Securing CAN-Based Cyber-Physical Systems

By

Abdulmalik Humayed

Submitted to the Department of Electrical Engineering and Computer Science and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Committee members

Prof.Bo Luo, Chairperson

Prof.Arvin Agah

Prof.Prasad Kulkarni

Prof.Heechul Yun

Prof.Prajna Dhar

Date defended: December 4, 2018

The Dissertation Committee for Abdulmalik Humayed certifies
that this is the approved version of the following dissertation :

Securing CAN-Based Cyber-Physical Systems

Prof.Bo Luo, Chairperson

Date approved: December 4, 2018

Abstract

With the exponential growth of cyber-physical systems (CPSs), new security challenges have emerged. Various vulnerabilities, threats, attacks, and controls have been introduced for the new generation of CPS. However, there lacks a systematic review of the CPS security literature. In particular, the heterogeneity of CPS components and the diversity of CPS systems have made it difficult to study the problem with one generalized model. As the first component of this dissertation, existing research on CPS security is studied and systematized under a unified framework. Smart cars, as a CPS application, were further explored under the proposed framework and new attacks are identified and addressed.

The Control Area Network (CAN bus) is a prevalent serial communication protocol adopted in industrial CPS, especially in small and large vehicles, ships, planes, and even in drones, radar systems, and submarines. Unfortunately, the CAN bus was designed without any security considerations. We then propose and demonstrate a stealthy targeted Denial of Service (DoS) attack against CAN. Experimentation shows that the attack is effective and superior to attacks of the same category due to its stealthiness and ability to avoid detection from current countermeasures.

Two controls are proposed to defend against various spoofing and DoS attacks on CAN. The first one aims to minimize the attack using a mechanism called ID-Hopping so that CAN arbitration IDs are randomized so an attacker would not be able to target them. ID-Hopping raises the bar for attackers by randomizing the expected patterns in a CAN network. Such randomization hinders an attacker's ability to launch targeted DoS attacks. Based on the evaluation on the testbed, the randomization mechanism,

ID-Hopping, holds a promising solution for targeted DoS, and reverse engineering CAN IDs, and which CAN networks are most vulnerable. The second countermeasure is a novel CAN firewall that aims to prevent an attacker from launching a plethora of nontraditional attacks on CAN that existing solutions do not adequately address. The firewall is placed between a potential attacker's node and the rest of the CAN bus. Traffic is controlled bi-directionally between the main bus and the attacker's side so that only benign traffic can pass to the main bus. This ensures that an attacker cannot arbitrarily inject malicious traffic into the main bus. Demonstration and evaluation of the attack and firewall were conducted by a bit-level analysis, i.e., "Bit banging", of CAN's traffic. Results show that the firewall successfully prevents the stealthy targeted DoS attack, as well as, other recent attacks. To evaluate the proposed attack and firewall, a testbed was built that consisted of BeagleBone Black and STM32 Nucleo-144 microcontrollers to simulate real CAN traffic.

Finally, a design of an Intrusion Detection System (IDS) was proposed to complement the firewall. It utilized the proposed firewall to add situational awareness capabilities to the bus's security posture and detect and react to attacks that might bypass the firewall based on certain rules.

Acknowledgments

I would like to thank all of those who have given me directions, advice, or encouragement. I begin by thanking my adviser and mentor Prof.Bo Luo, who have been patient and wise throughout the long PhD journey. Dr.Luo have pushed me to my limits and made me reach goals that I had not thought were possible.

I would also like to thank the committee members who have given me advice and support, without which I would not have been able to realize success. Thank you Prof.Arvin Agah, Prof.Prasad Kulkarni, Prof.Heechul Yun, and Prof.Prajna Dhar.

Thank you my lab mates who have made the journey more enjoyable. Thank you Dr.Lei Yang, Dr.Yuhao Yang, Dr.Meeyoung Park, Hao Xue, Qiaozhi Wang, Chris Seasholtz, and Sana Awan. A special thanks to Saad Adnan for the wonderful help with building the circuits that I used for various experiments. Also thanks for my KU friends who I have greatly enjoyed our scholarly conversations. Thank you Dr.Mohammed Alenazi, Dr.Ali Abushaiba, and Farhad Mahmood.

Huge thanks to my sponsor Jazan University, Saudi Arabia, for their support throughout the PhD journey. In addition, I would like to thank the Saudi Cultural Mission for their support.

I am grateful for people who I met outside KU in conferences and workshops across the US. This includes Dr.Yasser Shoukry at Maryland University, Rob Deker and Karl Heimer from the CyberAuto Challenge, and Robert Leale and Craig Smith from the Car Hacking Village. Also thanks to David Crocket and Eric Evenchick for their valuable email exchanges.

Special thanks to my mentors and colleagues at Grimm Co who I spent one of the most interesting and enjoyable summers. Thank you for the Critical Infrastructure's team: Matt Carpenter, Tim Brom, Aaron Cornelius, and Mitchell Johnson for the incredible advice and skills you have taught me.

My acknowledgment would not be complete without thanking my wonderful family for their infinite support and love. Thank you my parents and siblings for believing in me and supporting me all the way.

Saving the best for last, I would like to thank my wife, Maram, for her incredible support and

patience during my PhD study. Without your support and belief in me, I do not think I could have made it this far. Finally, thank for my three-months-old baby boy, Khalid, who has been a caffeine-free productivity tool. He made me have more appreciation for free time and thus increased my productivity immensely in the last three months.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, exponential growth has been witnessed in the development and deployment of various types of Cyber-Physical Systems (CPS). They have impacted almost all aspects of our daily life; for instance, in electrical power, oil and natural gas distribution, transportation systems, health-care devices, household appliances, and many more. Many of the newly proposed CPS systems are critical for national infrastructure, life support devices, or essential to our daily life. Therefore, they are expected to be free of vulnerabilities and immune to all types of attacks, which, unfortunately, is practically impossible for all real-world systems.

The problem in CPS is the heterogeneity of the building components. CPS are composed of various components. There is different hardware such as sensors, actuators, and embedded systems. There is also a different collection of software products for control and monitoring. The reason this heterogeneity is considered a security problem is that every component can be a contributing factor of a CPS attack. Understanding the current CPS security vulnerabilities, attacks and protection mechanisms will provide us with a better understanding of the security posture of CPS. Consequently, we should be able to point out the limitations of CPS that make them subject to different attacks and devise approaches to defend against security attacks.

In this dissertation, we introduced CPS, and how they are different from both old control systems and IT systems. Recognizing the difference is key to understanding CPS security problems.

We surveyed four CPS applications: Industrial Control Systems (ICS), Smart Grids, implantable and wearable medical devices, and smart cars. We then investigated the security posture in each application and identified the causes for such problems. Then we propose a cyber-physical framework that highlights the correlations among threats, vulnerabilities, attacks, and controls. In addition, we apply the framework to smart cars and show how different attacks evolved by threats that exploited existing vulnerabilities and what countermeasures could be used to prevent them.

After surveying security in CPS, generally, and smart cars, specifically, we recognized the security vulnerabilities in Controller Area Network (CAN), the popular serial communication protocol used in a plethora of applications ranging from small and large vehicles, to ships and planes, and even in artificial limbs, drones, radar systems, and submarines. CAN was developed in 1986 for automobiles by Robert Bosch to replace the complex point-to-point wiring used in cars at the time. CAN is flexible, efficient, robust, and cost-effective, however, security was not considered at the time it was developed. Therefore, many security attacks have been exploited [32, 92, 38].

In this dissertation, we propose a stealthy targeted denial of service (DoS) attack against CAN. The attack prevents CAN nodes from normal operations in a stealthy and selective manner, in the sense that it cannot be detected with the current solutions because of its seemingly-legitimate behavior. Targeted so that the attack only affects the intended targets while the unintended work normally. This attack is distinguished from proposed attacks due to its stealthiness. Other attacks result in shutting down a CAN node or inducing errors that make the attack stand out. We evaluated the attack by simulating CAN bus traffic using BeableBone Black microcontrollers to simulate CAN node, and a STM32 Nucleo-144 development board as the attacker. The evaluation shows that the attack successfully achieved its goal by preventing CAN nodes selectively and stealthily from a successful transmission. The impact of such an attack could be crucial. CAN is used in safety-critical applications and such an attack could prevent some functions that are life-saving.

To address this attack, among many recent non-traditional attacks, we propose three simultaneous solutions to improve security in CAN using smart cars as a means of evaluation: ID-Hopping, a CAN Transceiver-based firewall and a firewall-complementing Intrusion Detection System (IDS).

For the ID-Hopping, we propose a memory-efficient and cost-effective mechanism to prevent targeted DoS attacks on CAN. The idea is to alter the network's behavior when an attacker performs a DoS attack against a particular CAN node. The change in behavior results in thwarting the attack's danger.

In addition, a firewall that is based on a CAN transceiver is proposed to prevent attacks that are not addressed by current countermeasures. Its distinguishing feature is that it monitors and is able to prevent traffic from one or more potentially-malicious ECUs one bit at time, instead of one frame at a time commonly used in conventional firewalls. Using the same evaluation platform, we added another STM32 Nucleo-144 to implement the firewall's functionality. The evaluation shows the firewall effectiveness against our proposed attack in addition to other recent attacks.

Finally, we propose a novel IDS design that complements the proposed firewall. It aims to detect attacks that the firewall might fall short of preventing due to their seemingly-legitimate nature. Certain attack patterns are recognized by the IDS which then notifies the firewall to prevent them. To the best of our knowledge, the proposed IDS is the first one that detects attacks by bit banging, unlike traditional IDS that perform analyses at the frame level.

1.1 Motivation

We chose smart cars as a representative CPS application. Smart cars have received a tremendous amount of attention recently in academia and the media due to their importance and relevance to our lives and safety [33, 92, 64]. Smart cars are now connected more than ever before, and although connectivity has increased in smart cars, they still heavily rely on legacy protocols, such as CAN, and resource-limited embedded systems. In addition, despite increasingly added safety and comfort features, such as *Adaptive Cruise Control* (ACC), *Lane Keep Assist*, *Collision Prevention*, and *Comfort Park Assist*, smart cars have become vulnerable to remote attacks that could potentially exploit such "safety" and "comfort" features for malicious and potentially safety-critical attacks.

Numerous papers and reports have shown how vulnerable smart cars are to remote and local

attacks that range from privacy invasion to life-threatening cyber-physical attacks [33, 92, 30, 32, 31]. Most of the proposed solutions focus on cryptographic mechanisms [176, 51, 171, 139], Intrusion Detection Systems (IDS) [96, 127, 154], and redesigning automotive embedded systems and networks [157]. Applying cryptography could be cost-inefficient in such resource-constrained systems. Although proposed IDS mechanism detect attacks, they do not prevent them. Also, most of IDS are anomaly-based that aim to detect attacks based on the frequency of messages. We believe more parameters must be included to detect more subtle attacks. Finally, redesigning the automotive systems, or the introduction of solutions that require redesign, is a very costly approach for manufacturers who already face cost and competition challenges.

1.2 Thesis Statement

CPS are widely ubiquitous and their impact on human lives is increasing. Although aimed at improving people's safety and convenience, CPS could be leveraged to perform malicious activities that would threaten safety and well-being. Taking smart cars as one application of CPS, we examine the current cars' security from a cyber-physical perspective. This would reveal unexpected safety-related attacks. From such revelations, we plan to draw conclusions on approaches to prevent cyber-physical attacks. Therefore, the thesis statement is as follows:

Providing a unified security framework for CPS facilitates understanding security issues, and therefore addressing them. In addition, due to the criticality of CPS applications, the detection and prevention of denial of service (DoS) attacks can greatly improve the overall security posture in such safety-critical applications.

1.3 Proposed Solution

One of the most likely attacks on CAN networks is DoS. An attacker could simply flood the network with a frame that has the highest priority over all other frames. Although devastating, such an attack is easily detected but difficult to prevent. However, a subtle DoS variant is where an

attacker targets a specific CAN node every time it sends a frame. This is considered a targeted DoS attack. We propose ID-Hopping mechanism to prevent such an attack by making all network's participating computers change their frames IDs in a way that is only understood by legitimate parties. Therefore, an attacker is left confused because the targeted computer does not send the same ID anymore, and thus does not trigger an attack to launch the targeted DoS attack.

Firewalls are commonly used for allowing and denying network traffic based on certain rules. In an automotive context, firewalls do not have the luxury they have in IT systems such as IP addresses and port numbers. Instead, they have various kinds of parameters due to the use of different protocols. We propose an automotive firewall that allows and prevents frames based on rules that take situational-awareness into consideration. For example, the firewall should prevent an attacker from injecting arbitrarily when a CAN node is transmitting. No more than one CAN node is supposed to transmit simultanously with other nodes except during the arbitration phase.

Finally, to complement the work of the firewall, we also propose an IDS that detects attacks that could bypass the firewall. Once an attack is detected, the IDS notifies the firewall which then blocks transmission of the attacking ECU. To evaluate the above mechanisms, we have built a hardware-based simulation environment that consists of four BeagleBone Black (BBB) microcontrollers to simulate regular CAN nodes and two STM32 Nucleo-144 development boards to simulate an attacker's behavior and implement the firewall's functionalities.

1.4 Contributions

In this work, our contributions are as follows:

1. Survey security aspects in four common CPS application: ICS, Smart Grid, medical devices, and smart cars.

2. Present potential threat sources to CPS and their motivations.

3. Identify the root causes of security problems in the four CPS with actual examples.

4. Present a taxonomy that highlights the cyber-physical aspects of security in CPS applications, called "Cyber-Physical Security Framework for CPS."

5. Design and implement an ID hopping algorithm to prevent targeted DoS attacks.

6. Design and implement an automotive a CAN Transceiver-based firewall.

7. Design an Intrusion Detection System to complement the firewall.

8. Evaluate the effectiveness and performance of the proposed ID-Hopping, firewall, and IDS by a hardware-based simulation environment.

1.5 List of Related Publications

1. **Abdulmalik Humayed** and Bo Luo. CAN Transceiver-Based Firewall to Defend Against DoS on CAN. In submission.

2. **Abdulmalik Humayed** and Bo Luo. The Stealthy Targeted Arbitration Denial Attack on CAN. In submission.

3. **Abdulmalik Humayed**, Jingqiang Lin, Fengjun Li, and Bo Luo. Cyber-Physical Systems Security – A Survey. In IEEE Internet of Things Journal - Special Issue on Security and Privacy in Cyber-Physical Systems, Volume: 4 Issue: 6, 2017.

4. **Abdulmalik Humayed**, and Bo Luo. Using ID-Hopping to Defend Against Targeted DoS on CAN. In International Workshop on Safe Control of Connected and Autonomous Vehicles (SCAV), in conjunction with CPS Week 2017, Pittsburgh, PA, 2017.

5. **Abdulmalik Humayed**, and Bo Luo. Poster Abstract: Cyber-Physical Security for Smart Cars – Taxonomy of Vulnerabilities, Threats, and Attacks. In 6th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), Seattle, WA, 2015 (CPS Week Best Poster Finalist).

6. **Abdulmalik Humayed**, and Bo Luo. Cyber-Physical Security for Smart Cars – Issues, Survey and Challenges. In 2nd International IFIP Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), Seattle, WA, 2015.

Chapter 2

Background

2.1 Cyber-Physical Systems

CPS are systems used to monitor and control the physical world. They are perceived as the new generation of embedded control systems such that CPS are networked embedded systems [1]. In addition, systems, where sensor and actuator networks are embedded, are also considered CPS [26]. Because of the reliance on IT systems, CPS could be defined as IT systems that are integrated into applications in the physical world [63]. This integration is a result of the advancements in the information and communication technologies (ICT) to enhance interactions with physical processes. All of these definitions highlight the heavy presence of the interactions between the cyber and the physical worlds.

2.2 CPS Applications

An increasing dependence on CPS has been growing in various applications such as energy, transportation, military, health-care, and manufacturing. CPS can be called different names, depending on the application using them. For example, a widely used application of CPS is the Supervisory Control and Data Acquisition (SCADA) which is used in Critical Infrastructure (CI) such as the Smart Grid and Industrial Control Systems (ICS). Other examples have emerged in medical de-

vices such as wearable and implantable medical devices. In addition, a network of small control systems are embedded in modern cars to improve fuel efficiency, safety, and convenience.

2.2.1 ICS

ICS refers to the control systems used to enhance the control, monitoring, and production in different industries such as the electrical grid, nuclear plants, water and sewage systems, and irrigation systems. Sometimes ICS is called Supervisory Control and Data Acquisition (SCADA) or Distributed Control Systems (DCS). For convenience, we will use the term ICS to refer to either of them. In ICS, different controllers with different capabilities communicate to achieve numerous expected goals. One of which is the Programmable Logic Controller (PLC), which is a microprocessor that is designed to operate continuously in hostile environments so that human operators find difficulty being present in [97]. This field device is connected to the physical world through sensors and actuators. Usually, it is equipped with wireless and wired communications that satisfy different configurations in different environments. It can also be connected to PC systems in a control center that controls and monitors its operations.

2.2.2 Smart Grid

The Smart Grid is the next generation of power grid that has been used for decades for electricity generation, transmission, and distribution. The smart grid provides several benefits and advanced functionalities. At the national level, it provides enhanced emission control and energy savings. Whereas at the local level, it provides home consumers better control over their energy use that would be beneficial economically and environmentally [113]. The smart grid is comprised of two major components: *power application* and *supporting infrastructure* [161]. The power application is where the core functions of the smart grid are provided, that are electricity generation, transmission, and distribution. Whereas the supporting infrastructure is the intelligent component that is mainly concerned with controlling and monitoring the core operations of the smart grid with a set of software, hardware, and communication networks.

2.2.3 Medical Devices

Medical devices have been improved by integrating cyber and physical capabilities to deliver better health care services. We are more interested in medical devices with cyber capabilities that have physical impact on patients. Such devices are either implanted inside the patient's body, called Implantable Medical Devices (IMDs), or worn by patients, called wearable devices. They are usually equipped with wireless capabilities to allow communication with other devices such as the *programmer*, which is needed for updating and reconfiguring the devices. Wearable devices communicate with each other or with other monitoring or control devices, such as a remote physician's system or a smartphone [151].

2.2.4 Smart Cars

Smart cars (intelligent cars) are cars that are more environment-friendly, fuel-efficient, safe, and have enhanced entertainment and convenience features. These advancements are made possible by the reliance on a range of 50 to 70 computers networked together, called Electronic Control Units (ECUs). ECUs are responsible for monitoring and controlling various functions such as engine emission control, brake control, entertainment (radio, multimedia players) and comfort features (cruise control and windows opening and closing).

2.3 CPS Communications

Communication technologies vary in CPS applications. Different application use different protocols (open and proprietary) and technologies (wired and wireless). Here we give an overview of the most common communication technologies and protocols in each of the four applications.

2.3.1 ICS

Two types of communication protocols are deployed in ICS, one is used for the automation and control such as Modbus, Distributed Network Protocol (DNP3), and the other is for interconnecting ICS control centers, such as Inter-Control Center Protocol (ICCP) [8]. Those protocols are used in addition to the general-purpose protocols such as TCP/IP.

2.3.2 Smart Grid

The networks are of two types: field device communications within substations using Modbus and DNP3, and recently the more advanced protocol, IEC 61850. The other type is control center communications, which rely on the Internet-Control Center Communication Protocol (ICCP).

In addition, smart meters and field devices use wireless communications to send measurements and receive commands from control centers. Smart meters, for example, use short-range frequency signals, e.g. Zigbee, for diagnostics operations by technicians or readings by digital smart readers.

2.3.3 Medical Devices

It is a necessary requirement that IMDs are updated and reconfigured wirelessly so no surgical device extraction is no needed on the patient's body. Therefore, wireless communication is the most common method of communication in medical devices. Both types rely on different communication protocols and technologies. For example, IMDs use LF signals specified by FCC, called Medical Implant Communication Service (MICS), that make it possible for IMDs and their programmers to communicate. On the other hand, wearable devices rely on another type of wireless communications, that is Body Area Network (BAN). BAN utilizes a number of wireless communication technologies such as Bluetooth and IEEE 802.15.4 (ZigBee) [35].

2.3.4 Smart Cars

All ECUs are connected to a bus network, and the network is divided into several bus subnetworks, each of which consists of a number of interconnected ECUs that perform certain functionalities. An example of such subnetworks are CAN (Controller Area Network), MOST (Media Oriented Systems Transport), LIN (Local Interconnect Network), and FlexRay. Depending on the nature of the tasks expected from each ECU, it is attached to the appropriate subnetwork. These different subnetworks can intercommunicate through gateways.

2.4 CPS Perspective

Fig. 2.1 shows a high-level view of any CPS, where it mainly consists of three parts: communication, computation and control, and monitoring and manipulation. The communication is of two types, wireless or wired, and it could connect CPS with higher-level systems, such as control centers, or with lower-level components in the physical world. The computation and control part is where the intelligence is embedded in the CPS, control commands are sent, and sensed measures are received. The monitoring and manipulation part connects CPS to the physical world through sensors, to monitor physical components, and actuators, to manipulate them.

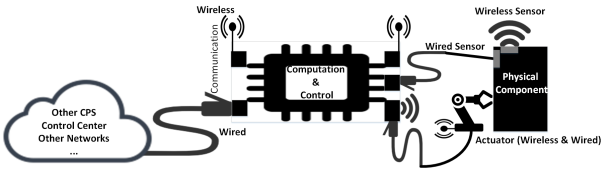


Figure 2.1: CPS abstract model

A CPS component might have the ability to communicate with control centers or other CPS components. This same component could also contain a sensor, an actuator, or both to connect to the physical world. Each one of these capabilities have different potential security implications that may result from the close interactions of the component's parts and their capabilities. For example, a CPS component's communicational and computational parts are not expected to affect

the physical world, and yet might be exploited to cause unexpected attacks with physical consequences. Similarly, the physical properties of this component, in addition to the physical properties of the object of interest in the physical world that the CPS controls and monitors, can also cause unexpected attacks that might result in non-physical attacks, such as misleading information sent to a CPS.

This heterogeneity of CPS among components, or within a component itself, results in a lack of understanding of new types of security threats that would exploit such heterogeneity. This urges CPS engineers to incorporate security-aware paradigms that consider the cyber-physical nature of potential attacks. The need to clearly distinguish between such aspects for security analysis and engineering arises. Thus, we view any CPS from three aspects: *cyber*, *physical*, and *cyber-physical*. The cyber aspect considers data computations, communications, and interactions that do not affect the physical world, whereas the cyber-physical aspect consider them when direct interactions with the physical world takes place. The cyber-physical aspect is where the cyber and physical world can connect. In addition, the physical aspect includes any security-related properties of the physical world, a CPS, or its components.

In Fig 2.2, we incorporated this CPS view in the abstract model shown in Fig 2.1. Referring to the Fig 2.2, (1) are the aspects that we consider cyber, whereas (2) are the cyber-physical aspects. Note the dashed line separating (1) and (2) shows how the same component can be considered cyber and cyber-physical at the same time depending on the presence or its the absence of the interaction with the physical world. (4) shows that the physical properties of any part of a CPS system could play a role in security issues. Therefore, we need to include it in the physical aspect.

In the following paragraphs, we present how our abstract model can capture the CPS aspects in the following applications: ICS, the Smart Grid, implantable and wearable medical devices, and smart cars. In each figure, we annotate the CPS aspects: (1)cyber, (2)cyber-physical, and (3)physical. This annotation distinguishes between the aspects in such a way that we can relate to it in the analysis of security issues.

ICS. Fig. 2.3 depicts the CPS aspects in a PLC scenario, where it is used for controlling the

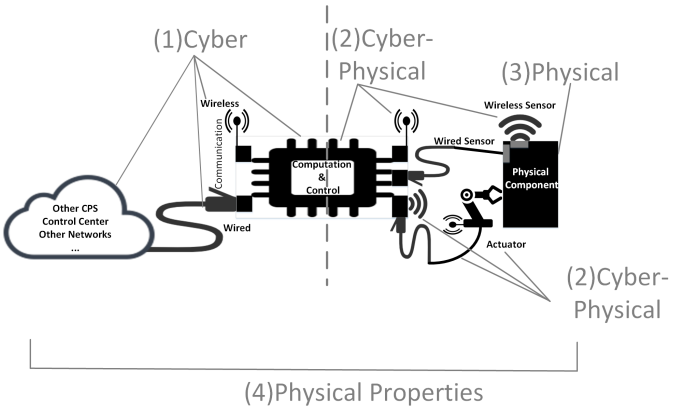


Figure 2.2: CPS Aspects

temperature in a chemical plant. The goal is to maintain the temperature within a certain range. If the temperature exceeds a specified threshold, the PLC is notified via a wireless sensor attached to the tank, which in turn, notifies the control center of the undesired temperature change. Alternatively, in closed-loop settings, the PLC could turn the cooling system on to reduce that tank's temperature within the desired range.

In this figure, the cyber aspects (1) are the interactions with the PLC such that there is no direct interaction with physical components, such as cooling fans or the tank. This involves any laptops that require direct access to connect directly to the PLC, communications with higher-level environments such as the control center and other remote entities, and the PLC's wireless interface that could be based on long or short-range frequencies. In addition, cyber-physical aspects (2) are those that connect cyber and physical aspects. The PLC, the actuator, and the sensor, are all cyber-physical aspects due to their direct interactions with the physical world. The wireless capabilities of the actuator and the sensor are also considered cyber-physical. Finally, the physical aspects are the physical objects that need monitoring and control, i.e. the cooling fans and the tank's temperature.

Smart Grid. Fig. 2.4 shows a typical scenario in the smart grid, more particularly, highlighting the interactions of the smart meters. A smart meter is attached to every house to provide utility companies with more accurate electricity consumption and customers with convenient ac-

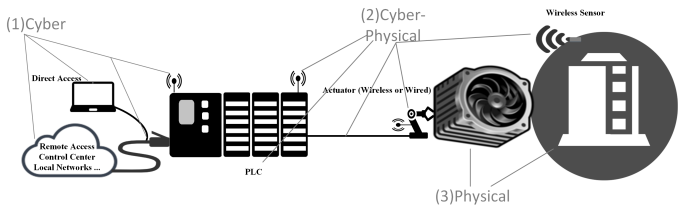


Figure 2.3: CPS aspects in ICS

cess to their usage. A smart meter interfaces a house's appliances and Home Energy Management Systems (HEMS) on the one hand, and interfaces with data collectors on the other. Wireless communications are the most common means to communicate with collectors, although wired communications, such as Power Line Communications (PLC), are also available. The meter is equipped with a diagnostics port that relies on short-range wireless interface for convenient access by digital meter readers and diagnostics tools [91]. The smart meter sends the measurements to a collector that aggregates all meters' data in a designated neighborhood. The collector sends the aggregated data to the distribution control center managed by the utility company. In particular, to the AMI headend server that stores the meters' data and shares the stored data with the Meter data management system (MDMS) that manages the data with other systems such as demand response systems, historians, and billing systems. The headend can connect/disconnect services by remotely sending commands to the meters. This feature is a double-edged sword such that it is very efficient way to control services, yet it could be exploited to launch large-scale blackouts by remotely controlling a large number of smart meters.

In Fig. 2.4, we highlight the CPS aspects in the involved components that have some interactions with the smart meters. Cyber aspects (1) are in the control center such that it stores smart meters' data, shares it, analyzes it, and makes informed decisions based on that. The control center can also have a cyber-physical aspect (2) when connect/disconnect commands are sent by the AMI headend to smart meters. In addition, the cyber-physical aspect (2) is also apparent in the smart meter itself due to its ability to perform cyber operations, such as sending measurements to utility, and physical operations, such as connecting/disconnecting electricity services. Other field devices in the generation, transmission automation, and distribution plants have a high presence

of the cyber-physical aspect due to their close interactions with physical aspects of the smart grid. We can consider those field devices as ICS devices due to the very similar environments to ICS applications. Smart meters are the distinguishing component that differentiate between ICS and the Smart Grid. Home appliances that are connected with smart meters are considered cyber-physical because of the their direct interaction with smart meters. A utility company can use smart meters to control the amount of energy consumed by particular home appliances when needed [133], which is a cyber-physical action (2).

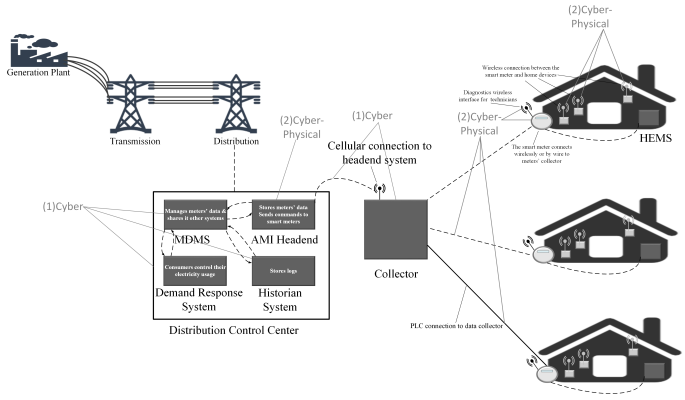


Figure 2.4: CPS aspects in the Smart Grid

Medical Devices. Fig. 2.5 demonstrates two of the most popular IMDs, the *insulin pump* and the *implantable cardioverter defibrillator* (ICD). The insulin pump is used to automatically or manually inject insulin injections for diabetics when needed, whereas the ICS is used to detect rapid heartbeat and response by delivering an electric shock to maintain a normal heartbeat [66]. The insulin pump usually needs another device, called the *continuous glucose monitor* (CGM), to receive blood sugar measurements. Both devices, the insulin pump and the CGM, require small syringes to be injected into a patient's body. The insulin pump receives measurements of glucose levels from the CGM. Based on the measurements, the pump decides whether the patient needs an insulin dose or not. The CGM sends the measurements through wireless signals to the insulin pump or other devices, such as a remote control or computer. In addition, some insulin pumps can be commanded by a remote control held by a patient or physician.

In this figure, the cyber (1) aspects are embodied in the monitoring computers in the hospital

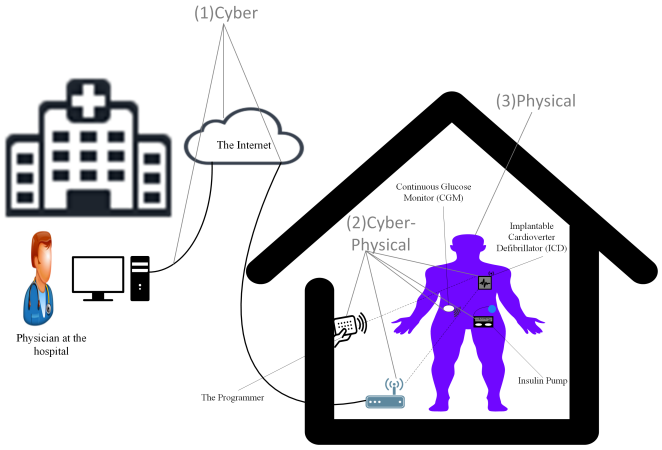


Figure 2.5: CPS aspects in medical devices

and the communications to the Internet. The cyber-physical aspects (2), on the other hand, are those devices that directly interact with the devices implanted in a patient. A patient is the physical (3) aspect in the context of medical devices. An IMD connects to the hospital by sending measurements through an in-home router. In order to reconfigure an ICD, a physical proximity is required to be able to do so using a device called the *programmer*.

Smart Cars. Fig. 2.6 shows the typical architecture of an in-car network. Depending on the nature of the tasks expected from each ECU, it is attached to the appropriate subnetwork. ECUs from different subnetworks can intercommunicate through gateways. In this paper we mainly focus on CAN bus for two reasons: 1) most security issues result from CAN network and 2) it has been required to be deployed in all cars in the U.S. since 2008 [92], thus it is in almost every car around us.

In Fig. 2.6, we annotated ECUs that do not have any interactions with physical components of the cars as cyber (1). Examples of which include the Telematics Control Unit (TCU) and the media player. The TCU has more than a wireless interface that allows advanced capabilities such as remote software updates by car manufacturers, phones pairing, receiving and making calls free of hands. The cyber-physical annotations are for ECUs that can legitimately interact with physical components and affect them, such as the BCM and RKE system. The RKE, for example, receives signals to make a physical impact on the car by locking/unlocking doors. Finally, physical compo-

nents such as the engine or the tires are physical (3).

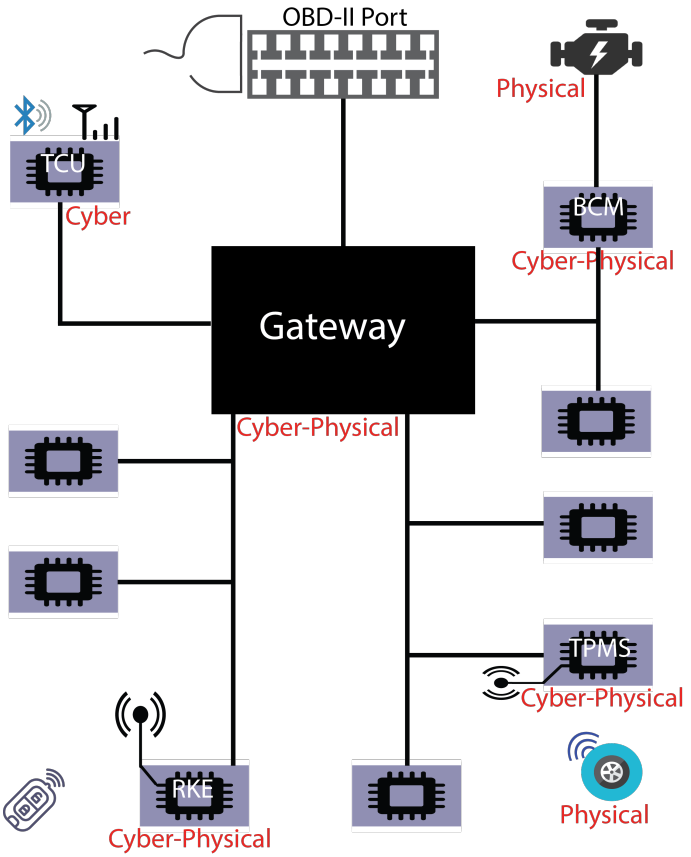


Figure 2.6: CPS aspects in smart cars

2.5 Security in CPS

Security is usually associated with mechanisms such as cryptography, access control, intrusion detection, and many other IT security solutions. Those mechanisms are very important in securing our information and communication technology's infrastructure. However, they are inadequate when solely applied to CPS. Different reported attacks on CPS applications show the inadequacy of the sole dependence on these mechanisms. Therefore, IT security solutions should complement security mechanisms that take the interactions between the physical and cyber aspects into consideration. In addition, prevention techniques do not address CPS resiliency when it is under an ongoing attack [24].

Security in ICS and Smart Grid. The lack or weakness of security in CPS could be catastrophic depending on the application. For example, if the security of CPS used in a nuclear plant has been compromised, a world-wide threat is the possible consequence. Furthermore, security violations in the Smart Grid could lead to the loss of services to the consumer and financial losses to the utility company. Because of the CPS's pervasiveness and its wide use in the critical infrastructure in particular, CPS security is of a critical importance. In fact, it is even suggested that ICS is not yet ready to be connected to the Internet [58]. This is due to the insecure communication networks and the vulnerabilities in the deployed control systems.

Security in Medical Devices. Security in medical devices is challenging for a number of reasons. Firstly, medical devices are resource-constrained in terms of memory, computation, communication, and energy. Therefore, adding security mechanisms, that consume such limited resources, to the current devices might be difficult. In addition, replacing devices that have been already implanted into patients' bodies requires surgery, which might carry potential health risks. Furthermore, security could be an obstacle in some emergency situations. For example, a patient might be in a situation that needs immediate intervention by a different health care provider who lacks access to the IMD. The unavailability of the IMD could be very dangerous [67].

Because of the different circumstances surrounding medical devices, the need for defining appropriate security goals arises. Halperin et al. [67] initiated the discussion of the security goals in medical devices by extending the standard security goals, confidentiality, integrity, and availability. Security goals include the authorized entities should be able to access accurate data, identify and configure devices, update software, and maintain the device's availability; whereas privacy goals include the protection of private information about a device's existence, type, unique ID, and patient's identification.

Security in Cars. Car manufacturers strive to come up with a variety of innovative technologies that would satisfy their customers by providing more functionalities and comfort. Typically cars are safe by design, but security, however, is not usually of great concern in the design phase. Safety ensures the car's ability to function during non-malicious incidents. Security, on the other hand,

has not been a design issue, but has been an add-on feature. The new features in cars require wireless communications and components with physical impacts. These two features alone result in most security vulnerabilities and attacks in smart cars.

Chapter 3

Security in CPS

In this chapter, we then survey the literature on CPS privacy and security under a unified framework, which consists of three orthogonal coordinates, as shown in Fig 3.1. First, from security perspective, we follow the well-known taxonomy of threats (Section. 3.1), vulnerabilities (Section. 3.2), attacks (Section. 3.3), and controls (Section. 3.4). Next, we discuss each main aspect following the CPS components perspective: cyber, physical, and cyber-physical. For instance, when we survey the attacks, we categorize them into cyber-attacks, physical-attacks, and cyber-physical-attacks. Last, from the CPS systems perspective, we explore general CPS features as well as representative systems, in particular, industrial control systems (ICSs), smart grids, medical CPS, and smart cars. Finally, we summarize the key threats, vulnerabilities, attacks and controls in each CPS aspect for each representative CPS system. Since this dissertation's focus is on smart cars, as a CPS exemplary application, in Chapter. 4, we will introduce smart cars' security in a dedicated chapter, Chapter. 4, so we can elaborate on the automotive security.

3.1 Threats

Securing CPS bears with it various challenges, one of which is understanding the potential threats [25]. We aim to tackle this challenge by identifying CPS potential threats and shedding light on them from different angles. First we discuss the general threats that almost any CPS application could

be vulnerable to. Then we dive into the specific threats to CPS applications.

Traditionally, in order for a system to be secure, it satisfies the three security requirements: confidentiality, integrity and availability. Due to the different nature of CPS and their direct interaction with the physical world, including human and critical environments, safety requirements are crucially important. Here we discuss the threats to both security and safety requirements. We follow the traditional approach in security research by providing a threat model that describes the threats based on a number of factors. Such an approach provides a systematic description of the threats in a way that would be useful for security procedures such as threat modeling and risk assessment.

3.1.1 Threat Model

The knowledge of who/what we protect a CPS against is equally important to the knowledge of the existing vulnerabilities and attack mechanisms. We first need to define what we mean by a *threat*. A security *threat* is defined as "a set of circumstances that has the potential to cause loss or harm" [143]. The potential aspect is key in this context, as we discuss potential threats that may not necessarily have occurred, but might. The loss might be in safety measures, confidentiality, integrity, or availability of resources, whereas the harm implies harming people, the environment, or systems. Note that due to the pervasiveness of the CPS applications, people are increasingly becoming a critical asset to protect, in addition to the other assets that are common in security literature.

We identify five factors about every threat: *source*, *target*, *motive*, *attack vector*, and *potential consequences*. Then we elaborate on each one showing the possible types that would be applicable to each factor.

Source. The source of a threat is the initiator of an attack. Threat sources fall into three types: (1) *adversarial threats* which pose malicious intentions. This type could be an individual, group (criminal groups, terrorists, and hackers), organization (industrial spies, possible competitors and customers), state-level groups (hostile nations); (2) *accidental threats* are threats

that have been caused accidentally or through legitimate CPS components; (3) *environmental threats* which include natural disasters (floods, earthquakes, and tornadoes), human-caused disasters (fires, explosions), and failures of supporting infrastructure (power outage or telecommunications loss) [134, 150, 168, 163, 87, 29, 162].

Target. Targets are CPS applications and their components or users. We will see specific examples for each application.

Motive. CPS attackers usually have one or more reasons to launch an attack: criminal, spying, terroristic, political, or cyberwar [156, 168].

Attack Vector. A threat might perform one type or more of four mechanisms for a successful attack: interception, interruption, modification or fabrication [143]

Consequence. Compromising the CPS's confidentiality, integrity, availability, privacy, or safety.

We investigate the potential threats to the four CPS applications using the proposed threat model. Then we highlight the threats that are specific to each application, with respect to the five factors: source, target, motive, vector, and consequence.

This paradigm of threat analysis provides insight to potential threats from different angles. given the complexity and heterogeneity of CPS, the threats have multi-faceted properties that traditional approaches, such as listing, cannot express. Our approach only highlights the most relevant and feasible threats.

3.1.2 ICS Threats

Criminal Attackers (motive). An attacker whose familiar with the system (source) could exploit wireless capabilities (vector) to remotely control an ICS application and possibly disrupt its operations (consequence).

Financially-motivated customers(motive). A capable customer (source) aiming to reduce a utility bill might be able to tamper with physical equipment or inject false data (vector) to misinform the utility (target) causing it to lose financially (consequence) [166].

Politically-motivated espionage (motive). Intelligence agencies (source) might perform recon-

naissance operations targeting a nation's critical infrastructure (target) possibly through spreading malware (vector) resulting in confidentiality violations of the critical data (consequence) [115, 125].

Politically-motivated cyberwar (motive). A hostile nation (source) could initiate a cyberwar against another nation (target) by remotely attacking its critical infrastructure, e.g. nuclear and chemical plants, and gas pipelines, by spreading malware or accessing field devices (vector) resulting in a plant's shutdown, sabotaging components, environmental pollution (consequence) [152, 165, 21, 87, 93].

In addition, the weak physical security around remote sensors and actuators can be exploited to affect the ICS desired operations.

Physical threats. An attacker (source) could spoof a sensor that measures the temperature of a particular environment (target) by applying heat or cold to it (vector) resulting in sending misleading false measurements to the control center (consequence).

3.1.3 Smart Grid Threats

Financially-motivated threats (motive). A customer (source) who wants to trick a utility company's billing system (target) might tamper with smart meters (vector) to reduce the electricity bill (consequence) [148, 114, 113, 121, 11]. Another example of this type of threat is when utility companies (source) might be interested in gathering customers' private information (target) by analyzing their electricity usage to infer habits and types of house appliances (vector) to sell such information for advertisement purposes resulting in privacy violation (consequence) [113, 145, 42, 161]. In addition, there is a possible scenario where criminals (source) extort by demanding a ransom (target) in exchange for not taking down a number of smart meters (vector) that might cause a blackout (consequence) [11].

Criminally or financially-motivated attacker (motive). Thieves (source) who aim to physically intrude into or rob a house (target) might be able to infer private information, such as a house inhabitant's presence, from the communication between the smart meter and the utility company

(vector) in order to perform a successful break in (consequence) [161].

Political threats (motive). A hostile nation (source) might initiate a cyberwar against another country's national power system (target) by gaining remote access to the smart grid infrastructure (vector) resulting in large scale blackouts, disturbance, or financial losses (consequence) [113].

3.1.4 Medical Devices Threats

Criminal threats (motive). A criminal hacker (source) might aim to harm a patient and affect his/her health condition (target) by using wireless tools to inject or replay commands (vector) in order to change the device's state and expected operations resulting in an undesired health condition (consequence) [67]. In addition, an attacker (source) might also be able to cause harm (target) by jamming the wireless signals exchanged between medical devices to maintain a stable health condition (vector) resulting in the unavailability of the device and its failure to deliver the expected therapies (consequence) [67, 70, 151].

Spying threats (motive). A hacker (source) aiming to reveal the existence of a disease, a medical device, or any other information that a patient considers private (target) by intercepting the communications of a patient's medial device via wireless hacking tools (vector), which results in a violation of privacy and confidentiality(consequence) [67]. In addition, as the medical devices communicate with the other parties, such as a hospital, a large amount of private data is stored in various locations. This could tempt an attacker (source) with spying motivations (motive) to gain an unauthorized access to such data (target) through penetrating the networks that connect among the involved legitimate parties (vector) resulting in privacy invasion (consequence) [98].

Politically-motivated threats (motive). Cyberwar has a new attack surface by which a hostile nation (source) could target political figures (target) by attacking their medical devices exploiting the devices wireless communications (vector) resulting in a potential critical health condition or eventual death (consequence) [169]. In fact, former U.S. Vice President Dick Cheney had the wireless capabilities disabled in his pacemaker because he was aware of the possible realistic assassination threats [151].

3.2 Vulnerabilities

In this section, we highlight causes of the existing vulnerabilities in CPS that we find consistent in the four applications: ICS, the smart grid, medical devices, and smart cars. Then we identify application-specific vulnerabilities that vary depending on the application. For example, not all smart grid vulnerabilities exist in medical devices and vice versa. Therefore, we need to distinguish between the generic and the application-specific vulnerabilities, so suitable solutions can be designed accordingly.

In addition, using the abstract CPS model proposed in Sec.2 to classify the vulnerabilities into into three types, according to which CPS aspect a vulnerability appears in: *cyber vulnerabilities*, *cyber-physical vulnerabilities*, and *physical vulnerabilities*.

3.2.1 Causes of Vulnerabilities

Isolation assumption. The trend of “security by isolation" has been dominant in most, if not all, CPS applications since their initial design. The focus has been on designing reliable and safe systems, whereas the security has not been of great importance. This is because the systems were supposed to be isolated from the outside world, and therefore, considered secure. For example, in ICS and power grid (before it became “smart"), the security relied on the assumption that systems are isolated from the outside world, and the monitoring and control operations were performed locally [27, 50, 106]. Furthermore, medical devices, such as IMDs, were originally designed to be isolated from networks and other external interactions [67]. In addition, the same isolation assumption is also present in smart cars where the security of the ECUs' intercommunications relied on their isolation from adversaries [95]. Recent and ongoing advances in CPS applications do not adhere to the isolation assumption, but rather more connectivity has been introduced. The increased connectivity increases the number of access points to cars, thus more attack surfaces arise.

