy compliant. A non legacy compliant countermeasure is to modify the encryption key negotiation protocol by securing it using the link key. The link key is a shared (and possibly authenticated) secret that should be always available before starting the entropy negotiation protocol. The new protocol should provide message integrity and might also provide confidentiality. Preferably, the specification should get rid of the entropy negotiation protocol and always use encryption keys with a fixed amount of entropy, e. g. 16 bytes. The implementation of these solutions only requires the modification of the Bluetooth controller (firmware).

9.7 Related Work

The security and privacy guarantees of Bluetooth were studied since Bluetooth v1.0 [94, 190]. Particular attention was given to Secure Simple Pairing (SSP), a mechanisms that Bluetooth uses to generate and share a long term secret (defined as the link key). Several attacks on the SSP protocol were proposed [160, 81, 24]. The Key Negotiation Of Bluetooth (KNOB) attack works regardless of security guarantees provided by SSP (such as mutual user authentication).

The most up to date survey about Bluetooth security was provided by NIST in 2017 [137]. This survey recommends to use 128 bit keys (16 bytes of entropy). It also describes the key negotiation protocol, and considers it as a security issue when one of the connected devices is malicious (and not a third party). Prior surveys do not consider the problem of encryption key negotiation at all [53] or superficially discuss it [176].

The various implementation of Bluetooth were also analyzed and several attacks were presented on Android, iOS, Windows and Linux implementations [15]. Our attack works regardless of the implementation details of the target platform, because if any implementation is standard-compliant then it is vulnerable to the KNOB attack.

The security of the ciphers used by Bluetooth has been extensively discussed by cryptographers. The SAFER+ cipher used by Bluetooth for authentication purposes was analyzed [100]. The E_0 cipher used by Bluetooth for encryption was also analyzed [61]. Our attack works even with perfectly secure ciphers. For our implementation of the custom Bluetooth security procedures (presented in Section 9.4) we used as main references the specification of Bluetooth [162] and third-party hardware [101] and software [113] implementations.

Third-party manipulations of key negotiation protocols were also discussed in the context of WiFi, for example key reuse in [178]. Compared to those attacks, our attack exploits not only implementation issues, but a standard-compliant vulnerability of the specification of Bluetooth.

Protocol downgrade attacks were discussed in the context of TLS[123], where the two parties are negotiating the cipher suite to use. We note that in contrast to our scenario, for TLS the application developers have commonly direct control over the cipher suites that will be offered by their applications. Therefore, avoiding a fallback

to legacy encryption standards can be prevented by the developers. To the best of our knowledge, this is not the case for Bluetooth, as the protocols does not enforce any mandatory checks on the encryption key's entropy.

9.8 Conclusion

In this chapter we present the Key Negotiation Of Bluetooth (KNOB) attack. Our attack is capable of reducing the entropy of the encryption key of any Bluetooth BR/EDR connection to 1 byte (8 bits). The attack is standard-compliant because the specification of Bluetooth includes an insecure encryption key negotiation protocol that supports entropy values between 1 and 16 bytes. As a main consequence, an attacker can easily negotiate an encryption key with low entropy and then brute force it. The attacker is effectively breaking the security guarantees of Bluetooth without having to posses any (pre-shared) secret material. The attack is stealthy because the vulnerable entropy negotiation protocol is run by the victims' Bluetooth controller and this protocol is transparent to the Bluetooth host (OS) and the Bluetooth application used by the victims. We expect that the attack could be run in parallel to target multiple devices and piconets at the same time.

We demonstrate that the KNOB attack can be performed in practice by implementing it to attack a Nexus 5 and a Motorola G3. In our attack we decrypt a file transmitted over an authenticated and link-layer encrypted Bluetooth connection. Brute-forcing a key with 1 byte of entropy introduces a negligible overhead enabling an attacker to decrypt all the ciphertext and to introduce valid ciphertext even in real-time.

We evaluate the KNOB attack on more than 14 Bluetooth chips from different vendors such as Broadcom, Qualcomm and Intel. All the chips accept 1 byte of entropy except the Apple W1 chip that accepts (at least) 7 bytes of entropy. Frankly, we were expecting to find more non standard-compliant chips like the Apple W1. Before submitting our work, we reported our findings to the Computer Emergency Response Team (CERT) and the Bluetooth Special Interest Group (SIG). Both organizations acknowledged the problem and we are collaborating with them to solve it. After our responsible disclosure, we plan to release the tools that we developed to implement the attacks as open-source.

The KNOB attack is a serious threat to the security and privacy of all Bluetooth users. We were surprised to discover such fundamental issues in a widely used and 20 years old standard. We attribute the identified issues in part to ambiguous phrasing in the standard, as it is not clear who is responsible for enforcing the entropy of the encryption keys, and as a result no-one seems to be responsible in practice. We urge the Bluetooth SIG to update the specification of Bluetooth according to our findings. Until the specification is not fixed, we do not recommend to trust any link-layer encrypted Bluetooth BR/EDR link. In Section 9.6.4 we propose legacy and non legacy compliant countermeasures that would make the KNOB attack impractical. We also recommend the Bluetooth SIG to create a dedicated procedure enabling researchers to securely submit new potential vulnerabilities, similarly to what other companies, such as Google, Microsoft and Facebook, are offering.