Increased connectivity. CPS are more connected than ever before. Manufacturers have im-

proved CPS by adding services that rely on open networks and wireless technologies. For example, ICS and the smart grid are connected to control centers which are connected to the Internet or some business-related networks. In fact, most ICS attacks have been internal until 2001; after that most of the attacks originate from external sources [21]. This is clearly due to the increase connectivity deployed in ICS. In addition, some field devices are directly connected to the Internet, for faster incident response and more convenience [99, 158]. Medical devices have wireless capabilities for easier reconfiguration and monitoring. Different wireless technologies are deployed for such needs such as LF Bluetooth. Smart cars have more connectivity so they are referred to as “connected cars". This connectedness relies on wireless communications such as Bluetooth, cellular, RFID, and satellite radio communications.

Heterogeneity. CPS consist of components that are usually heterogeneous such as COTS, third party, and proprietary components are integrated to from a CPS application. CPS are almost always multi-vendor systems, and each product has its own security problems. For example, a component might have been manufactured, specified, or implemented by different entities, and eventually integrated by the system deployers. Hence, the building components of CPS are more integrated rather than designed [50]. This integration invites vulnerabilities that exist within each product [9]. For example, one step of the Stuxnet attack was to exploit the default password in Siemens PLC to access a computer running a Windows OS [115]. The internal details of the integrated components are unknown, and thus they may produce unexpected behavior to the deployer. In fact, most of the bugs that led to successful attacks in smart cars, for example, were found at the boundaries of components, where the incorrect assumptions interact.

Many stakeholders. The number of CPS stakeholders is relatively large. This includes manufacturers, implementors, operators, administrators and consumers. Their activities and privileges differ, and hence need to be properly managed [9]. The large number of stakeholders, as well as the heterogeneous CPS components, require *change management*. This is another issue that we observe to be somewhat ignored. When changes occur in some CPS components, some coordination is required at some level by the stakeholders. Such changes are replacing hardware, updating

or changing software, and adding new capabilities [106]. Any uncoordinated change might alter the initial assumptions about the CPS security, and therefore violate them and potentially introduce new vulnerabilities.

3.2.2 Cyber Vulnerabilities

ICS Vulnerabilities.

Open communication protocols. ICS reliance on open standards protocols, such as TCP/IP and ICCP, is increasing. This invites these protocols' vulnerabilities to ICS applications. TCP/IP's vulnerabilities have been studied and investigated for many years, but the protocol still has security issues because it was not intended to be secure by design [14, 71]. Another protocol is Remote Procedure Call (RPC), which also has a number of security vulnerabilities, one of which contributed to the well-known Stuxnet [5]. In addition, Inter-Control Center Communications Protocol (ICCP), which interconnects control centers, lacks basic security measures such as encryption and authentication [134].

Wired communications. This includes fiber-optic and Ethernet. Ethernet is usually used for local area networks in substations. Because the communication using Ethernet uses the same medium, it is vulnerable to traffic analysis/interception, man-in-the-middle (MITM) attack [81]. An inside attacker, for example, could exploit the privilege of access to the local network and perform ARP spoofing to impersonate legitimate components and possibly inject false data to the traffic or disclose classified information [142, 173]. ICS-CERT [79] highlights ICS's vulnerability to ARP spoofing, and suggests some appropriate countermeasures.

Wireless communications. Wireless communications are either long-range or short-range. For short-range, they are usually performed within the ICS plant assuming an adversary is not able to get close enough to capture the wireless traffic due to the presence of security personnel. However, the traffic is still vulnerable to be capture, analysis or manipulation by malicious insiders or even a capable-enough outsider [46]. In addition, when employees connect their own, probably unsafe, devices to the ICS wireless network, they expose it to potential threats, so that an attacker could use

their personal computers as an attack vector [82]. In addition, long-range wireless such as cellular, microwave, and satellite are also used in ICS. The security of this type is somewhat absent in the literature and needs investigation. In general, wireless networks are more vulnerable to various attacks, including passive and active eavesdropping, replay attack, unauthorized access and others discussed thoroughly in the literature as in [175, 88].

Web-related vulnerabilities. A web-related vulnerability is SQL injection, where attackers can access databases records without authorization [142, 185]. The database that are connected directly or indirectly to ICS servers, such as SCADA, contain important data such as historical data and users' information. Furthermore, emails can also contribute to malware spreading to the network. In[129], a number of email-based attacks are shown by experimentation. In addition, gathering security credentials for ICS-connected computers is a very enticing target for attackers interested in gaining access to a secured network. If the employees are not alert and cautious about suspicious emails, they could be spoofed by phishing emails and their credentials are at risk, and the network and the ICS as a whole. Finally, vulnerabilities in Internet-exposed devices that are connected to the local network, such as servers in the control center, employees' portable devices like laptops, and smartphones might be exploited to perform malicious activities that affect the desired operations of the control devices [28].

Smart grid vulnerabilities. The large scale of the smart grid could result in more vulnerabilities, and make it unrealistic to ensure security everywhere in the grid. The grid's components now become more accessible in every household, such as smart meters, and hence provide a potential access point to it for malicious attackers [121]. In addition, the smart grid consists of heterogeneous components run by different entities. For example, the generation plant of the grid interacts with the transmission plant, where the transmission, in turn, interacts with the distribution, and finally the distribution delivers the electricity to the end users. Each type of interaction is usually run and administered by different companies, as a result of the deregulation trend [54, 73, 121]. In general, the smart grid has two types of vulnerabilities, one inherited from the existing power system infrastructure; whereas, the other results from the new kind of interactions among utilities

and customers, that is the information infrastructure. This includes monitoring and controlling operations of the smart grid through enormous connected networks.

DoS. In the context of the smart grid, DoS attacks involve large scale blackouts, loss of billing and customers' information. Since, the smart grid relies heavily on communication technologies, a DoS attack is not something far-reached for attackers using the Internet as an attack vector.

Communication vulnerabilities. The information infrastructure in the smart grid relies on a number of standardized Internet protocols, such as TCP/IP, Telnet, and SNMP. Each of these protocols has vulnerabilities that could be used to launch different attacks on communications. Different protocols are used according to the desired requirements in a particular application and based on its nature in terms of location and interconnections with other components. For example, TCP/IP is used for general-purpose connectivity to the Internet. The Internet-faced networks are, sometimes directly or indirectly, connected to the smart grid's monitoring and control networks due to network misconfiguration [43]. The connectivity itself is considered a vulnerability, let alone vulnerabilities in the open protocols. In addition, ICCP, which is the standardized protocol for data exchange between control centers, has a critical buffer overflow vulnerability [185].

Software vulnerabilities. Almost the same software vulnerabilities in ICS hold in the smart grid in addition to smart grid-specific vulnerabilities. For example, widespread smart meters that are remotely upgradeable, invite a critical vulnerability. An attacker can make use of such features to cause blackouts by controlling the meters, either from the control center, or the meters individually. This vulnerability can also be exploited by a software bug [11]. Some vendors leave backdoors in smart meters. Santamarta, in [153], was able to discover a backdoor in some smart meters that would result in full control of the meter, including pricing modifications. The meters can be connected to via *Telenet* protocol. In addition, this vulnerability can also be exploited to affect other smart meters in the grid to launch coordinated attacks.

Customers privacy vulnerabilities. A new type of vulnerabilities has emerged as a result of two-way communications between smart meters at customers' locations and utility companies. Attackers may be able to intercept the vast amount of traffic generated from smart meters and infer

private information about customers[40]. The kind of information attackers could be interested in is, for example, daily habits, occupation of targeted houses.

Medical devices vulnerabilities

Security through obscurity paradigm. Because of the lack of *mandatory* security standards for manufacturers of medical devices, some resort to designing their own protocols, relying on their secrecy as security measure [98]. This paradigm, a.k.a "security through obscurity" has always failed to thwart attackers.

Wireless communications. Because most medical devices rely on wireless communications, this suggests their vulnerability to a range of wireless-based attacks such as jamming, noise, eavesdropping, replay, and injection attacks. The communications between ICDs and their programmers are vulnerable to eavesdropping due to the lack of encryption. Besides this confidentiality violation, the lack of encryption allows replay attacks [66]. In addition, patients with IMDs or wearable devices can be vulnerable to a number of privacy invasion attacks ranging from discovering the existence of the devices, the devices type, to some physiological measures gathered by the device. In addition, if the devices' unique information is inferred, a patient could be tracked because of this tractability vulnerability [67].

Software vulnerabilities. The literature on software security in medical devices is limited, and it is a critical arena that needs further investigation. The role of software has been growing in medical devices, and so has the likelihood of more software vulnerabilities. As a result, recalls of medical devices due to software-related defects has increased [70, 61]. Failure of the device due to a software flaw could result in critical health conditions. Furthermore, Hanna et al. [70] presented the first publicly known software security analysis for medical devices. They found that one type of medical devices, namely Automated External Defibrillator (AED), to have four vulnerabilities: buffer overflow resulting in arbitrary code execution, weak authentication mechanism, improper storing of credentials, and the improper deployment of Cyclic Redundancy Check (CRC) that could lead to illegitimate firmware update. The device has no wireless capabilities, so the attacks would be relatively difficult. In addition, certain assumptions by device designers could lead to

undesired consequences. For example, Li et al. [101] show how a CRC check in the code can lead to dangerous attacks such as replaying outdated measures and sending illegitimate commands.

3.2.3 Cyber-Physical Vulnerabilities

ICS.

Communication protocols.

ICS relies on protocols that used to be proprietary such as Modbus and DNP3. Such protocols have been designed with the isolation assumption in mind. Thus they lack a considerable amount of security. Modbus and DNP3 protocols are widely used in ICS applications to monitor and send control commands from the control center to sensors and actuators that are directly attached to physical objects, therefore we consider them cyber-physical. ModBus protocol, the de facto standard communication protocol used in many ICS, lacks basic security measures that make it vulnerable to a plethora of attacks. Its lack of encryption exposes the plain text traffic to eavesdropping attacks [20, 8]. It also lacks integrity checks making data integrity questionable [20, 129]. In addition, no authentication measures are implemented, suggesting the feasibility of manipulating the data addressing actuators to make them act undesirably, or with data coming from sensors so the controllers can be spoofed by false data [165, 185]. Similarly, DNP3 protocol also does not implement any kind of encryption or authentication mechanisms [49, 78]. It has, however, a simple integrity measure using CRC. Although CRC is relatively simple, it is better than no integrity check altogether, like in Modbus. East et al. [49] analyzed the DNP3's vulnerability to at least 23 attacks that exploit the absence of encryption, authentication, and authorization. In addition, they also propose a taxonomy of the attacks that would improve their analysis. Their work was inspired from [78], Huitsing et al. proposed similar taxonomy but for Modbus attacks.

Connected field devices. Direct access to remote field devices such as RTUs and PLCs used in the smart grid is also a vulnerability that might be overlooked by the smart grid's operators. Some field devices might be left with default passwords [121]. Furthermore, a large number of PLCs were found to be directly connected to the Internet [99]. In fact, Leverett identified 7,500 field

devices that are directly connected to the Internet [100]. Those devices are also used in the smart grid, thus the smart vulnerability is also applicable to it.

Secondary access points. Sometimes, in case of failure of the primary communications, it is useful to have a secondary communication channels to access field devices such as PLCs and RTUs. Dial-up is a typical example of a secondary channel. It provides a direct connection with field devices, which in turn directly connected with the sensors and actuators [8]. This poses an opportunity for attackers to take control over the field devices without the need to exploit other more advanced communication links, especially in the presence of default logins and simple authentication mechanisms.

RTOS vulnerabilities. The operating systems in ICS devices, such as PLCs and RTUs, are Real-Time OS (RTOS), and they do not implement any access control measures. Therefore, all users are given the highest privileges, i.e, root access. This is fundamentally insecure, and clearly makes the devices vulnerable to various kinds of attacks due to the lack of separation of privileges [185]. Some attacks are realized by exploiting buffer overflow vulnerabilities in the OS running in the control center [165, 185]. Buffer overflow vulnerabilities are the most commonly reported vulnerabilities to ICS-CERT [79].

General-purpose OS vulnerabilities. Applications that are used for controlling and monitoring field devices such as PLCs can become attack surfaces for various types of attacks. These applications are running on computers that use general-purpose OS such as Windows OS. They are connected to field devices, either directly or indirectly. If the hosting computers or laptops have vulnerabilities in the OS or running software, they posses a potential attack vector on the connected field devices and their processes. An example of such exploitable vulnerabilities are two Windows OS vulnerabilities that were exploited in the Stuxnet attack. The first one is a vulnerability in the *Print Spooler Service*, which is vulnerable to remote code execution such that it could allow an attacker to remotely execute code by sending a specially crafted print request to a vulnerable system connected to the print spooler over remote procedure call (RPC) [2]. This particular vulnerability allowed Stuxnet to copy itself onto the vulnerable computer [36]. The other exploited vulnera-

bility was in *Windows Server Service* that also was vulnerable to remote code execution through sending a specially crafted RPC request [5]. Using this vulnerability, Stuxnet connected to other computers [36].

Software vulnerabilities. We consider programs running on general-purpose OS for controlling and monitoring controllers as cyber-physical software. This is because the controllers are exposed to physical components and controlled by cyber components. An example of such programs is *WinCC*, which is a Siemens software used for controlling PLCs. In the Stuxnet attack, the first step was to target vulnerable computers running WinCC software [36]. The computer's vulnerability is exploited so Stuxnet can copy itself onto the computer, it then installs a rogue driver DLL file that is used by both WinCC software and the PLC. Once the driver DLL installed, a rogue code is sent to the PLC. The critical vulnerability in the controller that allows such an action is the lack of the code digital signature. The controller treats any code that is syntactically correct as legitimate [94]. This is a design flaw that must be corrected. The problem, however, with such a controller is its limited computational capabilities, and hence cryptographic measures are computationally expensive. Another class of cyber-physical software covers software products running on field devices such as PLCs and other controllers. As we pointed out earlier, the presence of COTS products in CPS is one of the contributing factors for the increased number of vulnerabilities. Leverett and Wightman [99] revealed an authentication vulnerability in a very common COTS product that is available in 200 PLC models. This vulnerability allows the attacker to bypass the authentication and consequently take control of the PLC. What makes matters worst is how easy it is to scan the Internet for such PLCs. The authors conducted multiple scans and discovered a surprisingly large number of PLCs that directly connect to the Internet. In addition, some vendors leave backdoors in some field devices. This makes it possible for attackers to gain access and full control over the device when valid credentials are gathered [153].

Web-related vulnerabilities. Web-based vulnerabilities result from deploying web interfaces directly on cyber-physical components. For example, it is a new feature in modern PLCs and RTUs to have web-based access for configuration and monitoring. Sadly, some are left with their default

passwords [166]. This feature makes these field devices vulnerable to attacks such as DoS, where an attacker floods the devices by requests resulting in unavailability of the service for the legitimate users. Furthermore, the web-interface is directly connected to the communication protocol stack such as TCP/IP. If the stack is improperly implemented, it is possible to exploit that.

The Smart Grid.

Network intrusion. The smart grid's networks are potentially vulnerable to intrusions.

The cyber-physical components could be controlled when successful network intrusion succeeds. The field devices, such as AVR and state estimators, are connected to a LAN network that is not supposed to be connected to the Internet. However, it is possible to get into the network by malware infection, through a USB stick for example. In addition, it is also possible to inject the network with bogus packets that aim to flood it, resulting in DoS attack, or to inject false information, resulting in decisions based on false information [161].

Communication protocols. The power system infrastructure in the smart grid relies on almost the same protocols in ICS, such as Modbus and DNP3, thus the same related security issues to the protocols still hold in the smart grid. In addition to these protocols, IEC 61850 has also been introduced recently as an advancement of these protocols in substations' communications. The lack of some security properties in these protocols has a different impact in the context of the smart grid. For example, protocols that lack encryption, make the data in transit vulnerable to eavesdropping, which results in a number of attacks such as the inference of customers' usage patterns [121, 111], or even injection of false information due to the lack of authentication [173, 146]. The communications in the smart grid can be also vulnerable to false data injection attacks, where an attacker injects false measurements to cause disruptions and financial losses [180].

Smart meters. Smart meters rely on two-way communications, which contributes to a number of new security concerns about an attacker's abilities to exploit such interaction [89]. For example, a smart meter may have a backdoor that an attacker could exploit to have full control over the device. In [153], the author analyzed the meter's available documentation and found out that there is a "Factory Login" account. Aside from the customers' accounts with limited capabilities for

basic configurations, this factory login account gives full control to the user over the smart meter. What's more, the communication is transmitted through *telnet* which is well-known for major security weakness, i.e, sending data in clear text without encryption. This vulnerability is clearly discovered due to the readily available documentation of the meter that made the vulnerability easily discovered. An important question is what can the attacker do once full control over the smart meter is gained?

The author summarized three potential attacks: 1) power disruption, either directly by malicious interactions with other devices to change their desired activities in terms of power consumptions and, or indirectly by the injection of false data such that the control center receives false information and consequently makes wrong decisions, 2) using the meter as "bot" to launch different attacks possibly against other ICS devices or systems within the ICS network, and 3) the meter's collected data could be tampered with so that the bill reflects false data to reduce the cost.

Medical Devices.

Communications. The reliance on wireless communications in wearable and IMDs invites vulnerabilities that have a physical impact on patients. Examples are jamming the communications, replaying and forging sensors' measurements and actuators' commands. If medical devices fail to transmit or receive expected packets, a patient's health is at risk of incorrect operations performed by the medical device, and an undesired health condition. Medical devices communicate with other devices or with their programmers. In either case, the communication should not be eavesdropped, and therefore should be encrypted. Unencrypted traffic makes the devices vulnerable to numerous attacks.

DoS. The wireless nature of the communication the devices rely on invites jamming attacks, where an attacker can block the communications to achieve undesired goals. For example, when an insulin pump does not receive periodic updates from the associated CGM, it assumes the patient's condition is stable, and no need for an insulin injection. This leads to an undesirably high glucose level [101, 147]. Furthermore, attackers can render devices unavailable for the patient. The attack takes different forms, but eventually results in the unavailability of the device's services. For

example, some devices are vulnerable to battery exhausting attacks, where an adversary exhausts the devices computational or communicational resources to withdraw the battery's reserves [66, 151]. In addition, jamming the communications between devices could lead to DoS attacks.

Packets injection. Injection of unauthorized commands or false data by injecting a specially-crafted packet. Halperin et al. [66] and Gollakota et al. [62] demonstrated the ICDs' vulnerabilities to injection attacks by exploiting wireless vulnerabilities. In addition, Li et al. [101] demonstrated the insulin pump's vulnerability to be remotely controlled by intercepting the device's communications with its remote control. In addition, Radcliffe [147] uncovered a vulnerability in the insulin pump device that would allow injection attacks. The device requires its serial number to be part of the command packet as an authentication measure. An attacker equipped with the number can inject unauthorized commands to the device.

Replay attacks. This type differs from packet injection attacks. Replay attacks don't require knowledge of the underlying protocols, instead, an attacker only needs to capture legitimate measurements or command packets, and retransmits them to the component of interest. Li et. al [101] revealed a vulnerability in an insulin pump that would allow replay attacks so that the pump receives a dishonest reading of the glucose level, and therefore the patient decides mistakenly to inject the wrong amount of insulin such that the decision might threaten the health condition. In addition, Radcliffe [147] revealed that a CGM device was vulnerable to replay attack. By retransmitting pre-captured packets to the CGM the author was able to cause incorrect values. In addition, besides the violation of confidentiality, lack of encryption allows replay attacks [66].

Programmer unauthorized usage and impersonation. An attacker can use a commercial programmer with no authorization as a result of the implicit trust [66]. This makes medical devices vulnerable to safety-critical attacks without the attacker's technical knowledge. In addition, some attacks do not need programmers. Instead, Universal Software Radio Peripheral (USRP) is sufficient to replace a programmer and send malicious packets as shown in [66].

3.2.4 Physical Vulnerabilities

In this section we review vulnerabilities in physical components that would cause cyber impact when exploited. Generally, physical tampering with a physical component or environment surrounding them, such as reducing/increasing the water level in a canal, or applying unexpected heat to temperature sensors in some hazardous environments. Most of the vulnerability analysis work in the literature focuses on cyber attacks with physical impact. Very few, one the other hand, has been done to analyze physical attacks with cyber impact such as in [108].

ICS.

The physical exposure of many ICS components, such as RTUs and PLCs, that scatter around the ICS plants and outside it, is a vulnerability by itself. With insufficient physical security provided to these components, they become vulnerable to physical tampering or even sabotage. For example, a water canal's sensors rely on solar panels as a source of energy so they can communicate with the control center. These panels were stolen, and therefore, the control center lost critical data necessary for the desired operations [10].

The Smart Grid.

Field devices in ICS and Smart Grid are placed in unprotected environments. This makes them vulnerable to different sorts of attacks that leverage this nature of unprotected exposure. An adversary could, for example, physically sabotage the devices causing interruptions to the expected operations and eventually DoS attack. The attack seems simple, and indeed it is.

A huge amount of physical components of the Smart Grid are highly exposed without physical security, and thus vulnerable to direct physical destruction. For example, power lines are vulnerable to malicious, accidental, and natural attacks. For example, overgrown trees caused a large blackout affecting over 50 million people in Northern Ohio [165]. In addition, smart meters attached to buildings, houses, and remote areas make them an easy target to various physical attacks. Mo et al. [121] suggest the infeasibility of the physical protection of all assets in the Smart Grid. Therefore, it is necessary to devise prevention and detection solutions.

Medical Devices.

Medical devices, whether implantable or wearable, sometimes could be vulnerable to physical access. That is, the attacker's ability to physically deal with the medical device. For example, for maintenance purposes, an attacker exploiting the absence of the device's owner, tampers with it, and potentially performs malicious activities such as malware installation, or modification of the configurations such that the device delivers unadvised treatment that could threaten the patient's health. In addition, by physical access to the device, it is also possible to get the device's serial number, which is useful for some attacks [147]. In general, physical access to the device opens up many possibilities to various attacks. Hanna et al. recommends protecting medical devices from physical access by any potential attacker [70]. Another subtle vulnerability to consider is the mobility of medical devices' users. It could be a vulnerability by itself. As the device's designers cannot control the patients' surroundings, the devices could be vulnerable to unpredicted physical attacks when a patient is in an unsafe location. This is especially true for politically-motivated attacks [169].

3.3 Attacks

We review reported cyber, cyber-physical, and physical attacks on the four CPS applications that exploited the aforementioned vulnerabilities in Section 3.2. In general, publicly known attacks are rare [144], and it is infeasible to find attacks that represent exploitations of all vulnerabilities in section 3.2. Instead, we consider attacks that have been realized by experimentation or in real life. Then we describe at the end of this section the cyber-physical attacks using the taxonomy that Yampolskiy et al. presented in [182, 183] in tables 3.2, 3.3, and 3.4 for attacks on ICS, the Smart Grid, and wearable and IMDs, receptively. Their proposal dissects CPS attacks into six-dimensional description, by which we gather more insight about each attack. The description includes the attacked object (Influenced Element), the resulting changes on the attacked object from the attack (Influence), indirectly affected components (Affected Element), changes on the

CPS	Vulnerability	Type	Cause
ICS			
	Open communication protocols	C	I, O
	Wired communications	C	I, H, S
	Wireless communications	C	I, C
	Web-based attacks	C	Con, H
	Insecure protocols	CP	I, Con
	Interconnected & exposed field devices	CP	I, Con
	Insecure secondary access points	CP	I, Con
	Insecure OS & RTOS	CP	I, Con, H
	Software	CP	Con, H
	Equipments' physical sabotage	P	I
Smart Grid			
	Blackouts	CP	I, Con, H
	Communication protocols	C	I, C
	Software	C	I, C, O, S
	Customers' privacy invasion	C	I, Con, H
	Interconnected field devices	CP	I, Con, H
	Insecure protocols	CP	I, Con
	Insecure smart meters	CP	I, Con, H
	Equipments' physical sabotage	P	I
Wearable & IMDs			
	Jamming & noise	P	Con
	Replay & injection attacks	C, CP	I, Con
	Patient's privacy invasion	C	I, Con
	Software	C, CP	I, H
	DoS	CP	Con

Table 3.1: Summary of vulnerabilities. C: Cyber, CP: Cyber-Physical, P: Physical; I: Isolation assumption, C: Connectivity, O: Openness, H: Heterogeneity, S: Many stakeholders

CPS application (Impact), how the attack took place (Method), and preceding attacks needed to make an attack successful (Precondition). We also integrate our CPS framework into this taxonomy to add additional insight about attacks. In particular, we highlight CPS aspects in the attack tables with *C*, *CP*, and *P*, for cyber, cyber-physical, or physical aspects, respectively.

In this section, we categorize the attacks based on the damages' location. Attacks that do not reach sensors/actuators are considered purely cyber, while attacks that directly impact physical components are physical. Whereas attacks that indirectly impact physical components, through cyber components, are cyber-physical.

3.3.1 Cyber Attacks

ICS

Communication protocols. A number of attacks have exploited vulnerabilities in the communication protocols. For example, spoofing attacks on Address Resolution Protocol (ARP) were demonstrated on SCADA system [167, 82].

Espionage. *DuQu* and *Flame* are two examples of ICS attacks with spying purposes [37, 125]. Flame, for example, targeted various ICS networks in Middle East and was discovered in 2012. The malware's main goal was to collect corporations' private data such as emails, keyboard strokes, and network traffic [125]. Although the intention of the attack was not clear, such information leakage could result in undesired consequences such as a hostile nation acquiring industrial secrets that could be used for cyberwar or industrial competition.

In addition, in 2013, a group of hackers, known as *Dragonfly*, targeted energy firms in the U.S. and Europe. The attacker's main goal seems to have been gathering private information. To do that, they needed to infect systems in the targeted firms with malware that grants remote access. They started by sending phishing emails to the personnel of the targeted firms containing malicious PDF attachments. Then the attack vector escalated to exploiting *watering hole* vulnerabilities in victims' browsers by directing victims to visit malicious websites hosted by the attackers. Both delivery mechanisms infected targeted systems with a malware that allowed attackers to gather

private information in the infected systems[149].

Unintentional attack. Although software updates are critical to ensuring a desired security posture, it can be a source of service disruption. For example, in a nuclear plant, one computer in the control center was updated and rebooted thereafter. The reboot erased critical data on the control system, which misinterpreted the lack of it, resulting in a shutdown of the plant [27].

Web-based attacks. *Night Dragon* attack, in 2011, targeted sensitive information from private networks of a number of energy and oil companies [8]. The attack combined a number of vectors to succeed. It exploited a SQL injection vulnerability in a Windows server, social engineering, and malware injection [8, 115].

The Smart Grid

DoS. The traffic in the smart grid is time-critical, so delays in it may result in undesired consequences. Flooding the network at different layers is the probable approach to achieve DoS. The impact of DoS on the smart grid substations was evaluated in [104]. The authors found that the network performance only gets affected if the flooding is overwhelming. In addition, at the physical layer, the deployment of wireless communications increases and therefore, jamming attacks' are possible as shown in [105].

False data injection. Introducing false data in smart grid traffic leads to different consequences such as service disruption and financial losses. In [103], a simulated false data injection was demonstrated to evaluate the impact on the state estimation in the smart grid. The authors assumed the attacker's pre-intrusion to the control center for a successful attack, which aimed to ultimately inject false measurements to the meter's to disrupt the state estimate process. Such disruption leads to financial losses for the operating utilities [173].

Customers information. Attackers can analyze network traffic in the smart grid between smart meters and data centers to infer private information about customers. For example, an attacker can determine if a target is available at home at particular times and dates. In addition, it is also possible to deduce lifestyle in terms of sleeping times and quality, preferred home appliances, and many more as shown in [122].

Untargeted malware. In 2003, the Slammer worm resulted in disabling the traffic between field devices and substations. Although that malware was not intended to the affect the energy sector, it had an effect because of the interconnectedness of the smart grid networks. The malware consumed a significant amount of the time-critical traffic, but did not cause service outages [21].

Medical Devices

Most, if not all, of the reported attacks on medical devices have been performed in experimental environments. However, the possibility of the attacks in this section, should raise an alarm to stimulate the efforts in improving security in medical devices. Although some attacks are specific to certain devices, such as insulin pumps, the same attack techniques could be applicable to other IMDs and wearable devices due to the similarities in the communication links and hardware components.

Replay attacks. By exploiting a vulnerability in the insulin pump, replaying eavesdropped packets is possible by incorporating a previously intercepted device PIN [101]. In addition, replay attacks could result in misinformed decisions regarding insulin injection [147]. For example, by replaying an old CGM packet to the insulin pump, the patient receives a dishonest reading of the glucose level, and therefore decides mistakenly to inject the wrong amount of insulin such that the decision might threaten the health condition.

Privacy invasion. Attacks violating patients' privacy have different goals and consequences. For example, for the remote control attack on the insulin pump in [101], an attacker needs to learn the device type, PIN, and legitimate commands sent from the remote control. The authors successfully performed this attack and revealed three types of privacy-related information, namely, the devices' existence, its type, and finally the PIN. In addition, Halperin et al. [66] demonstrated similar attacks on an ICD medical device such as revealing patient's private personal and medical information, and the device's unique information.

3.3.2 Cyber-Physical Attacks

ICS

Legacy communication channels. Dial-up connections that provide direct access to field devices and sometimes overlooked. In 2005, billing information of a water utility was accessed by exploiting the dial-up connection in a canal system [166]. Although this attack did not have a physical impact, it could have due to the control capabilities provided by the dial-up connection.

Disgruntled insiders. A huge financial loss for utility companies with undesired environmental impacts could result from attacking a water and sewage system. In 2000, an ex-employee intentionally disrupted the operations of a sewage treatment system in Maroochy Water Services in Queensland, Australia. The attacker exploited his knowledge, as a previously-legitimate insider, to change configurations in pumping stations using a laptop and a radio transmitter. The consequence of the attack caused a huge amount of raw sewage to flood into the streets and taint the environment [158].

Modbus worm. An alarming work on the targeted malware presented in [129]. The authors crafted malware that exploits the lack of authentication and integrity vulnerabilities in the Modbus protocol. The worm aims to perform two attacks: DoS, by identifying sensors or actuators and sending them DoS-inducing messages, and command injection, by sending unauthorized commands to the sensors or actuators.

Malware. Some malware target specific systems to achieve goals like interception and interruption of operations. They exploit software vulnerabilities in systems hosting software to control field devices. A well-known example of that is *Stuxnet*. This attack is considered one of the most sophisticated attacks on ICS that clearly embodies cyberwar. Stuxnet exploits software vulnerabilities to achieve physical consequences [182]. Because the targeted networks were off the Internet, it is believed that the delivery mechanism was a USB stick. The attack can be generally summarized into two phases: 1) spreading and determining targets, and 2) PLCs hijacking [94]. The first step was realized by exploiting two zero-day Windows vulnerabilities, i.e, one in the shared printing server and the other was in Windows Server Service. Both vulnerabilities would allow remote

code execution using RPC. Stuxnet used the first vulnerability to install itself in the system and the other to connect to other systems to also install itself in an iterative fashion. This process led to about 100,000 infected systems worldwide, however, because the attack was targeted on specific PLCs, the infection did not have an influence on systems that were not connected to the targeted PLCs. Once Stuxnet is installed, it looks for a specific software used for monitoring and sending commands to the PLCs, that is Seimens WinCC. It goes through a thorough analysis to ensure that WinCC is connected to one of specific Siemens PLCs (315 and 417 PLCs) [94]. Once determined, each type is treated slightly differently to inject the malicious code that aims to alter the PLC's configuration. Once that achieved, the final objective of the attack was realized, which is, most likely, to damage the centrifuges used for uranium enrichment. For a detailed Stuxnet analysis, we refer to [135].

Web-based attacks. A group of hackers exploit a web-based interface that is directly connected to field devices like PLCs. They opened multiple connections and left them opened until the authorized users could not access them, resulting in a DoS attack. In addition, they also sent a web page to the controller/ field device that contains malicious Java script code designed to exploit a bug in the TCP/IP stack causing resetting of the controller [166].

The Smart Grid

Cyber extortion. This type of attack is rare, at least publicly, where attackers take control over the target smart grid and make demands as a price of not causing a large-scale blackout [130].

Blackouts. In the context of the smart grid, a blackout is considered a DoS attack. The availability of the smart grid is probably the most important security goal to maintain, and attacks aiming to compromise this requirement could result in a nationwide impact such as a large-scale blackout. In 2007, Idaho National Laboratory (INL) demonstrated an experiment on how a generator could be damaged as a result of a cyber attack [4]. The experiment proved the feasibility of such attacks. For example, in 2003, it is believed that the two blackouts in Ohio and Florida were caused by a Chinese politically-motived group, the People's Liberation Army [72]. In addition, about 800 blackouts in the U.S. occurred in 2014 for unknown reasons [179]. Some speculations suggest

that such mysterious outages may have resulted from cyber-physical attacks launched by hostile nations [72].

Medical Devices

DoS attacks. This attack, when successful, could lead to critical health condition to patients. In [66], Halerin et al. were able to disable the ICD therapies by replaying a previously recorded "turn off" command sent by the programmer.

False data and unauthorized commands injection. Li et al. [101] were able to remotely control an insulin pump's remote control and successfully stopped and resumed the insulin injection from a 20 meter distance.

Replay attacks. By exploiting a software vulnerability in the replay attacks countermeasure, any packet can be retransmitted to the CGM and insulin pump [101]. In addition, Radcliffe showed similar attacks in [147].

3.3.3 Physical Attacks

ICS

Untargeted attacks. Zotob worm, although not targeted on ICS, caused manufacturers to shut down their plants. For example, US-based DaimlerChrysler had to shutdown 13 of their manufacturing plants for about an hour [165]. Such an incident stimulated efforts such as in [129], where Fovino et al. performed a simulated study of the impact of malware, intended for tradition IT systems, on ICS networks. The malware's effects varied from causing ICS servers to reboot, open a potential arbitrary code execution vulnerability, infection of personal computers, and DoS.

The Smart Grid

Natural and environmental incidents. We give a few examples of power blackouts in 2014 that resulted from natural causes to show the impact of physical exposure and unreliability of the smart grid components. An ice storm in Philadelphia affected 750,000 people for several days with no electricity, whereas a tornado hit the New York area affecting 500,000 people [179]. Furthermore,

the widespread power transmission lines around various environments and conditions contributed to unexpected attacks such as overgrown or falling trees. For example, some overgrown trees caused a large blackout affecting over 50 million people in Northern Ohio [165]. This incident, however, is controversial and security analysts suggest that it resulted from a cyber-physical attacks originating from China [72]. In addition, in 2014, wild animals caused 150 blackout in the U.S. by eating and damaging cables [179].

Theft. Copper wires and metal equipment are profitable targets for financially-motivated thieves. For examples, theft caused a blackout with an impact on 3,000 people in West Virginia [179].

Car accidents. In 2014, 356 outages in the U.S. were caused by cars hitting transmission towers, transformers, or power poles [179].

Vandalism. Attackers can physically damages parts of the smart grid such as cables, poles, generators, smart meters, and transformers. An example of that is an incident in 2013 where a sniper in California shot more than a hundred shots at a transmission substation, leaving 17 transformed damaged [132].

Terrorist attacks. In 2014, the first terrorist attack on a power grid occurred in Yemen. The attackers launched rockets to destroy transmission towers and caused a nationwide blackout affecting 24 million people [83].

Medical Devices

Acquiring unique IDs. Obtaining devices' serial numbers is an example of attacks that require physical access to the target devices [147].

3.4 Controls

We briefly describe the research trends in CPS controls in two distinct paths. The first one is the solution that targets CPS in general, regardless of the application. The second is application-specific solutions that are specifically designed for some applications. In addition, we will highlight, whenever applicable, some solutions that can be cross-domain. That is, for example, some solutions

Name	IE [1]	I [2]	AE [3]	Impact	Method	Precondition	Ref
Maroochy	Pumps	Pumps work undesirably	Correct settings in pumping stations manipulated	Raw sewage flood in streets, tainted environment, and financial losses	Used a laptop and a radio transmitter to manipulate pumps	Insider knowledge	[158]
Modbus worm	ICS network	Infected ICS network	Connected field devices	Servers' rebooting, DoS, & unauthorized commands injection	Inject malware code into ICS network traffic	Access to ICS traffic	[129]
Stuxnet	Centrifuges PLCs	Exaggerated rotation of centrifuges	Centrifuges' rotors	Lifetime reduction & physical damage	Illegitimate commands from PLCs sent to centrifuges	Infected PLC by Stuxnet	[182, 36, 94, 135]
Web-based attacks	Field devices (e.g. PLCs)	Field devices web interface feature	The physical environments controlled by devices	Legitimate personnel unable to connect to field devices remotely or locally (DoS)	Leave devices with open connection state	Devices are directly exposed to the Internet	[166]
Web-based attacks	Field devices (e.g. PLCs)	Field devices web interface feature	The physical environments controlled by devices	Devices lost configurations	Malware injection	TCP/IP vulnerability in a COTS implementation	[166]

Table 3.2: ICS Cyber-Physical attacks

designed for cars could be applied to medical devices, or vice versa. CPS controls should consider the unique properties of CPS, such as time-criticality, and cyber-physical interaction.

3.4.1 General Controls

Here we review controls that consider securing CPS, regardless of the application. Addressing the vulnerability causes is the first step in the solution.

More connectivity controls. Therefore, new security considerations must be taken into account to secure the access point from unauthorized access. Furthermore, the communication protocols used for realizing such connectivity are either proprietary protocols, such as Modbus and DNP3 in deployed ICS and Smart Grid, or open protocols such as TCP/IP. The proprietary protocols are

Name	IE	I	AE	Impact	Method	Precondition	Ref
Cyber extortion	Power delivery	Utilities lose control over their grid system	Customers	Lost of services & financial losses	Exploit Internet-connected smart grid components	Inside knowledge	[130]
Aurora experiment	Circuit breakers (P)	Change relay behavior (CP)	Power generators and power-fed substations (CP)	Physical damage to generators & inability to deliver electricity (P)	Unexpected opening & closing of circuit breakers (CP)	Access & inside knowledge (CP)	[184]

Table 3.3: Smart Grid Cyber-Physical attacks

burdened with a lot of vulnerabilities due to the isolation assumption initially [9].

Communication controls. Security solutions at the communication level in ICS should consider the differences with traditional IT solutions. For example, Intrusion Detection Systems (IDS) should be time-critical, and long delays are intolerable [120]. Mitchel and Chen [116, 119] focus on designing IDS solutions for CPS. In addition, they provide a comprehensive survey on IDS solutions in CPS applications [120].

Device Attestation. CPS components running software need to verify the authenticity of the code they are running. This verification helps significantly minimize malware. A number of attestation proposals can be used, however, with trade-offs. For example, hardware-based solutions such as Trusted Platform Module (TPM), where security is proved verifiably because of the presumably, untampered with, TPM component. This technique assumes the physical security of the CPS component, which is infeasible to guarantee in some CPS applications such as ICS and Smart Grid. The problem with TPM is its associated cost to CPS limited computational and energy resources. Therefore, probably a new generation of TPM could be used for CPS purposes. It should not be resource-expensive, so it does not consume devices' resources.

Name	IE	I	AE	Impact	Method	Precondition	Ref
DoS	A medical device	The device is turned off	Patients	Patient does not receive expected therapy (DoS)	Retransmit “turn off" command	Capture “turn off" previously sent by the programmer	[66]
False data injection (FDI)	Insulin pump	False measurements sent to insulin pump	Patient’s therapeutic decisions	Wrong decisions	Impersonate CGM and by sending similar packets with false data	Interception of CGM and insulin pump communications	[101]
Unauthorized commands injection	Insulin pump	Unauthorized commands sent to insulin pump	Patient’s safety	Dangerous health condition	Impersonate insulin pump remote control by sending similar packets with unauthorized commands	Interception communications between insulin pump and its remote	[101]

Table 3.4: Medical devices Cyber-Physical attacks

3.4.2 ICS Controls

New design. ICS needs security designed specifically for ICS, taking into consideration the cyber-physical interactions, and the heterogeneity of components and protocols. Cardenas et al [25] suggest that, most of the solutions in ICS aim to provide reliability, i.e, make ICS reliable in the presence of non-malicious failures. Although important, malicious cyber attacks are now possible more than ever, and must be considered when designing new solutions.

Secure communications. Security solutions at the communication level in ICS should consider the differences from traditional IT solutions. For example, some key differences are the periodic nature in traffic and real-timeliness. Periodic traffic results in identifiable patterns, that, in turn, facilitates the attacker’s task. For real-timeliness property solutions, such as cryptography can cause undesired delay. Therefore, the solutions should strike a balance between such properties and security solutions.

Protocols with add-on security. A number of proposals that rely on modifications of currents protocols, such as Modbus, DNP, and ICCP, that aim to integrate security. Fovino et al. [57]

proposed the *Secure Modbus* framework. It provides authentication, non-repudiation, and thwart replayed packets.[109] proposed *DNPSec* framework, which adds confidentiality, integrity, and authenticity.

IDS. The complexity of designing IDS for securing ICS is relatively less than it is in traditional IT security. This is due to the predictability of the traffic and the static topology of the network [93]. Zhu and Sastry [186] defined a set of goals that IDS in ICS are expected to monitor. They must detect any access, including sending commands, to communication links between controllers and sensors/actuators. In addition, any modifications for sensor settings, or any physical tampering with actuators through cyber components, should be detected. Another effort that considers the physical exposure of the wireless communications within an ICS plant is *WildCAT* presented in [46]. WildCAT is a prototype for securing ICS from cyber attacks that exploits wireless networks. The idea of WildCAT is to install it in security guards' cars and it will collect wireless activities in the physical perimeter of the plant. The collected data is sent to an analysis center, which, in turn, detects any suspicious activities and direct the guards to the location causing such activities. For further analysis of the current ICS-specific IDS solutions, we refer you to [186, 18, 120, 93, 34, 13].

Remote access to field devices. Fernandez and Fernandez [53] suggest that only authorized personnel can remotely access field devices. In addition, the access should be strictly secured by using a designated laptop through a VPN. In addition, Turk [166] suggests a simple control for web-based DoS attacks that field devices with web access features are vulnerable to. The author suggests to close idle connections. In addition, it could be a good measure to disallow multiple connections simultaneously. Usually, no more than one legitimate employee tries to access such a resource at the same time.

Encryption and key management. There is undoubtedly a need for encryption in ICS networks, one of the associated problems with encryption is the delay, which is not desirable in a time-sensitive environment. Choi et al [41] proposed a key management solution that does not cause delay and ICS-specific. In addition, Cao et al. [23] proposed a layered approach aiming to protect

sensitive data in the widespread ICS environments. Their technique relies on Hash Chains to provide: 1) layered protection such that ICS is split into two zones: high and low security levels, and 2) a lightweight key management mechanism. Thanks to the layered approach, an attacker with full control of a device in the low security level cannot eavesdrop on data from higher security level zones.

Software controls. Regularly patching security vulnerabilities in operating systems and various applications running on them is a critical security practice. For example, Windows released Stuxnet-related security patches that could have prevented it from installing the worm's dropper [94]. However, vendors of ICS applications must also keep up with the patching and release compatible versions of their applications. This ensures that ICS operators don't resort to older versions of vulnerable OS to be able to use the compatible ICS application [86].

Standardization. The National Institute of Standards and Technology (NIST) is one of the leading firms in the standardization realm. Following their, and other, standards should significantly contribute to securing ICS. One example of NIST standards for ICS is in [163], where Stouffer et al. provided a comprehensive guidelines for ICS security. They provide guidelines for technical controls such as firewalls, IDS, and access control, and operational controls such as personnel security, and awareness and training. In fact, technical and operational controls must always go hand in hand, and the negligence of one leads to serious attacks. For example, lack of awareness could make employees vulnerable to social engineering attacks such as phishing. ICS-CERT reported that most of the attacks on ICS originated from phishing emails with malware-infected attachments [8]. In addition, security experts evaluated the security of an ICS corporation, and were able, through social engineering and phishing, to gain employees' credentials [134]. Sommestad et al.[159] conducted a comparison, based on keywords mining, and concluded that the standards focus either on the technical controls, or the operational controls, but not both. In addition, some standards somewhat neglect ICS-specific properties, and focus on IT security countermeasures alone.

3.4.3 Smart Grid Controls

DoS controls. Attacks on communication components should be prevented or, at least, detected. On one hand, at the network layer, prevention of attacks like DoS is achieved by rate-limiting, filtering malicious packets, and reconfiguration of network architecture. The first two are possible, while reconfiguring the network might be difficult due to its relatively static nature. Whereas techniques at the physical layer aim at preventing attacks of the nature of wireless jamming. On the other hand, detection techniques that aim at detection DoS attacks are categorized into 4 types: signal-based, packet-based, proactive, and hybrid. The above discussion is summarized from [173], which we refer you to consult for further experimental analysis of the detection and prevention mechanisms in the Smart Grid.

IDS. IDS for Smart Grid is still an ongoing problem that is not that mature yet. Designing IDS for the Smart Grid is a complex task due to the enormous size of the grid and the heterogeneity of its components [161]. In addition, IDS built for traditional IT systems will not necessarily work for the Smart Grid. They must be specifically designed for the Smart Grid to reduce the likelihood of false detection rates. Examples of Smart Grid IDS are in [85, 118].

Low-level authorization and authentication. A common problem in a large system like the smart grid is authentication and authorization of users to gain access to low-level layers such as field devices. Commonly, all field devices share the same password that employees share. This results in the impossibility of the non-repudiation security requirement. A malicious employee could gain access to a field device and make undesired changes to the system, and there is no way to trace who did it. Therefore, Vaidya et al. [170] proposed an authentication and authorization mechanism that provides legitimate employees the ability to access field devices in the substation automation systems in the smart grid. Their proposal relies on elliptic curve cryptography due to its low computation and key size requirements compared with other public key mechanisms.

New designs. New security issues require that various aspects of the Smart Grid be approached differently. The cyber-physical nature of the systems needs to be considered. Mo et al. [121] proposed *Cyber-Phyiscal Security*, a new approach that combines systems-theoretic and cyber security

controls. They provide two examples showing the applicability of their approach on two attacks in the Smart Grid: replay attacks, and stealthy deception. They emphasize the need for considering those two types of components, cyber- and physical-components, when designing Smart Grid controls. Most of the work done is extending existing protocols and systems to capture security properties. This might work as a temporary solution, but bottom-up redesign is desired.

Security extensions. The trend of adding-on security to existing components of the Smart Grid has been emerging. Protocols like DNP3, IEC 61850 and IEC 62351, are extended to capture security properties. For example, Secure DNP3 protocol is DNP3 extended to have basic authentication, integrity and confidentiality services. The security features are added by inserting a security layer in the communication stacks of these protocols [173].

Privacy-preserving controls. Lack of confidentiality in data aggregation protocols might result in privacy invasion of consumers' private information such as billing information and usage patterns [113], while the lack of integrity could result in disruption in state estimation and consumption reports [161]. Therefore, a number of privacy-preserving techniques have emerged to provide aggregated data with confidentiality and integrity when in transit between smart meters and control centers. For discussions on such techniques, we refer you to [173, 52].

Standardization. A number of bodies, such as he International Electrotechnical Commission (IEC) and NIST, have developed a set of standards for securing Smart Grid communications. For example, IEC TC57 and IEC in IEC 62351 standards were developed by the IEC [43], whereas NIST has developed smart grid guidelines in report 7628 [138].

Smart meters' disabling protection. To prevent remote attackers who exploit the disabling feature in smart meters, Anderson [11] suggests that smart meters should be programmed to notify customers in enough time in advance, before the command takes effect and disables the smart meter. This measure helps in the early detection of DoS attempts before they take place.

Physical security. As smart meters are physically exposed, they must be physically protected. NIST standards [138] state that smart meters must have cryptographic modules in addition to physical protection. Also smart meters are sealed in tamper-resistant units that should prevent

unauthorized parties from physically tampering with them.

3.4.4 Medical Devices Controls

Authentication. Halperin et al. proposed a cryptographic-based authentication and key-exchange mechanism to prevent unauthorized parties from accessing IMDs [66]. Both mechanisms do not require battery consumption as a source of energy. Instead, they rely on external Radio Frequency (RF) as a source of energy. In addition, Out-of-Band (OBB) authentication is deployed in some wearable and implantable devices. By which, authentication is performed using additional channels, other than the channels used for communication, such as audio and visual channels [151]. In fact, biometrics, such as electrocardiograms, physiological values (PVs), heart rate, glucose level and blood pressure, can all be used for key generation for encrypted communication between the body sensor network (BSN) [151].

Intrusion Detection Systems. Halperin et al. proposed a detection mechanism that alarms patients of unauthorized communication attempts with their IMDs [66]. In addition, not only does the *Shield* proposed in [62] detect malicious wireless-based attacks on IMDs, it also prevents them. Although the *Shield* is not designed specifically as an IDS, it certainly serves as one. On the other hand, Mitchell and Chen [117] aim to detect compromised sensors and actuators that pose threat to patients' safety through behavior rule-based IDS. Their proposal is not intended for IMDs or wearable devices. Rather, it is mainly for stand alone medical devices, such as *vital sign monitor* and *cardiac device*. Thus, there is a need for IDS solutions that consider implanted and wearable devices' special properties, e.g., communication protocols, the physical interaction with human bodies, and limited resources.

Secure communications. IEEE 802.15.6 is the latest BAN standard that provides security services such as authentication and encryption [6].

Location-based controls. Some solutions use the distance between medical devices and their legitimate communicating parties such as programmers and other medical devices. For example, Distance-Bounding protocols rely on the physical distance between communicating devices

so that remote attackers cannot launch attacks remotely. The distance is determined by various techniques such as ultra sound signals, received signal strength (RSS), electrocardiography (ECG) signals, and body-coupled communication (BCC). This technique provides authentication but not authorization, and hence other techniques must be incorporated [151].

Thwarting active and passive attacks. Li et al. [101] proposed the use of Body-Coupled Communication (BCC), were they experimentally investigate the BCC's ability to prevent passive and active attacks against insulin delivery systems for the first time. This type of communication thwarts most the passive and active attacks because of its dependence on the human body as its transmission medium, as opposed to conventional wireless communication where the air is the communication medium that is easily intercepted. When the human body becomes the transmission medium, an attacker needs very close proximity to the patient or even direct body contact, which significantly mitigates the attacks and raises the bar for the attackers. Their work is the first that experimentally investigates the BCC's ability to prevent passive and active attacks against insulin delivery systems.

Shifting security to external wearable devices. Incorporating security into the current IMDs and wearable devices has its own risks and challenges. One of which is the health risk associated with IMDs' surgical extraction from patients in order to update or replace with more secure IMDs. In fact, if we assumed there are no health risks for extracting IMDs, the cryptographic operations required for any secure system are expensive in terms of computational, memory, and battery resources. Therefore, the intuitive solution is to add another device built specifically to add security. Several proposals that deploy some cryptographic and anti-jamming-attack mechanisms utilize external wearable devices to implement such mechanisms. For example, Xu et al. [181] propose *IMDGuard* to defend against jamming and spoofing attacks. In addition, Gollakota et al. [62] propose an external wearable device, the *shield*, to detect and prevent any unauthorized commands sent to an IMD. They evaluated the *shield* on two modern IMDs, i.e., ICD and cardiac resynchronization therapy device (CRT). This device jams any signals initiated by unauthorized party.

Cross-domain solutions. Li et al. [101] proposed the adoption of the *rolling code* encryption mechanism used in RKE in cars. Smart cars and medical devices both share almost the same features in terms of computation limitations, power, and data bandwidth constraints. Therefore, using rolling code encryption should be an effective solution to prevent eavesdropping replay attacks.

Standardization and recommendations. The Food and Drug Administration (FDA) is the leading body in medical device standardization. It has issued a number of standards and guidelines for the manufacturers of medical devices. For example, in 2005, the FDA highlighted that potential vulnerabilities might result from using COTS software equipped with remote access capabilities [55]. Another recommendation was published in 2014 about cybersecurity in medical devices [56]. However, the recommendations are not detailed enough nor mandatory, rather "non-binding recommendations". Therefore, manufacturers have the liberty to choose not to follow them, which certainly would contribute to the production of less secure medical devices.

Leashing vs. unleashing remote functionalities In order to prevent attackers from penetrating networks that make the interaction between remote physicians with patients' devices possible, manufacturers disable remote capabilities from being sent through the network. They only allow remote parties to receive measures and logs, but not send commands. Although this is a good security practice to prevent attackers from sending remote commands, it limits the full utilization of such devices [98]. Therefore, there is a need to strike a balance between security and usability without posing remote threats to patients.

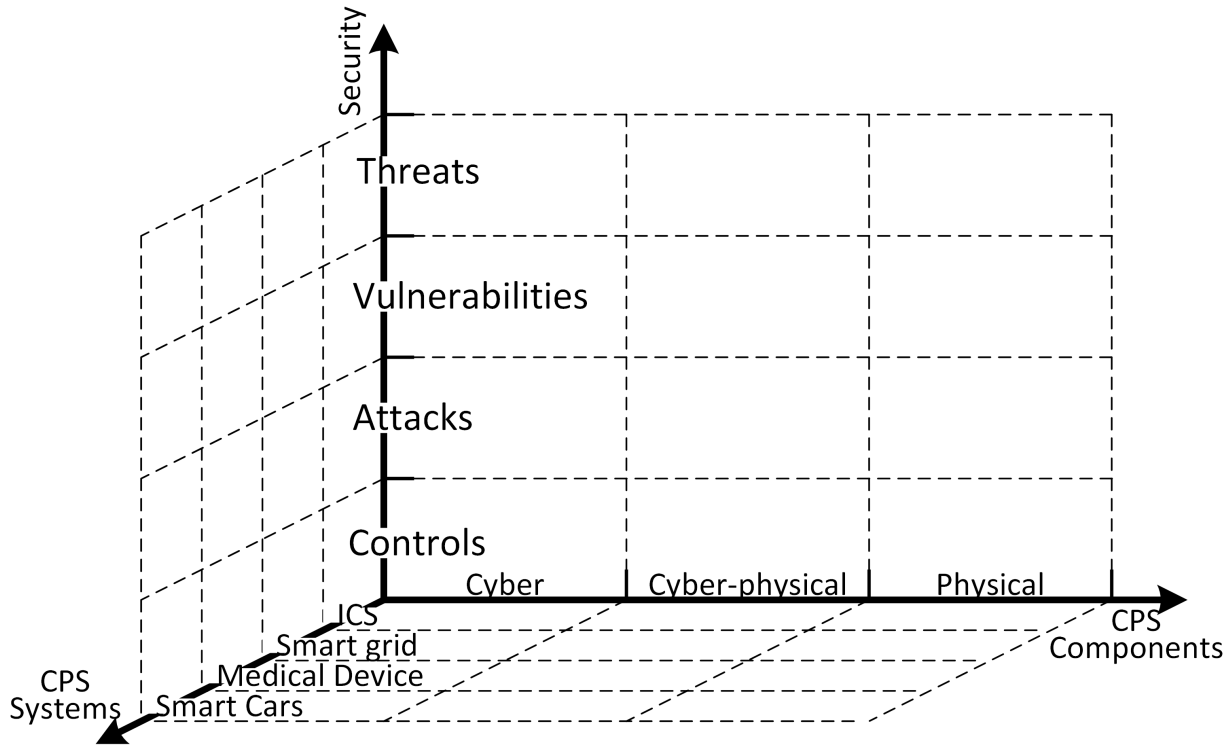


Figure 3.1: PS security framework with three orthogonal coordinates: security, CPS components, and representative CPS systems

Chapter 4

Security in Cars

In this chapter, we follow the same methodology in reviewing security used in Chapter 3. However, because smart cars are the focus of this research, we elaborate and explain in more details.

4.1 Types of Vehicle Communications

Communications related to smart cars can be classified into three types: 1) in-vehicle, 2) vehicle-to-vehicle, and 3) vehicle-to-infrastructure communications. The first type is the most common where internal ECUs can communicate with one another or with external devices. Whereas second type is currently less common, but it embodies future cars such that road and traffic information can be exchanged between cars. The third type is where cars can communicate with traffic lights or road sensors to alarm about risky weathers [177]. In this work, the focus is mainly on the first type, i.e., in-vehicle communications and more particularly on CAN.

4.2 CAN Bus Protocol

Controller Area Network (CAN) is a serial communications protocol used in various applications such as small and large vehicles, ships, planes, and industrial automation. It is even used in some drones, radar systems, and submarines. Its prevalence mainly comes from its low cost, robustness,

and flexibility. CAN protocol provides two services when referring to the IOS networking model: 1) at the physical layer, it allows the transmission of frames as voltages that do not get influenced by interfering magnetic fields and 2) at the data link layer where each frame is formatted in a well-defined format as shown in Fig4.1. At the physical layer, two types of signals are used, low-speed, specified in ISO11519-2, and high-speed, specified in ISO11898. Whereas at the data link layer, the most common specifications on how the protocol should work are Bosch's CAN2.0A and CAN2.0B [15]. After that, a higher level protocol is needed to handle the data contained in the frames and deal with the semantics. Examples of such protocols are CANOpen and Device Net. The latter is common in industrial networks [131].

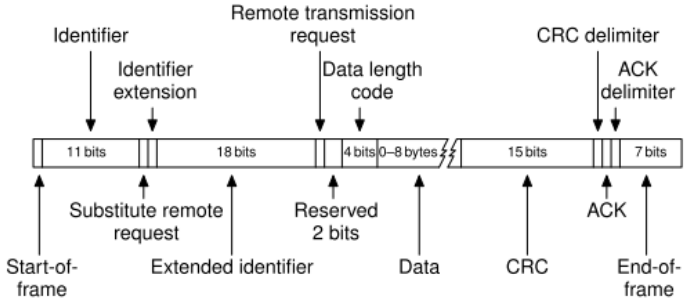


Figure 5. CAN packet structure. Extended frame format is shown. Base frame format is similar.

Figure 4.1: CAN Format from [92]

4.2.1 CAN Subnetworks

Typically, CAN networks are divided into three subnetworks: 1) powertrain, 2) comfort, and 3) infotainment. The powertrain subnetwork consists of ECUs that monitors and control operations related to the engine, brakes, and other critical operations. Whereas the comfort subnetwork consists ECUs that open/close windows, control HVAC, and seats' adjustments, to name a few.

The traffic in CAN bus is divided into two bandwidths: high-speed and low-speed bandwidth. When ECUs from different bandwidth need to intercommunicate, they need to do so through a gateway that transmits the exchanged messages from different traffic. This is realized by BCM (Body Control Module) which acts as a gateway between the two bandwidth speeds in addition to

OSI Layer	CAN Bus	CAN Standard/Specification
Physical	High tolerance to noise and bidirectional communications	ISO11519-2 & ISO11898
Data Link	Error detection & correction, multiple bus masters, priority levels of messages	CAN2.0A & CAN2.0B
Application	Determining the type of services each ECU provides and how ECUs interpret IDs	CANOpen & SAE J1939

Table 4.1: OSI Model VS. CAN bus

other functions such as controlling external lights and providing information to passengers.

In addition to the in-car bus network, smart cars have a number of wireless interfaces that are of two kinds: short-range (Bluetooth, Remote Keyless Entry, Tire Pressure Monitoring System, and RIFD keys) and long-range (cellular channels) [33].

4.2.2 CAN VS. OSI

If we look at CAN bus from an OSI 7-layers' perspective, we find that CAN bus provides services at the physical and data link layers. In addition, it can be controlled and monitored by application-level applications such as CANOpen and SAE J1939.

In order to implement CAN bus communications, we need three components [174]:

1. Physical layer transceiver: to translate CAN frames signals to/from wires or cables.
2. CAN controller: data-link layer implementation that adheres to standards such as ISO 11898.
3. Application level software: to translate the software application data to/from CAN frames.

Table.4.1 shows CAN bus's functionalities provided at the physical and data link layers [174].

4.2.3 Types of ECUs

Classifying ECUs is useful for security evaluation of the network. For example, safety-critical ECUs could be classified as critical group, whereas other ECUs as non-critical group. Regardless of the classification approach, it is useful to distinguish between ECUs in terms of security impact.

ECUs can be classified as *sensors*, *actuators*, and *monitoring and control* ECUs [139].

In [65], the authors introduce *trusted communication groups*, where each group of ECUs that exchange data can do so in a way that data is confidential and ECUs are authenticated. What is important in this section is that the authors classified every group of ECUs that need to exchange data as a group.

ECUs that belong to certain groups can have an ID that can be identified during communications. If an ECU that only accepts frames coming from an ECU in the same group, the ID is checked to decide [171].

Another approach to classify ECUs is based on their potential direct exposure to external connections [172]. For example, the telematics unit and OBD-II port are considered low-trust group, whereas all other ECUs are high-trust. ECUs in the high-trust group share a symmetric key used for generating MACs of sent messages and verifying received messages. Therefore, ECUs residing in the low-trust group cannot send nor receive message containing key-generated messages. HAving only one key for all ECUs in the entire group facilitates key management and storage. However, I think it is too many ECUs to include in the high trust group. It could have been better to only include the cyber-physical ECUs in the high-trust group and have a medium-trust group that could have less security restrictions.

4.2.4 ECUs and CAN

When ECUs send frames at the same time, there is no master node or arbitrator. Instead, each ECU is responsible to check that the bus is available to transmit a frame. A frame has a unique arbitration ID that is not used throughout the network by other ECUS. When two ECUs send a frame at the same time, the frame with the lowest arbitration ID wins and get transmitted. Based on the arbitration ID, the corresponding ECUs distinguishes the frame and accepts it. In addition, all ECUs must transmit at the same data rate, otherwise communications issues will arise.

4.3 Security in Cars

Car manufacturers strive to introduce different sorts of innovative technologies that provide more functionalities and convenience in smart cars. Typically, cars are safe by design, but security, however, is not usually of a great concern. Manufacturers neglect security, which results in its inadequacy at the cost of adding new features. In addition, most of the new features utilize wireless communications and components with physical impact. These two features alone result in most security vulnerabilities and attacks.

4.3.1 Automotive VS. IT Security

Using popular solutions in the IT applications such as encryption, IDS, firewall, Access Control mechanisms is not possible in the automotive space. This is due to a number of fundamental differences between the two spaces. First of all, systems and networks in IT applications are dynamic and in constantly changing environments. On the other hand, automotive systems and networks are almost static during their lifetimes [155].

In addition, automotive systems are time-critical, in contrast to IT systems [178]. Therefore, applying the same time-consuming encryption solutions would not be the best option. Instead, automotive-specific solutions are needed that considers such fundamental difference.

4.3.2 When is a car considered secure?

Wolf [177] has been an automotive security advocate since 2004. He considers car communications to be secure if the following criteria are satisfied:

1. Integrity: altered messages must be detected
2. Authenticity: source message must be legitimate in a verifiable fashion
3. Privacy: the privacy of the car's owner must be preserved. The privacy might be violated by tracking the car though the GPS system. Other privacy concerns are discussed in**??**.

In addition to the aforementioned properties, resilience is another critical property that need to be considered.

4.3.3 ECUs modifications for security

One may suggest that we can just modify ECUs in a way that makes them more secure, vulnerability-free, and attack-resistant. However, this suggestion would be very costly and time-consuming, since it requires countless recalls, major changes in the deployed hardware and software, and alterations in the protocols such that they accommodate the new changes [140].

On the flip side, CAN bus is also difficult to modify. Its format cannot be changed, cannot contain extra information such as authentication codes and any other extra data. Such modification requires introducing changes in the CAN transceiver that are attached to every ECU [154].

4.4 Security Threats

Securing smart cars bears with it various challenges; one of which is understanding the potential threats surrounding them. In addition, given the CPS nature, this understanding becomes of a greater importance [25]. In this section, we aim to tackle this challenge by presenting six examples of potential threats and shed light on them from five angles: *source*, *target*, *motive*, *attack vector*, and *potential consequences* [168].

Criminal hacking. A hacker (source) may target internal ECUs (target) through the car's communication interfaces (vector), and cause a collision or inability to control (consequences) [33].

Cyberwar. A hostile nation or terrorists (source) may target national transportation infrastructure and their commuters (target), through fully compromised cars (vector), to cause large-scale collisions and potential critical injuries (consequences) [33].

Espionage. Intelligence agencies (source) may target individuals (target), through exploiting vulnerabilities in traceable GPS components (vector), to obtain the targets' private location information (consequences) [19, 84, 33].

Driver profiling. Companies (source) may seek to obtain drivers,Äô driving habits (target) through analyzing stored information in ECUs (vector), which is a privacy and confidentiality violation (consequences) [77].

Thieves. Thieves (source) whose motive is stealing (motive) a car, or its exterior or interior parts (target).

Vandals. Vandals (source) aim to damage (motive) a car or its parts, such as anti-theft system(target). Both threats, thieves and vandals, exploit a car's physical exposure (vector) resulting in financial losses (consequence) [19].

4.5 Security Vulnerabilities

In this section, we categorize, review the vulnerabilities, and present in Table 4.2 a summary that shows the relationships between them and their causes.

4.5.1 Causes of Vulnerabilities

4.5.1.1 More connectivity vs. assumed isolation

The wide use of cyber channels, in addition to the car's unreadiness to connecting to the outside world, opened up a new attack vector that manufacturers may not have anticipated. This unreadiness resulted from the previously-correct isolation assumption that ECUs are secure from cyber attacks due to the lack of cyber channels [137]. However, despite the unreadiness, manufacturers have equipped smart cars with new features that improve maintainability, e.g., remote diagnostics, and convenience, e.g., Remote Keyless Entry (RKE) system. Most of such features rely on wireless communication channels. This increased connectivity has opened up new attack vectors as never before, and increased the likelihood of new vulnerabilities. In addition, attacks exploiting cyber vulnerabilities could compromise the safety of the passengers due to the physical impact that some ECUs have [31].

4.5.1.2 Heterogeneity

The heterogeneity of the cars' components along with the integration of COTS components contribute to most of the identified vulnerabilities. In addition, manufacturers integrate many components implemented by third-parties, which makes smart cars even more heterogeneous [45]. The internal details of the integrated components are unknown, and thus they may produce unexpected behavior to the manufacturer. In fact, most of the bugs that led to successful attacks were found at the boundaries of components, where the incorrect assumptions interact.

4.5.1.3 CAN Vulnerabilities

- DoS attacks: CAN is extremely prone to DoS attacks due to the arbitration ID nature. Therefore, an attacker could easily flood the network with messages that have the lowest ID in the network. Such messages will always occupy the network denying other ECUs from receiving expected frames. When ECUs can no longer receive frames, the kind of attack is considered a *starvation attack* [69].
- Messages modification [102]
- Spoofing/injecting new frames/messages

4.5.2 Categorization of Vulnerabilities

. We systematically categorize smart cars' vulnerabilities by the type of CPS aspects a threat may exploit. For example, a vulnerability in the Bluetooth protocol is considered cyber, whereas a vulnerability that exists in the Tire Pressure Monitoring System (TPMS) is cyber-physical due to its interaction with the physical world, i.e., with tires through sensors. Finally, any vulnerability that results from the physical properties of any component is physical such as the exposure of sensors or actuators to different threats.

4.5.2.1 Cyber Vulnerabilities

.

General wireless vulnerabilities. Wireless communication, e.g., Bluetooth, WiFi, cellular, RKE, and TPMS, implies the broadcasting nature of the communication. This makes any wireless communication potentially vulnerable to eavesdropping, spoofing, and Denial-of-Service (DoS) attacks.

Bluetooth vulnerabilities. The Bluetooth is one of the most vulnerable attack vectors in smart cars [31]. When a passenger wants to pair a phone with the car, the only authentication measure is a Personal Identification Number (PIN), prompted by the car's Telematics Control Unit (TCU). This measure is insufficient, and attackers can brute-force the PIN, intercept it, or even inject a false PIN by spoofing the Bluetooth's software. In addition, the Bluetooth connections could expose the car to traceability attacks if an attacker successfully extracts the Bluetooth's Media Access Control (MAC) address, which is unique and potentially traceable [33].

Vulnerabilities in the media player. The media player has the ability to directly connect to the CAN bus. This implies that any vulnerability in the player can affect other ECUs because of this connection. Checkoway et al. [33] identified two vulnerabilities: 1) a malicious specially-crafted CD could affect the media player's ECU and "reflash" it with malicious data, and 2) the media player is vulnerable to an arbitrary code execution, thanks to its ability to parse different media files.

Vulnerabilities in cellular communications. TCUs provide cars with cellular communication channels, among others, including the Bluetooth, and WiFi. Privacy concerns have emerged from using the cellular interface as a tracking tool where both Global Positioning System (GPS) and the microphone are parts of the TCU. This connection reveals the target's whereabouts, or can become a spying tool via eavesdropping on the in-car conversations by exploiting the microphone [33, 31].

Software vulnerabilities. Software is at the heart of every ECU, and smart cars reliance on it has significantly increased. This, in turn, increases the likelihood of software bugs and security vulnerabilities [74]. If a software is vulnerable to, say a malicious code injection, it would expose

the car to various attacks depending on the injected software.

4.5.2.2 Cyber-Physical Vulnerabilities

. *Communication protocols*. Smart cars are vulnerable to numerous kinds of malicious attacks since security was not taken into consideration when manufacturers designed the cars [95]. In-vehicle communication protocols, such as CAN and FIN, suffer from lack of encryption, authentication and authorization. Here we review the vulnerabilities in the most common protocol, that is the CAN bus. Because the CAN bus links cyber with physical components, it is a cyber-physical communication protocol. The CAN bus protocol has a number of vulnerabilities that contribute to most of the attacks on smart cars. Many vulnerabilities in the CAN bus can cause a significant number of attacks. For example, CAN bus protocol lacks critical security properties such as encryption, authentication, and has weak authorization and network separation. In addition, the protocol's broadcast nature increases the likelihood of DoS attacks [92]. Another security property, common in computer security literature, is non-repudiation, where there is no way to identify the sender of a particular message [77]. Clearly, these vulnerabilities, especially the lack of encryption and authentication, result from the isolation assumption discussed earlier.

Comfort ECUs. More advanced features are continuously added to ECUs to improve safety and comfort. For example, ECUs like *Adaptive Cruise Control* (ACC), *Lane Keep Assist*, *Collision Prevention* provide safety, where *Comfort Park Assist*, RKE are examples of ECUs that provide comfort. Although these components have a great impact on improving the driving experience in terms of safety and comfort, they pose a new type of attacks, that is cyber-physical attacks. These components are part of the CAN bus, and there is a threat of attacking them and compromising their expected functions by exploiting directly their vulnerabilities, or vulnerabilities in other ECUs residing in the same network [31]. We take one example of these ECUs and investigate the possible vulnerabilities that could cause cyber-physical attacks. ACC is the next generation of *cruise control* which used to be, and is still in many cars, manual and quite primitive. That is the driver presses the "set" button to maintain a particular speed the car has already reached, and increases or decreases

the speed by pressing plus or minus buttons. This kind of cruise control is completely manual and the driver's intervention is required at all times. On the other hand, ACC is the new generation of cruise control, and it has new features to give drivers more safety and comfort with minimal manual intervention. The ACC has the ability to detect the speed of the cars ahead and "adaptively" reduce the current speed to maintain a safe distance between cars. The detection is accomplished using laser or radar sensors. A well-equipped attacker might be able to interrupt those sensors' operations by either introducing noise so they cause the ACC to reduce or increase the speed unexpectedly, or spoofing the sensors such that the ACC reduce/increase the speed in a way that could cause collisions. The threat here is the ability to tamper with the sensors externally, or with the ACC ECU itself internally, possibly through other ECUs that are potentially vulnerable to remote attacks such as TPMS or RKE.

TPMS tracking, spoofing and eavesdropping. TPMS is vulnerable to eavesdropping and spoofing due to the lack of encryption [84]. In addition, tracing a car is possible by exploiting the unique ID in the TPMS communications.

TPMS replay attack. As a consequence of the eavesdropping vulnerability, attackers can launch replay attacks so that they record legitimate TPMS signals containing a safe air pressure in a particular tire. Then the attackers would physically reduce the air pressure to potentially cause an accident without the driver noticing the unsafe tire's measures.

Replay attacks. The lack of encryption and/or authentication results in replay attack vulnerabilities. Sometimes encryption solutions are in place, but are weakly implemented. This type of attack relies on recording messages intercepted from the network, and retransmitting them at later times to cause the same effect in the legitimate scenarios. For example, when a driver presses the unlock button in the key fob, an attacker might be able to record the communications between the key fob and the receiving component in the car. Then the same message is played back to unlock the car. This attack is applicable RKE to achieve unauthorized access to the car, and also to the TPMS to send old tire pressure data, for example.

X-by-wire. An emerging trend in smart cars is the "X-by-wire", that aims to gradually re-

place the mechanically-controlled components in the car, such as the steering wheel and the brake pedal, by electronic or electro-mechanical components that would make drivers control the relevant functionality by the press of a button. Steer-, Drive-, Brake-, Shift-, and Throttle-by-wire are all examples of this trend [7]. This implies new opportunities for attackers to launch cyber-physical attacks exploiting such new functionalities. However, this technology relies on FlexRay communication protocol, which is more advanced than CAN bus in terms of speed and safety features. However, it is more costly and less likely to widespread in the near future [7]. Once the x-by-wire technology is widely deployed, we should expect cyber-physical attacks.

4.5.2.3 Physical vulnerabilities

Smart cars, if not physically protected, can be vulnerable to numerous attacks that do not necessarily require cyber-capabilities. For example, TPMS external parts could be destroyed resulting in a DoS attack such that the TPMS sensors cannot send the tires' air pressure to the designated ECU. In addition, exposing the car to any kind of physical access is another vulnerability that could cause critical attacks. For example, a mechanic can get physical access to car's internal parts through the OBD-II port without the need for sophisticated attacks [3]. Furthermore, some external parts, such as the exterior mirrors, can be used to access critical components in the car's CAN bus [77].

Vulnerability	Description	Aspect [1]			Cause [2]		Ref
		C	CP	P	Con	H	
TPMS interception	Easy interception on TPMS's communications		CP		C		[84]
Tracking	A traceability vulnerability might be exploited for tracking	C			Con		[84]
Bluetooth	Authentication flaw resulting in spoofing and packets injection		CP		Con		[84]
Replay Attacks	Retransmission of recorded commands/messages captured wirelessly or by physical access		CP		Con	H	[30, 77, 92]
Software-based packet injection	Software vulnerabilities in communication technologies, e.g., WiFi and cellular, lead to packet injection	C			Con		[33]
Insecure CAN bus	CAN bus lacks basic security measures: encryption, authentication, and access control		CP		Con		[3, 92, 137]
Exploitation of Media Player	Specially-crafted code on a CD could cause inject malicious code		CP			H	[77]
Physical Tampering	Physically unprotected components facilitates compromising ECUs			P	Con		[77]

Table 4.2: Summary of Vulnerabilities

4.6 Security Attacks

We reviewed about two dozen papers that discuss the security of smart cars. Most of the work done is abstract, theoretical, or simulation-based. Only a few present results of actual experiments on real cars [77, 92, 33, 30].

In order for a successful attack on a car, an attacker needs to gain access to the internal network physically, through the OBD-II port, media player, or USB ports, or wirelessly, through the Bluetooth or cellular interfaces. Once an attacker gets into the car's internal network, a plethora of attacks opportunities are open.

4.6.1 Cyber Attacks

4.6.1.1 DoS

DoS attacks can take on different forms whose impacts vary in safety-criticality. Koscher et al. [92] disabled CAN communication from and to the Body Control Module (BCM) which resulted in a sudden drop from 40 to 0 MPH on the speedometer. In addition, this attack also resulted in freezing the whole Instrument Panel Cluster (IPC) in its current state. For example, if the speedometer was at 60 MPH before the attack, and the driver increases the speed, there will be no change in the speedometer.

4.6.1.2 False Data Injection (FDI)

An example of this attack is displaying a false speed on the speedometer. An attacker would first intercept the actual speed update packet sent by the BCM, and then transmit a modified packet that had the false speed [92]. In addition, an attacker can forge the real status of the airbag system to appear healthy, even if the airbag had malfunctioned or was removed [77].

4.6.1.3 Privacy invasion

Checkoway et al. were able to exploit the cellular interface in the TCU and eavesdropped on in-car conversations [33]. In addition, a report published by a U.S. senator reveals that car manufacturers store a large amount of private information such as driving history and cars' performance [110].

4.6.2 Cyber-Physical Attacks

.

DoS. Hoppe et al. [77] demonstrated other forms of DoS attacks. One where the attack prevents passengers from closing opened windows. Another is to disable the warning lights. The authors also performed another DoS attack on the theft alarming system, such that it will not go off during a burglary [77].

4.6.2.1 Malware injection via Bluetooth

Checkoway et al. [33], conducted an attack that exploits compromised devices connected to the car through Bluetooth. The authors assume the attacker's ability to first compromise a connected(paired) device with the car via Bluetooth, and then launch the attack exploiting the connectivity to the other ECUs. This was realized by implementing a Trojan Horse that captures the Bluetooth connections and then sends a malicious payload to the TCU. Then once the TCU is compromised, that attacker can communicate with safety-critical ECUs, such as the Electronic Brake Control Module (EBCM) that controls brakes.

4.6.2.2 Malware injection via cellular network

The cellular channel in the TCU is exploitable and vulnerable to malware injection attacks. The attack was realized by calling the target car and injecting the payload by playing an MP3 file [33].

4.6.2.3 Malware injection via OBD-II

Attackers need physical access to the OBD-II port to launch other attacks that rely on malware injection. Examples are a DoS attack on the ECUs of the window buttons or the warning lights, and the spoofing of the car's owner by not showing the nonexistent airbags that thieves have stolen [77].

4.6.2.4 Bypassing the gateway via OBD-II

Hoppe et al. [76] presented an exemplary attack on an in-vehicle gateway. The attack shows how it is possible to expose some potentially private data from isolated subnetworks and extract them via a USB-to-OBD cable plugged in the OBD-II port during a diagnostic session. The OBD-II port is connected directly to the gateway by which other subnetworks are isolated for functionality purposes rather than security, arguably [33].

4.6.2.5 Packet injection

This attack requires previous access to the CAN bus. Once an attacker gets into the network, physically or wirelessly, a large number of attacks are possible. For example, through the OBD-II port, it is possible to increase the engine's Revolutions Per Minute (RPM), disturb the engine's timing, disable the engine's cylinders, and disable the engine itself. In addition, attacks on brakes are also possible by injecting random packets to the EBCM such that it locks and releases the brakes resulting in unsafe driving experiences [92]. All of these attacks are possible by exploiting the trusted diagnostic service and bypassing the weak access control mechanism.

4.6.2.6 Replay attacks

This attack requires two steps: 1) intercepting the CAN bus traffic when certain functions are activated, and 2) retransmitting the observed packet to reactivate the same function. In [92], the authors successfully were able to disable the car's interior and exterior lights by sending previously-eavesdropped packets.

Name	IE	I	AE	Impact	Method	Precondition	Ref
DoS 1	BCM	Sudden drop in speedometer	IPC	Frozen IPC and failure in turning car on/off	Disabling communication to/from BCM	Physical access to CAN bus	[92]
DoS 2	Windows	Window closing failure	Windows control buttons	Discomfort and frustration	Reverse engineering and fuzzing	Physical access	[92, 77]
Malware injection 1	Bluetooth ECU	Ability to connect to other ECUs	Safety-critical ECU with cyber-phyiscal capabilities	Loss of control and potentially collision	Malware injection to other ECUs via Bluetooth's ECU	Vulnerability in Bluetooth pairing mechanism	[33]
Malware injection 2	Telematics unit cellular interface	Malware injection	Loss of control over other ECUs	Remote control and cyber-physical attacks	Call car and inject malware payload	Knowledge of car's specifics	[33]
Malware injection 3	An ECU	ECU becomes an attack vector to inject undesired packets	CAN bus traffic becomes vulnerable to malicious packets	Other ECU behaves undesirably affecting cyber-physical components	Transmit malware in CAN packets	Physical access or vulnerable wireless interfaces	[92, 77]
Packets injection	Varies, depending on target ECU	Varies	Other ECUs and physical components	False data injection, loss of control, DoS, and safety-critical consequences	Compromised ECUs inject packets	Malware injection	[92, 77]
Replay Attack	Lights ECUs	Lights turned off	Driver, passengers, and surrounding cars	Safety-critical situation	Eavesdrop and retransmit legitimate commands	Access to CAN bus network	[92]
Car's spying	TCU	Gain control of car	All other ECUs can be remotely controlled	Stealthily turn on car's microphone	Call the car	Buffer overflow vulnerability and flaw in authentication protocol	[33]
Relay attack	RKE system	Open, and start a car without owner's knowledge	Target car	Theft and unauthorized access	Capture and relay LF beacon signals from car to key fob, and relay resulting UHF signal from key fob to car	Attacker needs relaying tools e.g., antennas and amplifiers among other devices	[59]

Table 4.3: Smart Cars Cyber-Physical attacks

4.6.3 Physical Attacks

Relay attacks. This kind of attacks targets the RKE, where an attacker relays the communications between the car and its key fob. The attack exploits the periodical Low Frequency (LF) beacon signal the car sends to detect if the key fob is in close range. The attacker captures and relays it, using an antenna, to the relatively far key fob, which is most likely in the car's owner pocket. The key fob gets activated by the relayed signal and sends an "open" Ultra High Frequency (UHF) signal to open the car. Once the car gets opened, the same attack is repeated from inside the car to start it. The attack was implemented successfully on ten different cars from eight manufacturers. In addition, it evades cryptographic measures because it targets communications at the physical layer [59].

4.7 Security Controls

Generally speaking, security solutions for cars can be classified into 2 types: 1)internal and 2)external [7]. On the one hand, the first type focus on security attacks that may result from in-car communications over different bus protocols, such as CAN, LIN, and Flexray. On the other hand, external security solutions focus on approaches that aim to prevent attacks originating from outside of the car. For example, attacks that exploit wireless vulnerabilities in Bluetooth and cellular communications need this types of protection to be prevented. Internal security include cryptographic solutions used for authentication, confidentiality, and integrity. Also, IDS for detecting certain behaviors or anomalies in the system, and firewalls to prevent them from escalating. Whereas external protection techniques should aim at protecting all channels that expose the car to the outside world.

Koscher et al. [92] state the importance of only trusting the trustworthy ECUs, and revoking the trust from those ECUs that should not be blindly trusted.

4.7.1 Defending external threats

Han et al. [68] propose an authentication mechanism to authenticate untrusted mobile devices that need to connect to a car. They assume a secure in-vehicle communication, and the need for security is between the network's gateway and external devices. The gateway is the only means of communication to the car. The gateway could be wired, e.g., USB, or wireless, e.g., WiFi or Bluetooth. Not only does the gateway authenticates any external devices, it also authenticates its data requests thereafter.

Bouard et al. [17] propose an IP-based approach that deploy a proxy to prevent attacks launched from mobile devices. The approach is more suitable for next-generation cars equipped with IP-based capabilities.

Woo et al. [178] propose an encryption mechanism that aims to secure the communications between external devices and the in-vehicle network. In particular, they propose a key management techniques to realize the secure communications with external devices.

4.7.2 Unimplemented promising controls

A number of security controls have been proposed to secure the in-car network, most of which have not been implemented. For example, Wolf et al. [3] proposed three controls that would secure the bus network: authentication gateway, encryption, and firewalls. In addition, Larson and Nilsson [95] call for redesigning security in cars, and propose embracing of the *defense-in-depth* security paradigm, i.e, prevention, detection, deflection, countermeasures, and recovery.

4.7.3 Software security

It is critical to update ECUs' software securely. Compromised software can lead to countless attacks. In addition, a new trend that would open up a new attack vector that is remote updating for software. Therefore, different efforts have been proposed to ensure software security.

The hardware modules such as HSM and SHE can be used as a trust root that validates the code in ECUs [7].

Isolating less-trusted code from affecting the bus can prevent many remote attacks. This idea would have prevented a remote attack that exploited a vulnerability in the wireless communication protocol such as in [32]. In [22], Cankaya et al. proposed how the idea of isolation can realized by utilizing a hypervisor, such as XtrantuM, to create different virtual machines.

4.7.4 Heterogeneity of components

It is difficult to suggest that all components produced by different OEMs should be replaced by components produced only by cars' manufacturers. This might be an impractical solution given the complexity and highly skilled specialty involved in designing different components that are integrated with cars. Instead, both parties must be in accord in terms of security requirements, assessment and testing.

4.7.5 Cryptography

The use of cryptography provides a number of security properties such as confidentiality, integrity, and authentication. However, cryptography mechanisms are relatively computationally-expensive in the computationally-limited environment. Thus, the deployment of efficient solutions is vital. Wolf and Gendrullis [176] and Escherich et al. [51] propose hardware-based solutions that are designed specifically for cars' security. Wolf and Gendrullis [176] designed and implemented the Hardware Security Module (HSM). They show its applicability to secure communications of ECUs withing a car, or even in Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. Escherich et al. [51] presented the Secure Hardware Extension (SHE), a standard for adding security properties, such as secret keys' protection and secure boot, to ECUs.

The use of cryptography would be the ultimate solution for all security problems in cars. ECUs and massages can be authenticated and communications get encrypted. The problem though with cryptography is that it is computationally-expensive especially in the automotive environments.

ECUs have limited processing power. Therefore, traditional cryptographic solutions are not possible.

Various cryptography-based approaches have been proposed to achieve authentication and confidentiality.

Herrewege et al. [171] proposed *CANAuth*, an authentication protocol that provides message authentication, replay attack resistance, group keys, and backward compatibility. The group key is the ability for a group of ECU to share a key for authentication operations. The protocol relies on sending authentication data through an out-of-band channel. This ensures that the normal CAN network is not interrupted by any overwhelming authentication frames, i.e, backward compatible protocol. Such mechanism is possible by the use of CAN+ protocol presented in [187]. One possible drawback of the CANAuth is that it is readily compatible with current CAN networks without having CAN+ capability in the CAN controllers.

Another proposal in [172] where the authors proposed an authentication mechanism that relies on symmetric keys to encrypt and decrypt messages in the same group or subnetwork. A shared symmetric key is stored in every ECU of a group. For ECUs to be identified, a byte-sized ID is assigned to each one, which is written to ECUs' flash memory during a diagnostic session along with the group's symmetric key. This can accommodate up to 256 ECUs. If a new ECU is added to the a group, no more keys are needed, since all ECUs share the same key. As the symmetric key provides message authentication service for ECUs, another technique aims at thwarting replay attacks. Usually replay attacks are prevented by time stamping packets. However, since ECUs do not have clocks, "session numbers" are introduced. To uniquely identify a CAN message, a *session number* and *message counter* are used. When ECUs communicate, an ECU broadcast/send 2 messages: data message and authentication message. The data message is the normal message that ECUs usually send, where the authentication message is where the node ID, message counter, and the message authentication code (MAC) are included. This proposal needs to be implemented between ECU's CAN interfaces and the application layer such as CANOpen.

Regardless of cryptography adopted to encrypt and authenticate frames in a CAN network, the

network will be vulnerable to DoS attacks as long as frame's IDs are sent in clear text and an attacker can see them. By seeing them, the attacker could simply flood the network with high-priority IDs that prevent legitimate frames from being sent. Therefore, Han et al. [69] propose a new protocol, Anonymization for CAN (IA-CAN), that aims to hide frames' IDs through an anonymization mechanism. This will prevent any unauthorized parties from eavesdropping on the traffic and learn the IDs. The IA-CAN anonymizes IDs in the sense that they have no meaning for an eavesdropper. With some reverse engineering skills and patience, an attacker can infer CAN IDs and then launch various attacks. IA-CAN prevents the inference because the IDs are changed randomly, from the attacker's view, so that only legitimate ECUs can send and receive frames with certain IDs.

Ansari et. al. [12], propose an approach to detect malicious frames sent by malicious ECUs. Their proposal does not need any modification on the currently widely used CAN protocol. Instead, they extend the CRC mechanism to include detection of malicious frames in addition to its main function, i.e., error detection.

The authors use stream cipher (RC4) to encrypt either the data and CRC fields, or only the data field. With every transmission across the CAN bus, all legitimate ECUs will update their secret keys. Upon a frame's arrival to an ECU, the ECU decrypts the encrypted field (data and CRC or data only). If the decryption results in correct data and CRC, the frame is accepted, otherwise, it gets rejected.

In the proposal, every frame with the expected ID will be received, decrypted, and then verified. This poses a DoS attack to ECUs. For example, an attacker could simply flood the network with frames that have certain IDs where an ECU or more expect. Then each one will receive the malicious frame and eventually drops it. However, flooding the network with a large number of malicious frames could result in DoS attacks and rendering ECUs unavailable.

4.7.5.1 Hardware-Based Cryptography

The most efficient way to implement cryptographic mechanisms in in-vehicles' networks is through hardware. Efficiency is key in these networks given the real-time nature and the limited power capabilities.

There have been a number of proposals that aim to introduce hardware-based cryptography in in-vehicle networks. For example, Wolf and Gendrullis [176] designed and implemented the Hardware Security Module (HSM). The module is designed to provide cryptographic primitives. Depending on the desired security services, the HSM comes into three variants: 1)full, for V2V communications, 2)medium, for in-vehicle communications and 3)light, for ECUs and sensors/actuators communications.

Escherich et al. proposed another hardware-based solution, called Secure Hardware Extension (SHE). It is a mainly built for protecting cryptographic key from software attacks [51]. It does not provide as many security features as the HSM.

Although HSM and SHE seem very promising and reliable, they require a costly re-design of the existing common architectures that OEMs use. That is, an ECU that is needed to be protected has to be extended to realize such solutions. In addition, the aim of such solutions is to replace the existing ECUs, which is very costly and time consuming. Our proposal aims to maintain existing ECUs and implement security in a firewall/gateway.

One might suggest using TPM for car. However, TPM is expensive for cars' microcontrollers and most likely will introduce latency. Cars cannot always connect with the TPMs' remote servers for attestation [139]. However, Oguma et al. [139] propose an TPM-like alternative for cars. They assume the availability of high-resourced ECUs, called *masters*. A masters verifies the authenticity of other ECUs using Key Predistribution System (KPS) and then the ECUs communicate securely by encrypting their frames using the predistributed keys.