Chapter 10

Conclusion about Wireless Systems Security

Wireless technologies, such as Wi-Fi and Bluetooth, are pervasive in our society and they are used to exchange sensitive information and to monitor and control remote systems. In the second part of the thesis we focused on securing these technologies. We defined three problems that we wanted to tackle: evaluation and usability of deployed physical layer features as security defenses, improvement of the accessibility of complex (and proprietary) wireless technologies, and verification (and improvement) of the security posture of widely adopted wireless systems.

We revised recent amendments of the IEEE 802.11 (WLAN) standard [9] because 802.11 includes several advanced physical layer features such as MIMO, spatial diversity and spatial multiplexing that might be used as defense mechanisms. We concluded that indeed these features are capable of providing some benefits, e. g. beamforming reduces the SNR of a passive eavesdropper that is far from the main lobe of transmission.

Nearby Connections is an API to include proximity based services into Android and Android Things application. A single vulnerability in this API can be exploited on millions of devices. Despite the wide attack surface of Nearby Connections, before our work there were no information about its security mechanisms. In [11][1] we present the first security analysis of Nearby Connections. This technologies presents several significant problems such as the lack of mandatory authentication both at the link layer and the application layer. The takeaway message is that "encryption can be meaningless without authentication" and the attacks that we demonstrated are a direct consequence of that.

Bluetooth is a pervasive wireless technology used in many scenario such as mobile, IoT, car, medical, and industrial. The Bluetooth specifications is extremely complex, there is no reference implementation available and no detailed security analysis of its components. In [12][2] we describe an high impact vulnerability at the architectural level of Bluetooth that we found while revising the Bluetooth standard. Our finding allows an attacker to reduce the entropy of the encryption key of any Bluetooth BR/EDR connection as low as 1 Byte. The attacker does not have observe the Bluetooth secure simple pairing. We call our attack the KNOB (Key Negotiation Of Bluetooth) attack, and we demonstrate that "the KNOB is broken" by exploiting a broad set of Bluetooth

[1] https://francozappa.github.io/project/rearby/
[2] https://francozappa.github.io/project/knob/

chips from Intel, Broadcom, Qualcomm, and Apple. We collaborated with the industry to fix the Bluetooth specification[3].

10.1 Lessons Learnt

During this PhD I've learnt many valuable lessons. Here I'm sharing some of them regarding wireless system security:

- Security through obscurity is still a problem for wireless systems, especially in the case of SoC and related firmwares.
- Controlled experiments and attacks over the air are difficult to execute.
- Architectural attacks on any protocol have devastating consequences.

10.2 Future Work

These are the main research directions that we would like to see in the future:

- Development of security-oriented physical layer features.
- Case studies about already deployed physical layer features that can used for defensive purposes.
- Security evaluations of popular (proprietary) technologies that are managing sensitive data.
- Development of low cost software and hardware tools to ease (security) evaluations of wireless technologies.

[3]See Chapter 5 for our conclusion about cyber-physical systems security.

Bibliography

[1] Sridhar Adepu and Aditya Mathur. "An investigation into the response of a water treatment system to cyber attacks". In: *Proc. of Symposium on High Assurance Systems Engineering (HASE)*. IEEE. 2016.

[2] Sridhar Adepu and Aditya Mathur. "Detecting multi-point attacks in a water treatment system using intermittent control actions". In: *Proc. of the Singapore Cyber-Security Conference (SG-CRC)*. 2016.

[3] Sridhar Adepu and Aditya Mathur. "Distributed Detection of Single-Stage Multipoint Cyber Attacks in a Water Treatment Plant". In: *Proc. of the Asia Conference on Computer and Communications Security (ASIACCS)*. 2016.

[4] Narendra Anand, Sung-Ju Lee, and Edward W Knightly. "Strobe: Actively securing wireless communications using zero-forcing beamforming". In: *INFOCOM, 2012 Proceedings IEEE*. IEEE. 2012.

[5] Android Developers. *Overview of Google Play Services*. https://developers.google.com/android/guides/overview, Accessed: 2018-01-26.

[6] Daniele Antonioli. *MiniCPS public repository*. https://github.com/scy-phy/minicps.

[7] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. "Towards high-interaction virtual ICS honeypots-in-a-box". In: *Proceedings of the ACM Workshop on Cyber-Physical Systems Security and Privacy (co-located with CCS)*. https://dl.acm.org/citation.cfm?id=2994493 **Research excellence award by ST Electronics FIRST workshop 2017**. ACM. 2016, pp. 13–22.

[8] Daniele Antonioli, Giuseppe Bernieri, and Nils Ole Tippenhauer. "Taking control: Design and implementation of botnets for cyber-physical attacks with cps-bot". In: *arXiv preprint arXiv:1802.00152* (2018).

[9] Daniele Antonioli, Sandra Siby, and Nils Ole Tippenhauer. "Practical Evaluation of Passive COTS Eavesdropping in 802.11b/n/ac WLAN". In: *Proceedings of the Cryptology And Network Security (CANS) conference*. 2017.

[10] Daniele Antonioli and Nils Ole Tippenhauer. "MiniCPS: A toolkit for security research on CPS networks". In: *Proceedings of the ACM Workshop on Cyber-Physical Systems-Security and/or Privacy (co-located with CCS)*. https://dl.acm.org/citation.cfm?id=2808715, Repo: https://github.com/scy-phy/minicps. ACM. 2015, pp. 91–100.

[11] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. "Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2019.

[12] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. "The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation of Bluetooth BR/EDR." In: *Proceedings of the USENIX Security Symposium*. 2019.