4.7.5.2 Key distribution techniques

In the cryptography-based approaches, keys are essential for any security service. Therefore, distributing keys in a secure way is imperative. Here we review different techniques applied for key distribution in CAN networks.

One of the probably first proposals of applying cryptography to automotive networks was in [3]. Wolf et al. show an example of using symmetric and asymmetric keys. A central gateway interconnects different subnetworks. Each ECU has a symmetric subnetwork key, its own public and secret key, and the gateway's public key. The gateway holds the symmetric keys of every subnetwork, and hence provide an efficient encrypted communications between ECUs and the gateway. However, the authors do not explain how such approach can be implemented in an automotive network, given the limited capabilities and computational power.

Oguma [139] proposed a technique called "key predistribution system" that the author claims has a advantage over public key system in automotive applications. Because the number of ECUs is limited, the number of keys is also limited to the maximum number of possible ECUs. This provides the efficiency of symmetric key encryption.

4.7.6 Redefining trust

Koscher et al. [92] suggested two trust-related controls that would have prevented most, if not all, of their attacks. One in revoking trust from arbitrary ECUs so they cannot be able to perform diagnostic and reflashing operations. The other is that ECUs with diagnostic and reflashing capabilities must be authorized and authenticated before performing these tasks.

4.7.7 Restricted critical commands

A physical access is required to the car before any “dangerous" commands can be issued [92]. Although this could be an effective control, the term dangerous is relative, and its interpretation

varies from manufacturer to another, such that seemingly-benign commands could result in serious attacks. If manufacturers decide to restrict the amount of commands that require physical access, the flexibility and convenience will be affected. Therefore, we need solutions that consider all possible attacks resulting from critical or benign commands, while maintaining the existing flexibility.

4.7.8 Bluetooth

Bluetooth connections between devices and cars can be exploited to launch different attacks such as compromising the TCU and consequently other ECUs [33]. Cars need an additional security layer to defend against Bluetooth-dependent attacks. Dardanelli et al. [47] show the applicability of their proposed security layer to protect against smartphone-initiated Bluetooth attacks, with little impact on performance. Although their proposal was tested on a two-wheeled vehicle, it should also be applicable to cars.

4.7.9 Firewalls

In the IT security applications, firewalls rely on blocking certain port numbers or IP addresses. This is possible because of the IP addressing paradigm. However, specifying addresses is not possible in the automotive context because of the lack of addressing scheme in CAN. It is possible to block frames based on their IDs or other parameters such as data field length and data contents.

Wolf et al. [3] suggest implementing firewalls to complement the work of gateways. If a gateway is equipped with cryptographic services such as MACs and certificates, some rules can be implemented in a firewall to allow or deny certain ECUs to and from doing something. This will ensure that only legitimate ECUs can do certain actions. The firewall should also prevent diagnostic messages from being sent from any ECU during normal operation of the car.

A firewall is needed to monitor traffic and inspect frames. For traffic monitoring, the firewall monitors the traffic patterns and directions and prevents frames from reaching ECUs that they are

not supposed to. For frames' inspection, as frames' IDs are the main parameter that determines whether an ID should be forwarded to networks and be accepted by ECUs, the firewall should inspects more than IDs. Data field should be inspected and and validated against malicious contents. In addition, situational properties, such as the car's status (drive, reverse, or park), need to be inspected [164].

The topology of the network is critical for an effective firewall. Star topology that separates critical from non-critical ECUs is ideal. However, it could introduce a single point of failure.

4.7.10 Intrusion Detection Systems (IDS)

Generally speaking, IDS are usually classified as network-based and host-based. In the network-based IDS, the network traffic is analyzed and upon which, attacks are detected. Whereas in the host-based IDS, the host analyzes logs that are gathered from the running applications and the operating system. Usually host-based IDS detect insiders abusing privileges, whereas network-based IDS detects outsiders who aim at exploiting vulnerabilities in the system [48]. The detection mechanism varies, and could be performed either in a signature-based or anomaly-based fashion. For the signature-based, the traffic or activities are compared with a predefined patterns that represent attacks, or undesired activities. For the anomaly-based detection, traffic or activities are compared with normal or desired patterns, and deviations are detected.

4.7.10.1 Automotive VS. IT IDS

Patterns of anomalies and signatures are usually updated in the IT IDS. In automotive, the updating process is not a practical solution especially for the low-cost cars. This requires a regular connections to a database at the OEM location. Apart from the expected cost, this connection introduces a new attack vector [154].

4.7.10.2 Centralized VS. Distributed IDS

On the one hand, central IDS ensures a global view of a given network. On the other hand, ECUs' internal details are missing. Whereas a distributed IDS could be able to analyze internal details of ECUs as an IDS is installed in every ECU. This however requires a substantially high cost to have an IDS in each ECU. Therefore, striking a balance between the two types would be an ideal solution [77]. For example, instead of having one IDS for every single ECU, each subnetwork can have an IDS that, in turn, is connected to a central IDS.

4.7.10.3 Automotive IDS

There have been a number of IDS proposals in the automotive space. They range from signature- and anomaly-based detection to intrusion prevention systems. Here we review briefly each type and show what differs from our work.

Signature-Based Detection:

Larson et al. [96] proposed a host-based IDS such that a detector is embedded in every ECU. The authors chose this approach because a central IDS will not be able to distinguish sources and destinations of frames due to the lack of addressing in CAN. In addition, the detection relies on detecting deviations from the the protocol's specifications. In particular, two parameters are derive from the CANOpen specifications: 1)security specifications from the protocol stack to detect network activities and 2)ECU expected behavior from the object directory to detected attacks at the ECU-level. The behavior of ECUs is derived from CANOpen specifications, and any activities that differ from the specifications are considered attacks, and therefore detected.

One drawback in the proposed detection mechanism is the reliance on protocol's specifications assuming that they are carefully followed and implemented by manufacturers. Koscher et al. stated that deviations from the standards are common [92]. Therefore, such assumption limits the adoption of the proposal because of the proprietorial preference by manufacturers when it comes to use standards. Another assumption that might limit the effectiveness of their detection mechanism is that an attacker can control the gateway or other ECUs except the source and destination ECUs. If

an attcker is able to control all these ECUs, why not control the source or destination ECUs?

Muter et al. [128] designed another signature-based IDS that deploys eight sensors for collecting data for various purposes. Each sensor reports suspicious patterns that match defined ones in the IDS's database. Once a frame, or a sequence of frames, is detected, an alarm is raised. Each sensor checks for formality, location, range, frequency, correlation, protocol, plausibility, and consistency, respectively.

Schweppe and Roudier [154] propose an approach that is designed to prevent operating systems' attacks such as buffer overflows, and unauthorized access to other ECUs' functionalities or in-vehicle data. The authors use a binary tainting tool for detecting malicious instructions originating from unauthorized parties. For example, the approach prevents attacks that leverage the TPMS connection to access to the network and send malicious commands to other ECUs. The solution is not designed for current cars, but rather for next generation cars as needs periodic updates and high computation resources.

Miller and Valasek [31] demonstrated a proof-of-concept low-cost attack detection system that detects anomalies in the CAN network, and the great opportunities for implementing such a system at low cost and no manufacturing overhead. However, regardless of the low-cost and ease of application of their approach, when their detection mechanism detects an attack, it shuts down the whole network. We believe that shutting down the network is an overreaction and some safety-critical consequences could result.

Ling and Feng proposed an IDS that detects attacks that exploits two known vulnerabilities in CAN: 1) DoS and 2) error flags spoofing [102]. For the DoS attack, an attacker would flood the network with a frame whose ID has the highest priority, resulting in preventing frames with lower priority IDs from being transmitted. Whereas the error flag spoofing occurs when an attack crafts frames that looks like CAN's error flags which results in interruption of the normal sequence of CAN frames. Their detection mechanism keeps track of two types of frames' ID, known and unknown. For each kind, a counter and a flag are defined in addition to a threshold of the normal frequency of such type. When an ID exceeds its threshold, an alarm is created signifying the

detection of either a DoS or error flag spoofing attack. It is not clear how and why the authors choose certain IDs and on what basis a threshold is determined. Although the approach seems promising, it needs further explanation and justifications.

Anomaly-Based Detection:

Hoppe et al. proposed an anomaly-based IDS [77], which is network-based IDS monitoring the network and detecting any deviations from the normal traffic. The authors proposed three patterns that each of which signifies an attack. The patterns are: 1) increased message frequency such that when an attacker injects messages to the network, the number of usual message is observed as an indicator of an attack, 2) obvious misuse of message IDs, and 3) low-level communication characteristics.

Otsuka et al. [140] proposed a novel approach, "delayed-deciosn cycle detection", that needs no modifications on the current ECUs. Their anomaly-based approach relies on analyzing normal frequency of periodic messages that appear in cycles in a CAN network. If a message arrives at the gateway in a higher frequency than what is considered normal, the message is either dropped, if it is unauthorized, or forwarded later, if it is authorized. The approach is implemented in a gateway, and all other ECUs are intact. Comparing with similar approaches, this proposal has the lowest false positives and negatives. Although this approach is especially useful for detecting attacks that resemble periodical messages, it is effective with other attacks.

Seifert and Obermaisser [155] proposed a behavior-based approach, implemented in a *security gateway*, that focuses on ECUs' communications, rather than the specific protocols. They proposed a generic approach that derives rules from the relations between ECUs and timing properties of the exchanged messages. The approach is abstract enough to be applicable to any protocol, including CAN and Flexray.

Matsumoto et al. [112] propose an authorized frames' detection and prevention mechanism. The precondition is that no two ECUs can use the same ID at any given time. In other words, IDs are uniquely used by ECUs. Also, each ECU monitors the traffic consistently looking for frames that have the same ID of its own. If a detection occurs, the ECU transmits an *error frame* that

will have a higher priority and will override the unauthorized frame. Therefore, not only does the mechanism detects, it also prevents authorized frames from completing their transmission.

Müter and Asaj [127] propose a novel entropy-based IDS. An entropy is a measurement of the level of coincidence a dataset contains. The higher the entropy is, the more coincidences in the dataset, and vice versa. In the context of automotive networks, traffic behavior is mostly predictable and static, unlike traditional IT networks' traffic. Therefore, the authors propose an IDS that detects when the networks' entropy increases, which is a signal of deviation from the dataset representing normal traffic, indicating an attack.

Hybrid IDS:

Some IDS can be signature-based, anomaly-based, or both. Schweppe [154] propose a distributed IDS framework that attach distributed *sensors* to ECUs that might be used as an attack vector such as ECUs with USB ports or bluetooth connections. These sensors report to a "master node" to evaluate their data. This approach can be signature-based or anomaly-based depending on the designers' choice. The evaluator can also be stateless or stateful. Once an intrusion is detected, the evaluator creates an *event* which, in turn, triggers an *action*. The actions include modifications of the network's configurations and alerting the driver of an ongoing attack.

Intrusion Reaction: When an intrusion is detected, an alert could be sent to the driver. Hoppe et al. [75] proposed a three-levels alerting system that increases the reaction according to the attack's severity.

Intrusion Prevention: According to [90], there has not been an intrusion prevention system proposed. One reason for that is arguably safety-critical requirements prevent such active response that has physical consequences [75].

Chapter 5

Preliminaries

5.1 Introduction

CAN protocol has a special approach to handle addresses of senders/recipients of messages. There is not an IP-like addressing. Instead, the protocol uses an 11-bit field, and could be extended to 29 bits, that is called arbitration ID. Each CAN frame starts with an ID that determines the frame's purpose and priority. The purpose is what the frame means to ECUs, whereas the priority determines the frame's ability to win arbitration over using the network when another frame collides with it in the bus. The frame with the lower value ID will get higher priority and therefore dominate the network. All frames must have unique IDs in order to avoid errors caused by two frames transmitting simultaneously as a result of dominating the bus because of their identical IDs. Fig.**??** shows a simplified CAN frame that has an 11-bit arbitration ID.

A frame gets the right to access the bus if the ID has the lowest value. When a frame occupies the bus, all ECUs receive it and only the interested ones accept it. An ECU only accepts frames that it is configured to accept by recognizing their IDs. The choice of arbitration IDs is propriety and differs from an Original Equipment Manufacturer (OEM) to another. In addition, an arbitration ID is not the sender or the recipient's address. Rather, it is only an indicator of a message's contents and purpose.

5.2 CAN Node

Each ECU is composed of three components: 1) a microcontroller (MCU), 2) a CAN controller, and 3) a CAN transceiver. The microcontroller is responsible for high-level functions such as calculating the speed, sending an airbag command, warning the driver about the oil pressure. Whereas the CAN controller is responsible for encoding/decoding outgoing/incoming CAN frames to/from the CAN bus. The CAN transceiver converts the frames to/from physical-level bits. A common assumption in CAN communications is that all nodes adhere to CAN standards as shown in Fig. 5.1. This assumption is going to fall short in Chapter 6.

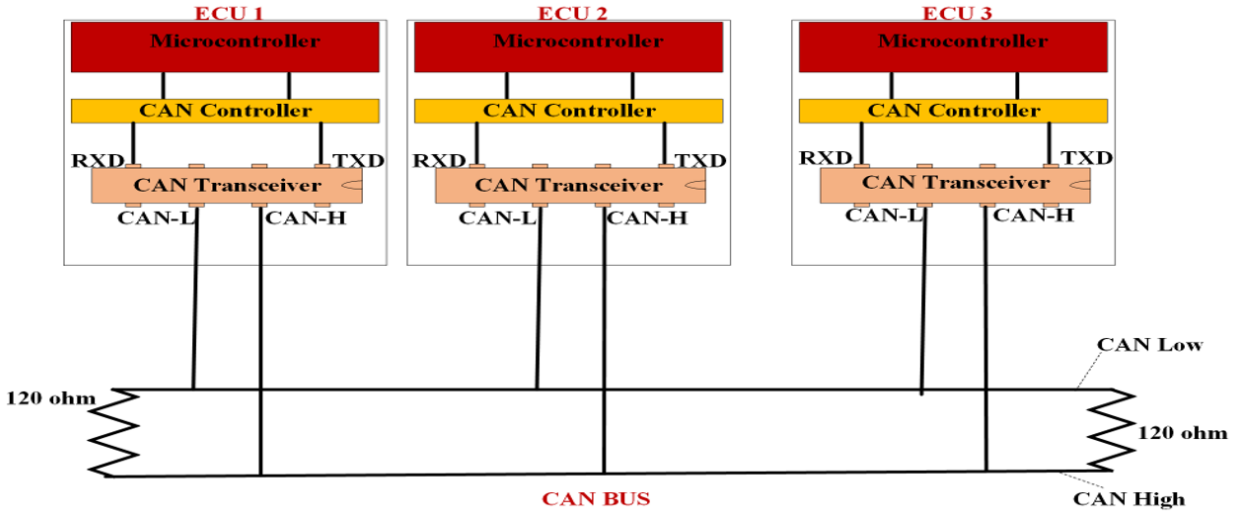


Figure 5.1: Components in Typical CAN Nodes

There are various CAN transceiver chips that implement CAN specifications. We use Microchip's MCP2551 and Fig. 5.2 shows how we connect it. Pins 6 and 7 are connected to the CAN bus whereas pins 1 and 2 are connected to a CAN controller or a microcontroller. When pin 1 (TX) receives a bit (0 or 1), it output the corresponding voltages from pins 6 and 7 so the corresponding value appears on the bus and receiving ECUs decode it correctly. On the other hand, when there is a frame being transmitted on the bus, the transceiver decodes the differential voltage level between CANH (pin 6) and CANL (pin 7) and then pin 4 (RX) sends 0 or 1.

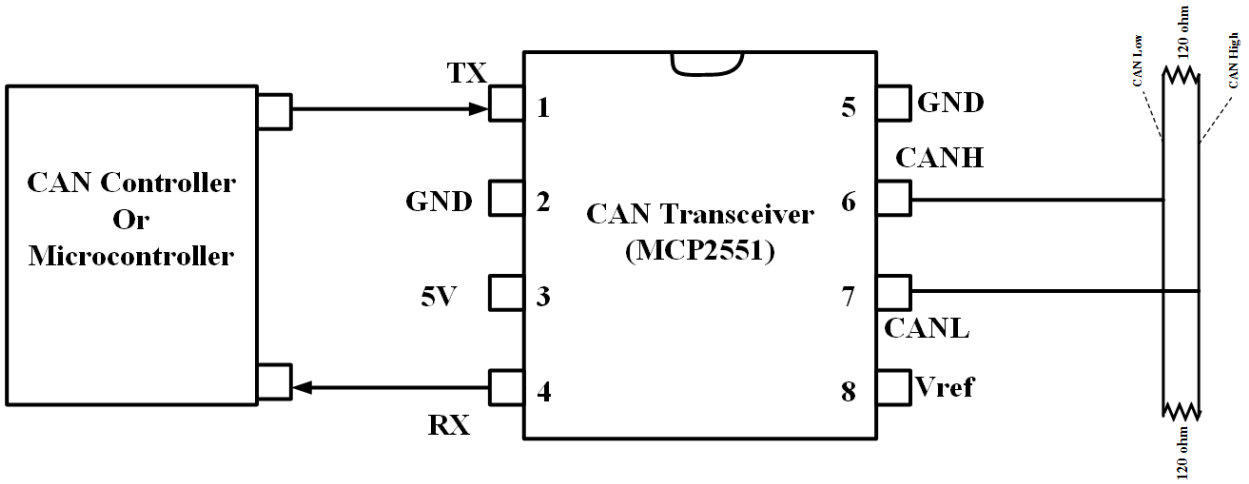


Figure 5.2: Wiring Diagram for MCP2551 CAN Transceiver

5.3 Dominant VS. Recessive Bits

CAN nodes can either send a 0 or 1. The 0 is called a "dominant" bit while the 1 is called "recessive". When 0 is sent, CAN transceivers are designed to output 3.5V on the CAN-High and 1.5V on the CAN-Low. The resulting differential voltage is 2V which corresponds to a 0 bit. On the other hand, when 1 is sent, the transceiver outputs 2.5V on both CANH and CANL resulting in a 0V. The receiving ECUs decode bits based on the voltage level on the bus using the CAN transceiver as shown in the previous section. When two ECUs try to send simultanously, if one of then is a dominant bit (0), then it will always appear on the bus. Fig 5.3.

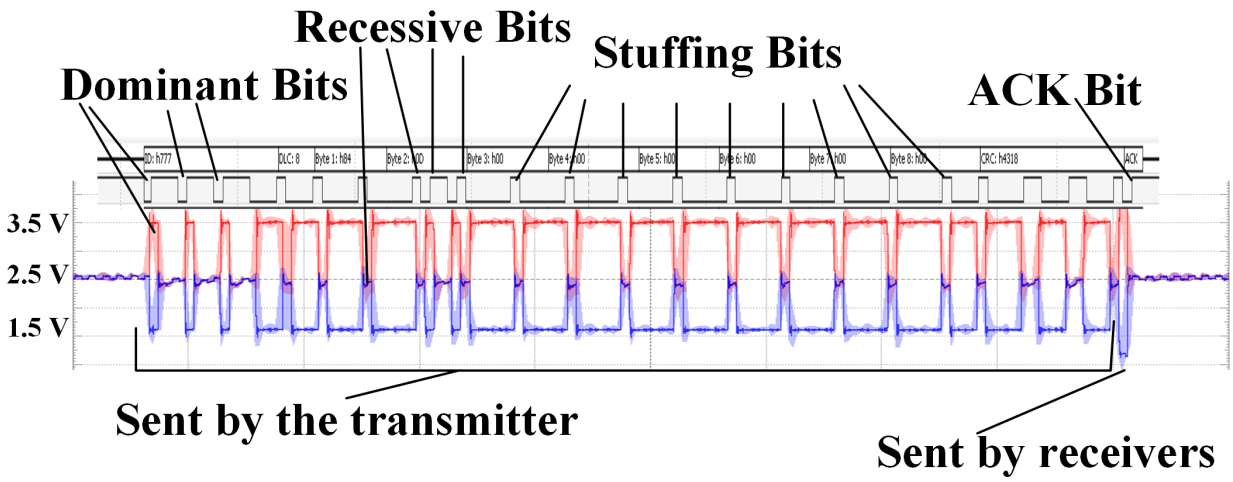


Figure 5.3: Dominant VS. Recessive Bits

5.4 CAN Arbitration ID

Arbitration ID determines which frame wins the arbitration phase and thus, transmit the rest of the frame that just won the arbitration. We mentioned that a CAN ID with the lowest value will have the highest priority during arbitration. We explain this further because its relevance to the upcoming chapters. Fig. 5.4 demonstrates how five ECUs try to transmit simultanously. They all start the transmission by sending the Start of Frame (SOF) dominant bit and then try to send the rest of the CAN ID. The important thing to note here is that they all send at precisely the same time. They send one bit at a time. We see that ECU_1 lost the arbitration phase early when it sent a recessive bit. This is because of the presence of the dominant bit in the bus that has a higher priority over recessive bits. ECU_2 loses the arbitration too when its recessive bit is faced by a dominant one on the bus. Every time an ECU loses, it switches to a receiving mode and stops transmission until the next chance. Eventually, ECU_5 wins the arbitration because it is sending the lowest possible value a CAN ID could ever have, i.e., 0x0. Once it wins the arbitration, it transmits the rest of its winning frame while the remaining ECUs only receive.

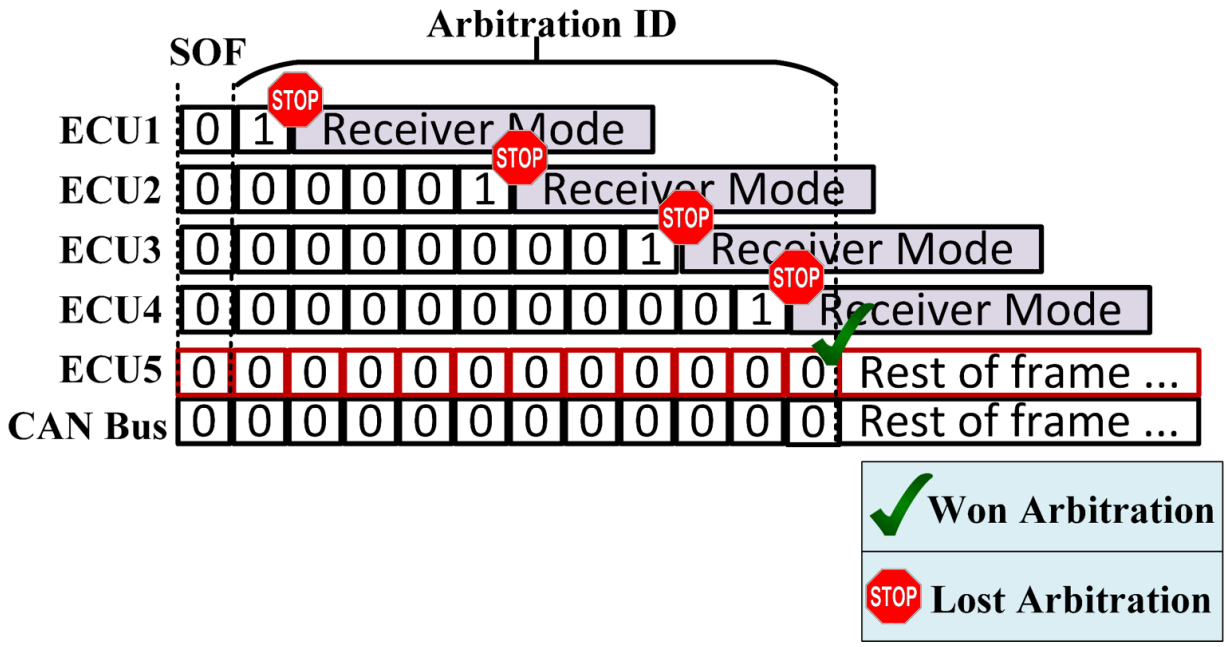


Figure 5.4: Arbitration Process

5.5 Error Handling

One of the main features in CAN is its ability to handle communication errors and resolve them in a semi-autonomous manner without the intervention of a third party. Each ECU is responsible for detecting two types of errors: transmission and receiving errors. Transmission errors occur when an ECU detects a problem while it is transmitting a frame that won arbitration. On the other hand, non-transmitting ECUs can detect errors that result from various reasons such as detecting an incomplete frame that has no delimiter.

5.5.1 Error Types

According CAN Specefication V2 [16], there are five types of errors that a CAN node should handle:

1. **Bit Error**: when a sender sees discrepancies with transmitted bit(s).

2. **ACK Error**: when a sender does not get at least an ACK bit.

3. **Stuff Error**: when a receiver detects 6 or more of the same consecutive bits (dominant or recessive).

4. **Form Error**: when a receiver detects violation of fixed fields.

5. **CRC Error**: when a receiver detects a discrepancy in the expected and received CRC.

5.5.1.1 Bit Error

When an ECU transmits to the bus it monitors it to verify that the transmitted bit is what is seen on the bus. When there is a discrepancy between the transmitted and observed bits, a bit error is detected, current frame's transmission stops, and an error flag is transmitted.

5.5.1.2 ACK Error

One critical bit in CAN frames is the acknowledgment bit (ACK bit) that a transmitting ECU can rest assured that its frame has been received by at least one other ECU. The ACK bit is sent by other ECUs at a precise time so it fits in the expected position in the CAN frame being transmitted. If an ACK is not detected by the transmitter, it aborts the transmission, an ACK error is detected, an error flag is transmitted, and then it reattempts sending the frame until an ACK bit is detected.

5.5.1.3 Stuff Error

Because CAN is an asynchronous protocol, no more than five consecutive bits are allowed to have the same value for synchronization purposes. After every five consecutive zeros (dominant bits) or ones (recessive bits), one bit of the opposite value is injected. Fig. 5.3 shows how this rule works. When this rule is violated, errors occur and a retransmission of the violating ECU is needed.

5.5.1.4 Form Error

A CAN frame has a few fields with fixed values that are always expected. When a fixed-valued bit field is detected by one of the receiving ECUs to be different than the expected value, a Form Error

is detected and an error flag is transmitted.

5.5.1.5 CRC Error

The CRC sequence field the is observed on the bus contains the result of the CRC value by calculated the transmitter. The receiving ECUs calculate the CRC sequence as well and a CRC Error is detected when the calculated CRC sequence is different than the one observed on the bus.

5.5.2 Error States

Each ECU keeps track of two important counters used for error handling, Transmit Error Counter (TEC) and Receive Error Counter (REC). Depending on the role of the ECU when an error is detected, one of the counters increases. When things go well in terms of successful transmission and reception, the counter decreases. Depending on the counter and its value, the ECU determines its error state according to the following rules:

- **Error-Active**: When TEC $<$ 128 or REC $<$ 128
- **Error-Passive**: When TEC $>$ 127 or REC $>$ 127
- **Bus-Off**: When TEC $>$ 255

The counters increase and decrease according to certain rules. In addition, each state entails some properties that the possessing ECU will have.

5.6 Summary

In this chapter, we introduced in details the most relevant CAN concepts to our upcoming chapters. Arbitration and error handling in CAN are considered corner stones for its robustness and fault-tolerance. However, these very same features could be a double-edged sword. We will show how they could be exploited and misused to result in devastating attacks.

Chapter 6

The Stealthy Targeted Arbitration Denial Attack

6.1 Introduction

CAN bus security has been an important aspect in automotive security recently. Typically, each CAN node is composed of three components: a microcontroller, CAN controller, and CAN transceiver. The microcontroller is responsible for the high-level functions such as calculating the speed, sending an airbag command, warning the driver about the oil pressure. Whereas the CAN controller is responsible for decoding/encoding outgoing/incoming CAN frames to/from the CAN bus. The CAN transceiver converts the frames to/from physical-level bits.

The presence of CAN controllers ensures the adherence to CAN standards. For example, once an ECU wins the arbitration phases, no other ECU can send until End of Frame (EOF) delimiter is seen on the bus. However, a new class of CAN attacks has emerged where an attacker relies on skipping CAN controllers in a way that directly connects microcontrollers to the bus through CAN transceivers, rendering CAN controllers useless or absence. This type of attacks allows attackers to communicate with CAN bus, and hence ECUs, without CAN controllers, giving them the full capability to manipulate the bus and ECUs without being tight to CAN controllers restrictions.

In this chapter, we review recent attacks where CAN controllers are not present in the attacking CAN node or certain features in the controllers are misused. Then we present a stealthy targeted DoS attack that also relies on similar techniques. However, based on the evaluation and experimentation, the proposed attack is superior to the surveyed ones due to its ability to achieve its goals without being noticeable.

6.2 Related Work

We have surveyed recent CAN attacks that belong to a new class of attacks that exploit CAN's design in terms of arbitration and error-handling. We classify these attacks into four categories based on the target: bus, ECU, arbitration, and frame.

6.2.1 Bus Denial

Preventing ECUs from transmitting can be done by ensuring that the bus is always occupied. There are two attack variants to do that: 1) transmit a stream of dominant bits or 2) transmit a high priority IDs at a high frequency. Fig 6.1 shows how each one of these attacks works. In the top attack, the attacker injects a continuous stream dominant bits to keep the bus busy. In the bottom attack, the attacker transmits an ID with the highest priority, i.e., 0x0. ECUs always lose the arbitration phase to this ID and the attacker's ECUs dominates the bus as long as its keeps transmitting the 0x0 ID at high frequency so that other ECUs can not transmit another ID when there is an idle time on an unoccupied bus.

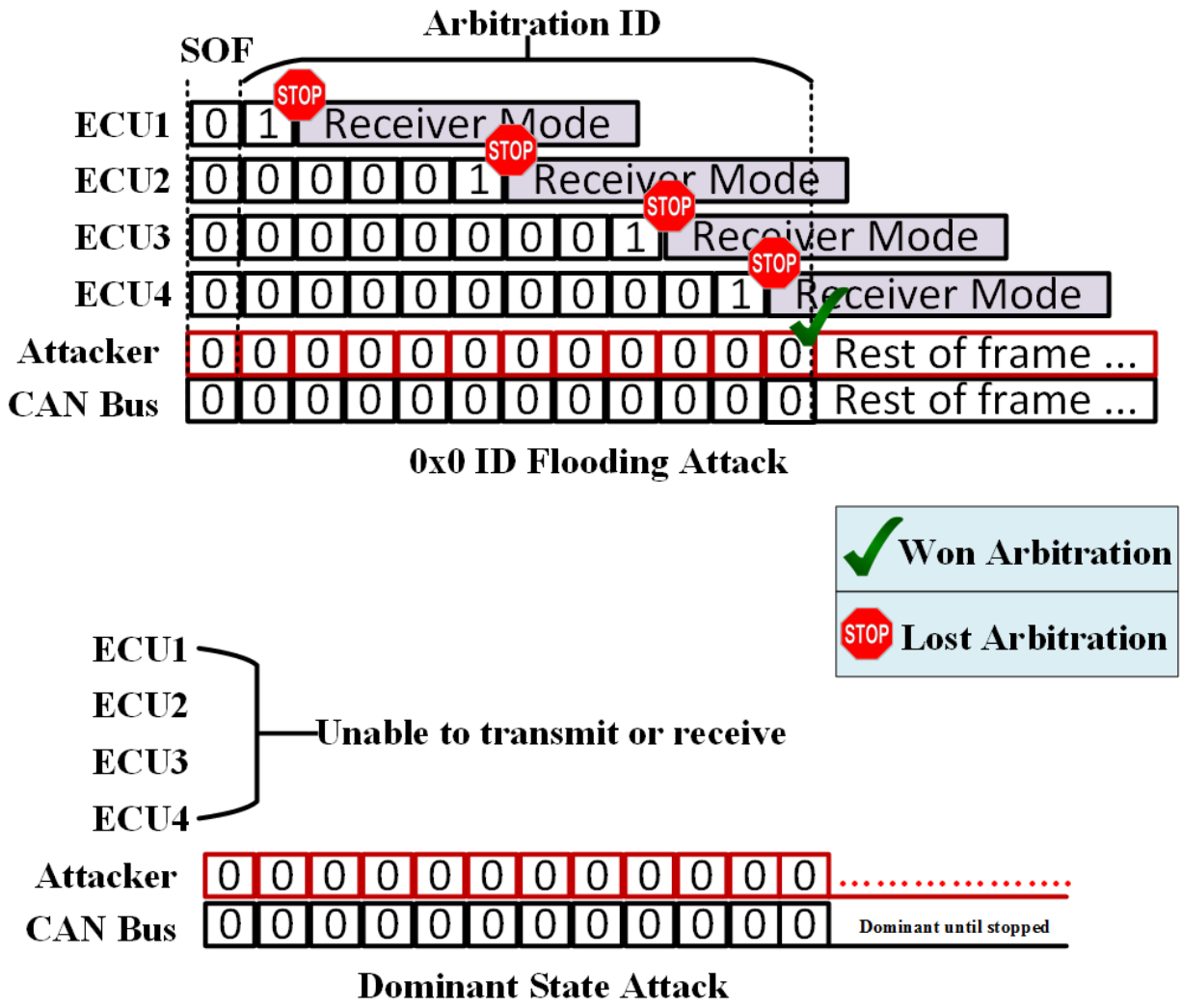


Figure 6.1: Bus Denial Attacks

6.2.1.1 Dominant Bits Stream (Bus Denial 1)

The physical bus connecting CAN-based ECUs can be rendered useless with one simple attack, i.e., the injection of a continuous stream of dominant bits. This will make ECUs retract every time they try to transmit a frame due to the bus's occupation from the point of view of ECUs. However, an attacker with access to a conventional ECU cannot launch the attack without violating CAN standards. As we mentioned in Sec. 4.2, CAN controllers ensure that the standards are followed among ECUs participating in a CAN bus. However, there are ways to overcome that. For example, Fröschle and Stühring [60] exploited a feature available in some CAN controllers called "Test

Mode" where it, when enabled, triggers the controller to flood the bus with dominant bits upon receiving at least one dominant bit and it would not stop until the mode is disabled. In addition, Murvay and Groza [126] built a malicious ECU that consists of an MCU and a CAN transceiver, missing a critical part of any ECU, i.e., CAN controller. This allowed the authors to launch the dominant bits stream attack without the restrictions of CAN controllers.

6.2.1.2 Flooding with 0x0 IDs (Bus Denial 2)

An attacker can flood the bus with a series of frames that have the highest possible priority IDs to prevent all other frames. This ID is 0x0 and it will always win arbitration against any other ID values. This attack can be thought of as a blind one in the sense that it prevents all non-0x0 ID frames [60]. Based on experiments, this attack must be done at a high rate to succeed.

Table. 6.1 summarizes the bus denial attacks and highlights the used tool, exploited vulnerability, and action taken to launch the attack.

Table 6.1: Bus Denial Attacks

Attack	**Tool**	**Vulnerability**	**Action**	**Ref**
Bus Denial 1	ECU	CAN controller's feature	Activate the Test Mode feature	[60]
Bus Denial 1	MCU & CAN transceiver	CAN design	Inject a continuous stream of dominant bits	[126]
Bus Denial 2	ECU	CAN design	Inject a 0x0 ID at high frequency	[60]

6.2.2 ECU Denial

Forcing a target ECU to a bus-off state is another category of the denial attacks. In this category, an attacker's goal is to prevent a target ECU from sending or receiving frames. As a result, all of the

functionalities managed by the target ECU will be lost. There are various approaches to achieve this attack as we will discuss here but they all share the final goal, i.e., driving an ECU to the bus-off state. By revisiting the *Error Counting Rules* in Sec. 5.5, we see that an ECU's error state will transition to the bus-off state only when its TEC counter reaches more than 255. In other words, only a transmitting ECU could transition to bus-off. On the other hand, the receiving ECUs never transition to the bus-off state because their worst-case scenario is the transition to the error-passive state whereby REC is more than 127. The following attacks rely on exploiting the error-handling mechanism in CAN where they carefully disrupt the target transmitting ECU in ways that force it to increment its TEC until it reaches more than 255.

6.2.2.1 CAN Controller's Misuse (Bus-Off Attack 1)

Fröschle and Stühring [60] exploited two features in some CAN controllers to achieve the attack: 1) ID Ready Interrupt and 2) "Test Mode". The first feature provides fast detection of the target ID, whereas the latter injects dominant bits to disrupt the frame with the detected ID. This results in an error flag triggered by a bit error that is sent by the target ECU as it detects the error as a result of the discrepancies between what it sends and what it sees on the bus. Then the stream of the dominant bits will trigger other errors for violating bit stuffing rules, no more than 5 consecutive bits can be transmitted and preventing error flags from being sent successfully. These errors gradually increase the transmitter's TEC to more than 255 and thus, the ECU's error state becomes bus-off. The activation of the "Test Mode" should not last for long to avoid affecting other non-targeted frames. Fig. 6.2 shows how this attack works for the case of *Attacker 1*.

6.2.2.2 Maliciously-Crafted Frames (Bus-Off Attack 2)

Another variant of this attack is to send a frame with identical contents except on recessive bit replaced by a dominant one. The premise of the success of this attack relies on the precise timing of the transmission of this malicious frame. It has to be done analogously to the target frame. This will result in a bit error by the transmitting ECU. This attack was shown in [38, 60] and depicted

in Fig. 6.3. The attack proposed by Cho et al. [38] only work on periodic frames.

6.2.2.3 CAN Transceiver-based Denial (Bus-Off Attack 3)

This attack avoids CAN controller's restrictions by connecting an MCU to a CAN transceiver that is connected to the bus. Similar to the previous attacks, this one aims to drive the target ECU to the bus-off error state by inducing bit errors. Palanca et al. [141] and Murvay and Groza [126] implemented an attack that detects certain target IDs and then overwrites a recessive bit by a dominant one to cause a bit error until the transmitting ECU goes to bus-off state. Iehira et al. [80] proposed a similar attack based on an FPGA board connected to a CAN transceiver. In addition, an active error flag can be sent once the target ID is detected to cause a bit error to the target ECU and a stuff error to the non-transmitting ECUs [80]. Fig. 6.2 shows how this attack works for the case of *Attacker 2*.

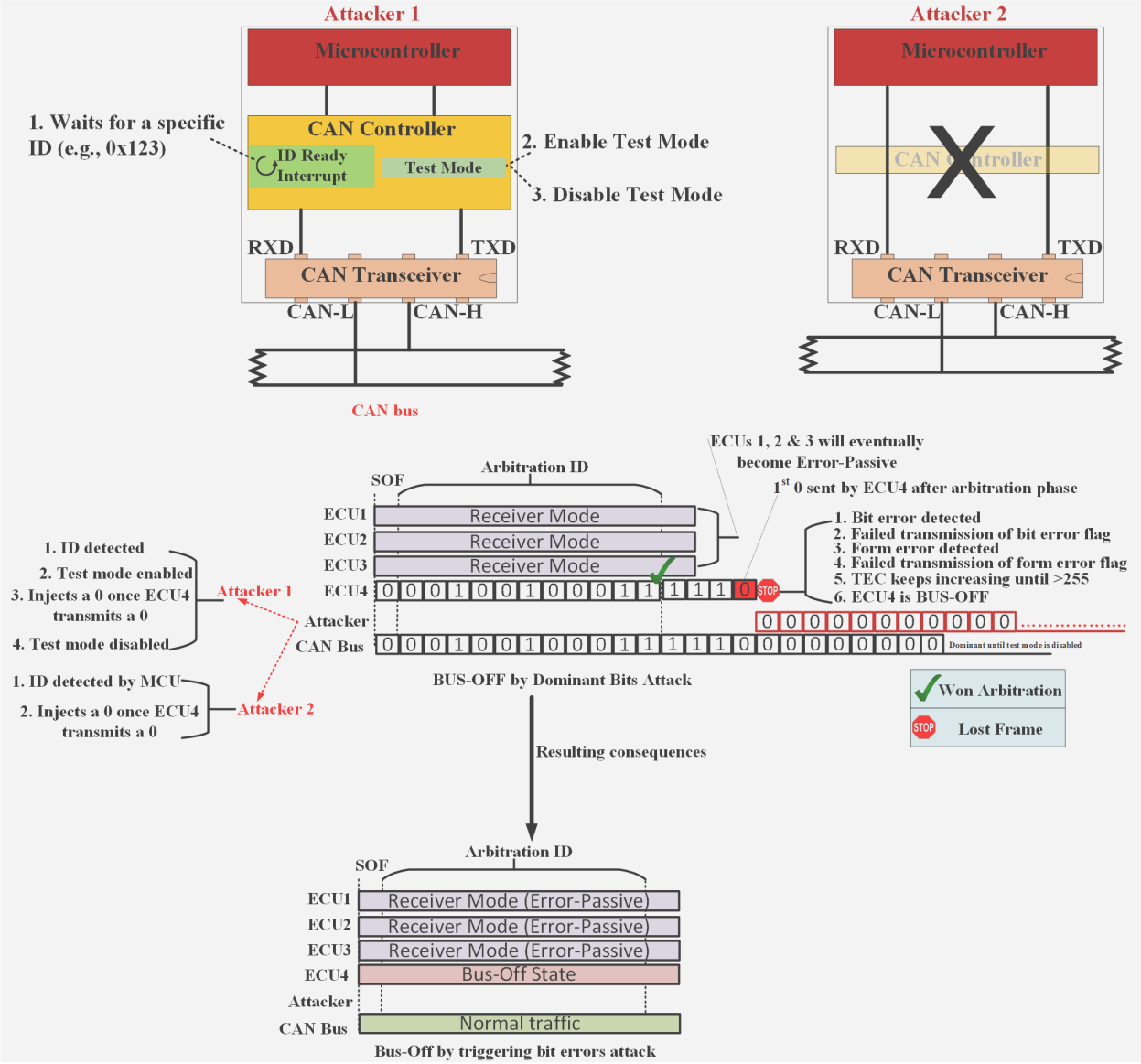


Figure 6.2: ECU Bus-Off Attacks 1 & 3

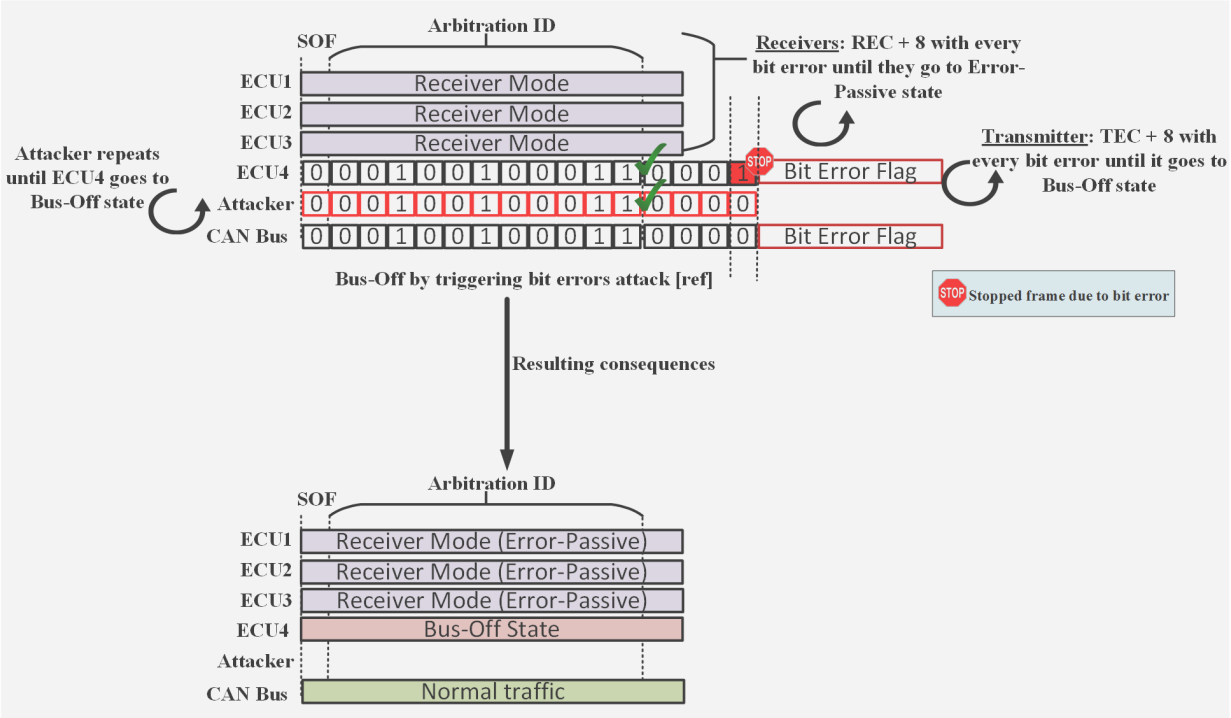


Figure 6.3: ECU Bus-Off Attack 2

Table. 6.2 summarize the ECU denial attacks and highlights how they are achieved.

6.2.3 Frame Denial

Targeting an ECU may not be the goal of some attacks. An attacker's goal could be prevention, modification, or masquerading of certain frames. An attacker with such goals would need to detect frames based on their IDs, then perform the attack based on the desired goals. Based on the surveyed attacks, we classify frame denial attacks to prevention, interruption, and impersonation attacks.

6.2.3.1 Prevention

Flooding with High Priority IDs. An attacker could transmit IDs with higher priority than the IDs used by the target frames. This could have a collateral damage in the sense that non-targeted frames could be affected [60]. For example, if the attacker's goal is to prevent frames with 0x123

Table 6.2: Bus-Off Attacks

Attack	**Tool**	**Vulnerability**	**Action**	**Ref**
Bus-Off 1	ID Ready & ECU	Controller's feature & error handling	Detect ID with ID Ready then transmit a frame with a dominant bit to cause a collision	[60]
Bus-Off 1	ID Ready & Test Mode	Misuse of Controller's features	1. Detect target ID with ID Ready 2. Enable Test Mode	[60]
Bus-Off 2	ECU	1. Real and fake frame's predictability 2. Error Handling: inducing bit error	1. Detect preceded IDs 2. Send crafted frames that cause bit error 3. If preceded IDs are lacking, then inject fake preceded IDs	[38]
Bus-Off 3	MCU & CAN transceiver	Error Handling: inducing bit error	1. Detect target ID by bit-level analysis 2. Send an identical frame to with a dominant bit difference to cause a collision, and hence, bit error	[141, 126]
Bus-Off 3	FPGA & CAN transceiver	Error Handling: inducing stuff error	1. Detect target ID by bit-level analysis 2. Send an active error flag to trigger a stuff error	[80]

IDs, she would flood the bus with 0x122 IDs. Although this should prevent 0x123 IDs from a successful transmission, any ID with a value greater than 0x122 could be affected too.

6.2.3.2 Interruption

Bit Errors. An attacker could prevent target frames from successful transmissions by triggering a bit error during the transmission of the frame [141, 126, 60, 80, 38].

Active Error Flags. When frames with certain IDs are detected, an attacker can inject an active error flag, 6 dominant bits, during transmission using the “Test Mode" feature [60] or an FPGA board connected to the bus by a CAN transceiver [80]. This forces the target frame to retract and potentially go to a bus-off state as mentioned in Sec. 6.2.2.

6.2.3.3 Impersonation

It is possible to impersonate an ECU by transmitting its own frames to achieve various goals such as sending false data.

Interrupt and Inject. Assuming the automatic retransmission is disabled, an attacker could interrupt a target frame and then transmit the fake frame with false data [60].

Drive to Bus-Off and Inject. An attacker could perform one of the bus-off attacks in Sec 6.2.2 and then send fake frames to impersonate the silenced ECU [60].

Tables. 6.3 and 6.4 summarize the frames’ attacks and show their intersection with the ECU denial attacks.

Table 6.3: Frames Denial Attacks

Tool	**Vulnerability**	**Action**	**Ref**
A combination of the following tools: • ID detection: -ID Ready -Periodicity -Preceded IDs • Test Mode feature for bits injection	CAN design	**Assumption:** Automatic retransmission is disabled 1. Detect target ID 2. Enable Test Mode to inject 6 dominant bits	[60]
ECU	CAN design	Send an ID with a high priority at a fast rate to block all IDs with lower priorities	[60]

Table 6.4: Impersonation Attacks

Tool	Vulnerability	Action	Ref
ECU	Controller's features	• Perform a bus-off attack against the ECU owning the target IDs • Inject fake frames	[60]
ECU	Controller's features	**Assumption:** Automatic retransmission is disabled • Suppress a frame with attack **x** • Inject fake frames	[60]

6.2.4 Arbitration Denial

This attack aims to prevent frames with target IDs from winning arbitration. Murvay and Groza [126] proposed an instance of this attack where the bus is monitored with an MCU connected to the bus by a CAN transceiver to detect the target ID and then inject a dominant bit that replaces a recessive bit in the ID field. As this attack relies on the injection of a single bit in the arbitration phase, the target ID loses arbitration but a form error is detected due to an incomplete frame. When the attack persists and the target ECU continues to lose arbitration, an error flag is generated. Fig 6.4 shows that the attack is monitoring the bus waiting for the least significant recessive bit of the target ID and then injects a dominant bit to replace it. This forces ECU4 to stop transmission due to the lost arbitration but results in a form error due to an incomplete frame observed on the bus.

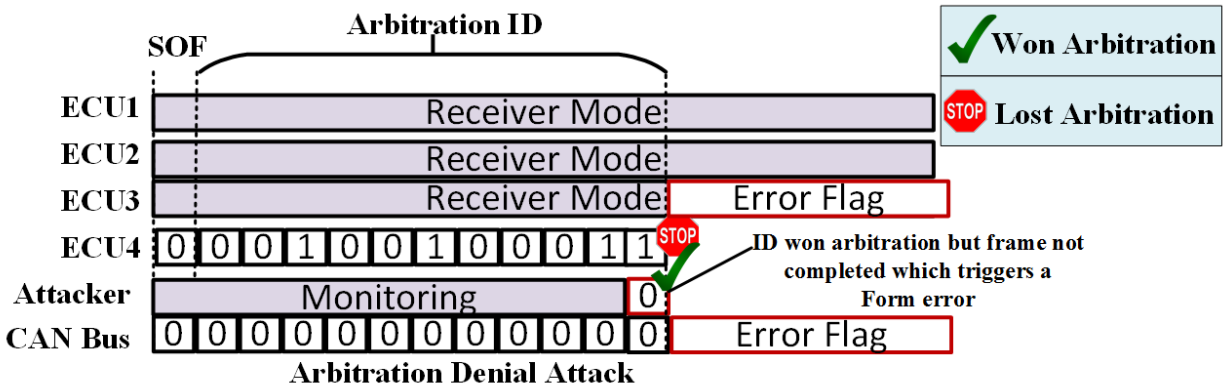


Figure 6.4: Arbitration Denial Attack

6.3 Stealthy Arbitration Denial Attack

We present a variant of the arbitration denial attack that is different and stealthier than the one discussed in Sec. 6.2.4. The attack shown in [126] triggers an error every time it is launched. The attack has two drawbacks: 1) it leads to a bus-off state even if the goal is not to shut off an ECU and 2) the abnormal behavior of injecting a single dominant bit could be detected by an IDS. On the other hand, our attack avoids these drawbacks and goes undetected because it does not violate the expected behavior of the ID arbitration process. Our attack passively monitors the bus in order to detect an ID of interest and then injects a dominant bit overwriting a recessive one. To avoid triggering an error flag as in [126], we complete the existing frame with a valid frame that completes the interrupted frame containing the target ID. Fig. 6.5 shows how ECU4 loses arbitration of ID 0x123 to what seems to be 0x122 ID. However, the attacker waits for the bit preceding the last recessive bit and overwrites it with a dominant one. The attacker then completes the rest of the fake 0x122 frame to avoid triggering error flags. The figure shows that when ECU4 is transmitting a frame with a 0x123 ID, the attacker is monitoring the bus waiting for the bit at the position 2, which precedes the last recessive bit. The attacker sees that the ID that is about to successfully win ID arbitration and transmit is most likely 0x123. Therefore, once bit 2 is detected, the attacker injects a dominant bit to force the sending ECU of the 0x123 ID to lose arbitration.

ECU4 stops transmitting and goes to receiving mode. If the retransmission is enabled, ECU1 will attempt to retransmit 0x123 frame when the bus is idle. If the retransmission is disabled, then ECU4 has lost the disturbed 0x123 frame. What differentiates this attack from similar DoS proposals is its compliance with CAN specification such that it does not cause error frames that might lead ECU4 to go bus-off. Fig 6.6 illustrates how the attack occurred on the scope.

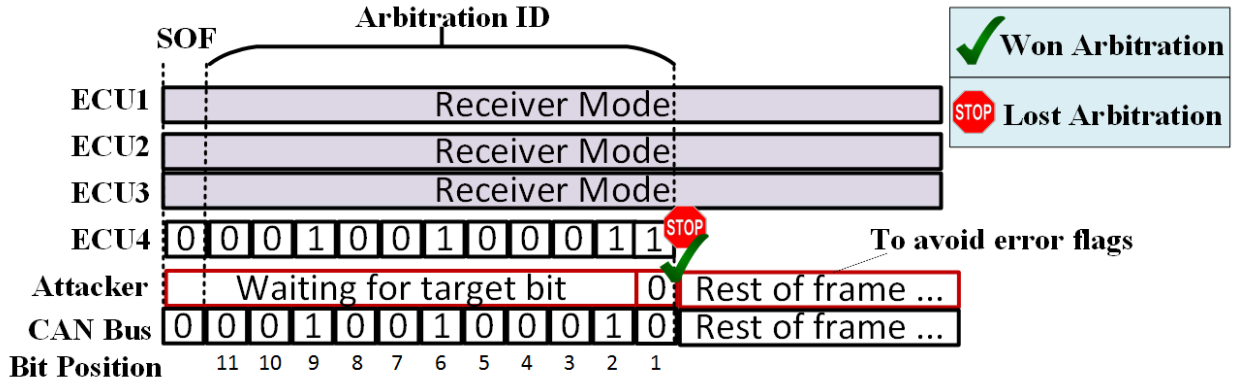


Figure 6.5: Stealthy Arbitration Denial

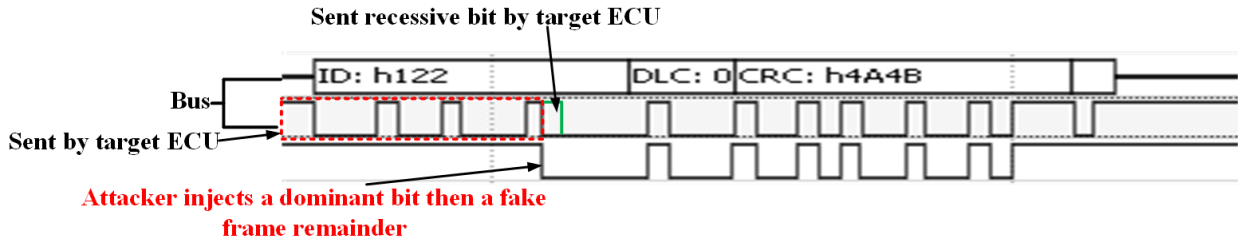


Figure 6.6: Arbitration Denial on The Scope

Table. 6.5 summarizes the attacks that aim to prevent frames with certain IDs from winning the arbitration phase. In both attacks, the frames with the target IDs lose arbitration. The difference is the consequences. In the attack proposed by Murvay et al. [126], an error frame is generated because the transmitting ECU could not complete the frame's transmission and hence, a error was generated. This could eventually drive the target ECU to bus-off state. On the other hand, our attack injects a crafted set of bits that would complete the interrupted frame so that no error are generated and the attack goes unnoticed.

Table 6.5: Arbitration Denial Attacks

Tool	**Vulnerability**	**Action**	**Ref**
MCU & CAN transceiver	CAN arbitration design	1. Detect target ID (partially) 2. Overwrite a recessive bit with a dominant bit 3. Target ID loses arbitration 4. Due to the incomplete frame, a form error is generated	[126]
MCU & CAN transceiver	CAN arbitration design	1. Detect target ID (partially) 2. Overwrite a recessive bit with a dominant bit 3. Inject fake bits to complete the started frame by the target ECU 4. Target ID loses arbitration 5. No errors are generated	Our work

6.3.1 Threat Model

The attacker we consider is assumed to be able to monitor the bus, injects CAN frames, and individual bits such that CAN standards are violated. We assume that an attacker could launch the discussed attacks by gaining physical access to the car or remotely. The physical access provides one of two attack surfaces: 1) the OBD-II port and 2) an ECU of the attacker's choice to be compromised [92]. or misused [60]. On the other hand, remote access has been shown in various works such as in [32] and [136]. The level of difficulty varies in the attack's surfaces, but the goals are the same. Table 6.6 compares between the attacker's means of attack and their difficulty.

Attack Surface	**Difficulty**	**Ref**
OBD-II Port	Easy	[92]
Compromised ECU	Medium	[92, 60]
Remote Exploitation	Difficult	[32, 136]

Table 6.6: Threat Model Comparison

6.3.2 Algorithm

The algorithm used for this attack consists of three steps:

1. Calculation of target ID's bit position to attack (replace by a dominant bit)
2. Detection of the target ID on the bus
3. Attacking the target ID's bit position

6.3.2.1 Bit Position

To determine the bit position, we simply calculate the position of the least recessive significant bit in the target ID in advance as shown in Algorithm 1. The algorithm finds the position using a simple bitmask and shift operations as shown in Fig 6.7. The given examples show that the bit position in 0x123 ID is 1 and 4 when 0x554 is the target ID.

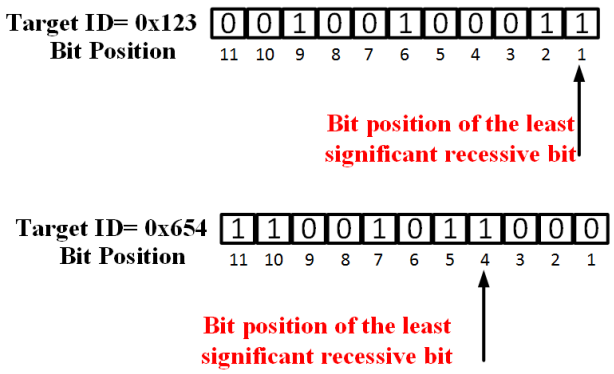


Figure 6.7: Bit Position Algorithm

Algorithm 1 Target Bit Position Calculation

The **bitPosition** function returns the target bit position in the target CAN ID
Input: *targetID*: This is the value of the target ID

1: position $= 1$
2: **while** (targetID$_{position} \neq 1$) **do**
3: position $++$
4: **end while**
5: bitPosition $=$ position ▷ position of the least significant recessive bit

6.3.2.2 Target ID's Detection and Attacking

Then we monitor the traffic waiting for an ID that matches the target ID up one bit prior to the target bit. After that, we inject a dominate bit to override the target bit that we determined its position in the previous step. It is, most likely, the recessive bit that would complete the target bit if not overridden. Algorithm. 2 shows the steps to detect and attack the target ID and Fig. 6.8 gives an example with 0x243 as the target ID.

Algorithm 2 Arbitration ID Detection and Injection

The **attackID** function performs a selective DoS on a particular ID
Input: *bitPosition*: This is the value of the target ID's target bit position

```
1: bitsToTargetBit = 11 − bitPosition
2: while matched ≤ bitsToTargetBit do
3:     if currentBit == targetID[(11 − bitsToTargetBit) + bitsToTargetBit] then
4:         matched++
5:         if matched == bitsToTargetBit then
6:             Transmit a dominant bit
7:             Transmit a fake frame's remainder
8:         end if
9:     else
10:        matched = 0                          ▷ End loop and wait for the next ID
11:    end if
12: end while
```

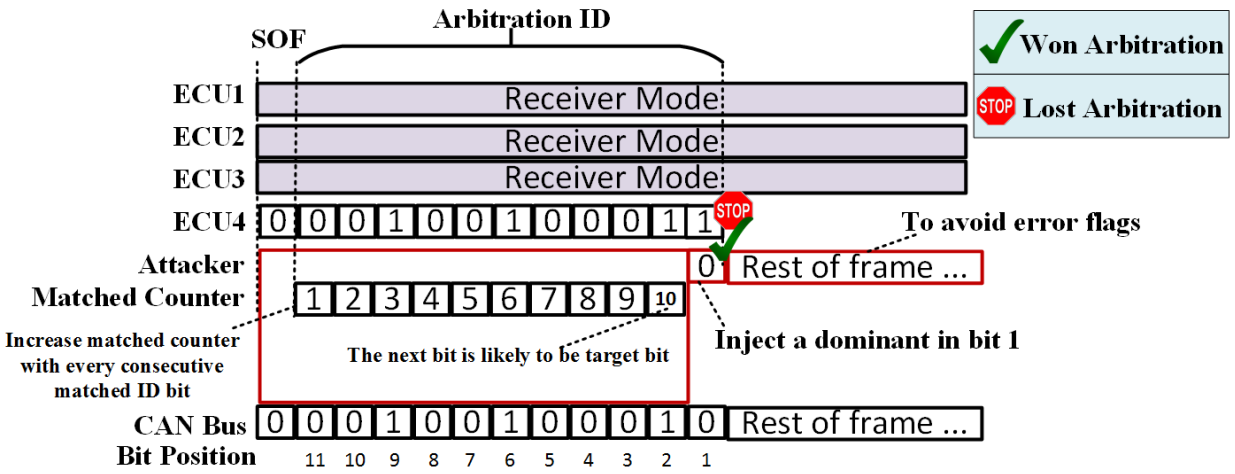


Figure 6.8: Attack Algorithm Demonstration

6.4 Evaluation

In order to evaluate the attack's effectiveness, our analysis needs to be rigorous. We need to analyze data that shows the attack's impact on the target ECU. Therefore, we analyze CAN error statistics in the target ECU. We need to learn from the CAN error statistics important measures such as Transmit Error Count (TEC), Receive Error Count (REC), and error state. These measures reflect

the magnitude of the attack. The less they are affected, the more successful the attack is in terms of stealthiness. When an attack triggers a large number of error frames, TEC and REC will increase until their ECUs become Error-Passive and potentially bus-off depending on their role when errors take place. TEC increases when an error occurs during a frame's transmission in the transmitter's ECU and REC in all receiving ECUs.

We evaluated the proposed attack and compared it with the other denial attacks discussed in Sec. 6.2. The evaluation shows that our attack is the stealthiest compared with the others. It achieves the arbitration denial without causing a single error, unlike the rest of the attacks.

The evaluation consists of two parts: 1) choice of hardware and 2) effectiveness metrics.

6.4.1 Choice of Hardware

Most of the current research on attacks that require precise timing such as the discussion above use automotive-grade microcontrollers or high-end tools that raises the entry level for researchers. Using such devices presents some challenges such as: 1) A certain set of skills is required, 2) there is a steep learning curve, and 3) these devices are costly. However, we implemented our attack, along with the other attacks, with an off-the-shelf microcontroller that is accessible in terms of cost and usability. We decided to use an intuitive, custumizable, and interactive tool designed specifically for CAN security testing when CAN controllers are not followed. This tool is called *CANT*, and it is an open source tool built for STM32 Nucleo-144 development boards. CANT is developed by GRIMM CO's automotive security team [44]. The tool is built to provide CAN security researchers a unique feature that is not easily achievable with low-cost devices, i.e., CAN testing without CAN controllers. In other words, we can do things that we cannot do with conventional CAN tools due to the fact that they were not built for security testing. The tool provided us with the flexibility to implement all of the attacks and countermeasures we wanted to evaluate.

CANT allows for bit-by-bit analysis for CAN frames. This allows us to inject bits at any time we decide to. CANT facilitates the synchronization with the bus and sampling of CAN frames as shown in Fig. 6.9. The figure illustrates how sampling works in CANT. CANT sets up an interrupt

waiting for a falling edge after a idle period, which is going to be an SOF. Once the interrupt is triggered, it gets disabled then sampling the incoming frame starts and upon the EOF bits, the sampling finishes and the interrupt is enabled back waiting for the next frame. From an attacker's viewpoint, injection could be done at any time during the sampling when certain conditions are met.

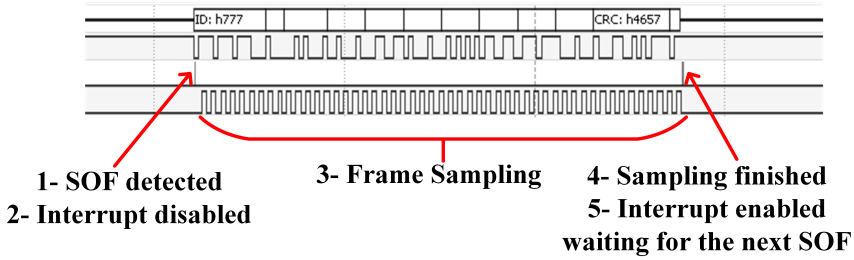


Figure 6.9: Sampling Frames with CANT

Before using CANT, we have explored others option to evaluate the aforementioned attacks. We started our evaluation with a BBB microcontroller and managed to implement a few of them. BBB has two real-time cores called, Programmable Real-Time Units (PRUs). However, due to the counter-intuitive nature of PRUs and the need to implement the functionalities of CAN controllers, we decided to use CANT. Although we use CANT, we still provide an overview of our experience of PRUs in Sec. B.

6.4.1.1 Evaluation Platform

Fig 6.10 shows the evaluation platform that we built. It consists of four BBBs to simulate normal ECUs and the attacker is represented by the STM32 Nucleo-144 development board in the left side of the figure. Each of the five microcontollers is connected to the CAN bus through an MCP2551 CAN transceiver. We build these boards to reduce clutter and loose wires.

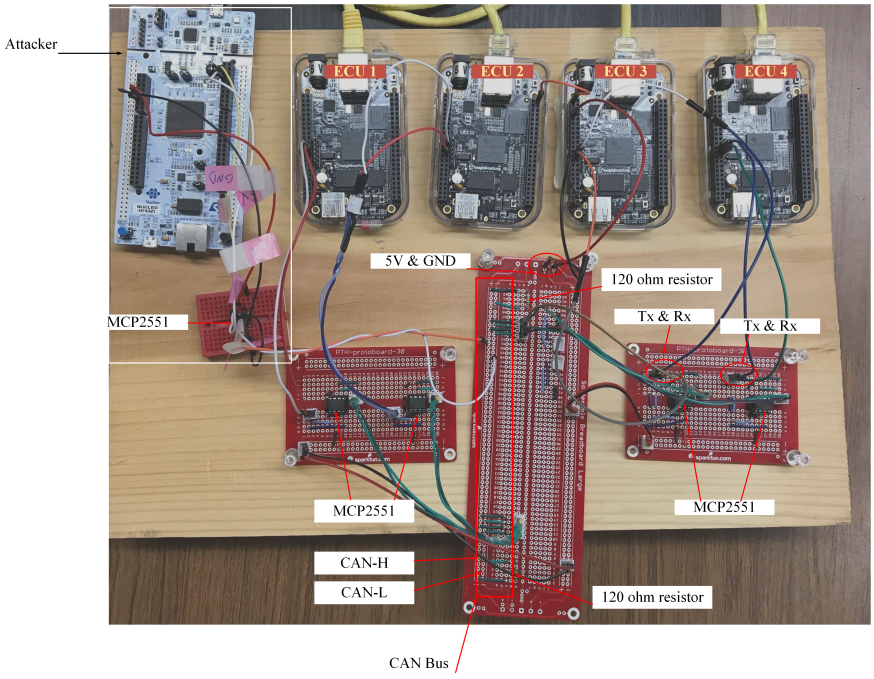


Figure 6.10: Evaluation Platform

6.4.2 Effectiveness Metrics

All of the discussed attacks, except the bus denial, exploit CAN's error-handling mechanism to achieve their goals. Therefore, it is reasonable to monitor error states of the targeted ECUs during an attack. To do this, we assign a BBB as a target ECU and capture its error states for further analysis. We are interested in the two error counters, REC and TEC. The following discussions show error states during each attack and based on them we learn about the attacks' levels of stealthiness.

6.4.3 Evaluated Attacks

Table. 6.7 summarizes the evaluated attacks.

No.	Attack Name	Description	Ref.
1	Bus Denial (Sec. 6.2.1.1)	Continuously forces the bus to be in a dominant state	[60, 126]
2	Bus Denial (Sec. 6.2.1.2)	Continuously transmits frames with 0x0 ID at a fast rate	[60]
3	ECU Denial (Sec. 6.2.2.3)	Drives an ECU transmitting the target ID to switch to Bus-Off state	[38, 60]
4	Frame Interruption (Sec. 6.2.3.2)	Triggers bit errors when the target frame is being transmitted until transmitting ECU switches to a Bus-Off state (same as attack *3*)	[141, 126, 60, 80, 38]
5	Arbitration Denial (Sec. 6.4)	Override the least significant recessive bit in the arbitration ID with a dominant one	[126]
6	Stealthy Arbitration Denial	In addition to attack *4*, a fake remainder of the frame is injected to avoid triggering form errors	Our work

Table 6.7: Summary of the Evaluated Attacks

6.4.4 0x0 Arbitration ID

This attack exploits the very feature of CAN, that is its physical-layer arbitration between dominant and recessive bits. Because of that, there are no errors detected as shown in Fig. 6.11.

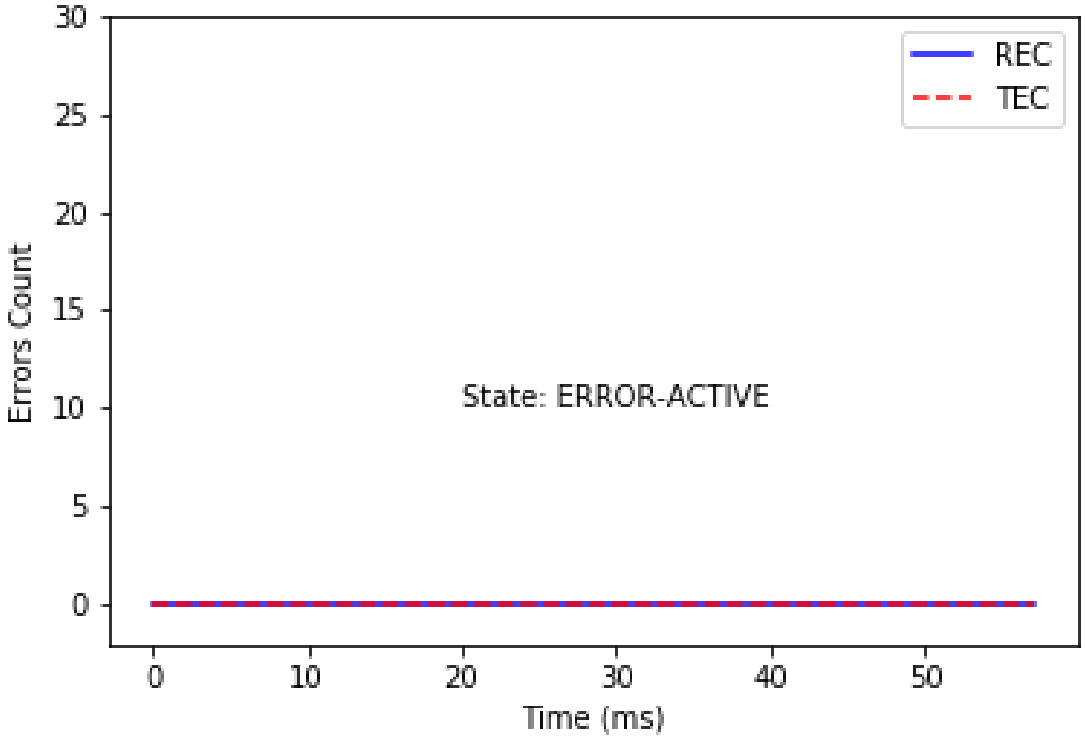


Figure 6.11: Errors During 0x0 ID Flooding Attack

6.4.5 Bus-Off Attack Evaluation

The attack proposed in [141, 38, 60, 126] aims to turn off an ECU's communications by forcing it to switch to Bus-Off error state. Fig. 6.12 shows how TEC increased sharply when the attack started at time 120 ms. After a few milliseconds, the error state switched to Error-Passive and then eventually to Bus-Off.

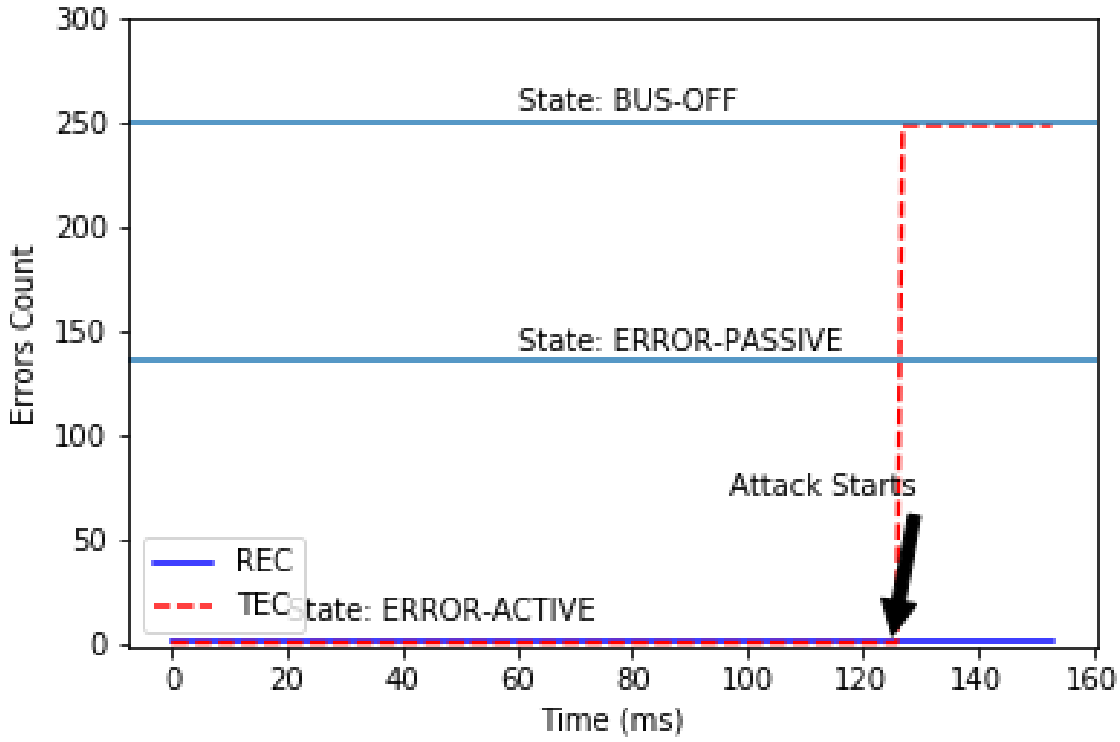


Figure 6.12: Errors During Bus-Off Attack

6.4.6 Arbitration DoS Evaluation

The arbitration denial attack proposed by Murvay and Groza [126] aims to prevent a frame with a target arbitration ID from a successful transmission by injecting a dominant bit that replaces a recessive one in the arbitration field. We implemented the attack with CANT and analyzed the impact on the transmitting ECU. Fig. 6.13 shows that the attack had no impact on TEC, but REC increases when the attack starts until it reaches 127, changing the error state to Error-Passive. The attack did not cause the ECU to shut down by switching to Bus-Off state like the attack discussed in Sec 6.2.2. However, it still caused errors that resulted in an Error-Passive state. The first observation is that TEC was not affected because the transmitting ECU switched to receiving mode once it lost arbitration, which means that whenever an error is detected, REC is going to increase, not TEC. This also means that the ECU cannot be shut down with this attack. We still

need to further investigate the type of error that caused REC's increase. Since the attack occurred during the arbitration phase, the error is not a bit error because it is expected to have multiple ECUs sending simultaneously. It has to be a form error because when the transmitting ECU lost arbitration due to the dominant bit replacing a recessive one, the frame was not complete. From receiving ECUs' perspective, this is a form error. Every time the attack occurs, a form error is triggered on one or more of the receiving ECUs, including the target transmitting ECU, which had switched to a receiving mode once it lost the arbitration phase.

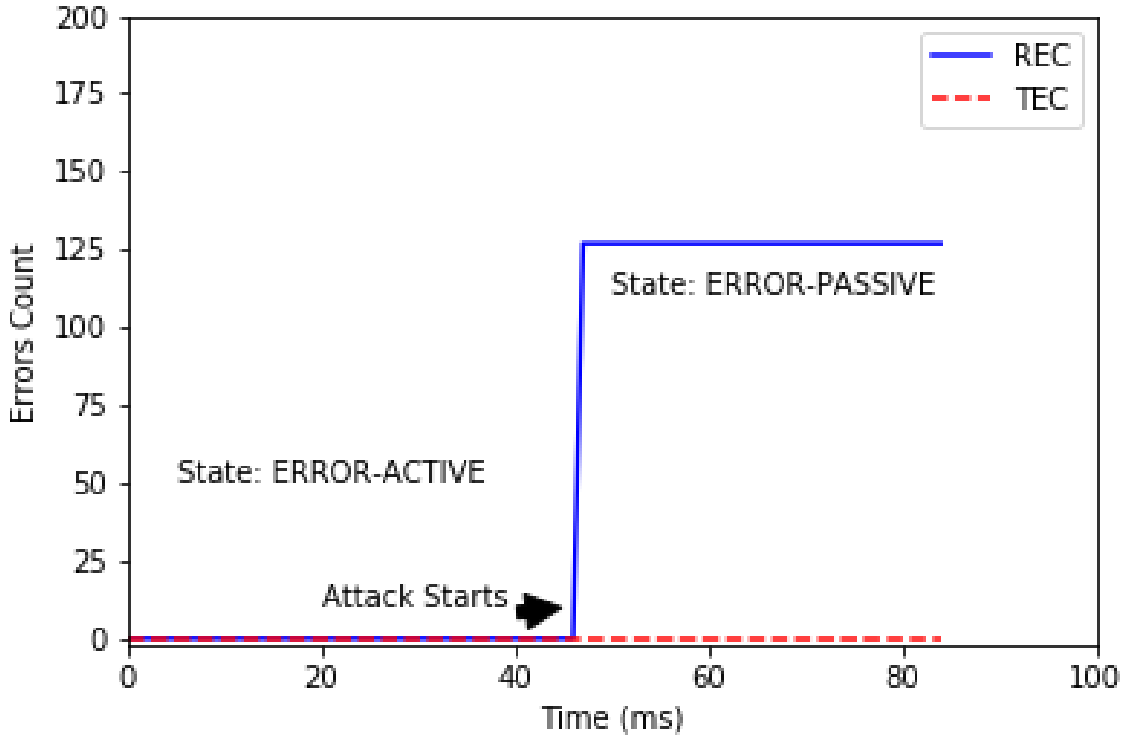


Figure 6.13: Errors During Arbitration DoS Attack

Fig. 6.14 shows the attack on the scope. We see at the top a partial 0x777 ID that is interrupted by a dominant bit that replaces the transmitted recessive bit and stops the completion of the frame. We show the intended frame with the location of injection in Fig. 6.15.

Figure 6.14: Arbitration DoS Attack on The Scope

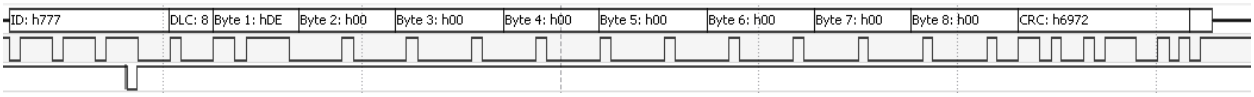


Figure 6.15: Target Frame and Dominant Bit Injection

6.4.7 Selective Arbitration DoS Evaluation

Here we compare the impact of the attack on the ECU sending the target ID. We show the following information that has been captured during the attack. We show the attack's impact on three IDs that represent 1) lowest possible ID to attack (0x1), 2) average ID (0x123), and 3) highest possible ID (0x7FF). Before we start capturing data on each ECU (sender and receiver), we start CAN interface with the error reporting option enabled to get the most amount of error-related information.

```
$ canconfig can0 bitrate 500000 ctrlmode berr-reporting on
$ canconfig can0 start
```

We also wrote a script that extracts error handling information: TEC and REC before, during, and after that attack and then plot the data to see whether or not the attack disrupted the bus and triggered errors. The complete script is provided in Sec. A.2.5.2. Fig 6.16 shows that the attack did not have any obvious impact on the transmitting ECU of the target ID apart from preventing it from transmitting frames with the target ID. We see that when the attack starts at time $\approx$ 15 ms, REC and TEC counters did not increase at all. This validates the effectiveness and stealthiness of our attack and shows that it is stealthier and more dangerous than the other attacks.

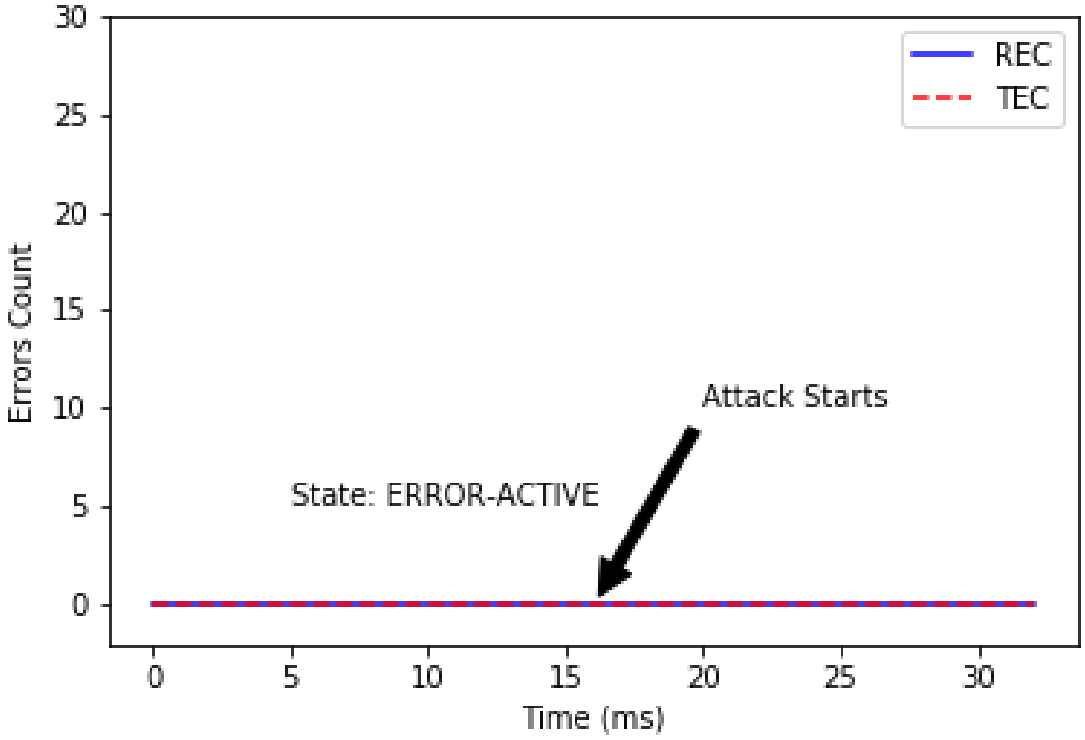


Figure 6.16: Errors During Selective DoS Attack

6.5 Summary

In this chapter, we surveyed recent attacks on CAN that use nontraditional approaches to realize their goals. These attacks either exploit CAN features such as its error-handling mechanism, or misuse CAN controllers to perform malicious actions. We implemented the attacks and compared them with a new attack that we proposed in this chapter, called *stealthy target arbitration denial attack*. The evaluation and comparisons show that show that our attack is superior in terms of stealthiness compared to the other attacks. The more errors an attack generates, the less stealthy it is considered. Our attack did not trigger any errors and thus we consider it the stealthiest.

Chapter 7

Using ID-Hopping to Defend Against Targeted DoS on CAN

7.1 Introduction

Nowadays, smart cars are more intelligent than ever before. Manufacturers embed numerous microcomputers to enhance the cars' safety, comfort, and entertainment features. Examples are collision avoidance systems, and active trace control (safety features); remote start and parking assistance systems (comfort features); and on-board Internet and satellite radio (entertainment features).

The recent advancements in smart cars, however, also introduced new security issues that cars' manufacturers have not anticipated. Numerous reports and research papers have shown the inadequacy of security in smart cars [92, 33, 30]. These security issues result from the unexpected behaviors that result from the interactions between the many heterogeneous components every smart car consists of [45]. This includes Commercial-Off-The-Shelf (COTS) products and components implemented by third parties. In addition, the increased wireless channels exposed the two-decades-old unconnected cars to numerous security problems.

With the complex cyber-physical interactions, as well as the increased communication chan-

nels, security attacks become imminent. Therefore, security solutions have been proposed to add security layers to smart cars. For example, cryptography-based solutions have been proposed to provide authentication, confidentiality, and integrity measures. Intrusion Detection Systems (IDS) have been proposed to detect internal attacks and firewalls to prevent external attacks.

Although these solutions improve security in smart cars, there is a very little attention given to DoS attacks. Automotive networks typically deploy a network protocol called Controller Area Network (CAN). This protocol has inherently security weaknesses such as the lack of authentication and confidentiality, weakness of integrity checking, and lack of nodes' identification. Network's nodes, called Electronic Control Units (ECUs), cannot identify each other. Instead, each frame has a unique ID, called *arbitration ID*, that signifies what the frame means and what priority it has. Due to the lack of security measures, an attacker can easily flood the network with a high priority ID to constantly dominate the network and prevent legitimate ECU from using it. This is considered as a DoS attack because ECUs cannot use the network. In addition, an attacker can easily use any ID and spoof other ECUs with illegitimate frames carrying an ID belonging to a legitimate ECU.

In this chapter, we consider a special type of DoS attacks where an attacker targets a certain ECU, or a set of ECUs, to prevent it from sending particular frames. For example, when the Anti-Brake Systems (ABS) sends a sensing data frame that is needed to mitigate a potential accident, an attacker would send a frame, or a set of frames, to ensure that the ABS's frame cannot use the network, and therefore the ABS fails. This scenario is safety-critical and proper solutions must be proposed.

We propose the ID-Hopping algorithm where we design a mechanism that hinders an attacker's ability to send malicious frames that would make targeted DoS attacks possible. Even if an attack succeeds, our mechanism detects such an attack and reacts immediately. The mechanism generates a set of alternative IDs that each ECU should use when an attack is detected. Our proposal is similar to Identity-Anonymized CAN (IA-CAN) proposed by Han et. al. [69], where they propose a mechanism that requires all communicating ECUs to calculate anonymous IDs before an ID is

sent or received. Our proposal is different such that we only require communicating ECUs to share a carefully calculated value (*offset*) such that it can be used when an attack occurs so the IDs are changed based on that value.

We evaluated the ID-Hopping mechanism by simulating the process of generating alternative IDs from the offset. Two critical conditions our proposal ensures when alternative IDs are generated: 1) alternative IDs are unique to the original ones and 2) IDs priorities are not compromised.

We believe that the ID-Hopping mechanism holds a promising solution to a variety of DoS attacks. In addition, if we deploy the mechanism to work natively in the network, we believe it should prevent reverse engineering attacks on IDs.

7.2 Background

All ECUs are connected to a bus-topology network that has a several subnetworks. Each subnetwork consists of a number of interconnected ECUs that perform certain functions. The most common protocol is Controller Area Network (CAN), which is a typical bus network through which ECUs can intercommunicate, monitor sensors, and control actuators.

7.2.1 CAN Protocol

CAN is the most common protocol because it has been mandated to be deployed in all cars in the US since 2008 [92]. CAN protocol mainly provides two services: 1) at the physical layer, it allows the transmission of frames as voltages that do not get influenced by interfering magnetic fields, and 2) at the data link layer, each frame is formatted in a well-defined format so ECUs can exchange messages in a meaningful manner. After that, a higher level protocol is needed to handle the data contained in the frames and deal with the semantics. These layers are specified by ISO 11898-1 through 11898-5.

7.2.2 Arbitration IDs and ECUs

As mentioned, a CAN frame with the lower value ID will get the highest priority and therefore dominate the network. All frames must have unique IDs in order to avoid errors caused by two frames transmitting simultaneously as a result of dominating the bus because of their identical IDs. If the two frames are not completely identical, errors will occur resulting in both frames' interruption. When a frame occupies the bus, all ECUs receive it and only the interested ones accept it. An ECU only accepts frames that it is configured to accept by recognizing their IDs. The choice of arbitration IDs is propriety and differs from an Original Equipment Manufacturer (OEM) to another. In addition, an arbitration ID is not the sender or the recipient's address. Rather, it is only an indicator of a message's contents and purpose.

Typically, each ECU is configured to use a certain set of IDs for its outgoing frames that would make sense for certain ECUs, and another set for incoming frames from other ECUs. For example, in Table.7.1, ECU_1 broadcasts a frame with an ID `0x002`. ECU_2 is configured to accept frames with the ID `0x002`.

ECU_1		ECU_2	
Send	Receive	Send	Receive
0x002	0x005	0x005	0x002
0x003	0x006	0x006	0x009
0x004	0x007	0x007	0x011

Table 7.1: IDs in ECUs 1 and 2

7.3 Related Work

Solutions that rely on cryptography have been proposed to provide authentication, confidentiality, and integrity measures [176, 171**?** , 172, 139]. In addition, IDS also have been proposed such as in [96, 127, 128, 154, 102, 140]. Furthermore, controls preventing attacks initiated from external devices have been introduced in [68, 17, 178, 47].