[13] Daniele Antonioli et al. "Gamifying ICS Security Training and Research: Design, Implementation, and Results of S3". In: *Proceedings of the ACM Workshop on Cyber-Physical Systems Security and Privacy (co-located with CCS)*. https://dl.acm.org/citation.cfm?id=3140253. 2017.

[14] William A Arbaugh et al. *Real 802.11 security: Wi-Fi protected access and 802.11 i*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[15] Armis Inc. *The Attack Vector BlueBorne Exposes Almost Every Connected Device*. https://armis.com/blueborne/, Accessed: 2018-01-26.

[16] Michael Backes, Sven Bugiel, and Erik Derr. "Reliable third-party library detection in android and its security applications". In: *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM. 2016, pp. 356–367.

[17] Xiaolong Bai et al. "Staying secure and unprepared: Understanding and mitigating the security risks of apple zeroconf". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 655–674.

[18] Elaine Barker, L Chen, and et al Roginsky A. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf. Recommendations of the NIST. 2018.

[19] Elaine Barker et al. "Recommendation for key management part 1: General (revision 3)". In: *NIST special publication* 800.57 (2012), pp. 1–147.

[20] David C. Bergman and David M. Nicol. "Test Bed for Evaluation of Power Grid Cyber-Infrastructure". In: *Real-Time Simulation Technologies Principles, Methodologies, and Applications*. Ed. by PJ Mosterman K Popovici. CRC Press, 2012.

[21] Massimo Bernaschi, Francesco Ferreri, and Leonardo Valcamonici. "Access points vulnerabilities to DoS attacks in 802.11 networks". In: *Wireless Networks* (2008).

[22] Daniel J. Bernstein and Tanja Lange. *SafeCurves: choosing safe curves for elliptic-curve cryptography*. https://safecurves.cr.yp.to, Accessed: 2018-07-16.

[23] Thirumalesh Bhat and Nachiappan Nagappan. "Evaluating the efficacy of test-driven development: industrial case studies". In: *Proc. of symposium on Empirical Software Engineering (ISESE)*. 2006, pp. 1–8. ISBN: 1595932186. URL: http://dl.acm.org/citation.cfm?id=1159787.

[24] Eli Biham and Lior Neumann. *Breaking the Bluetooth Pairing–Fixed Coordinate Invalid Curve Attack*. http://www.cs.technion.ac.il/~biham/BT/bt-fixed-coordinate-invalid-curve-attack.pdf, Accessed: 2018-10-30.

[25] Biondi, Philippe. *Scapy*. http://www.secdev.org/projects/scapy.

[26] Bluetooth SIG. *Bluetooth Network Encapsulation Protocol*. http://grouper.ieee.org/groups/802/15/Bluetooth/BNEP.pdf, Accessed: 2018-10-28. 2001.

[27] Nikita Borisov, Ian Goldberg, and David Wagner. "Intercepting mobile communications: the insecurity of 802.11". In: *Proceedings of the Annual international Conference on Mobile computing and networking (MobiCom)*. ACM. 2001, pp. 180–189.

[28] Elie Bursztein et al. "Webseclab Security Education Workbench". In: *Proc. of Conference on Cyber Security Experimentation and Test (CSET)*. 2010.

[29] Dániel István Buza et al. "CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot". In: *Proceedings of the Workshop on Smart Grid Security*. Springer. 2014, pp. 181–192.

[30] A. A. Cárdenas et al. "Attacks against process control systems: Risk assessment, detection, and response". In: *Proc. of the ACM Conference on Computer and Communications Security (CCS)*. 2011.

[31] Martin Casado et al. "SANE: a protection architecture for enterprise networks". In: *Proc. of the USENIX Security Symposium*. 2006, pp. 137–151.

[32] Defense Use Case. "Analysis of the cyber attack on the Ukrainian power grid". In: *Electricity Information Sharing and Analysis Center (E-ISAC)* (2016).

[33] John Henry Castellanos et al. "Legacy-Compliant Data Authentication for Industrial Control System Traffic". In: *Proceedings of the Conference on Applied Cryptography and Network Security (ACNS)*. 2017.

[34] Rohan Chabukswar et al. "Simulation of network attacks on SCADA systems". In: *First Workshop on Secure Control Systems* (2010).

[35] Haowen Chan, Adrian Perrig, and Dawn Song. "Random key predistribution schemes for sensor networks". In: *Proceedings of Symposium on Security and Privacy*. IEEE. 2003, pp. 197–213.

[36] Yu-Chung Cheng et al. "Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis". In: *Proc. of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. 2006. ISBN: 1-59593-308-5.

[37] Bill Cheswick. "An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied". In: *In Proceedings of the Winter USENIX Conference* (1992), pp. 163–174.

[38] Nicholas Childers et al. "Organizing large scale hacking competitions". In: *Proveedings of conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. 2010. DOI: 10.1007/978-3-642-14215-4_8.

[39] Cisco. *Cisco's Visual Networking Index Forecast Projects Nearly Half the World's Population Will Be Connected to the Internet by 2017*. https://newsroom.cisco.com/press-release-content?articleId=1197391. 2013.

[40] CISCO. *Industrial Ethernet: A Control Engineer's Guide*. www.cisco.com/web/strategy/docs/manufacturing/industrial_ethernet.pdf.

[41] David D Coleman and David A Westcott. *CWNA: Certified Wireless Network Administrator Official Study Guide: Exam CWNA-106*. Sybex, 2014.

[42] Crispin Cowan. "Defcon Capture the Flag: Defending vulnerable code from intense attack". In: *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX)*. 2003.