Most of the proposed solutions require ECUs to verify each arriving frame before deciding

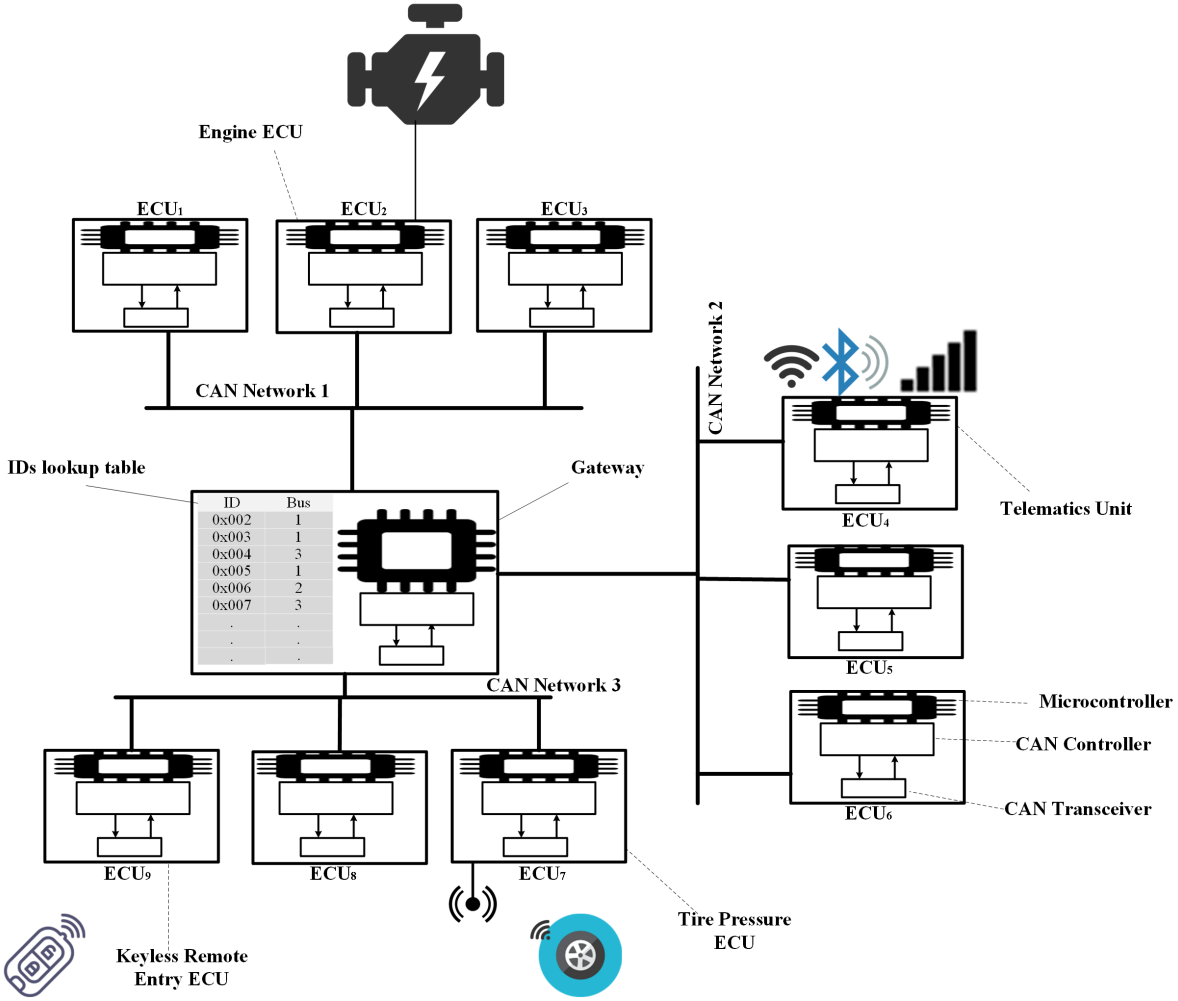


Figure 7.1: Overview of a Modern In-Vehicle Network

whether to accept it, for further processing, or drop it. This in itself poses ECUs to DoS attacks when an attacker floods the network with frames that has certain IDs but incorrect data. When this happens, a receiving ECU would check the ID first and then verify other measures such as the authenticity and integrity of the frame. This adds an unnecessary overhead in terms of the number of operations needed at the receiving side.

The most relevant work to ID-Hopping mechanism is the Identity-Anonymized CAN (IA-CAN) that Han et. al. have proposed [69]. They introduce a solution that provides authentication with minimal overhead as opposed to other CAN-based authentication approaches, e.g., [171**?** , 172], that require relatively more overhead.

Therefore, Han et. al. [69] propose an ID-anonymization approach that lets ECUs only accept legitimate frames based on their anonymous IDs. The IA-CAN anonymizes IDs in the sense that they have no meaning for an eavesdropper. With some reverse engineering skills and patience, an attacker could infer CAN IDs and then launch various attacks such as sending false information and safety-critical command injection. IA-CAN prevents such inference because the IDs are changed randomly, from the attacker's viewpoint, so that only legitimate ECUs can send and receive frames using IDs that they only can know.

IA-CAN works in three stages: 1) an ID is randomly generated for each frame before it is sent, 2) each receiving ECU has already pre-computed the ID, 3) when a frame with the expected ID arrives, the receiving ECU simply filters it in using an XOR operation. Each ID is used only once, which prevents replay attacks. The authors have four assumptions in order for their mechanism to work: 1) they assume there is only one CAN network where an attacker exists, 2) ECUs share a secret key before the communication takes place, 3) keys are securely pre-stored in a tamper-proof hardware, 4) there is a time ECU for synchronizing time.

In addition, an important feature in IA-CAN is IDs priority preservation. The authors maintain the priority bits intact in the arbitration field, and anonymize/randomize the remaining bits. The arbitration bits vary depending on the deployed standard. For example, SAE J1939 uses the first 3 bits of the 29-bit ID. In our proposal, we maintain the priority by considering all bits in the ID field.

In other words, we treat the whole ID as priority bits. Therefore, regardless of the used variant of CAN protocol, ID-Hopping should be applicable.

Using the same IDs across a large number of cars that use the same platform exposes them to a large-scale attacks that might affect a significant number of cars. As a matter of fact, Miller and Valasek [32] had the capability of attacking any car in the US that belongs to a certain fleet of cars that uses the same platform. They were able to uncover a vulnerability in a car that allowed them to take remote control over it by exploiting its cellular communication channel. Once they get in the car's internal network, they could inject CAN frames with IDs that they have already reverse engineered and known what their purposes are. Because all cars of this fleet use the same set of IDs in their CAN network, the attack was possible to be launched on more than a million cars across the US.

Therefore, another work along the lines of IDs randomization is proposed by Lukasiewycz et. al. [107]. They introduce an approach, called *Security-aware Obfuscated Priority Assignment*, that would prevent large-scale attacks on a fleet of cars that use the same platform and hence the same CAN IDs. Unlike our proposal and IA-CAN, this mechanism is aimed at a fleet of cars instead of an individual one. In addition, the IDs' priority assignment is going to be fixed in each car since the time it is manufactured. This might protect against a large-scale attacks, but not a targeted attack on one car. They use Quadratically Constrained Quadratic Program (QCQP) solving to generate obfuscated CAN IDs across different cars using the same platform. Unlike Lukasiewycz's et. al. [107] proposal, our ID-Hopping mechanism does not need to be integrated during the design phase. Instead, we claim that it should be applicable to existing CAN networks with no modification needed. The exception is the assumption of a shared secret key between ECUs and the central gateway in order to authenticate whenever IDs need to be changed.

7.4 DoS attacks

Due to the way ID arbitration works in CAN networks, DoS is very feasible. There are a number of DoS reported attacks. For example, Koscher et al. [92] disabled CAN communication from and to the Body Control Module (BCM) which resulted in a sudden drop from 40 to 0 MPH on the speedometer. In addition, this attack also resulted in freezing the whole Instrument Panel Cluster (IPC) in its current state. For example, if the speedometer was at 60 MPH before the attack, and the driver increases the speed, there will be no change in the speedometer. Hoppe et al. [77] demonstrated other forms of DoS attacks. One of which is where the attack prevents passengers from closing any opened window. Another is to disable the warning lights. The authors also performed another DoS attack on the theft alarming system, such that it will not go off during a burglary. DoS attacks can take on different forms whose impacts vary in safety-criticality such as traditional DoS, random DoS, and targeted DoS.

7.4.1 Types of DoS

7.4.1.1 Traditional DoS

An attacker could simply flood the network with frames that have the lowest IDs, i.e., `0x000`, such that their dominance of the network is guaranteed. Solving this kind of DoS is not possible at the data-link layer and above. Rather, a physical design modification is needed such that extra checks could be introduced before a high priority frame continuously dominates the network. In addition, the detection of this traditional DoS is not difficult as it will become obvious to the driver that the car is not responsive. Our proposal is designed towards a selective class of attacks that is subtle and less obvious than the traditional DoS.

7.4.1.2 Random DoS

The attacker in this type of DoS does not target a specific ECU, rather he would randomly send frames with randomly selected IDs aiming to disturb ECUs' normal operations. Although fuzzing

the network with random IDs might result in unintentional DoS, the attack is not targeted against a particular ECU.

7.4.1.3 Targeted DoS

An attacker could be interested in sabotaging certain ECUs for different reasons. For example, an attacker could target the airbag system to fail when a collision occurs. This could cause a life-threating consequences to the driver and passengers. Targeted DoS is a subtle attack and challenging to detect, let alone prevent. Therefore, targeted DoS is the focus of this paper.

7.4.2 Attacker Model

In this chapter, we consider a variant of DoS attacks where an attacker targets a certain ECU and aims to prevent its frames from using the network and eventually from arriving to the desired destinations. For example, in Fig.7.2, the engine's ECU (ECU_2) sends its periodic CAN frame containing some information to be displayed to the driver, such as speed and temperature. The frame has an arbitration ID with a value of `0x006`. At the time ECU_2 is sending this frame, it has the highest priority in *CAN network 1* and therefore should successfully occupy the network and reach the gateway so it can forward the information to the instrument cluster's ECU in another CAN network.

However, an attacker is assumed to have compromised ECU_3 and configured its controllers to be monitoring the network. ECU_3 gets triggered and floods the network with a frame whose ID has a lower value such as `0x005` whenever a frame with the ID `0x006` is observed in the network. This attack scenario could miss the first frame with the ID `0x006`, but the remaining frames are going to be affected. This will force ECU_2 to retract from sending `0x006` frame due to the network's occupancy with a higher priority ID, i.e., `0x005`. The latter frame has higher priority as it has the lowest value, and thus can dominate the bus. This attack ensures that ECU_2 cannot use the bus resulting in an unfair usage of resources by other ECUs, more particularly, the attacker's. This attack is often called *fairness/starvation attack*.

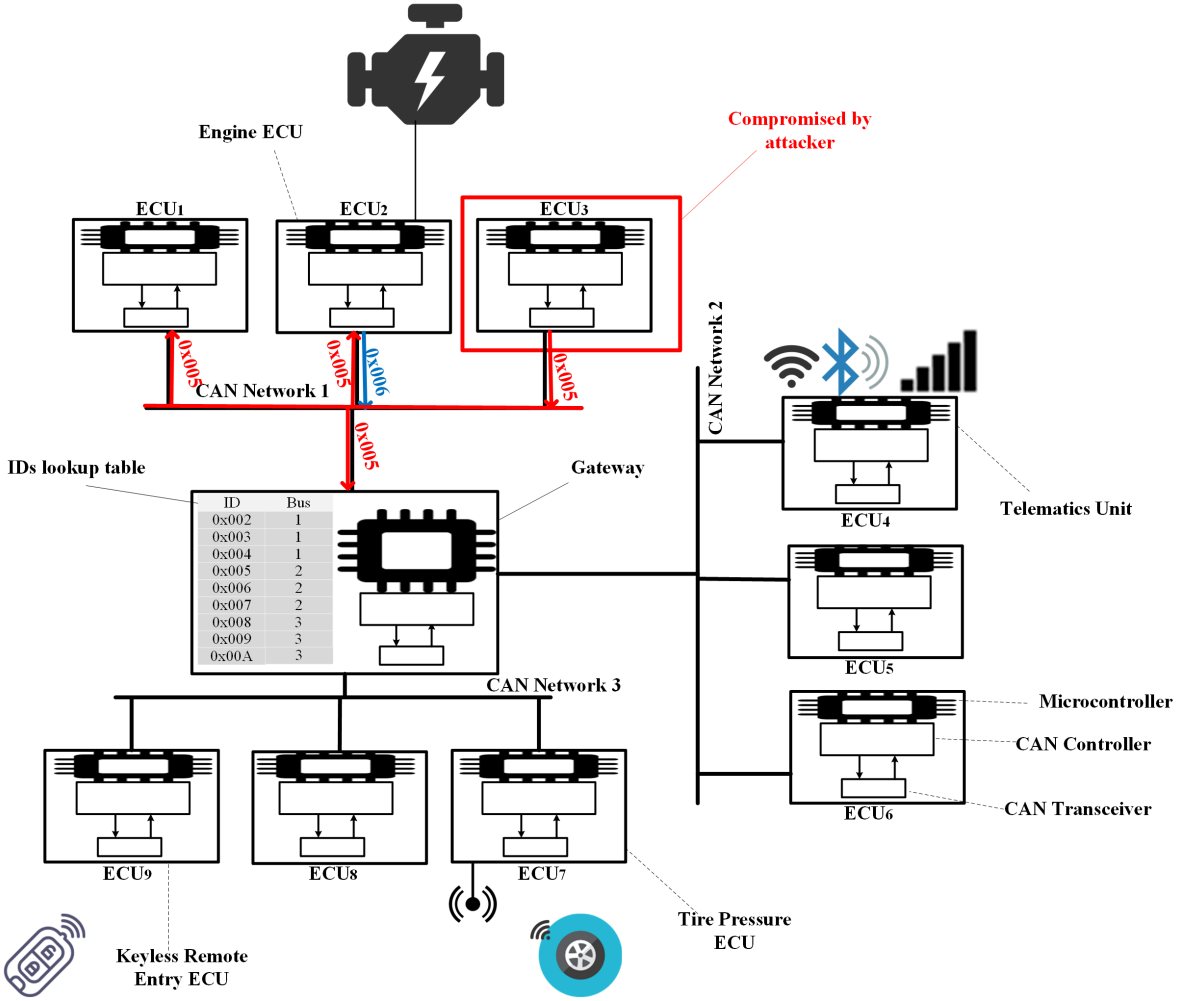


Figure 7.2: Scenario of a targeted DoS attack

7.5 ID-Hopping Mechanism

7.5.1 Requirements

The mechanism should satisfy the following requirements:

1. No modifications needed for current CAN protocol.
2. The mechanism must preserve the same level of priority for current IDs configured by OEMs.
3. Minimum to no performance overhead on the network.

7.5.2 Overview

When an ECU becomes a target of a DoS attack, it should detect that it is under attack and then switch its frames' ID to the proposed ID-Hopping mechanism. We propose configuring ECUs with two kinds of IDs: 1) IDs used during normal operations and 2) IDs used when an ECU or more is under a targeted DoS attack. The first type is currently deployed in all cars, whereas the latter is our proposal towards thwarting targeted DoS attacks against certain ECUs.

For the sake of simplicity, let's say that ECU_1 is sending a frame with ID `0x002`. An attacker immediately floods the network with a frame that has `0x001` ID in order to win the arbitration and cause the `0x002` frame to retract. After a few attempts of sending the `0x002` frame, ECU_1 notices that the frame loses the arbitration repeatedly due to the existence of a higher ID. Also, a host-based or a gateway-based mechanism should be implemented to recognize that the frame with `0x001` ID only gets triggered whenever ECU_1 sends `0x002` ID. These two patterns indicate that ECU_1 is under a targeted DoS attack.

To overcome such an attack, ECU_1 should switch to ID-hopping mode where it uses an alternative set of IDs in order to confuse the attacker. Now, ECU_1 sends a frame with `0x009` ID which carries the same data as the previous frame, except that it uses a different ID as shown in Table 7.2. This should thwart the attacker from targeting the ECUs until she learns the alternative set of IDs.

The attacker needs close monitoring and reverse engineering in order to uncover the alternative IDs. When this happens, a new set of alternative IDs is generated.

ECU_1		
ID	Alternative ID	Receive
0x002	0x009	0x005
0x003	0x00A	0x006
0x004	0x00B	0x007

Table 7.2: ECU_1 IDs

ECUs that receive frames from ECU_1 need to be aware of the alternative IDs in order for the car to function normally. We achieve that by having the alternative IDs in every ECUs' lookup table as long as the ECU is configured to receive the original ID before an attack takes place. For example, Table 7.3 shows that ECU_2 has the alternative ID 0x00C as well as the original ID 0x005.

Finally, when ECUs switch to the ID-hopping mode, the alternative IDs must not compromise the original priorities of the original IDs. ID-hopping mode must preserve the overall priority across all IDs. In other words, each original ID will have an equivalent alternative ID in terms of priority. So if two original IDs 0x100 and 0x101 are used in the alternative mode, they will be 0x111 and 0x112. Hence, the priority is still preserved even though different IDs are used. This can be achieved when all ECUs, including the gateway, use the alternative IDs when an ECU or more is under attack.

ECU_2		
ID	Alternative ID	Receive
0x005	0x00C	0x002
0x006	0x00D	0x003
0x007	0x00E	0x04

Table 7.3: ECU_2 IDs

7.5.3 The Algorithm

In the gateway, ID-Hopping algorithm takes a list of used IDs as input to generate the alternative IDs. Each resulting ID is unique and not used by any ECU in the network. This generation is a

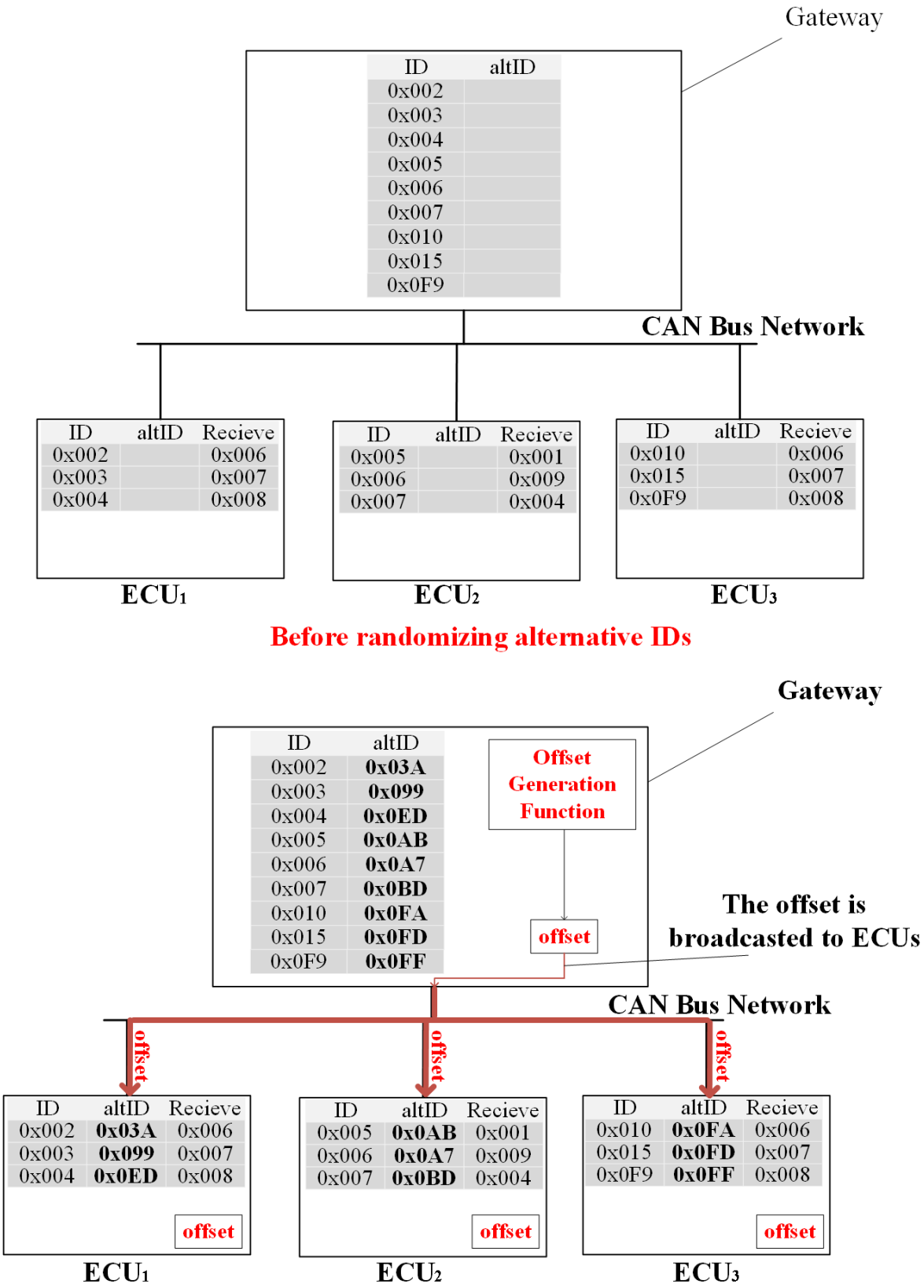


Figure 7.3: Overview of the network before and after the ID-Hopping takes place

result of adding a carefully selected offset to the current used IDs resulting in a new list of IDs that preserves the same level of priority as their original counterparts. Then each ECU receives the offset, in a secure way, and then adds it to every original ID it has resulting in an alternative ID corresponding to each original ID. Finally, when an ECU detects a targeted DoS attack, ID-Hopping is activated and all ECUs switch to the alternative IDs. We assume that ECUs share a secret key with the gateway in order for the gateway to broadcast the offset securely.

7.5.3.1 Phase One: IDs Generation

When an attack is detected, either by the gateway or an ECU, the ID-Hopping mode is activated by the ECU under attack. The gateway assigns a new and randomly generated ID for each used ID. The assigned IDs will not compromise the existing priorities configured by the car's manufacture. In the deployed algorithm, the basic idea is to shift the values of current IDs by a random offset. The offset itself must satisfy two conditions: 1) $offset \geq$ `0x001` and 2) $offset + maximum \leq$ `0x7FF`.

These two conditions ensure that no alternative ID results in an out-of-bound value, i.e., more than `0x7FF`, which is upper bound for an 11-bit ID. Once the random offset is generated, each ECU generates the alternative IDs by adding this offset to each one of the original IDs. The offset is added even with IDs that are not targeted in order to maintain the overall posture of priorities across the network that the manufacturer has intended.

Before adding the offset to the original IDs, we check whether the value of the highest ID is `0x7FF` . If it is not, we proceed with the described procedure above as shown in Fig.7.4. However, if the maximum ID is indeed `0x7FF`, this means there will be at least an alternative ID out-of-bound and will have more than 11-bit value. Therefore, the algorithm makes some additional steps before generating the alternative IDs as shown in Fig.7.5.

Firstly, we take the lowest available set of IDs and add the random offset to each ID. Although it could be enough to use these low value available IDs as alternative ID, we need to avoid detectable patterns in the alternative IDs such as being consecutive numbers. This is the case because our algorithm deals with IDs that have been already assigned by a manufacturer in a proprietary

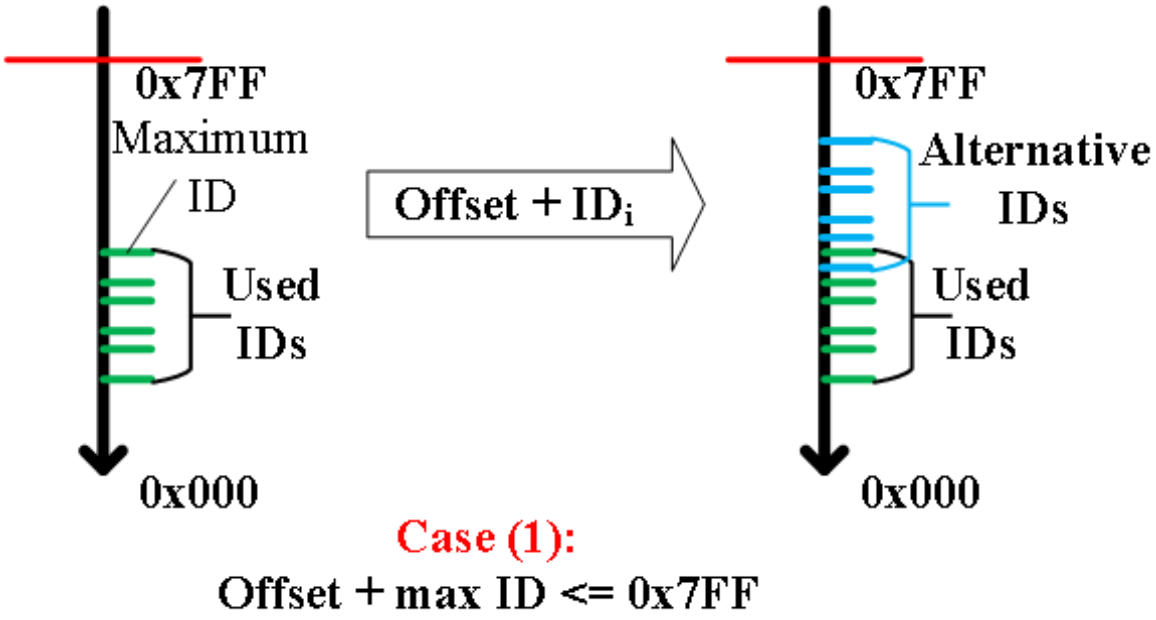


Figure 7.4: Alternative IDs' Generation: Case 1

manner. Therefore, we cannot predict how they have been assigned. There is one case though were the resulting alternative IDs are in sequence. Such pattern is expected if the value of the highest original ID is `0x7FE`, which is the upper bound `0x7FF` - 1. The algorithm can only generate offset = 1 because it is the only available number that satisfies the two conditions discussed above. However, we can easily change the offset requirements to ensure it has a value greater than one.

7.5.3.2 Phase Two: IDs Assignment

When the gateway generates the alternative IDs, all ECUs need to know about the alternatives of their IDs before an attack takes place. We considered two potential ways to distribute the alternative IDs: 1) sending a list of the corresponding alternative IDs to each individual ECU or 2) sending the generated offset instead, and then each ECU calculates the alternative IDs by adding the offset to each original ID. The first approach takes n transmissions, where n is the number of alternative IDs to be sent to each ECU. The latter approach only requires one transmission because the gateway broadcasts the offset and all ECUs receive it simultaneously. Therefore, we chose the latter as it significantly requires less network usage.

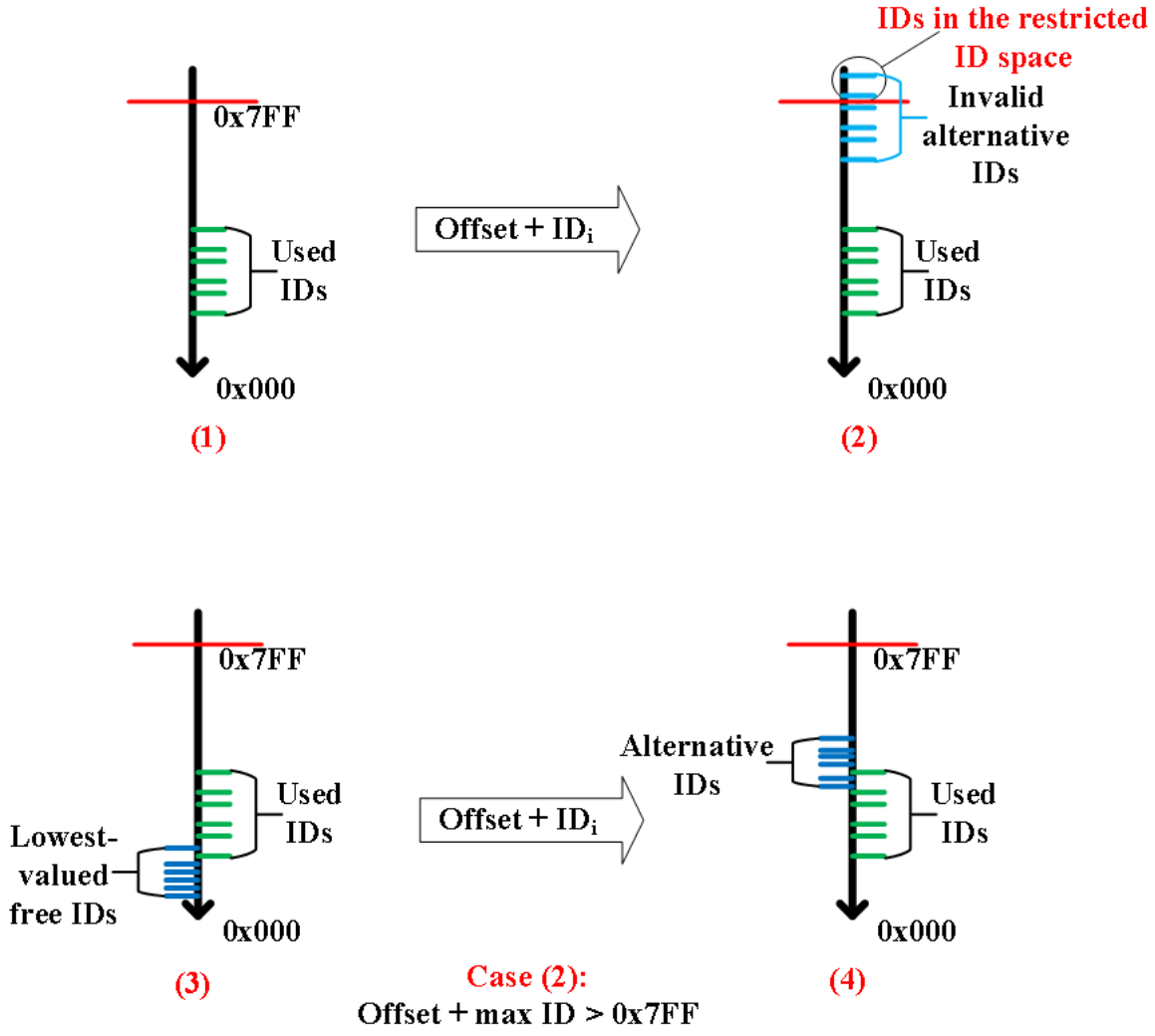


Figure 7.5: Alternative IDs' Generation: Case 2

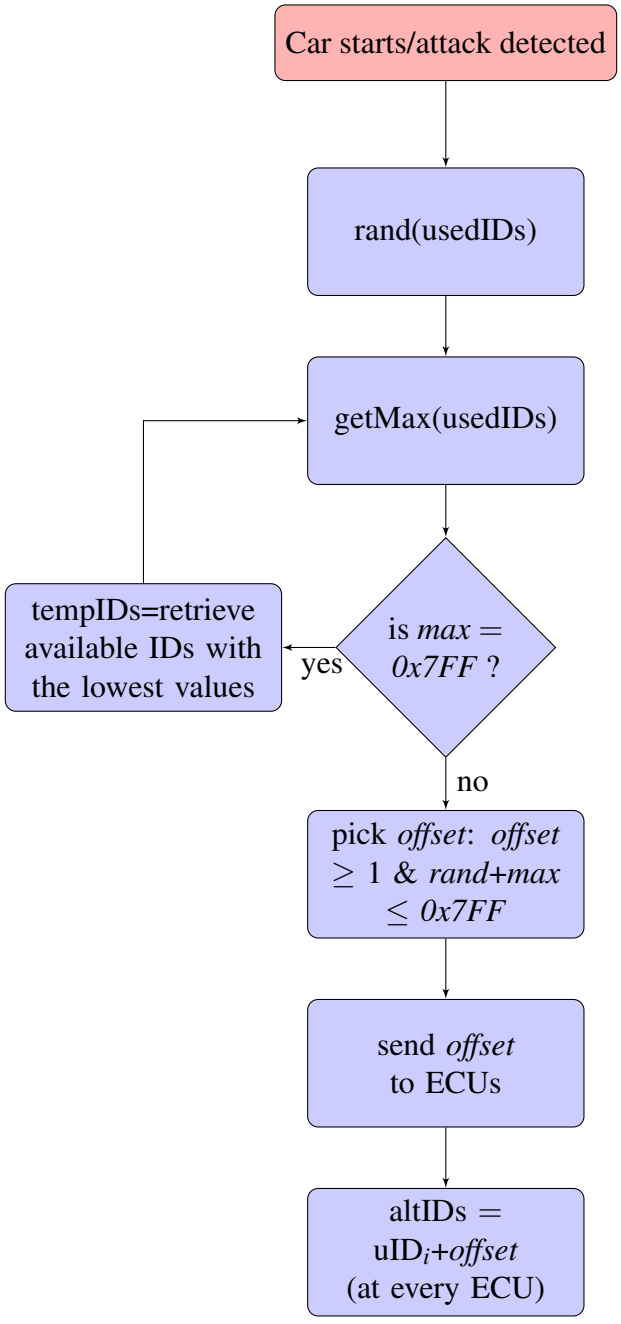


Figure 7.6: Flowchart of the ID-Hopping Algorithm

One might ask how the offset is sent to ECUs. What if an attacker pretends to be a legitimate ECU, which could potentially allow her to receive the offset and easily deduce the alternative IDs of the targeted original IDs? One of the most straightforward ways is to establish a secure connection between the gateway and ECUs. The encryption could be asymmetric, i.e., with a public/private key pair, or symmetric, i.e., with a shared secret key. The latter is a more efficient approach that does not introduce network overhead, especially if the encryption is only used for sending the offset. Therefore, we assume that an encryption mechanism is in place for sending the offset.

Because encryption operations are relatively expensive and could add network overhead, we limit the encryption to only this operation, i.e., sending the offset to ECUs once every time alternative IDs are generated. This operation is required at least once when the car starts, and once again when an attack is detected. Therefore, when each ECU receives the encrypted offset, encryption is no longer needed and the network should not suffer from any overhead.

7.5.3.3 Phase Three: ID-Hopping Mode

When an ECU detects that one (or more) of its IDs is no longer able to occupy the network, the ECU assumes that it is under a targeted DoS attack. Therefore, it resorts to the ID-Hopping mode where only alternative IDs are used. Receiving ECUs and the gateway will receive the same frame but with an alternative ID. As a result, they notice that an alternative ID is used, and hence they switch to ID-Hopping mode as well. The gateway broadcasts a message to all ECUs, including the unaffected onces, telling them to switch. The message is encrypted and containing a flag that indicates that ID-Hopping mode is activated. Therefore, all receiving ECUs switch to the ID-Hopping mode. The reason we force all ECUs in the network to switch is to preserve the IDs' priorities assigned by the car's manufacturer. If we only allow one alternative ID to be used, the overall IDs' priority order might get compromised. A flowchart of the algorithm is shown in Fig. 7.6. In the chart, *usedIDs* is a list of the original IDs, *altIDs* is a list of alternative ID, and *tempIDs* is a list of temporary IDs.

7.6 Security Analysis

7.6.1 Attack Prevention

We implemented a targeted DoS attack with the testing platform explained in Sec.7.7 and the ID-Hopping successfully detected the attack and made all ECUs switch to their alternative IDs and the network behaved normally. The mechanism proved its effectiveness to prevent targeted DoS attacks.

7.6.2 Collision-Free

Our evaluation showed that when all ECUs switch to the alternative IDs, the network is collision-free. This is because the ID-Hopping mechanism guarantees that there is no case where two ECUs have the same alternative IDs unless they had the same original ID assigned by the OEM.

7.6.3 Priority Preservation

ID-Hopping successfully maintains the priorities given to the assigned original IDs as long as all ECUs switch to the alternative IDs mode. This is because the mechanism is designed to carefully calculate an offset that guarantees two conditions: 1) original IDs' priorities are preserved and 2) alternative IDs are unique.

7.6.4 Protocol-Independent

Compared to Han et al.'s work [69] which needs to be configured differently across different CAN variants, ID-Hopping is independent of the used CAN variants because it treats the ID as a prior

7.6.5 Efficiency

ID-Hooping is more efficient than other mechanisms such as IA-CAN due to the minimal amount of encryption operations. Encryption is only used in two cases: 1) when the gateway broadcasts

the offset to ECUS and 2) when "swich to alternative IDs" frame is sent, either by the attacked ECU or the gateway.

7.6.6 Limitations

7.6.6.1 Original Frames are intact, except their IDs

One limitation in the ID-hopping mechanism is that when ECUs switch to the ID-hopping mode, the original frames remain with no modifications. In other words, the only change is in the IDs. This poses the risk for the attacker to cross match the contents of the frames and infer the alternative ID for the attacked frame. A possible solution is to modify the contents of the original frame, as well as the ID, possibly by shifting the values by the same offset used for IDs.

7.6.6.2 Attacker retrieves the offset

It could possible for an attacker to compromise an ECU and get access to the encrypted offset. We assume that such attack needs physical access to the ECU in order to succeed. To prevent such an attack we need to rely on tamper-proof ECUs that are physically-protected.

7.6.6.3 Potential limited CAN ID space

Because free CAN ID space is only known for OEMs, our mechanism might fall short when implemented in a network with little or no free CAN ID space. A potential solution that could be considered is the use extended CAN frames, where we can use 29-bit IDs rather than 11-bit. Better yet, we could improve our design to use 11-bit IDs by default, and 29-bit IDs when space is limited. Therefore, the mechanism can scale when free ID space is limited or the number of ECUs is large.

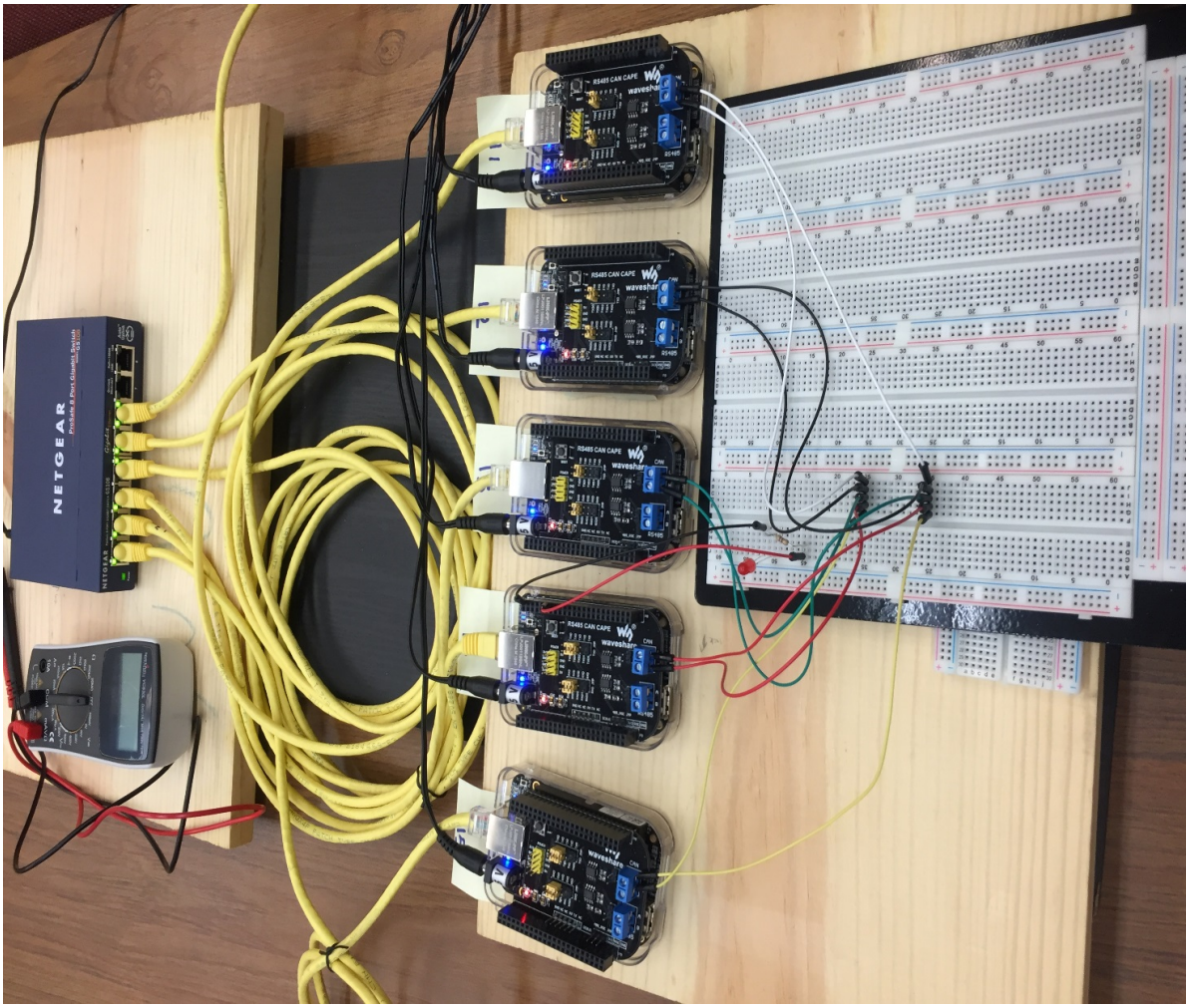


Figure 7.7: Testing Platform

7.7 Evaluation

Fig.7.7 shows our test platform which consists of five BeagleBone Black (BBB) microcontrollers, five CAN Capes, a breadboard, a switch, a multimeter, and an external LED. Four of the BBBs simulate regular ECUs, and the fifth simulates a gateway. The switch is used for accessing and configuring the BBBs.

The algorithm successfully generates alternative IDs that substitute the original IDs. Fig. 7.8 shows a real example of three of the ECUs' IDs before and after applying the offset to the original IDs. It appears from the figure that the priority of the original IDs is maintained in the alternative IDs. Furthermore, the alternative IDs maintain the same priority level as in the original IDs. The

current version of ID-Hopping is designed for a single CAN network and the size of IDs is 11 bits. We plan to perform a more realistic evaluation on a testing bench and test the mechanism with the extended 29-bit ID.

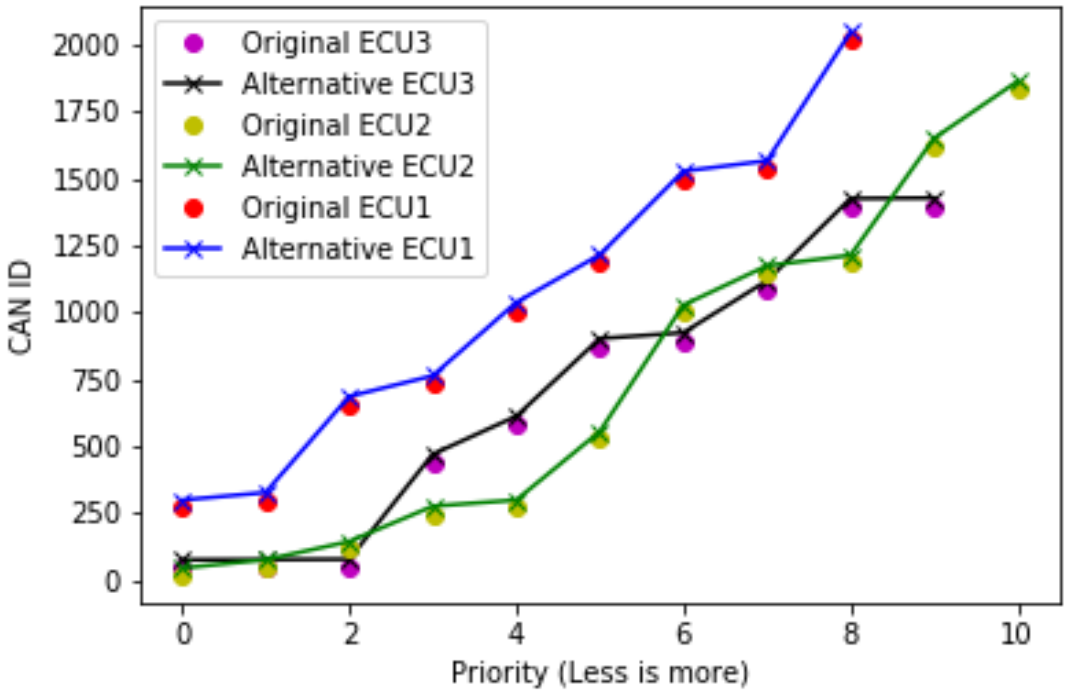


Figure 7.8: Original vs. Alternative IDs

7.8 Summary

While CAN is prone to various attacks due to the lack of security measures, smart cars are increasingly exposed to new communication channels. This makes it critical to design short-term and long-term security solutions. Our work serves as a short-term solution that works with current CAN networks such that only minimal modification is needed to the current infrastructure deployed in todays' smart cars. Long-term solutions take time to be adopted and require security considerations from early design phases.

In this chapter, we propose ID-Hopping mechanism to defend against targeted DoS on CAN

networks. We design the algorithm to provide ECUs with an efficient approach to generate alternative IDs that can be used when a DoS takes place. Of particular significance is that these alternative IDs maintain the same priorities of the original IDs regardless of the used CAN variant. In addition, the gateway only needs to send a single frame containing an offset that is carefully calculated to ECUs so they can calculate their alternative IDs.

We evaluated ID-Hopping with a testing platform consisting of five microcontrollers, simulating ECUs and a gateway, and a breadboard, simulating the CAN bus. The evaluation shows successful alternative IDs' generation, DoS detection, and prevention.

ID-Hopping holds a promising solution to defend against CAN IDs' reverse engineering, which is one of the most common attacks against CAN networks. Therefore, a future direction could include improving ID-Hopping to work with 29-bit CAN IDs and tackling multiple simultaneous targeted DoS against single/multiple ECUs.

Chapter 8

Firewall To Defend Against Denial Attacks on CAN

8.1 Introduction

In the previous chapter, a new class of attacks on CAN was introduced and a new attack was proposed and shown to be superior to the discussed attacks. This new class exploits the consequences when participating ECUs do not adhere to CAN standards. For example, an attacker could inject carefully-selected bits to achieve various types of denial attacks. The common theme of the discussed denial attacks is the disobedience of CAN standards. For example, an attacker does not wait for a specific minimum amount of time before attempting to start submission. Instead, an attacker could inject anything at any time thanks to the absence of CAN controllers or the misuse thereof.