[43] Bob Cromwell. *The Problem With Government-Imposed Backdoors*. https://cromwell-intl.com/cybersecurity/backdoors.html, Accessed: 2019-2-4.

[44] B P Crow et al. "IEEE 802.11 Wireless Local Area Networks". In: *IEEE Communications Magazine* (1997). ISSN: 01636804. DOI: 10.1109/35.620533. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=620533.

[45] *CTFtime*. https://defcon.org/. Accessed: 2016-10-19.

[46] Miller Damien and Friedl Markus. *Chroot in OpenSSH*. http://undeadly.org/cgi?action=article&sid=20080220110039.

[47] *DEF CON Hacking Conference*. https://defcon.org/. Accessed: 2016-10-19.

[48] Arnaud Delmas. *A C implementation of the Bluetooth stream cipher E0*. https://github.com/adelmas/e0, Accessed: 2018-10-28.

[49] Android Developers. *Utilities for encoding and decoding the Base64 representation of binary data*. https://developer.android.com/reference/android/util/Base64, Accessed: 2018-01-26.

[50] Danny Dolev and Andrew Yao. "On the security of public key protocols". In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[51] L Dong, A P Petropulu Z. Han, and H V Poor. "Improving wireless physical layer security via cooperating relays". In: *IEEE Transactions on Signal Processing* (2010).

[52] Xinshu Dong et al. "Software-Defined Networking for Smart Grid Resilience: Opportunities and Challenges". In: *In Proc. of The Cyber-Physical System Security Workshop (CPSS)*. Singapore, 2015.

[53] John Dunning. "Taming the blue beast: A survey of Bluetooth based threats". In: *IEEE Security & Privacy* 8.2 (2010), pp. 20–27.

[54] Chris Eagle and John L Clark. *Capture-the-flag: Learning computer security under fire*. Tech. rep. DTIC Document, 2004.

[55] Manuel Egele et al. "An Empirical Study of Cryptographic Misuse in Android Applications". In: *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security (CCS)*. Berlin, Germany: ACM, 2013, pp. 73–84. ISBN: 978-1-4503-2477-9.

[56] *Elasticsearch: Open Source, Distributed, RESTful Search Engine*. https://github.com/elastic/elasticsearch. Accessed: 2016-10-19.

[57] Ellisys. *Ellisys protocol test solutions*. https://www.ellisys.com/, Accessed: 2018-10-28.

[58] Ettercap Project. *Ettercap*. https://ettercap.github.io/ettercap/.

[59] Nicolas Falliere, L.O. Murchu, and Eric Chien. "W32. Stuxnet Dossier". In: *Symantec Security Response* (2011). URL: http://large.stanford.edu/courses/2011/ph241/grayson2/docs/w32{_}stuxnet{_}dossier.pdf.

[60] Nick Feamster, Jennifer Rexford, and Ellen Zegura. "The Road to SDN". In: *ACM Queue* 11.12 (2013), pp. 20–40. ISSN: 15427730. DOI: 10.1145/2559899.2560327. URL: http://dl.acm.org/citation.cfm?doid=2559899.2560327.

[61] Scott Fluhrer and Stefan Lucks. "Analysis of the E0 encryption system". In: *International Workshop on Selected Areas in Cryptography*. Springer. 2001, pp. 38–48.

[62] Aurélien Francillon, Boris Danev, and Srdjan Capkun. "Relay attacks on passive keyless entry and start systems in modern cars". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Eidgenössische Technische Hochschule Zürich, Department of Computer Science. 2011.

[63] Brendan Galloway, Gerhard P Hancke, et al. "Introduction to industrial control networks." In: *IEEE Communications Surveys and Tutorials* (2013).

[64] Hamid Reza Ghaeini et al. "State-Aware Anomaly Detection for Industrial Control Systems". In: *Proceedings of Symposium on Applied Computing (SAC)*. 2018.

[65] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[66] Google. *BoringSSL is a fork of OpenSSL that is designed to meet Google's needs*. https://boringssl.googlesource.com/boringssl/, Accessed: 2018-01-26.

[67] Google. *Conscrypt is a Java Security Provider*. https://www.conscrypt.org/, Accessed: 2018-01-26.

[68] Google. *Nearby Connections: Advertise and Discover*. https://developers.google.com/nearby/connections/android/discover-devices, Accessed: 2018-07-17. 2018.

[69] Google. *Nearby Connections: Get Started*. https://developers.google.com/nearby/connections/android/get-started, Accessed: 2018-07-17. 2017.

[70] Google. *Nearby Connections: Manage Connections*. https://developers.google.com/nearby/connections/android/manage-connections, Accessed: 2018-07-17. 2018.

[71] Google. *Nearby Connections: Strategies*. https://developers.google.com/nearby/connections/strategies, Accessed: 2018-07-17. 2017.

[72] Google. *Nearby Connections: v11 update*. https://developers.google.com/nearby/connections/v11-update, Accessed: 2018-07-17. 2017.

[73] Google. *Nearby Connections: Wi-Fi Issues*. ttps://stackoverflow.com/questions/49401461, Accessed: 2018-07-17. 2018.

[74] Google. *RockPaperScissors Sample App for Nearby APIs on Android*. https://github.com/googlesamples/android-nearby/tree/master/connections/rockpaperscissors, Accessed: 2018-07-17. 2018.

[75] Google. *Samples for Nearby APIs on Android*. https://github.com/googlesamples/android-nearby/tree/master/connections, Accessed: 2018-07-17. 2018.

[76] P. K. Gopala, Lifeng Lai, and H. El Gamal. "On the Secrecy Capacity of Fading Channels". In: *IEEE Transactions on Information Theory* (2008).

[77] Glenn Greenwald. *No place to hide: Edward Snowden, the NSA, and the US surveillance state*. Metropolitan Books, 2014. ISBN: 1250062586.

[78] Julian B. Grizzard, Sven Krasser, and Henry L. Owen. "The Use of Honeynets to Increase Computer Network Security and User Awareness". In: *Journal of Security Education* 1.2-3 (2005), pp. 23–37.

[79] Guardsquare. *ProGuard: The open source optimizer for Java bytecode*. https://www.guardsquare.com/en/products/proguard, Accessed: 2018-07-17. 2018.

[80] Ramakrishna Gummadi et al. "Understanding and mitigating the impact of RF interference on 802.11 networks". In: *ACM SIGCOMM Computer Communication Review* (2007).

[81] Keijo Haataja and Pekka Toivanen. "Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures". In: *IEEE Transactions on Wireless Communications* 9.1 (2010).

[82] Nikhil Handigol et al. "Reproducible Network Experiments Using Container-based Emulation". In: *Proc. of Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. CoNEXT '12. Nice, France: ACM, 2012, pp. 253–264. ISBN: 978-1-4503-1775-7. DOI: 10.1145/2413176.2413206. URL: http://doi.acm.org/10.1145/2413176.2413206.

[83] Alfred Hero. "Secure space-time communication". In: *IEEE Transactions on Information Theory* (2003).

[84] Guido R Hiertz et al. "The IEEE 802.11 universe". In: *IEEE Communications Magazine* (2010).

[85] T Holczer, M Félegyházi, and L Buttyán. *The design and implementation of a PLC honeypot for detecting cyber attacks against industrial control systems*. https://www.crysys.hu/publications/files/HolczerFB2015CN.pdf. 2015.

[86] Yih-Chun Hu, Adrian Perrig, and David B Johnson. "Wormhole attacks in wireless networks". In: *IEEE journal on selected areas in communications* 24.2 (2006), pp. 370–380.

[87] IEEE. *IEEE 802.11 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. http://standards.ieee.org/getieee802/download/802.11-2016.pdf. 2016.

[88] IETF. *Counter with CBC-MAC (CCM)*. https://www.ietf.org/rfc/rfc3610.txt, Accessed: 2018-10-28.

[89] Dragos Inc. *TRISIS Malware: Analysis of Safety System Targeted Malware*. https://dragos.com/blog/trisis/TRISIS-01.pdf. 2017.

[90] SANS institute. *The State of Security in Control Systems Today*. https://www.sans.org/reading-room/whitepapers/analyst/state-security-control-systems-today-36042. 2015.

[91] Internet Security Research Group (ISRG). *Let's Encrypt*. https://letsencrypt.org/.

[92] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. 1st ed. Springer Publishing Company, Incorporated, 2008. ISBN: 0387717595, 9780387717593.

[93] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. "Practical invalid curve attacks on TLS-ECDH". In: *European Symposium on research in computer security*. Springer. 2015, pp. 407–425.

[94] Markus Jakobsson and Susanne Wetzel. "Security weaknesses in Bluetooth". In: *Cryptographers' Track at the RSA Conference*. Springer. 2001, pp. 176–191.

[95] Avinash Kak. *BitVector.py*. https://engineering.purdue.edu/kak/dist/BitVector-3.4.8.html, Accessed: 2018-10-28.

[96] Eunsuk Kang et al. "Model-Based Security Analysis of a Water Treatment System". In: *Proc. of Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*. 2016.

[97] Bounpadith Kannhavong et al. "A survey of routing attacks in mobile ad hoc networks". In: *IEEE Wireless communications* 14.5 (2007).

[98] Karl M Kapp. *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

[99] Chris Karlof and David Wagner. "Secure routing in wireless sensor networks: Attacks and countermeasures". In: *Proceedings of the Workshop on Sensor Network Protocols and Applications*. IEEE. 2003, pp. 113–127.

[100] John Kelsey, Bruce Schneier, and David Wagner. "Key schedule weaknesses in SAFER+". In: *The Second Advanced Encryption Standard Candidate Conference*. 1999, pp. 155–167.

[101] Paraskevas Kitsos et al. "Hardware implementation of Bluetooth security". In: *IEEE Pervasive Computing* 1 (2003), pp. 21–29.

[102] Constantinos Kolias et al. "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset". In: *IEEE Communications Surveys & Tutorials* (2016).

[103] Sandeep Kumar et al. "Breaking ciphers with COPACOBANA–a cost-optimized parallel code breaker". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2006, pp. 101–118.

[104] Perry Kundert. *Communications Protocol Python Parser and Originator*. https://github.com/pjkundert/cpppo. [Online; accessed 14-June-2015].

[105] Bob Lantz, Brandon Heller, and Nick McKeown. "A Network in a Laptop: Rapid Prototyping for Software-defined Networks". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010. ISBN: 978-1-4503-0409-2. DOI: 10.1145/1868447.1868466.

[106] S. K. Leung-Yan-Cheong and Martin E. Hellman. "The Gaussian Wire-Tap Channel". In: *IEEE Transactions on Information Theory* (1978).

[107] Michael Liljenstam et al. "RINSE: The Real-time Immersive Network Simulation Environment for network security exercises". In: *Proc. of Workshop on Principles of Advanced and Distributed Simulation (PADS)*. 2005, pp. 119–128. ISBN: 0-7695-2383-8. DOI: 10.1109/PADS.2005.23.