Conventional controls do not address these attack adequately because they mainly work at a higher level than where these attacks are. In other words, current solutions first require all ECUs to accept a frame and then process it to verify its validity. In addition to the introduced unnecessary latency and waste of resources, attacks that does not require transmitting complete frames, bypass such controls. Therefore, a need for countermeasure that addresses this kind of attack emerges.

In this chapter, we propose a novel automotive firewall that aims to prevent the discussed

denial attacks in the previous chapter. The firewall could be located between a single or multiple potentially-malicious ECUs and the rest of the bus.

8.2 CAN Transceiver-Based Firewall

Since most of the surveyed and proposed attacks rely on either the misuse CAN controllers or bypassing them, a countermeasure needs to behave in a similar fashion. In other words, we propose using a countermeasure that does not rely on a CAN controller nor adhere to CAN standards. Instead, it uses an MCU that is located between a potentially vulnerable ECU and the bus. A CAN transceiver is needed so the MCU can analyze outgoing bits from ECUs that could be potentially become a source of attacks. The concept of analyzing serial communications with software is known as *bit-banging*.

8.2.1 Firewall Architecture

The nature of the attacks we aim to prevent requires a firewall that closely monitors traffic coming from the attacker. However, because CAN bus is broadcast-based, it is not possible to identify the source of any transmission on the bus with the current bus's architecture. Consider the bus example in Fig. 8.1, ECU_2 is malicious and could launch various attacks. If we introduce a firewall to the bus by simply plugging it to the same bus, it can not know that ECU_2 is 'the source of an attack should it occur. We need a firewall that can identify the source of an attack with high certainty.

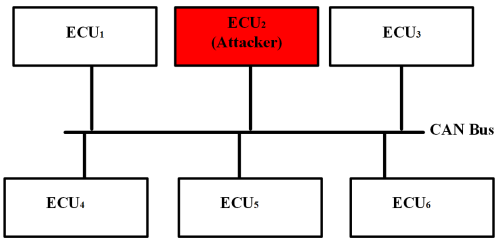


Figure 8.1: Typical CAN Bus

In order to be able to identify a source of an attack, we propose a slight modification on the

architecture in Fig 8.1 so that an attacker's traffic could be identified and prevented. In Fig. 8.2, we see two instances of the modified architecture. In the left example, a firewall is located between ECU_2 and the rest of the bus, whereas it is located between two additional ECUs in the example in right. In the first, the firewall can know with 100% certainty when ECU_2 is the source of the top port's traffic. On the other hand, in the right example, the firewall has a 33% probability of knowing the source of a frame or malicious injections. In either case, the firewall should be able to detect and prevent attacks reaching to CAN_1 originating from CAN_2.

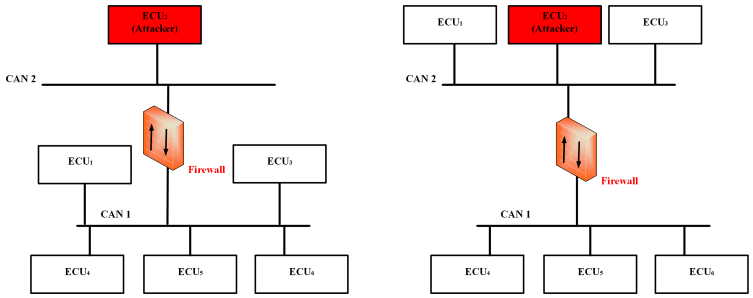


Figure 8.2: Firewall Architecture

8.2.2 Firewall Requirements

8.2.2.1 Low Latency

CAN-based systems are typically time-critical, and not only any added latency to the network is very undesired, but it could also result in safety-related incidents. Therefore, any controls need to make real-time decisions. Our firewall's rules need to be simple enough to accommodate real-time requirements. It should be seamless to honest ECUs so their communications do not get affected. Meanwhile, an attacker's malicious activities should be detected and prevented with the lowest possible impact on the honest ECUs.

8.2.2.2 Low Cost

For an easy adoption, the firewall needs to be as cheap as possible.

8.2.2.3 Compatibility

The firewall needs to work with any CAN-based network with minimum modifications.

8.2.3 Firewall Goals

We analyzed the aforementioned attacks and deduced a set of attack patterns that we desire to prevent with the least amount of false positives or negatives. As each category of the aforementioned attacks shares common features, we categorize the attacks to prevent as the same categories of the attacks: *bus, ECU, arbitration, and frame denial attacks*.

8.2.3.1 Bus Denial

In the current version of the firewall, we focus on preventing the continuous dominant state attack, discussed in Sec. 6.2.1.1. The firewall should prevent any injected dominant bits that violate CAN standards. For example, a dominant bit that is not a SOF should not be allowed. On the other hand, because 0x0 ID flooding attack, discussed in Sec. 6.2.1.1, do not violate CAN standards, it is slightly harder to prevent. If the attacker uses a conventional ECU that follows CAN standards, then the firewall could launch a bus-off attack against the malicious ECU. However, if the attacker's ECU does not follow the standards, then further measures are needed to prevent such an attack.

8.2.3.2 ECU and Frame Denial

Attacks discussed in Sec. 6.2.2 and 6.2.3.2 violate CAN standards by injecting a dominant bit to override a carefully selected recessive bit in the target frame while in transmission. To prevent such an attack, we need to block any injections from the attacker's side during any transmission on the main bus. Therefore, the firewall should prevent any transmission if a frame is being transmitted. A frame on the main bus is assumed to have already won arbitration and no other ECUs are supposed to send unless an error is detected. Since other ECUs on the main bus detect errors, the potential attacker's ECU will not be able to send error flags. It is a trade-off that seems to be worth preventing

the attacker from sending error flags.

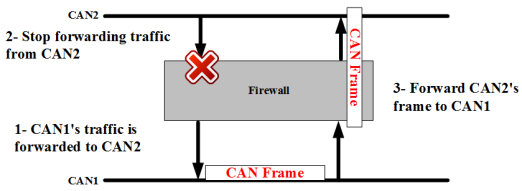
8.2.3.3 Arbitration Denial

The attack discussed in Sec. 6.2.4 is realized by injecting a dominant bit to replace a recessive one in the arbitration ID of the target frame. Prevent this attack is realized by disallowing single dominant bits during the arbitration phase.

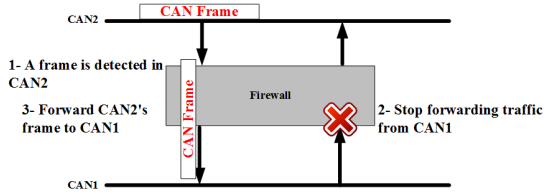
8.3 Algorithm

In this section, we introduce our algorithm for preventing the denial attacks. As shown in Fig 8.2, the firewall faces two CAN buses: 1) the main bus where the honest ECUs reside (CAN_1) and 2) the potentially-malicious bus where an ECU or more reside (CAN_2). The aim is to allow traffic to flow from CAN_1 bus to CAN_2 bus and vice versa without any collisions. The algorithm consists of two main parts:

1. Traffic forwarding from CAN_1 to CAN_2.
2. Traffic forwarding from CAN_2 to CAN_1.



Case 1: CAN1 to CAN2



Case 2: CAN2 to CAN1

Figure 8.3: Firewall Demonstration

Since the attacks discussed in Chapter 6 exploit the absence of CAN controller. They could inject malicious traffic arbitrarily such that CAN standards are not followed. The proposed firewall ensures that once the arbitration phase is won, no ECU on the potentially-malicious bus can interrupt the transmission. Fig 8.3 shows the mutually-exclusive forwarding scheme. We see in the top case, *Case 1*, that when the firewall detects a CAN frame on CAN_1 bus, it disables the interrupt waiting for a frame on CAN_2's side before forwarding the traffic. The second step is critical to avoid an unnecessary infinite loop. Once CAN_1's frame is forwarded successfully to CAN_2, the interrupts are enabled again on both interfaces. Algorithm. 3 illustrates the steps of forwarding between the two buses.

Algorithm 3 Allow and Block Traffic Between CAN_1 and CAN_2

```
1: while true do                                              ▷ Infinite loop
2:     if A frame is detected on CAN_1 then
3:         Disable frames' detection on CAN_2
4:         Forward traffic to CAN_2
5:     else if A frame is detected on CAN_2 then
6:         Disable frames' detection on CAN_1
7:         Forward traffic to CAN_1
8:     end if
9: end while
```

8.4 Evaluation

The attacks that we aim to prevent with the proposed firewall require a bit-by-bit analysis due to the lack of CAN standards adherence from the attackers. Therefore, we use the same mechanism in the attack proposed in Chapter .6. In other words, we build on CANT to monitor and allow/block the traffic coming from the attacker's ECU. As shown in Fig. 8.4, the proposed firewall consists of 3 components: a microcontroller and two CAN transceivers. One transceiver interfaces the attacker's

ECU, where the other interfaces the bus. The microcontroller in the middle analyzes traffic coming from the attacker's ECU and blocks it when an attack is detected. On the other hand, the traffic coming from the bus is forwarded to the ECU without any intervention as we assume that the bus is not malicious and therefore will not transmit malicious traffic. We only monitor traffic coming from the attacker's ECU to protect the bus, i.e., other ECUs from the ECU behind the firewall.

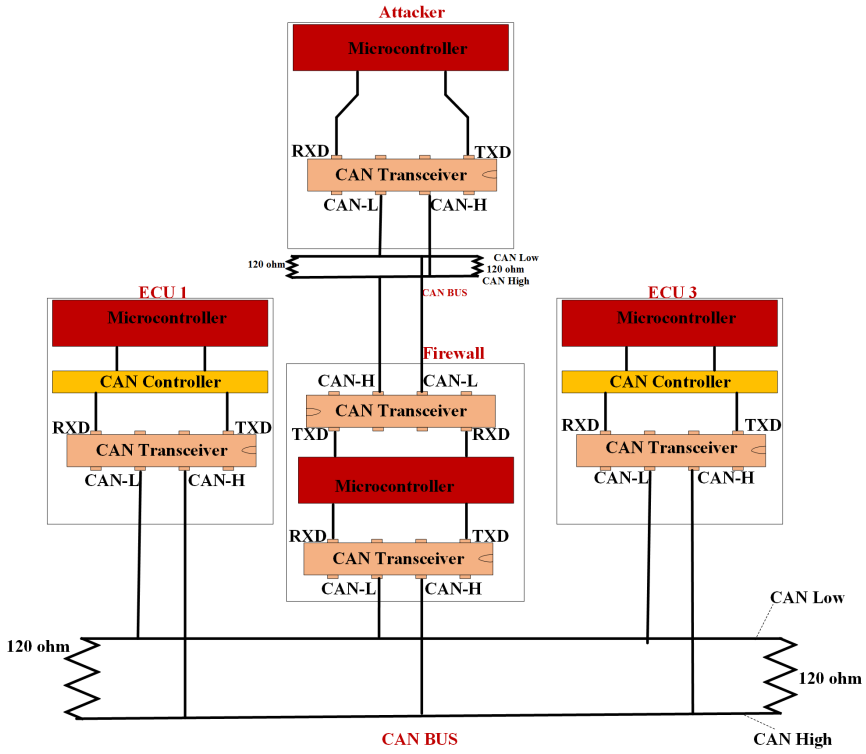


Figure 8.4: Firewall Example

8.4.1 Testing Platform

The testing platform consists of two STM32 Nucleo-144 microcontrollers, one acts as an attacker and the other as the firewall, and four BBB microcontrollers as conventional ECUs.

As shown in Fig **??** the attacker is separated from the main bus and is connected to a small CAN bus that works as a bridge with the firewall. PA15 and PB12 GPIOs are connected to the CAN transceiver's Tx and Rx pins, respectively.

The firewall has more connections because it interfaces tow CAN buses, CAN_1 and CAN_2.

Four GPIOs are connected to two CAN transceivers as follows: PXXX and PXXX are connected to the main bus's CAN transceiver PXXX and PXXX are connected to the attacker bus's CAN transceiver

8.4.2 Traffic To Attacker (CAN_1 to CAN_2)

Our experiments show no delays caused by this forwarding. An important observation to note is that the main bus has to consist of two ECUs or more for successful transmission of frames because they require the ACK bit which is sent by at least one receiving ECU on the same bus.

When the firewall synchronizes with the main bus, it waits for the SOF's interrupt to start forwarding to the attacker's bus.

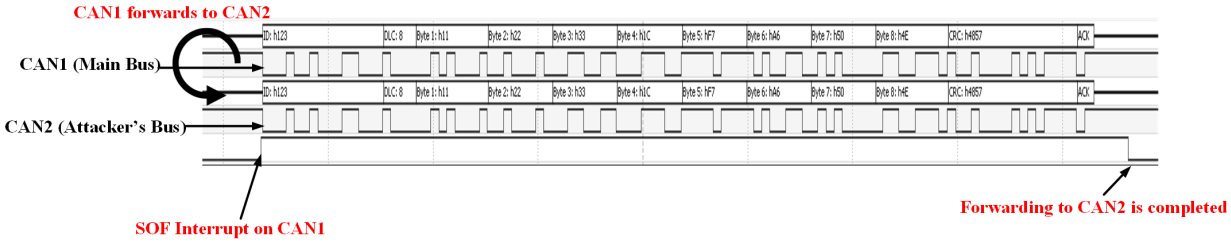


Figure 8.5: Firewall Forwards Traffic From CAN1 to CAN2

8.4.3 Traffic From Attacker (CAN_2 to CAN_1)

Another interrupt is configured on a different GPIO to detect an SOF from the attacker's side. Once it is detected, the traffic coming from the main bus is blocked and the attacker's traffic is forwarded to the main bus. No forwarding is allowed while a frame is being transmitted, and forwarded, from the main bus to the attacker's. This prevents attacks 1, 3, 4, 5, and 6. The attacker cannot arbitrarily inject dominant bits while a frame is being transmitted on the main bus.

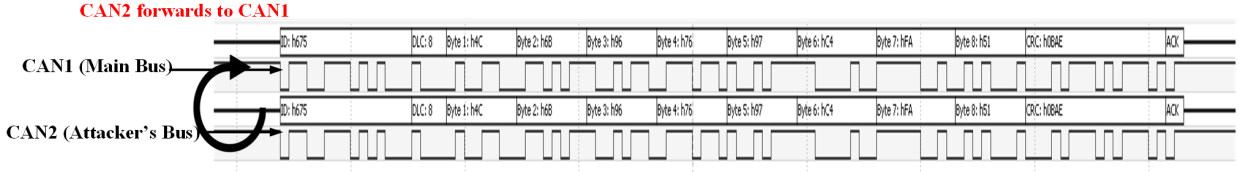


Figure 8.6: Firewall Forwards Traffic From CAN2 to CAN1

8.4.4 Prevention of Bus Denial

The firewall successfully prevents an attack from launching the bus denial attack discussed in Sec 6.2.1.1. The firewall prevents injecting arbitrarily to the bus. Any attempts from the attack to inject dominant bits at arbitrary times are prevented.

8.4.5 Prevention of ECU and Frame Denial

In Fig. 8.7, the bus-off attack, discussed in Sec. 6.2.2.2, starts at time $\approx$ 40 ms and the firewall is not activated. TEC at the target ECU increases until it reaches more than 127 and 255 causing its error state to change to Error-Passive and then Bus-Off, respectively. When the firewall is activated at time $\approx$ 80 ms, we see a sharp drop in TEC until the error state is back to Error-Active. We then turned the firewall off for a brief period of time and the same behavior occurs until we reactivate the firewall to prevent the attack.

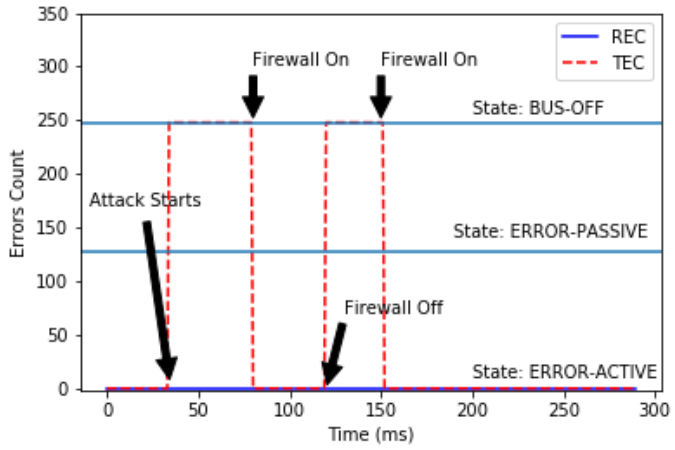


Figure 8.7: Bus-Off Attack with and Without the Firewall

8.4.6 Prevention of Arbitration Denial

Fig. 8.8 demonstrate the effectiveness of the firewall against the arbitration denial attack discussed in Sec. 6.2.4. We see that at time $\approx$ 30 ms REC starts to sharply increase until it stops increasing at the Error-Passive state. When we activate the firewall, REC decreases until the error state goes

back to Error-Active. The reason that TEC was not affected is when a CAN ID loses arbitration due to the injected bit by the attacker, it switches to a receiving mode. Then the interrupted ECU that lost the arbitration does not complete the interrupted frame causing an uncompleted frame on the bus. In other words, the interrupted frame violates CAN standards such that the frame is not terminated. As a result, one or more of the receiving ECUs generates an error flag caused by a form error.

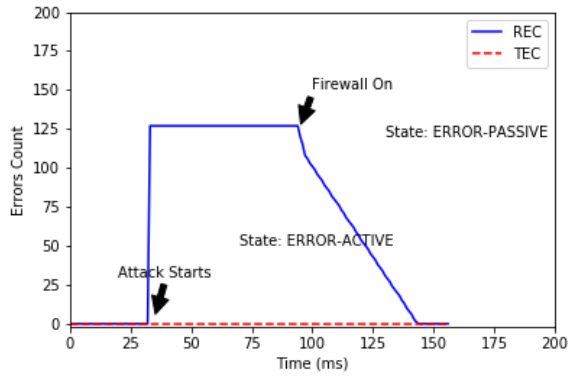


Figure 8.8: Firewall with The Arbitration Denial Attack

8.4.7 Prevention of The Stealthy Targeted Arbitration Denial

Finally, we show the effectiveness of the firewall against the proposed stealthy targeted arbitration attack that does not affect the error states as we show in Fig. 6.16. Since the error states do not change during the attack, we show the impact on the targeted CAN ID 0x777. In Fig. 8.9, once the attack starts, 0x777 ID disappears from the bus and a lower ID replaces it every time it tried to retransmit. This ID is 0x776 which results from replacing the least significant bit in the target CAN ID with a dominant bit. When the firewall is activated, the ECU sending the target0x777 is able to transmit it even though the attacker is still attempting to attack it.

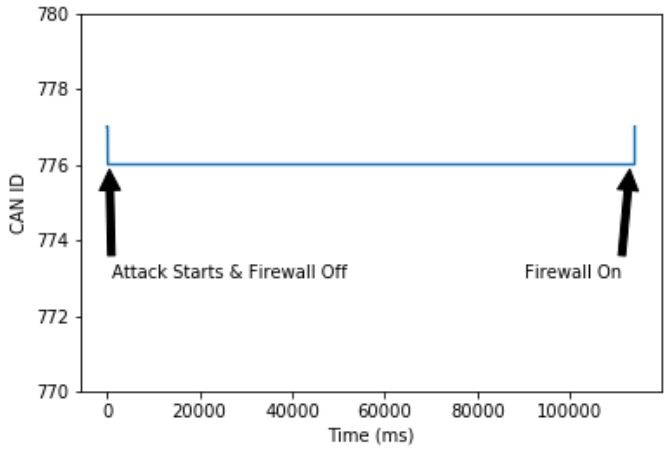


Figure 8.9: Firewall and The Stealthy Targeted Arbitration Denial

8.5 Related Work

Current solutions assume that an attacker uses an ECU that adheres to CAN standards in terms of transmission and receiving. Therefore, most of them detect and prevent malicious frame based on their CAN IDs. However, the frame-less attacks discussed in Chapter 6 can not be detected nor prevented with such solutions [141]. Therefore, the countermeasures for similar attacks are relatively limited.

Cho et al. [38] proposed a three-stepped approach to detect and prevent the bus-off attack discussed in Sec. 6.2.2. When a minimum occurrence of two consecutive active error flags during a transmission of a CAN frame, followed by a transmission of a frame with the same CAN ID, a bus-off attack is detected. To prevent it, the attacked ECU resets to avoid increasing its TEC to 255 and therefore, switching to a bus-off state. This countermeasure ensures the targeted ECU preserves its error-active state and thus, transmit and receive successfully. This solution is fine-grained towards thwarting the bus-off attack only. It does not address other attacks like our proposed firewall.

Another solution that to the bus-off attack is to counter-attack the attacker's ECU. Souma et al. [160] proposed to perform a bus-off attack against the malicious bus-off frame itself. This will cause the attacker's TEC to keep increasing until it reaches 255, and thus, a bus-off state. This

solution is also specific to bus-off attacks, unlike our firewall that targets a plethora of attacks. In addition, this solution assumes that the attacker launches the attack using an ECU with a CAN controller so TEC could be targeted. However, the countermeasure is defenseless against CAN controller-less attacks.

8.6 Summary

In this chapter, we introduced a novel CAN firewall for addressing attacks on CAN that violate CAN standards and misuse the error-handling mechanism to achieve various malicious goals. The firewall is placed between any potentially-malicious ECU and the rest of the bus. The evaluation shows the effectiveness of the firewall against the attacks discussed in Chapter. 6 with no latency. The firewall opens up new opportunities to address a plethora of attacks that either misuse CAN design or use bit-banging for achieving their goals. We build on CANT, the open source tool used for the Stealthy Targeted Arbitration Denial Attack.

Chapter 9

Firewall-Complementing Intrusion Detection System (IDS)

9.1 Introduction

The current version of the firewall prevents any injection that occurs after the arbitration takes place in the main bus. This prevents the ECU denial attacks in Sec. 6.2.2, frame denial attacks in Sec. 6.2.3, and the arbitration denial attacks in Sec. 6.2.4 and our proposed stealthy targeted denial attack in Sec. 6.3. However, the firewall might fall short of detecting and preventing attacks that perform injection when the bus is idle. This include the *bus denial 1* attack, Sec. 6.2.1.1, that could be launched when the main bus is idle. Another potential attack that might bypass the firewall is the transmission of incomplete frames. Unterminated frames trigger errors at the receiving ECUs due to the form error generated from observing incomplete frames. Although this attack does not lead to shutting off ECUs (bus-off), it is still a valid attack that we need to consider preventing. Adding more logic to the firewall might introduce overhead that could affect its effectiveness. Therefore, we need to complement the firewall with another mechanism that detects and reacts to such attacks, i.e., an Intrusion Detection System (IDS).

In this chapter, we propose a novel IDS design that aims to detect attacks based on certain rules

and then inform the firewall to prevent them. To the best of our knowledge, the IDS is the first one that detects attacks by bit banging, unlike traditional IDS that perform analyses at the frame level.

9.1.1 IDS Detection Rules

9.1.1.1 Bus Denial by Dominant Bits Stream

The SOF bit signals the start of a frame from the attacker's side and the frame is forwarded to the main bus. The firewall stops forwarding when the EOF delimiter is recognized. However, an attacker could perform the bus denial by dominant bits stream by starting the attack when the bus is idle, i.e., no traffic on either bus, CAN_1 nor CAN_2. To detect and then prevent such an attack the IDS needs to recognize it by its identifying pattern. It starts with an SOF, and more than four consecutive dominant bits follow the presumably SOF bit. When this happens, the IDS should immediately know that this is a potential bus denial by dominant bits stream.

9.1.1.2 Bus Denial by 0x0 ID flooding

This attack is easy to detect because the IDS waits for the entire arbitration ID to complete and then sees a 0x0 ID is being transmitted. If 0x0 ID is not expected to be used from the attacker's ECU, then it must be prevented.

9.1.1.3 Incomplete Frames Injection

Although this attack does not lead to a bus-off error state, it is still worth preventing. The attack occurs when an attacker starts sending a frame correctly by sending a CAN ID that wins the arbitration phase so the rest of the ECUs are in receiving mode. However, once the arbitration is complete and the dominance of the bus is gained, the attacker does not transmit the rest of the frame resulting in *form errors* that could lead the receiving ECUs to be in Error-Passive states. The reason of this attack's success at bypassing the firewall is that it starts with a SOF bit which the firewall detects and then starts forwarding the traffic from the attacker's bus to the main bus.

9.1.2 IDS and Firewall Integration

When the IDS detects the discussed attacks, it should notify the relevant firewall to block the attacker for a random period of time. When the same attacker reattempts the attack, it gets blocked for a longer time. In some cases, performing a bus-off attack might be considered to shut down the attacker's ECU.

9.1.2.1 Prevention of Bus Denial by Dominant Bits Stream

IDS informs the firewall to block traffic from the attacker's ECU and not forward it to the main bus. Fig 9.1 shows that how the attack bypasses the firewall and then how the IDS detects such an attack when the pattern is recognized.

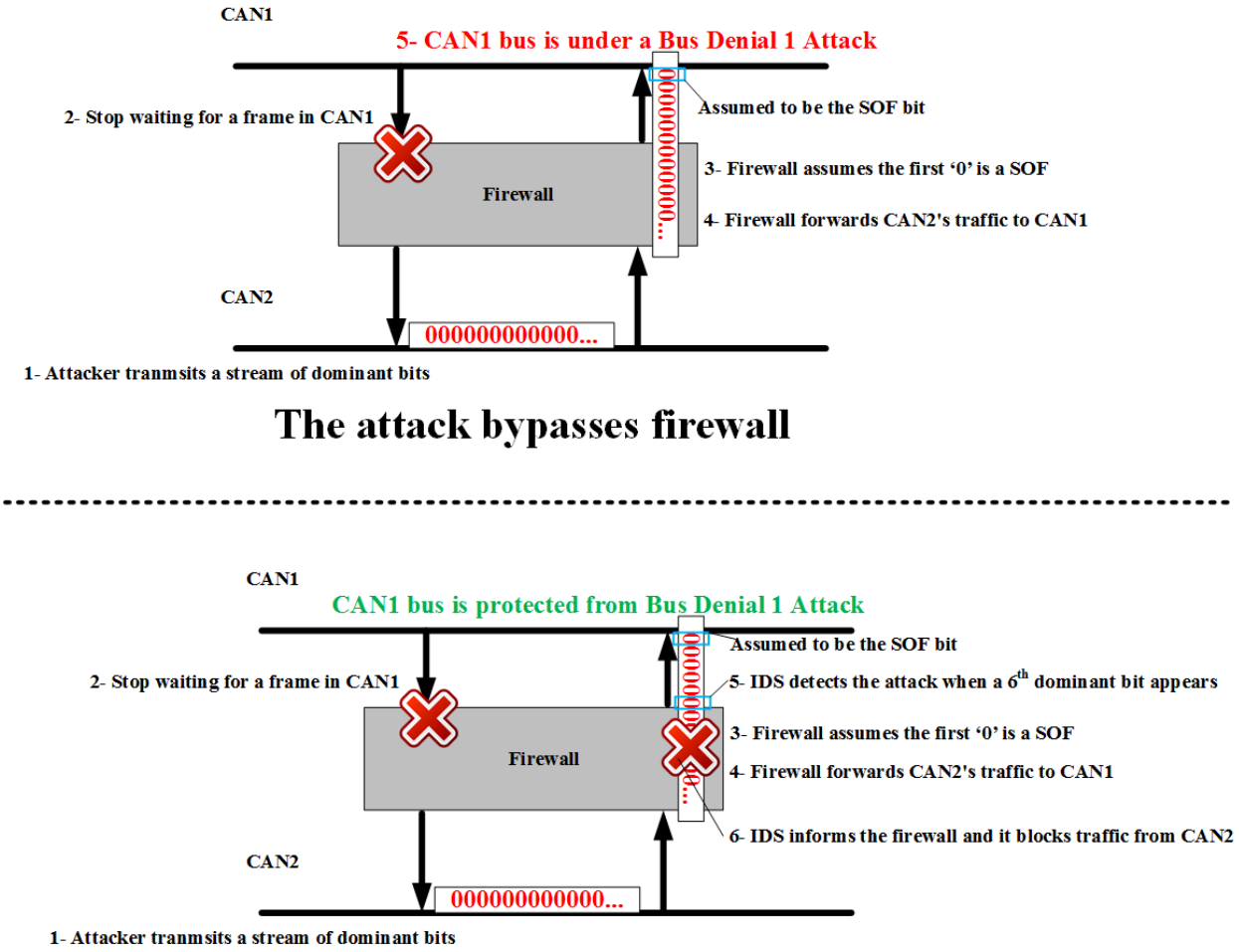


Figure 9.1: Bus Denial 1 Attack Demonstration

9.1.2.2 Prevention of Bus Denial by 0x0 ID flooding

When 0x0 ID flooding is detected, the IDS informs the firewall about the attack. Consequently, if the attacker's ECU is a conventional one with the a CAN controller, then a bus-off attack could be performed to drive the attacker's ECU to the bus-off state. If CAN controller is not used, then blocking the traffic is a safer option. Fig 9.2 illustrates how the attack bypasses the firewall and how the IDS detects it and helps preventing it.

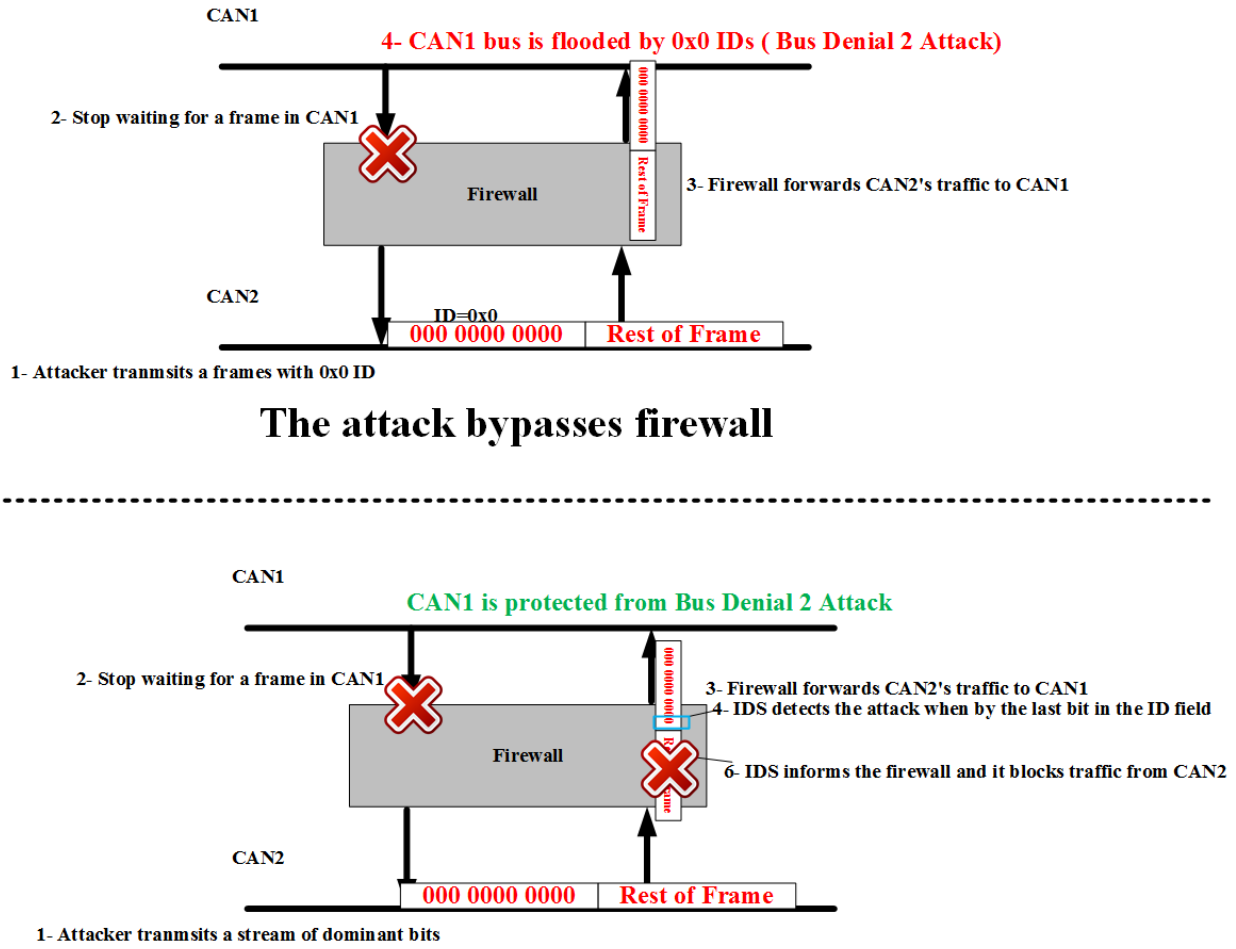


Figure 9.2: Bus Denial 2 Attack Demonstration

9.1.2.3 Prevention of Incomplete Frames Injection

When the IDS detects the attack's pattern, the firewall could do one of several options. For example, it could complete the incomplete frame so that no form errors are generated and thus, receivers do not switch to the error-passive state. Another option is to block the traffic from the attacker's ECU for a random period of time.

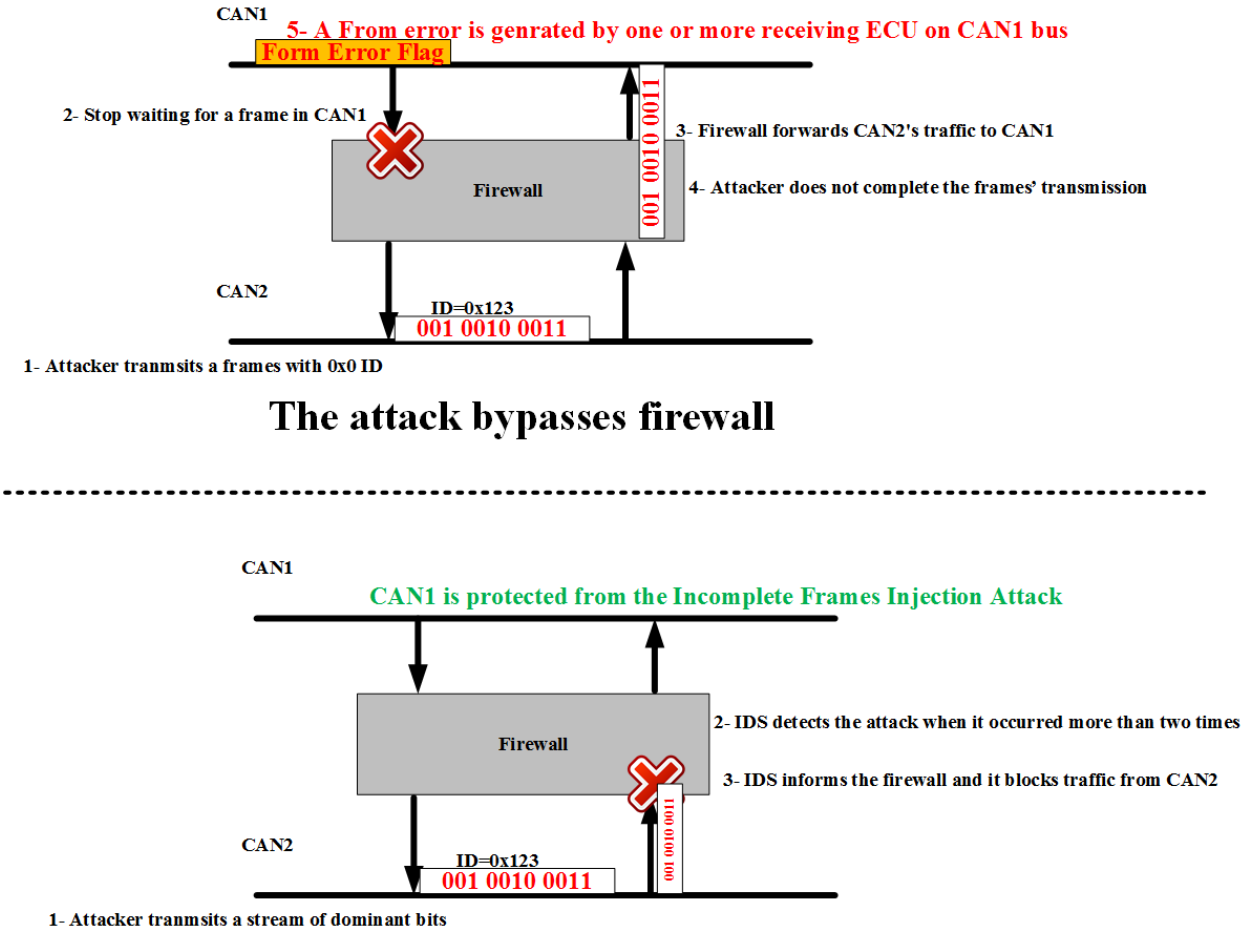


Figure 9.3: Bus Denial 2 Attack Demonstration

9.1.3 IDS Architecture

Fig. 9.4 illustrates the architecture of the proposed IDS. The IDS is connected to every firewall with three connections, two from the firewall and one to the firewall. The connections from the firewall provides the IDS the means for real-time analysis for attack detection. One is used for reading the Rx in the CAN transceiver facing the attacker's bus and another for the Rx in the main bus's transceiver. On the other hand, the connection to the firewall is used to inform the firewall's decisions for preventing ongoing attacks from the attacker's bus.

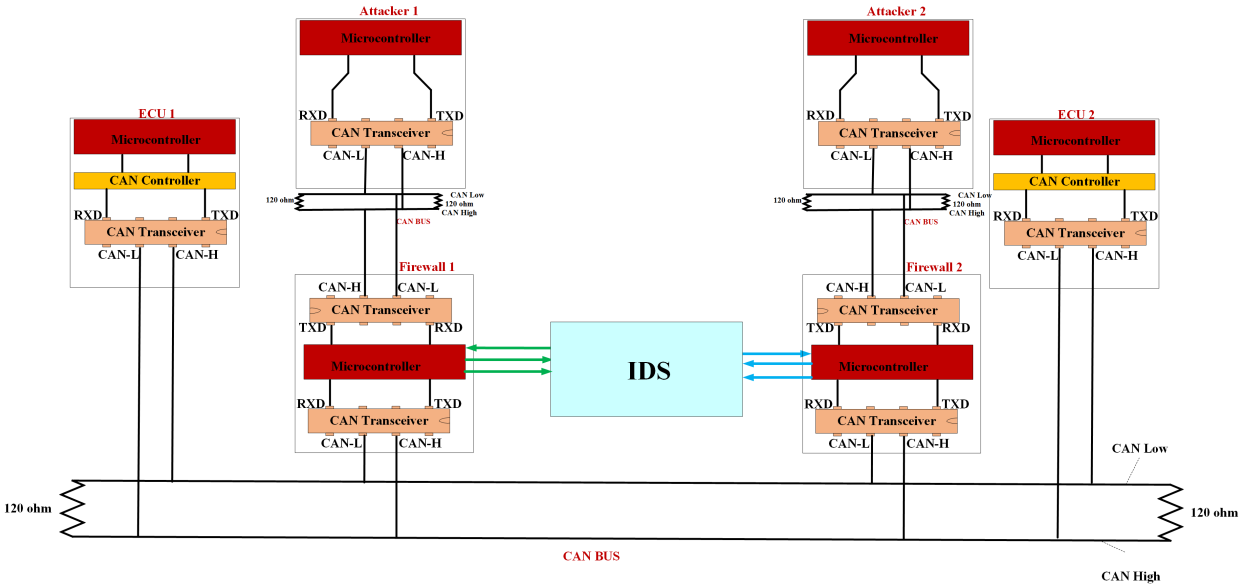


Figure 9.4: IDS Architecture

9.2 Related Work

The conventional approaches detect based on analyses frames' frequency and contents. Frames based on their CAN IDs and data fields are analyzed and attacks are detected. A detailed review of the current IDS is in Sec 4.7.10. However, we propose the first IDS that perform the analysis at a frame-less level. In other words, our IDS detects attacks of a nature that current IDS is not designed to detect.

9.3 Summary

In this chapter, we introduced a novel IDS design for detection of three of the denial attacks discussed in Chapter 6. Because these attacks initially follow CAN standards, the proposed firewall, proposed in Chapter. 8, does not prevent them. The proposed IDS is connected bidirectionally with each firewall. It monitors each firewall's traffic from the attacker's side to detect attacks. Once an attack is detected by the IDS, the firewall receives a notification from the IDS in order to prevent it by blocking the traffic from the attacker's side. This proposal complements the firewall and more

attacks could be easily added.

Chapter 10

Conclusions and Future Work

In this dissertation, we surveyed four CPS application and presented a framework to capture their security issues from a cyber-physical perspective. We then presented a stealthy targeted denial attack on CAN that could have cyber-physical implications. In addition, we proposed three simultaneous solutions to detect and prevent DoS attacks on CAN. The first solution is the ID-Hopping mechanism. Denial attacks rely on the detection of target CAN IDs to launch an attack. When an attacker learns about a CAN ID and that it is used for a certain function that they target, the attacker will wait for this CAN ID to start a denial attack.

10.1 Conclusions

In Chapters 3 and 4, we surveyed the literature on security and privacy of cyber-physical systems in four representative CPS applications: ICS, smart grids, and medical devices, with a special focus on smart cars. We presented a taxonomy of threats, vulnerabilities, known attacks and existing controls. A cyber-physical security framework was also presented that incorporated CPS aspects into the security aspects. The framework captured how an attack of the physical domain of a CPS can result in unexpected consequences in the cyber domain and vice versa along with proposed solutions. Using our framework, effective controls can be developed to eliminate cyber-physical attacks. For example, we identified that the heterogeneity of CPS components contributes signifi-

cantly to many attacks. Therefore, for an effective solution, special attention should be paid when heterogeneous components interact. Another example that the dissertation is based on is the DoS attack, which is one of the most common attacks with safety-critical implications in CPS. When a cyber-physical system undergoes a DoS attack, life-threatening impacts are expected. For example, car brakes ECU must be secure against DoS so that an attacker cannot harm passengers and their safety.

Therefore, in addition to the surveyed attacks in the four CPS applications, we then further explored a new class of DoS attacks on CAN. CAN is the de facto communication protocol in many CPS applications, most notably smart cars. These attacks render targeted ECUs useless when they are most needed. For example, an attacker could disable the *collision avoidance system* when an accident is about to be avoided by targeting the ECU's CAN IDs. Therefore, an important question was raised: How we can detect and prevent such an attack in a protocol that lacks the most basic security measures such as identification and authentication? Furthermore, CAN is broadcast-based so that CAN frames are identified only by their CAN IDs that determine their priority to occupy the bus and the relevance for the receiving ECUs. Introducing controls that are common in IT applications such as authentication and access control are not directly applicable and might add a non-negligible overhead and delays in such limited-spaced CAN frames that are expected to perform in real-time conditions.

Therefore, in Chapter 6, we surveyed the non-traditional denial attacks on CAN and classified them into four categories: bus, ECU, frame, and arbitration denial. All the surveyed attacks either exploited existing design vulnerabilities, such as the bit-wise arbitration and error-handling mechanism, or misused features in CAN controllers such as the "Test Mode." Before addressing the attacks, we proposed and demonstrated in the same chapter a new stealthy targeted arbitration denial attack that avoids detection by existing countermeasures due to its adherence to CAN specifications. The attack is demonstrated and compared with the other denial attacks. Our experiments show that it is superior to the other denial attacks in terms of stealthiness.

In Chapter 7, we propose the ID-Hopping mechanism to defend against CAN ID-based DoS

attacks. ID-Hopping randomizes CAN IDs in a way that deters an attacker. When an attack is detected, the targeted ECU transmitting the target ID signals the other ECUs to use a different set of IDs that does not include the target ID that the attacker is seeking for. This results in the disappearance of the attack's trigger, and therefore prevention of that attack. We evaluated ID-Hopping using a set of micro-controllers that act as normal ECUs and an attacking ECU. Because the normal ECUs simulate legitimate CAN nodes, we used four BeagleBone Black micro-controllers where each one includes a CAN controller that is connected to an external CAN transceiver. For the attacker, we implemented the attack using an STM32 Nucleo-144 development board due to its support for CANT - an open source CAN security testing tool that allows for testing security when CAN specifications are not followed. The evaluation shows that the attack was successfully prevented when ID-Hopping was activated because it replaced the target ID by another one that the attacker was not aware of.

In Chapter 8, we propose a novel CAN transceiver-based firewall that prevented the stealthy targeted denial attack in addition to most of the surveyed denial attacks. The firewall is located as a gateway between a potential attacker,Äôs ECU and the rest of the CAN bus. The traffic is forwarded bi-directionally between the main bus and the attacker. The firewall prevents the attacker from performing any malicious attacks when there is a frame being transmitted in the main bus. The attacker's ECU is only permitted to participate with the main bus when CAN specifications are followed. The specifications states that an ECU can only start transmission when the bus is idle [16]. In other words, when the main bus is occupied by a legitimate ECU transmission, the attacker's ECU is not allowed to transmit until the bus is idle. We evaluated the proposed firewall and demonstrated its effectiveness against most of the denial attacks with the use of BBBs as legitimate ECUs and two STM32 Nucleo-144 development boards; one is used as an attacker while the other as the firewall.