[108] J. Lin et al. "On false data injection attacks against distributed energy routing in smart grid". In: *Conference on Cyber-Physical Systems (ICCPS)*. 2012.

[109] Samuel Litchfield et al. *Poster: Re-thinking the Honeypot for Cyber-Physical Systems*. Poster at IEEE Symposium on Security and Privacy. 2016.

[110] Donggang Liu, Peng Ning, and Rongfang Li. "Establishing pairwise keys in distributed sensor networks". In: *ACM Transactions on Information and System Security (TISSEC)* 8.1 (2005), pp. 41–77.

[111] Eric Luiijf. "Cyber (In-) security of Industrial Control Systems: A Societal Challenge". In: *International Conference on Computer Safety, Reliability, and Security (SafeComp)*. Springer. 2015.

[112] Eric Luijif and Bert Jan te Paske. *Cyber Security of Industrial Control Systems*. TNO technical report. https://www.tno.nl/ics-security/. 2015.

[113] Musaria K Mahmood et al. "MATLAB Implementation of 128-key length SAFER+ Cipher System". In: ().

[114] Dennis Mantz. *InternalBlue*. https://github.com/seemoo-lab/internalblue, Accessed: 2018-10-30.

[115] Konstantinos Markantonakis et al. "Practical relay attack on contactless transactions by using NFC mobile phones". In: *Proceedings of Workshop on Radio Frequency Identification System Security (RFIDsec)* 12 (2012), p. 21.

[116] Stephen Martin. *Directional Gain of IEEE 802.11 MIMO Devices Employing Cyclic Delay Diversity*. 2013. URL: https://apps.fcc.gov/kdb/GetAttachment.html?id=zx796foayVA0TnNkVOgKjg%3D%3D&desc=OET%2013TR1003%20Directonal%20Gain%20of%20802%2011%20MIMO%20with%20CDD%2004%2005%202013&tracking_number=49466.

[117] James L Massey, Gurgen H Khachatrian, and Melsik K Kuregian. "Nomination of SAFER+ as candidate algorithm for the Advanced Encryption Standard (AES)". In: *NIST AES Proposal* (1998).

[118] John C Matherly. *SHODAN the computer search engine*. https://www.shodan.io. Accessed: 2016-08-01.

[119] Aditya Mathur and Nils Ole Tippenhauer. "A Water Treatment Testbed for Research and Training on ICS Security". In: *Proceedings of Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater)*. Apr. 2016.

[120] Nick Mckeown et al. "OpenFlow: enabling innovation in campus networks". In: *Computer Communication Review* 38.2 (2008), pp. 69–74. ISSN: 01464833. DOI: 10.1145/1355734.1355746.

[121] Martin Mink and Rainer Greifeneder. "Evaluation of the offensive approach in information security education". In: *Proc. of IFIP International Information Security Conference (IFIP SEC)*. 2010.

[122] Arunesh Mishra, Minho Shin, and William Arbaugh. "An empirical analysis of the IEEE 802.11 MAC layer handoff process". In: *ACM SIGCOMM Computer Communication Review* (2003).

[123] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. *This POODLE bites: exploiting the SSL 3.0 fallback*. https://www.openssl.org/~bodo/ssl-poodle.pdf, Accessed: 2019-02-04. 2014.

[124] James R. Moyne and D.M. Tilbury. "The Emergence of Industrial Control Networks for Manufacturing Control, Diagnostics, and Safety Data". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 29–47. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.887325.

[125] Amitav Mukherjee and A Lee Swindlehurst. "Robust Beamforming for Security in MIMO Wiretap Channels With Imperfect CSI". In: *IEEE Transactions on Signal Processing* (2013).

[126] Bruce Roy Munson et al. *Fluid mechanics*. Wiley Singapore, 2013.

[127] Muhammad Naveed et al. "Inside Job: Understanding and Mitigating the Threat of External Device Mis-Binding on Android." In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2014.

[128] NOXRepo.org. *The POX controller*. https://github.com/noxrepo/pox. [Online; accessed 14-June-2015].

[129] ODVA. *Ethernet/IP Technology Overview*. https://www.odva.org/Home/ODVATECHNOLOGIES/EtherNetIP.aspx. Accessed: 2016-08-01.

[130] Frédérique Oggier and Babak Hassibi. "The secrecy capacity of the MIMO wiretap channel". In: *IEEE Transactions on Information Theory* (2011).

[131] R.L.S. de Oliveira et al. "Using Mininet for emulation and prototyping Software-Defined Networks". In: *Proc. of Conference on Communications and Computing (COLCOM)*. 2014, pp. 1–6. DOI: 10.1109/ColComCon.2014.6860404.

[132] Eng Hwee Ong et al. "IEEE 802.11 ac: Enhancements for very high throughput WLANs". In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*. IEEE. 2011.

[133] Open Networking Foundation. "Software-Defined Networking: The New Norm for Networks [white paper]". In: *ONF White Paper* (2012), pp. 1–12.

[134] Michel Ossmann. *HackRF: low cost SDR platform*. https://greatscottgadgets.com/hackrf, Accessed: 2019-05-01.

[135] Ossmann, Michael. *Project Ubertooth*. https://github.com/greatscottgadgets/ubertooth/.

[136] Yin Minn Pa Pa et al. "IoTPOT: Analysing the Rise of IoT Compromises". In: *9th USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2015.

[137] John Padgette. "Guide to bluetooth security". In: *NIST Special Publication* 800 (2017), p. 121.