Finally, in Chapter 9, we introduced a novel IDS design for detection of three of the denial attacks discussed in Chapter 6. Because these attacks initially follow CAN standards, the proposed firewall, proposed in Chapter. 8, does not prevent them. The proposed IDS is connected

bi-directionally with each firewall and monitors each firewall's traffic from the attacker's side to detect attacks. Once an attack is detected by the IDS, the firewall receives a notification from the IDS to prevent it by blocking the traffic from the attacker's side. This proposal complements the firewall and more attacks could be easily added.

Since the ID-Hopping mechanism prevented the stealthy targeted denial attack, it should also prevent any ID-based attacks due to the randomization of the IDs that are only known to legitimate ECUs. The mechanism's strength lies in the secrecy of the offset. In addition, the proposed firewall was able to prevent most of the denial attacks by ensuring that the CAN specifications are followed by preventing an attacker from injecting dominant bits during frame transmissions on the main bus. This prevents the interruption of legitimate frames, and therefore any consequences of the interruption.

CAN is used real-time in a plethora of applications ranging from small and large vehicles, ships, planes, and even in artificial limbs, drones, radar systems, and submarines. Therefore, prevention of various types of denial attacks in CAN-based systems is crucial to prevent potential safety-critical consequences. Several proposals attempted to address denial attacks, some by randomizing CAN IDs so an attacker cannot detect them, and therefore attack them [68]. Others use a type of counterattack so that an attacker,Äôs ECU shuts off and stops sending to the CAN bus affecting the target ECUs [38, 160]. However, there is not yet a control that could intrusively prevent malicious ECUs from performing denial attacks with little to no latency and computation overhead. We claim that our firewall contributes to addressing this gap. It is inexpensive, so its adoption would not be costly and it is compatible with any CAN-based application regardless of the higher layer protocols.

10.2 Future Work

10.2.1 CPS Applications

CPS is becoming more popular and its applications are growing. Therefore, it would be a beneficial future direction to apply the proposed CPS security framework to emerging CPS applications such as smart cities wherein more than CPS applications integrate to make security issues even more complex.

10.2.2 Firewall for Other Attacks

The focus in this dissertation has been in detecting and preventing denial attacks. However, a promising future direction would be the prevention of false data injection attacks that are easily done in CAN-based applications due to the broadcast nature and lack of authentication.

10.2.3 Evaluation of the IDS

Due to time constraints, we did not evaluate the proposed IDS thoroughly. A future direction would be to implement the design and evaluate it.

10.2.4 Detection of Subtle Attacks

An IDS could have more analytical capabilities so it could prevent attacks with a completely legitimate format but with malicious purpose. For example, the IDS should be provided with the CAN IDs that each ECU is expected to transmit, especially for the ECUs that are sitting behind the firewall. Then, it should monitor the traffic coming from that ECU and detect abnormal/unexpected IDs so it could inform the firewall to block them.

10.2.5 Fingerprinting ECUs

Bit banging could be used to identify ECUs based on characteristics. Due to the imperfection of the ECUs clock skews, there will be unique offset that makes a CAN transceiver-based firewall or IDS able to fingerprint ECUs [39].

References

[1] Critical physical infrastructures. http://varma.ece.cmu.edu/cps/Presentations/Working-Group-Summaries/Critical%20Physical%20Infrastructures.pdf.

[2] Microsoft security bulletin summary for september 2010. https://technet.microsoft.com/library/security/ms10-sep.

[3] Security in automotive bus systems. 2004.

[4] Sources: Staged cyber attack reveals vulnerability in power grid. http://www.cnn.com/2007/US/09/26/power.at.risk/, September 2007.

[5] Microsoft security bulletin ms08-067 - critical. https://technet.microsoft.com/library/security/ms08-067, 2008.

[6] Ieee standard for local and metropolitan area networks - part 15.6: Wireless body area networks. *IEEE Std 802.15.6-2012*, pages 1–271, Feb 2012.

[7] *Survey on security threats and protection mechanisms in embedded automotive networks*, 2013.

[8] Cristina Alcaraz and Sherali Zeadally. Critical control system protection in the 21st century: Threats and solutions. 2013.

[9] S. Amin, G.A. Schwartz, and A. Hussain. In quest of benchmarking security risks to cyber-physical systems. *Network, IEEE*, 27(1):19–24, 2013.

[10] Saurabh Amin, Xavier Litrico, S Shankar Sastry, and Alexandre M Bayen. Stealthy deception attacks on water scada systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 161–170. ACM, 2010.

[11] Ross Anderson and Shailendra Fuloria. Who controls the off switch? In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 96–101. IEEE, 2010.

[12] M. R. Ansari, Shucheng Yu, and Qiaoyan Yu. Intellican: Attack-resilient controller area network (can) for secure automobiles. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), 2015 IEEE International Symposium on*, pages 233–236, Oct 2015.

[13] Rafael Ramos Regis Barbosa. Anomaly detection in scada systems: a network based approach. 2014.

[14] Steven M Bellovin. Security problems in the tcp/ip protocol suite. *ACM SIGCOMM Computer Communication Review*, 19(2):32–48, 1989.

[15] CAN Bosch. Specification version 2.0. *Published by Robert Bosch GmbH (September 1991)*, 1991.

[16] Robert Bosch et al. Can specification version 2.0. *Rober Bousch GmbH, Postfach*, 300240:72, 1991.

[17] Alexandre Bouard, Johannes Schanda, Daniel Herrscher, and Claudia Eckert. Automotive proxy-based security architecture for ce device integration. In *Mobile Wireless Middleware, Operating Systems, and Applications*, pages 62–76. Springer, 2013.

[18] Linda Briesemeister, Steven Cheung, Ulf Lindqvist, and Alfonso Valdes. Detection, correlation, and visualization of attacks against critical infrastructure systems. In *Privacy Security*

and Trust (PST), 2010 Eighth Annual International Conference on, pages 15–22. IEEE, 2010.

[19] R. R. Brooks, S. Sander, J. Deng, and J. Taiber. Automotive system security: Challenges and state-of-the-art. In *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, CSIIRW '08, pages 26:1–26:3, New York, NY, USA, 2008. ACM.

[20] Eric Byres, Matthew Franz, and Darrin Miller. The use of attack trees in assessing vulnerabilities in scada systems. In *Proceedings of the International Infrastructure Survivability Workshop*, 2004.

[21] Eric Byres and Justin Lowe. The myths and facts behind cyber security risks for industrial control systems. In *Proceedings of the VDE Kongress*, volume 116, 2004.

[22] H Cankaya, C Grepet, A Groll, J Holle, and M Wolf. Towards a shared digital communication platform for vehicles. In *Proc. of ITS World Congress*, 2011.

[23] Huayang Cao, Peidong Zhu, Xicheng Lu, and A. Gurtov. A layered encryption mechanism for networked critical infrastructures. *Network, IEEE*, 27(1):12–18, January 2013.

[24] A.A. Cárdenas, S. Amin, and S. Sastry. Secure control: Towards survivable cyber-physical systems. In *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on*, pages 495–500, 2008.

[25] Alvaro Cárdenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and Shankar Sastry. Challenges for securing cyber physical systems. In *Workshop on future directions in cyber-physical systems security*, 2009.

[26] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and

response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 355–366. ACM, 2011.

[27] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. Research challenges for the security of control systems. In *HotSec*, 2008.

[28] Alvaro A Cardenas, Tanya Roosta, and Shankar Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems. *Ad Hoc Networks*, 7(8):1434–1447, 2009.

[29] James L Cebula and Lisa R Young. A taxonomy of operational cyber security risks. Technical report, DTIC Document, 2010.

[30] Chris Valasek Charlie Miller. Adventures in automotive networks and control units. A SANS Whitepaper, August 2013.

[31] Chris Valasek Charlie Miller. A survey of remote automotive attack surfaces. Black Hat USA 2014, 2014.

[32] Chris Valasek Charlie Miller. Remote exploitation of an unaltered passenger vehicle, 2015.

[33] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*, 2011.

[34] M. Cheminod, L. Durante, and A. Valenzano. Review of security issues in industrial networks. *Industrial Informatics, IEEE Transactions on*, 9(1):277–293, Feb 2013.

[35] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor C Leung. Body area networks: A survey. *Mobile Networks and Applications*, 16(2):171–193, 2011.

[36] Thomas M Chen and Saeed Abu-Nimeh. Lessons from stuxnet. *Computer*, 44(4):91–93, 2011.

[37] Eric Chien, Liam OMurchu, and Nicolas Falliere. W32.duqu: The precursor to the next stuxnet. In *Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, Berkeley, CA, 2012. USENIX.

[38] Kyong-Tak Cho and Kang G Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055. ACM, 2016.

[39] Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *USENIX Security Symposium*, pages 911–927, 2016.

[40] Shinyoung Cho. Privacy and authentication in smart grid networks. 2014.

[41] Donghyun Choi, Hakman Kim, Dongho Won, and Seungjoo Kim. Advanced key-management architecture for secure scada communications. *Power Delivery, IEEE Transactions on*, 24(3):1154–1163, 2009.

[42] Richard Chow, Ersin Uzun, Alvaro A Cárdenas, Zhexuan Song, and Sung Lee. Enhancing cyber-physical security through data patterns. In *Workshop on Foundations of Dependable and Secure Cyber-Physical Systems (FDSCPS)*, page 25, 2011.

[43] Frances M Cleveland. Iec 62351 security standards for the power system information infrastructure. `http://iectc57.ucaiug.org/wg15public/Public%20Documents/White%20Paper%20on%20Security%20Standards%20in%20IEC%20TC57.pdf`, 2012.

[44] GRIMM CO. Cant. In *url=https://github.com/bitbane/CANT*.

[45] Jeffrey Cook. Cyber-physical systems and the twenty-first century automobile. In *NSF Workshop on Cyber-Physical Systems*, pages 1–3, 2006.

[46] A. D'Amico, C. Verderosa, C. Horn, and T. Imhof. Integrating physical and cyber security resources to detect wireless threats to critical infrastructure. In *Technologies for Homeland Security (HST), 2011 IEEE International Conference on*, pages 494–500, Nov 2011.

[47] Andrea Dardanelli, Federico Maggi, Mara Tanelli, Stefano Zanero, Sergio M Savaresi, R Kochanek, and T Holz. A security layer for smartphone-to-vehicle communication over bluetooth. 2013.

[48] Sajal K Das, Krishna Kant, and Nan Zhang. *Handbook on Securing Cyber-Physical Critical Infrastructure*. Elsevier, 2012.

[49] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. A taxonomy of attacks on the dnp3 protocol. In Charles Palmer and Sujeet Shenoi, editors, *Critical Infrastructure Protection III*, volume 311 of *IFIP Advances in Information and Communication Technology*, pages 67–81. Springer Berlin Heidelberg, 2009.

[50] G.N. Ericsson. Cyber security and power system communication 2014;essential parts of a smart grid infrastructure. *Power Delivery, IEEE Transactions on*, 25(3):1501–1507, July 2010.

[51] R Escherich, I Ledendecker, C Schmal, B Kuhls, C Grothe, and F Scharberth. She: Secure hardware extensionófunctional specification, version 1.1. *Hersteller Initiative Software (HIS) AK Security, April*, 2009.

[52] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart gridóthe new and improved power grid: A survey. *Communications Surveys & Tutorials, IEEE*, 14(4):944–980, 2012.

[53] John D Fernandez and Andres E Fernandez. Scada systems: vulnerabilities and remediation. *Journal of Computing Sciences in Colleges*, 20(4):160–168, 2005.

[54] Terry Fleury, Himanshu Khurana, and Von Welch. Towards a taxonomy of attacks against energy control systems. In Mauricio Papa and Sujeet Shenoi, editors, *Critical Infrastructure Protection II*, volume 290 of *The International Federation for Information Processing*, pages 71–85. Springer US, 2008.

[55] Food and Drug Administration (FDA). Cybersecurity for networked medical devices containing off- the-shelf (ots) software. http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm077823.pdf.

[56] Food and Drug Administration (FDA). Cybersecurity for networked medical devices containing off- the-shelf (ots) software. http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM356190.pdf.

[57] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. Design and implementation of a secure modbus protocol. In *Critical Infrastructure Protection III*, pages 83–96. Springer Berlin Heidelberg, 2009.

[58] Guillermo A Francia III, David Thornton, and Joshua Dawson. Security best practices and risk assessment of scada and industrial control systems.

[59] Aurélien Francillon, Boris Danev, Srdjan Capkun, Srdjan Capkun, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.

[60] Sibylle Fröschle and Alexander Stühring. Analyzing the capabilities of the can attacker. In *European Symposium on Research in Computer Security*, pages 464–482. Springer, 2017.

[61] Kevin Fu and James Blum. Controlling for cybersecurity risks of medical device software. *Commun. ACM*, 56(10):35–37, October 2013.

[62] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. They can hear your heartbeats: Non-invasive security for implantable medical devices. *SIGCOMM Comput. Commun. Rev.*, 41(4):2–13, August 2011.

[63] Dieter Gollmann. Security for cyber-physical systems. In *Mathematical and Engineering Methods in Computer Science*, pages 12–14. Springer, 2013.

[64] Andy Greenberg. Hackers remotely kill a jeep on the highway?with me in it. *Wired, 21July*, 2015.

[65] André Groll and Christoph Ruland. Secure and authentic communication on existing in-vehicle networks. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 1093–1097. IEEE, 2009.

[66] D. Halperin, T.S. Heydt-Benjamin, B. Ransford, S.S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W.H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 129–142, May 2008.

[67] Daniel Halperin, Thomas S. Heydt-Benjamin, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing*, 7(1):30–39, 2008.

[68] Kyusuk Han, Swapna Divya Potluri, and Kang G Shin. On authentication in a connected vehicle: secure integration of mobile devices with vehicular networks. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*, pages 160–169. IEEE, 2013.

[69] Kyusuk Han, André Weimerskirch, and Kang G Shin. Automotive cybersecurity for in-vehicle communication. *IQT QUARTERLY SUMMER 2014*, 2014.

[70] Steven Hanna, Rolf Rolles, Andrés Molina-Markham, Pongsin Poosankam, Kevin Fu, and Dawn Song. Take two software updates and see me in the morning: The case for software security evaluations of medical devices. In *Proceedings of the 2Nd USENIX Conference on Health Security and Privacy*, HealthSec'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[71] B Harris and R Hunt. Tcp/ip security threats and attack methods. *Computer Communications*, 22(10):885–897, 1999.

[72] Shane Harris. Chinaís cyber militia. *National Journal Magazine*, 31, 2008.

[73] Jeffrey Lloyd Hieb. *Security hardened remote terminal units for SCADA networks*. ProQuest, 2008.

[74] Greg Hoglund and Gary McGraw. *Exploiting software: how to break code*. Pearson Education India, 2004.

[75] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Applying intrusion detection to automotive it-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4(6):226–235, 2009.

[76] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats. In *Computer Safety, Reliability, and Security*, pages 145–158. Springer, 2009.

[77] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive can networks — practical examples and selected short-term countermeasures. In *Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security*, SAFECOMP '08, pages 235–248, Berlin, Heidelberg, 2011. Springer-Verlag.

[78] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1(0):37 – 44, 2008.

[79] ICS-CERT. Common cybersecurity vulnerabilities in industrial control systems. `http://ics-cert.us-cert.gov/sites/default/files/recommended_practices/DHS_Common_Cybersecurity_Vulnerabilities_ICS_2010.pdf`, 2010.

[80] Kazuki Iehira, Hiroyuki Inoue, and Kenji Ishida. Spoofing attack using bus-off attacks against a specific ecu of the can bus. In *Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual*, pages 1–4. IEEE, 2018.

[81] Guillermo Francia III, David Thornton, and Thomas Brookshire. Cyberattacks on scada systems. 2012.

[82] Guillermo Francia III, David Thornton, and Thomas Brookshire. Wireless vulnerability of scada systems. In Randy K. Smith and Susan V. Vrbsky, editors, *ACM Southeast Regional Conference*, pages 331–332. ACM, 2012.

[83] InvestmentWatch. First time in history, a terrorist attack on the electric power grid has blacked-out an entire nation ñ in this case yemen. `http://investmentwatchblog.com/first-time-in-history-a-terrorist-attack-on-the-electric-power-grid-has-blacked-ou`

[84] Rob Millerb Ishtiaq Roufa, Hossen Mustafaa, Sangho Ohb Travis Taylora, Wenyuan Xua, Marco Gruteserb, Wade Trappeb, and Ivan Seskarb. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium, Washington DC*, pages 11–13, 2010.

[85] Xuan Jin, John Bigham, Julian Rodaway, David Gamez, and Chris Phillips. Anomaly detection in electricity cyber infrastructures. In *Proceedings of the International Workshop on Complex Networks and Infrastructure Protection, CNIP*, 2006.

[86] Robert E Johnson. Survey of scada security challenges and potential attack vectors. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pages 1–5. IEEE, 2010.

[87] Dong-Joo Kang, Jong-Joo Lee, Seog-Joo Kim, and Jong-Hyuk Park. Analysis on cyber threats to scada systems. In *Transmission Distribution Conference Exposition: Asia and Pacific, 2009*, pages 1–4, 2009.

[88] Tom Karygiannis and Les Owens. Wireless network security. *NIST special publication*, 800:48, 2002.

[89] Himanshu Khurana, Mark Hadley, Ning Lu, and Deborah A Frincke. Smart-grid security issues. *Security & Privacy, IEEE*, 8(1):81–85, 2010.

[90] Pierre Kleberger, Tomas Olovsson, and Erland Jonsson. Security aspects of the in-vehicle network in the connected car. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 528–533. IEEE, 2011.

[91] Eric D Knapp and Raj Samani. *Applied Cyber Security and the Smart Grid: Implementing Security Controls Into the Modern Power Infrastructure*. Newnes, 2013.

[92] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462, May 2010.

[93] Maryna Krotofil and Dieter Gollmann. Industrial control systems security: What is happening? In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 670–675. IEEE, 2013.

[94] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, 9(3):49–51, 2011.

[95] Ulf E. Larson and Dennis K. Nilsson. Securing vehicles against cyber attacks. In *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, CSIIRW '08, pages 30:1–30:3, New York, NY, USA, 2008. ACM.

[96] Ulf E Larson, Dennis K Nilsson, and Erland Jonsson. An approach to specification-based attack detection for in-vehicle networks. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 220–225. IEEE, 2008.

[97] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. Lee & Seshia, 2011.

[98] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alex Roederer, et al. Challenges and research directions in medical cyber–physical systems. *Proceedings of the IEEE*, 100(1):75–90, 2012.

[99] Éireann Leverett and Reid Wightman. Vulnerability inheritance in programmable logic controllers. `https://ics-cert.us-cert.gov/content/cyber-threat-source-descriptions`.

[100] Eireann P Leverett. Quantitatively assessing and visualising industrial system attack surfaces. *University of Cambridge, Darwin College*, 2011.

[101] Chunxiao Li, A. Raghunathan, and N.K. Jha. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*, pages 150–156, June 2011.

[102] Congli Ling and Dongqin Feng. An algorithm for detection of malicious messages on can buses. In *2012 National Conference on Information Technology and Computer Science*. Atlantis Press, 2012.

[103] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[104] Zhuo Lu, Xiang Lu, Wenye Wang, and Cliff Wang. Review and evaluation of security threats on the communication networks in the smart grid. In *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010*, pages 1830–1835. IEEE, 2010.

[105] Zhuo Lu, Wenye Wang, and Cliff Wang. From jammer to gambler: Modeling and detection of jamming attacks against time-critical traffic. In *INFOCOM, 2011 Proceedings IEEE*, pages 1871–1879. IEEE, 2011.

[106] Matthew E. Luallen. Critical control system vulnerabilities demonstrated - and what to do about them. A SANS Whitepaper, November 2011.

[107] Martin Lukasiewycz, Philipp Mundhenk, and Sebastian Steinhorst. Security-aware obfuscated priority assignment for automotive can platforms. *ACM Trans. Des. Autom. Electron. Syst.*, 21(2):32:1–32:27, January 2016.

[108] Doug MacDonald, Samuel L Clements, Scott W Patrick, Casey Perkins, George Muller, Mary J Lancaster, and Will Hutton. Cyber/physical security vulnerability assessment integration. In *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*, pages 1–6. IEEE, 2013.

[109] Munir Majdalawieh, Francesco Parisi-Presicce, and Duminda Wijesekera. Dnpsec: Distributed network protocol version 3 (dnp3) security framework. In *Advances in Computer, Information, and Systems Sciences, and Engineering*, pages 227–234. Springer, 2006.

[110] Ed Markey. Tracking and hacking : Security and privacy gaps put american drivers at risk. `http://www.markey.senate.gov/imo/media/doc/2015-02-06_MarkeyReport-Tracking_Hacking_CarSecurity%202.pdf`.

[111] Daisuke Mashima and Alvaro A. Cárdenas. Evaluating electricity theft detectors in smart grid networks. In *Proceedings of the 15th international conference on Research in Attacks, Intrusions, and Defenses*, RAID'12, pages 210–229, Berlin, Heidelberg, 2012. Springer-Verlag.

[112] Tsutomu Matsumoto, Masato Hata, Masato Tanabe, Katsunari Yoshioka, and Kazuomi Oishi. A method of preventing unauthorized data transmission in controller area network. In *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, pages 1–5. IEEE, 2012.

[113] Patrick McDaniel and Stephen McLaughlin. Security and privacy challenges in the smart grid. *Security & Privacy, IEEE*, 7(3):75–77, 2009.

[114] Anthony R. Metke and Randy L. Ekl. Security technology for smart grid networks. *IEEE Trans. Smart Grid*, 1(1):99–107, 2010.

[115] Bill Miller and Dale Rowe. A survey scada of and critical infrastructure incidents. In *Proceedings of the 1st Annual conference on Research in information technology*, pages 51–56. ACM, 2012.

[116] R. Mitchell and Ing-Ray Chen. Survivability analysis of mobile cyber physical systems with voting-based intrusion detection. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International-/*, pages 2256–2261, 2011.

[117] Rob Mitchell and Ray Chen. Behavior rule based intrusion detection for supporting secure medical cyber physical systems. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7. IEEE, 2012.

[118] Robert Mitchell and I-R Chen. Behavior-rule based intrusion detection systems for safety critical smart grid applications. 2013.

[119] Robert Mitchell and Ing-Ray Chen. Effect of intrusion detection and response on reliability of cyber physical systems. *IEEE Transactions on Reliability*, 62(1):199–210, 2013.

[120] ROBERT MITCHELL and INGRAY CHEN. A survey of intrusion detection techniques for cyber physical systems.

[121] Yilin Mo, T.H.-H. Kim, K. Brancik, D. Dickinson, Heejo Lee, A. Perrig, and B. Sinopoli. Cyber-physical security of a smart grid infrastructure. *Proceedings of the IEEE*, 100(1):195–209, 2012.

[122] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM, 2010.

[123] Derek Molloy. Exploring beaglebone.

[124] Derek Molloy. *Exploring BeagleBone: tools and techniques for building with embedded Linux*. John Wiley & Sons, 2014.

[125] Kate Munro. Deconstructing flame: the limitations of traditional defences. *Computer Fraud & Security*, 2012(10):8 – 11, 2012.

[126] Pal-Stefan Murvay and Bogdan Groza. Dos attacks on controller area networks by fault injections from the software layer. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 71. ACM, 2017.

[127] Michael Müter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 1110–1115. IEEE, 2011.

[128] Michael Müter, André Groll, and Felix C Freiling. A structured approach to anomaly detection for in-vehicle networks. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 92–98. IEEE, 2010.

[129] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. An experimental investigation of malware attacks on scada systems. *International Journal of Critical Infrastructure Protection*, 2(4):139–145, 2009.

[130] Ellen Nakashima and Steven Mufson. Hackers have attacked foreign utilities, cia analyst says. *Washington Post*, 19, 2008.

[131] Nicolas Navet and Françoise Simonot-Lion. A review of embedded automotive protocols. *Automotive Embedded Systems Handbook, Industrial Information Technology Series, pages*, pages 4–1, 2008.

[132] FOX News Network. Threat to the grid? details emerge of sniper attack on power station. http://www.foxnews.com/politics/2014/02/06/2013-sniper-attack-on-power-grid-still-concern-in-washington-and-for-utilities/.

[133] Clifford Neuman. Challenges in security for cyber-physical systems. In *DHS Workshop on Future Directions in Cyber-Physical Systems Security*. Citeseer, 2009.

[134] Andrew Nicholson, S Webber, S Dyer, T Patel, and Helge Janicke. Scada security in the light of cyber-warfare. *Computers & Security*, 31(4):418–436, 2012.

[135] Liam O. Nicolas Falliere. W32.stuxnet dossier, 2011.

[136] Sen Nie, Ling Liu, and Yuefeng Du. Free-fall: Hacking tesla from wireless to can bus. In *Black Hat*, pages 1–16. Black Hat, 2017.

[137] Dennis K Nilsson, Phu H Phung, and Ulf E Larson. Vehicle ecu classification based on safety-security characteristics. In *Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference, IET*, pages 1–7. IET, 2008.

[138] NIST Nistir. 7628: Guidelines for smart grid cyber security. Technical report, Technical report, 2010.

[139] Hisashi Oguma, XAkira Yoshioka, Makoto Nishikawa, Rie Shigetomi, Akira Otsuka, and Hideki Imai. New attestation based security architecture for in-vehicle communication. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–6. IEEE, 2008.

[140] Satoshi Otsuka, Tasuku Ishigooka, Yukihiko Oishi, and Kazuyoshi Sasazawa. Can security: Cost-effective intrusion detection for real-time control systems. Technical report, SAE Technical Paper, 2014.

[141] Andrea Palanca, Eric Evenchick, Federico Maggi, and Stefano Zanero. A stealth, selective, link-layer denial-of-service attack against automotive networks. In *International Conference*

on Detection of Intrusions and Malware, and Vulnerability Assessment, pages 185–206. Springer, 2017.

[142] T. Paukatong. Scada security: A new concerning issue of an in-house egat-scada. In *Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES*, pages 1–5, 2005.

[143] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing (4th Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[144] Ludovic Pièetre-Cambacédès, Marc Tritschler, and Goran N Ericsson. Cybersecurity myths on power control systems: 21 misconceptions and false beliefs. *Power Delivery, IEEE Transactions on*, 26(1):161–172, 2011.

[145] Elias Leake Quinn. Privacy and the new energy infrastructure. *Available at SSRN 1370731*, 2009.

[146] Amir Hamed Mohsenian Rad and Alberto Leon-Garcia. Distributed internet-based load altering attacks against smart power grids. *IEEE Trans. Smart Grid*, 2(4):667–674, 2011.

[147] Jerome Radcliffe. Hacking medical devices for fun and insulin: Breaking the human scada system. In *Black Hat Conference presentation slides*, volume 2011, 2011.

[148] M.A Rahman, P. Bera, and E. Al-Shaer. Smartanalyzer: A noninvasive security threat analyzer for ami smart grid. In *INFOCOM, 2012 Proceedings IEEE*, pages 2255–2263, March 2012.

[149] Symantec Security Response. Dragonfly: Western energy companies under sabotage threat. `http://www.symantec.com/connect/blogs/dragonfly-western-energy-companies-under-sabotage-threat`.

[150] Ronald S. Ross. Nist, guide for conducting risk assessments. nist special publication 800-30 revision 1. Technical report, US Dep. of Commerce, 2012.

[151] Michael Rushanan, Aviel D Rubin, Denis Foo Kune, and Colleen M Swanson. Sok: Security and privacy in implantable medical devices and body area networks.

[152] Dae Hyun Ryu, HyungJun Kim, and Keehong Um. Reducing security vulnerabilities for critical infrastructure. *Journal of Loss Prevention in the Process Industries*, 22(6):1020 – 1024, 2009. Papers Presented at the 2007 and 2008 International Symposium of the Mary Kay O'Connor Process Safety Center and Papers Presented at the {WCOGI} 2007.

[153] Ruben Santamarta. Here be backdoors: A journey into the secrets of industrial firmware. `https://media.blackhat.com/bh-us-12/Briefings/Santamarta/BH_US_12_Santamarta_Backdoors_WP.pdf`, 2012.

[154] Hendrik Schweppe. *Security and privacy in automotive on-board networks*. PhD thesis, Télécom ParisTech, 2012.

[155] Stefan Seifert and Roman Obermaisser. Secure automotive gateway– secure communication for future cars. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, pages 213–220. IEEE, 2014.

[156] Roberto Setola. Cyber threats to scada systems, 2011.

[157] Shanker Shreejith and Suhaib A Fahmy. Security aware network controllers for next generation automotive embedded systems. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[158] Jill Slay and Michael Miller. Lessons learned from the maroochy water breach. In *Critical Infrastructure Protection*, pages 73–82, 2007.

[159] Teodor Sommestad, Göran N Ericsson, and Jakob Nordlander. Scada system cyber security—a comparison of standards. In *Power and Energy Society General Meeting, 2010 IEEE*, pages 1–8. IEEE, 2010.

[160] Daisuke Souma, Akira Mori, Hideki Yamamoto, and Yoichi Hata. Counter attacks for bus-off attacks. In *International Conference on Computer Safety, Reliability, and Security*, pages 319–330. Springer, 2018.

[161] Siddharth Sridhar, Adam Hahn, and Manimaran Govindarasu. Cyber–physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2012.

[162] Gary Stoneburner, Alice Y. Goguen, and Alexis Feringa. Sp 800-30. risk management guide for information technology systems. Technical report, Gaithersburg, MD, United States, 2002.

[163] Keith A. Stouffer, Joseph A. Falco, and Karen A. Scarfone. Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc). Technical report, Gaithersburg, MD, United States, 2011.

[164] Ivan Studnia. *Détection d'intrusion pour des réseaux embarqués automobiles: une approche orientée langage*. PhD thesis, Institut national des sciences appliquées de Toulouse, 2015.

[165] Rose Tsang. Cyberthreats, vulnerabilities and attacks on scada networks. *University of California, Berkeley*, 2010.

[166] Robert J Turk. *Cyber incidents involving control systems*. 2005.

[167] V. Urias, B. Van Leeuwen, and B. Richardson. Supervisory command and data acquisition (scada) system cyber security analysis using a live, virtual, and constructive (lvc) testbed. In *MILITARY COMMUNICATIONS CONFERENCE, 2012 - MILCOM 2012*, pages 1–8, 2012.

[168] US-CERT. Cyber threat source descriptions. `https://ics-cert.us-cert.gov/content/cyber-threat-source-descriptions`.

[169] Lisa Vaas. Doctors disabled wireless in dick cheney's pacemaker to thwart hacking. http://nakedsecurity.sophos.com/2013/10/22/doctors-disabled-wireless-in-dick-cheneys-pacemaker-to-thwart-hacking/, 10 2013.

[170] B. Vaidya, D. Makrakis, and H.T. Mouftah. Authentication and authorization mechanisms for substation automation in smart grid network. *Network, IEEE*, 27(1):5–11, January 2013.

[171] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In *ECRYPT Workshop on Lightweight Cryptography 2011*, 2011.

[172] Qiyan Wang and Sanjay Sawhney. Vecure: A practical security framework to protect the can bus of vehicles. In *Internet of Things (IOT), 2014 International Conference on the*, pages 13–18. IEEE, 2014.

[173] Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer Networks*, 57(5):1344 – 1371, 2013.

[174] Conal Watterson. Controller area network (can) implementation guide. *Application Note AN-1123, Analog Devices, Inc*, 2012.

[175] Donald Welch and Scott Lathrop. Wireless security threat taxonomy. In *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, pages 76–83. IEEE, 2003.

[176] Marko Wolf and Timo Gendrullis. Design, implementation, and evaluation of a vehicular hardware security module. In *Information Security and Cryptology-ICISC 2011*, pages 302–318. Springer, 2012.

[177] Marko Wolf, André Weimerskirch, and Thomas Wollinger. State of the art: Embedding security in vehicles. *EURASIP Journal on Embedded Systems*, 2007(1):074706, 2007.

[178] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A practical wireless attack on the connected car and security protocol for in-vehicle can. *Intelligent Transportation Systems, IEEE Transactions on*, 16(2):993–1006, 2015.

[179] Eaton Powering Business Worldwide. Power outage annual report: Blackout tracker. `http://www.eaton.com/blackouttracker`.

[180] Le Xie, Yilin Mo, and Bruno Sinopoli. False data injection attacks in electricity markets. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 226–231. IEEE, 2010.

[181] Fengyuan Xu, Zhengrui Qin, C.C. Tan, Baosheng Wang, and Qun Li. Imdguard: Securing implantable medical devices with the external wearable guardian. In *INFOCOM, 2011 Proceedings IEEE*, pages 1862–1870, April 2011.

[182] Mark Yampolskiy, Peter Horvath, Xenofon D Koutsoukos, Yuan Xue, and Janos Sztipanovits. Taxonomy for description of cross-domain attacks on cps. In *Proceedings of the 2nd ACM international conference on High confidence networked systems*, pages 135–142. ACM, 2013.

[183] Mark Yampolskiy, Péter Horváth, Xenofon D. Koutsoukos, Yuan Xue, and Janos Sztipanovits. A language for describing attacks on cyber-physical systems. *IJCIP*, 8:40–52, 2015.

[184] Mark Zeller. Myth or reality?does the aurora vulnerability pose a risk to my generator? In *Protective Relay Engineers, 2011 64th Annual Conference for*, pages 130–136. IEEE, 2011.

[185] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on scada systems. In *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, ITHINGSCPSCOM '11, pages 380–388, Washington, DC, USA, 2011. IEEE Computer Society.

[186] Bonnie Zhu and Shankar Sastry. Scada-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[187] Tobias Ziermann, Stefan Wildermann, and Jürgen Teich. Can+: A new backward-compatible controller area network (can) protocol with up to 16× higher data rates. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.*, pages 1088–1093. IEEE, 2009.

Appendix A

BeagleBone Black Configuration

A.1 Static IP Configuration

Although BBB are accessible via USB cable, it is cumbersome when you have more than one BBB. Therefore, we connect to the BBBs through a switch and configure each BBB to have a unique static IP address.

A.1.1 Host Machine IP Configuration

The following configurations need to be written in the */etc/network/interfaces* file on the host machine:

```
$ auto eth0
$ iface eth0 inet static
$ address 192.168.1.10
$ netmask 255.255.255.0
$ network 192.168.1.0
$ gateway 192.168.0.1
```

A.1.2 BBB IP Configuration

I configured each BBB by accessing through the USB cable only once as follows:

```
$ ssh root@192.168.7.2
```

Then add the following to the */etc/network/interfaces* file:

```
$ auto eth0
$ iface eth0 inet static
$ address 192.168.1.11
$ netmask 255.255.255.0
$ network 192.168.1.0
$ gateway 192.168.0.1
```

The above example is for BBB1, and for the BBB2 through BBB5, I change the address to 192.168.1.12, 192.168.1.13, 192.168.1.14, 192.168.1.15, respectively.

A.1.3 Connecting to BBBs

I open a separate terminal, a new terminal or with *tmux*, for each BBB as follows:

Connecting to BBB1:

```
$ ssh root@192.168.1.11
```

Connecting to BBB2:

```
$ ssh root@192.168.1.12
```

Connecting to BBB3:

```
$ ssh root@192.168.1.13
```

Connecting to BBB4:

```
$ ssh root@192.168.1.14
```

Connecting to BBB5:

```
$ ssh root@192.168.1.15
```

A.2 CAN Configuration

BBB has two built-in CAN controllers, DCAN0 and DCAN1. I will use DCAN1 on each BBB because it is more straightforward than DCAN0. DCAN0 needs more work to be done such as disabling i2s pins.

A.2.1 DCAN1 Configuration

1. Copy the Device Tree shown in Fig.A.1 to each BBB[1].

2. Execute the following command:

   ```
   $ dtc -O dtb -o BB-DCAN1-00A0.dtbo -b 0 -@ BB-DCAN1-00A0.dts
   ```

   This will create the overlay binary BB-DCAN1-00A0.dtbo.

3. To use the overlay, copy it to /lib/firmware as follows:

   ```
   $ sudo cp BB-DCAN1-00A0.dtbo /lib/firmware
   $ echo BB-DCAN1 > /sys/devices/bone_capemgr.*/slots
   ```

A.2.2 CAN Transceiver Wiring

A.2.3 CAN Interface Configuration

```
$ sudo modprobe can
$ sudo modprobe can-dev
$ sudo modprobe can-raw
```

[1] Found at `http://www.embedded-things.com/bbb/enable-canbus-on-the-beaglebone-black`

BB-DCAN1-00A0.dts

```
/dts-v1/;
/plugin/;

/ {
    compatible = "ti,beaglebone", "ti,beaglebone-black";

    /* identification */
    part-number = "dcan1pinmux";

    fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            dcan1_pins_s0: dcan1_pins_s0 {
                pinctrl-single,pins = <
                    0x180 0x12  /* d_can1_tx, SLEWCTRL_FAST | INPUT_PULLUP | MODE2 */
                    0x184 0x32  /* d_can1_rx, SLEWCTRL_FAST | RECV_ENABLE | INPUT_PULLUP | MODE2 */
                >;
            };
        };
    };

    fragment@1 {
        target = <&dcan1>;
        __overlay__ {
             #address-cells = <1>;
             #size-cells = <0>;

             status = "okay";
             pinctrl-names = "default";
             pinctrl-0 = <&dcan1_pins_s0>;
        };
    };
};
```

Figure A.1: DCAN1 Device Tree

A.2.4 SocketCAN Utilities

Get and build can-utils

```
$ git clone https://github.com/linux-can/can-utils.git
$ cd can-utils/
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

Set up the bus speed and enable it:

```
$ sudo ip link set can0 up type can bitrate 500000
$ sudo ifconfig can0 up
```

A.2.5 Scripts and Commands

Here we include some of the useful and time-saving commands that we have used in our work.

A.2.5.1 Start CAN with Error Reporting

We start CAN interface with the error reporting option enabled to get the most amount of error-related information. Below is the used command:

```
$ canconfig can0 bitrate 500000 ctrlmode berr-reporting on
$ canconfig can0 start
```

A.2.5.2 Script for Extracting Errors' Information

We wrote this script to extract error handling information: error states, TEC and REC before, during, and after that attack and then plot the data to see whether or not the attack disrupted the bus and triggered errors.

Figure 23-19. CTL Register

31	30	29	28	27	26	25	24
RESERVED						WUBA	PDR
R-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			DE3	DE2	DE1	IE1	InitDbg
R-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h
15	14	13	12	11	10	9	8
SWR	RESERVED	PMD				ABO	IDS
R/WP-0h	R-0h	R/W-5h				R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
Test	CCE	DAR	RESERVED	EIE	SIE	IE0	Init
R/W-0h	R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Figure A.2: Showing the settings to confirm the disabling took effect

A.2.6 CAN Simulation with 2 BBBs

We have two BBBs connected to a breadboard as the physical CAN bus. Both BBBs exchange CAN frames under normal operations when the attack is not present. The configurations are as follows for each BBB:

1. ssh to BBB

   $ ssh root@192.168.1.1x

   x is the BBB's number. 1 and 2 for BBB1 and BBB2, respectively.

2. Configure CAN

   $ canconfig can0 bitrate 500000

3. Start CAN

   $ canconfig can0 start

A.2.7 DCAN

Using *canconfig* utility, we start the CAN controller and by default the automatic transmission is enabled. We can directly access the CTL register and set the 5th bit to disable automatic retransmission, *DAR* as shown in Fig. A.2.

A.2.8 Automatic Retransmission

By desging, CAN controllers have an automatic retransmission capability that ensures the successful transmission of frames that get disrupted or lose arbitration. However, with frames that are time-sensitive or perodic, the automatic retransmission is not desired [**?**]. Therefore, we have evaluated the attack under two circumstances: when the automatic retransmission is enabled and disabled. We can configure the built-in CAN controller in the BBB by accessing the registers with *devmem2* [**?**]. Using *devmem2* we can directly access the registers and modify their values. After consulting with the Technical Reference Manual (TRM), section 23.4, we found DCAN registers' addresses [**?**].

A.2.8.1 Disable Automatic Retransmission

Figures A.3, A.4, and A.5 show the CTL's values during setting the automatic retransmission.

```
root@beaglebone:~/devmem2# ./showDCAN1Settings.sh
/dev/mem opened.
Memory mapped at address 0xb6efb000.
Value at address 0x481D0000 (0xb6efb000): 0xE
```

Figure A.3: The register's setting before disabling automatic retransmission

```
root@beaglebone:~/devmem2# ./disableRetransmission.sh
/dev/mem opened.
Memory mapped at address 0xb6fa3000.
Value at address 0x481D0000 (0xb6fa3000): 0xE
Written 0x2E; readback 0x2E
```

Figure A.4: The register's value after disabling the retrinamission

```
root@beaglebone:~/devmem2# ./showDCAN1Settings.sh
/dev/mem opened.
Memory mapped at address 0xb6f2b000.
Value at address 0x481D0000 (0xb6f2b000): 0x2E
```

Figure A.5: Showing the settings to confirm the disabling took effect

A.2.8.2 Show DCAN1 Settings

Using *devmem2*, we can see the current settings of DCAN1 with the following command:

- $ ./devmem2 0x481d0000

Please note this step cannot work if the CAN has not been started.

Appendix B

Programmable Real-Time Units (PRUs)

Before we decided on using CANT for attack's implementations, we had stared using BBB's real-time capabilities and had satisfactory results. However, CANT was a better option because it provided us with the exact capabilities we were aiming to realize in BBBs but with a more intuitive approach. We added this appendix as a future reference researchers.

BBB has real-time capabilities that sets it apart from other reasonably-priced microcontrollers. In addition to the core 1 GHz AM335x processer, BBB has two independent real-time cores, called Programmable Real-Time Units (PRUs). Each one which has a 32-bit 200 MHz RISC processor core with an 8 kB of RAM, in addition to a shared 12 kB of RAM between the two PRUs. Also, they have a direct access to the BBB's input/output pins with only a cost of one cycle. A single cycles takes only 5 nanoseconds. The PRUs are designated for specific tasks that require real-time capabilities. For examples, input/output manipulation, implementation of custom communication interfaces such as UART and SPI [**?**]. PRUs can be programmed in the PRU Assembly Language (PASM) or C. Either of the two languages can be compiled into a machine language by Texas Instruments' compiler written for the PRUs.

Configuration and programming PRUs is somewhat complex and time-consuming. Alanwar et al. **??** proposed a framework that greatly facilities the configurations and programming efforts from a PRU-specific environment to an accessible one. The framework gives three options for

programming the PRUs, Javascript-like language, C, and assembly. In this paper, we chose to work with C and partially assembly. Mostly we used C for its familiarity and assembly whenever C is not fast enough. Cyclops provides an easy way to access the PRU's GPIO pins. This allowed us to focus on the CAN-related aspects, instead of worrying about the configurations and the platform-specific instructions.

B.1 Download a Customized Image

To work with PRUs, one can either build the OS image or use a customized one. We downloaded the imaged named *4.1.12-bone-rt with QoT* from Cyclops[1].

B.2 Load Devicetree Overlays

Devicetree is a way to describe a processor with data structures. Upon booting, the kernel will learn the current processor's configurations such as input/output pins. Although useful, this devicetrees require rebooting with every reconfiguration. Therefore, overlays emerged in BBB to allow developers to reconfigure pins from the user-space without the need to reboot the system.

1. Loading NESL-PRU overlay:

```
$ echo "NESL–PRU" > /sys/devices/bone_capemgr.9/slots
```

2. Load universala overlay:

```
$ config-pin overlay cape-universala
```

B.3 Configure Pins

Cyclops provide an easy way to configure the GPIO pins on the BBB using the following commands:

[1] `https://github.com/nesl/Cyclops-PRU/tree/master/FullImage/images`

Input pin configuration:

```
$ config-pin P8_15 pruin
```

Output pin configuration:

```
$ config-pin P8_11 pruin
```

To check the pin's status:

```
$ config-pin -q P8_15
```

It is important to consult with the pins' mapping before assigning them. One excellent resource is Derek Molloy's book [124] and website [123]. In the website, two comprehensive tables are given showing all of the possible options pins can be configured to. These tables can be found at these two links: P8 header, `http://exploringbeaglebone.com/wp-content/uploads/resources/BBBP8Header.pdf`, and P9 header, `http://exploringbeaglebone.com/wp-content/uploads/resources/BBBP8Header.pdf`. For example, the pins 11 and 15 on the header P8 was configured as an input pin for the PRU because the *mode 6* of the pinmux shows that it is possible to do so. Without Cyclops, this operation is time-consuming and somewhat complex.

B.4 Programming

Cyclops provides a convenient way to program PRUs. Our implementation is built on the GPIO example in the examples directory. It contains the necessary code examples to get started with PRU programming with C programming. The most important files to consider are:

- host_main.c

  - This file contains the components that run on the Linux side such as interrupts.

- pru_main.c

  - This file contains the actual tasks that you need to be done in real-time by the PRU.

- build.sh

  - In addition to compiling your c code in the above two files, you can determine which PRU to use. This is useful when you want to use both PRUs: PRU_0 and PRU_1.

- gen directory

  - This directory contains the executable files that will actually run the PRU. Change directory to *gen* and then execute *host* file as instructed.

B.4.0.1 PRU Configuration

1. ssh to BBB

   $ ssh root@192.168.7.2

2. Configure the PRU

   $ cd ucla-nesl-pru-lib/examples/myGpio

   $ ./loadAll.sh

   - This is a script to load the overlays and configure pins.

3. The attack's code resides in pru_main.c. To run it:

   $ cd gen

   $ ./host

   - Now the PRU is running