[138] Vern Paxson. "Bro: a system for detecting network intruders in real-time". In: *Computer Networks* (1999), pp. 2435–2463. ISSN: 13891286.

[139] Kostas P Peppas, Nikos C Sagias, and Andreas Maras. "Physical layer security for multiple-antenna systems: A unified approach". In: *IEEE Transactions on Communications* (2016).

[140] Eldad Perahia and Robert Stacey. *Next generation wireless LANs: 802.11 n and 802.11 ac*. Cambridge University Press, 2013.

[141] Adrian Perrig, John Stankovic, and David Wagner. "Security in wireless sensor networks". In: *Communications of the ACM* 47.6 (2004), pp. 53–57.

[142] Tom Phinney. *IEC 62443: INDUSTRIAL NETWORK AND SYSTEM SECURITY*. https://www.isa.org/pdfs/autowest/phinneydone/.

[143] Christina Pöpper et al. "Investigation of Signal and Message Manipulations on the Wireless Channel". In: *Proc. of the European Symposium on Research in Computer Security*. 2011.

[144] Kevin Poulsen. "Slammer worm crashed Ohio nuke plant net". In: *The Register* (2003).

[145] Vinay Uday Prabhu and Miguel RD Rodrigues. "On wireless channels with-antenna eavesdroppers: Characterization of the outage probability and-outage secrecy capacity". In: *IEEE Transactions on Information Forensics and Security* (2011).

[146] Niels Provos. "A virtual honeypot framework". In: *Proc. of the USENIX Security Symposium*. 2004.

[147] J Radcliffe. *Capture the flag for education and mentoring: A case study on the use of competitive games in computer security training*. http://www.sans.org/reading-room/whitepapers/casestudies/capture-flag-education-mentoring-33018. 2007.

[148] Ole André Ravnås. *Frida: Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers*. https://www.frida.re/, Accessed: 2018-01-26.

[149] Hamid Reza Ghaeini and Nils Ole Tippenhauer. "HAMIDS: Hierarchical Monitoring Intrusion Detection System for Industrial Control Systems". In: *Proc. of Workshop on Cyber-Physical Systems Security & Privacy (SPC-CPS)*. 2016.

[150] Jordan Robertson and Michael Riley. *The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies*. https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies, Accessed: 2018-10-30.

[151] Pieter Robyns et al. "Exploiting WPA2-enterprise Vendor Implementation Weaknesses Through Challenge Response Oracles". In: *WiSec*. ACM, 2014.

[152] Marco Rocchetto and Nils Ole Tippenhauer. "On Attacker Models and Profiles for Cyber-Physical Systems". In: *Proc. of the European Symposium on Research in Computer Security (ESORICS)*. 2016. DOI: 10.1007/2F978-3-319-45741-3_22. URL: \url{http://link.springer.com/chapter/10.1007\%2F978-3-319-45741-3_22}.

[153] Ronacher, Armin. *Flask (A Python Microframework)*. http://flask.pocoo.org/.

[154] Andrew Ruef et al. "Build It, Break It, Fix It: Contesting Secure Development". In: *Proc. of the ACM Conference on Computer and Communications Security (CCS)*. 2016. URL: http://arxiv.org/abs/1606.01881.

[155] Agostino Ruscito. *Pycomm: a collection of modules used to communicate with PLCs*. https://github.com/ruscito/pycomm. [Online; accessed 14-June-2015].

[156] Mike Ryan. "Bluetooth: With Low Energy Comes Low Security". In: *Proceedings of USENIX Workshop on Offensive Technologies (WOOT)*. Vol. 13. 2013, pp. 4–4.

[157] Floris A Schoenmakers. *Contradicting paradigms of control systems security: how fundamental differences cause conflicts*. 2013.

[158] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. "NexMon: A Cookbook for Firmware Modifications on Smartphones to Enable Monitor Mode". In: *arXiv preprint arXiv:1601.07077* (2015).

[159] Charlie Scott. *Desigining and Implementing a Honeypot for SCADA Network*. White paper published by SANS Institute Infosec Reading Room. 2014.

[160] Yaniv Shaked and Avishai Wool. "Cracking the Bluetooth PIN". In: *Proceedings of the conference on Mobile systems, applications, and services (MobiSys)*. ACM. 2005, pp. 39–50.

[161] Anmol Sheth et al. "MOJO: A distributed physical layer anomaly detection system for 802.11 WLANs". In: *Proceedings of the 4th international conference on Mobile systems, applications and services*. ACM. 2006.

[162] Bluetooth SIG. *Bluetooth Core Specification v5.0*. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043, Accessed: 2018-10-28. 2016.

[163] Jill Slay and Michael Miller. "Lessons Learned from the Maroochy Water Breach". In: *International Conference on Critical Infrastructure Protection*. Springer. 2007.

[164] Dominic Spill and Andrea Bittau. "BlueSniff: Eve Meets Alice and Bluetooth". In: *Proc. of USENIX Workshop on Offensive Technologies (WOOT)*. USENIX, 2007.

[165] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Reading, 2002, pp. 0–321. ISBN: 0321108957.

[166] Lance Spitzner. "The honeynet project: Trapping the hackers". In: *IEEE Security & Privacy* 1.2 (2003), pp. 15–23.

[167] Cliff Stoll. *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Simon and Schuster, 2005.

[168] Keith Stouffer, J Falco, and Karen Scarfone. *Guide to Industrial Control Systems (ICS) Security*. http://industryconsulting.org/pdfFiles/NISTDraft-SP800-82.pdf. Recommendations of the National Institute of Standards and Technology. 2006.

[169] Keith Stouffer et al. *Guide to Industrial Control Systems (ICS) Security (Revision 2)*. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf. Recommendations of the National Institute of Standards and Technology. 2015.

[170] V. F. Taylor et al. "Robust Smartphone App Identification via Encrypted Network Traffic Analysis". In: *IEEE Transactions on Information Forensics and Security* 13.1 (2018), pp. 63–78. ISSN: 1556-6013. DOI: 10.1109/TIFS.2017.2737970.

[171] OpenWrt Developer Team. *OpenWrt Wireless Freedom*. https://openwrt.org/.

[172] Erik Tews and Martin Beck. "Practical attacks against WEP and WPA". In: *Proceedings of the second ACM conference on Wireless network security*. ACM. 2009, pp. 79–86.

[173] The Honeynet Project. *Conpot*. http://conpot.org/.

[174] Thenewstack.io. *SDN Series*. 2015. URL: http://thenewstack.io/defining-software-defined-networking-part-1/.

[175] Nils Ole Tippenhauer et al. "On the requirements for successful GPS spoofing attacks". In: *Proceedings of the ACM conference on Computer and communications security (CCS)*. ACM. 2011, pp. 75–86.

[176] Juha T Vainio. "Bluetooth security". In: *Proceedings of Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Seminar on Internetworking: Ad Hoc Networking, Spring*. Vol. 5. 2000.

[177] B. Van Veen and K. Buckley. "Beamforming: A Versatile Approach to Spatial Filtering". In: *IEEE ASSP Magazine* (1988).

[178] Mathy Vanhoef and Frank Piessens. "Key reinstallation attacks: Forcing nonce reuse in WPA2". In: *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM. 2017, pp. 1313–1328.

[179] András Varga et al. "The OMNeT++ discrete event simulation system". In: *Proc. of the European simulation multiconference (ESM)*. sn. 2001, p. 65.

[180] Giovanni Vigna. "Teaching network security through live exercises". In: *Security education and critical infrastructures*. Springer, 2003.

[181] Joao P Vilela et al. "Wireless secrecy regions with friendly jamming". In: *IEEE Transactions on Information Forensics and Security* 6.2 (2011), pp. 256–266.

[182] E.K. Wang et al. "Security Issues and Challenges for Cyber Physical System". In: *Proc. of Conference on Cyber, Physical and Social Computing (CPSCom)*. 2010, pp. 733 –738.

[183] J. Wang, J. Lee, and T. Q. S. Quek. "Best Antenna Placement for Eavesdroppers: Distributed or Co-Located?" In: *IEEE Communications Letters* (2016).

[184] S. Weerakkody, Yilin Mo, and B. Sinopoli. "Detecting integrity attacks on control systems using robust physical watermarking". In: *Proc. of Conference on Decision and Control (CDC)*. IEEE, 2014.

[185] Joseph Werther et al. "Experiences in Cyber Security Education: The MIT Lincoln Laboratory Capture-the-flag Exercise". In: *Proc. of the Conference on Cyber Security Experimentation and Test (CSET)*. San Francisco, CA: USENIX Association, 2011.

[186] Sean Whalen. *An introduction to ARP spoofing*. machacking.net/kb/files/arpspoof.pdf. 2001.

[187] Matthias Wilhelm et al. "Short paper: reactive jamming in wireless networks: how realistic is the threat?" In: *Proceedings of the fourth ACM conference on Wireless network security*. ACM. 2011, pp. 47–52.

[188] Kyle Wilhoit. *The SCADA that didn't cry wolf*. Whitepaper. 2013.

[189] Kyle Wilhoit. *Who's really attacking your ICS equipment?* http://www.edgis-security.org/honeypot/whos-really-attacking-your-ics-devices/. Whitepaper. 2013.

[190] Ford-Long Wong and Frank Stajano. "Location privacy in Bluetooth". In: *European Workshop on Security in Ad-hoc and Sensor Networks*. Springer. 2005, pp. 176–188.

[191] A. D. Wyner. "The Wiretap Channel". In: *Bell System Technical Journal* (1975).

[192] Nan Yang et al. "Transmit antenna selection for security enhancement in MIMO wiretap channels". In: *IEEE Transactions on Communications* (2013).

[193] Wanqing You and Kai Qian. "OpenFlow Security Threat Detection and Defense Services". In: *Int. J. Advanced Networking and Applications* 2351 (2014), pp. 2347–2351.

[194] A. Zaalouk et al. "OrchSec: An orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions". In: *Proc. of Network Operations and Management Symposium (NOMS)*. 2014, pp. 1–9.

[195] Mark Zeller. "Myth or reality-does the aurora vulnerability pose a risk to my generator". In: *Protective Relay Engineers, 2011 64th Annual Conference for*. IEEE. 2011, pp. 130–136.

[196] B. Zhu, A. Joseph, and S. Sastry. "A Taxonomy of Cyber Attacks on SCADA Systems". In: *Proc. of Conference on Cyber, Physical and Social Computing*. 2011, pp. 380–388.

[197] S. Zonouz et al. "SCPSE: Security-Oriented Cyber-Physical State Estimation for Power Grid Critical Infrastructures". In: *Smart Grid, IEEE Transactions on* 3.4 (2012), pp. 1790–1799.

[198] Yulong Zou et al. "Improving physical-layer security in wireless communications using diversity techniques". In: *IEEE Network* (2015).