University of Greenwich
Faculty of Architecture, Computing and Humanity
Department of Computing & Information Systems

Cyber-physical intrusion detection for robotic vehicles

Tuan Phan Vuong

A thesis submitted in partial fulfilment for
the degree of Doctor of Philosophy
March 2017

DECLARATION

I certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of Doctor of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations except where otherwise identified by references and that I have not plagiarised the work of others.

Student: Tuan Phan Vuong

Date:

Signature:

Supervisor: Dr George Loukas

Date:

Signature:

Dedication

To my wife Tue-Anh, my daughter Truc-Anh and her grandparents.

You mean the world to me.

Acknowledgements

My greatest gratitude is dedicated towards my supervisors Dr George Loukas and Dr Diane Gan. Without their dedication, passion and support through this tough process, I would not have been able to complete this research. I am in great debt to their guidance and teaching.

I would like to thank my other collaborators (in alphabetical order) Anatolij Bezemskij, Avgoustinos Filippoupolitis, Ryan Heartfield, Georgia Sakellari and Yongpil Yoon who have inspired me to explore diverse fields of machine learning and cloud computing. Their expertise and inputs made our publications possible and significant. I would like to thank the members of CSAFE research centre, especially Dr Dave Chadwick who provided important guidance on different aspects of my struggle.

It has been a challenging time and a great deal of work throughout the years. Without my family and friends' support, I would not have had strengths to pursue this professional aspiration. I want to say thank-you to Carl, Kabir, Cain, Thaddeus, Joseph, Bill, Carmen, Thaya, Babak, Anatolij, Ali, Tausifali, and William who have been very supportive and encouraging for all the good and not-so-good days. I appreciate all constructive feedback and advice from my long-time friends Minh-Tri and Quoc-Long on my research journey. I would like to thank to Dave and Helen for their continued support over the years.

I am eternally indebted to my family including both parents and my parents-in-law for putting my welfare before theirs. My deepest thanks to Tue-Anh and Truc-Anh for making life so much more loving and meaningful. Without you, I would not even have enough courage to take the initial steps on this blessing journey.

This work was thankfully supported by a University of Greenwich (UoG) Vice chancellor's PhD grant and a student bursary from Dr George Loukas's REF funding.

Abstract

Intrusion detection systems (IDS) designed for conventional computer systems and networks are not necessarily suitable for mobile cyber-physical systems (CPS), such as robots, drones and automobiles. They tend to be geared towards attacks of different nature and do not take into account mobility, energy consumption and other physical aspects that are vital to a mobile cyber-physical system. This work provides two different approaches for addressing the problem of detecting attacks against vehicles, using a small-scale robotic vehicle as a testbed. The first approach is based on decision trees and the second on deep learning. Both use a combination of cyber and physical features that can be measured by its onboard systems and processes. Experimental evaluation on a variety of scenarios involving denial of service, command injection and two different types of malware infections demonstrated the feasibility of the approaches.

Decision tree algorithm is one of the most lightweight machine learning techniques, yet sufficiently powerful in many areas of applications, because it can naturally account for non-linearities in the data. Decision trees produce sets of simple rules, which can be easily checked onboard even the most resource-constrained of robotic vehicles. In the case of our vehicle, this approach was able to achieve high accuracy rate for denial of service attacks, but less so for the other attacks tested.

Due to their processing resource constraints, cyber-physical systems, such as robotic vehicles, tend to be limited to lightweight mechanisms, such as decision trees and other statistical machine learning techniques. We show that considerably higher accuracy rates can be achieved if one utilises techniques from the field of deep learning. In particular, we use a recurrent neural network architecture, benefiting from a long short-term memory layer, which is highly appropriate for real-time data. To address the processing limitations, we turn to computational offloading, which is a technique particularly common for mobile devices, for largely the same reasons: to save energy and to have access to greater processing resources. We show both experimentally and mathematically in which cases offloading the periodic task of deep learning based intrusion detection to a remote server can be practical, especially in relation to the time the whole process takes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robotic vehicles, unmanned aerial vehicles (UAVs) and automobiles feature a tight coupling between their cyber and physical properties. The term cyber is used to refer to their computation and communication processes, and physical to refer to their mobility, power consumption and any other physical manifestation of their operation. Focus here is on examples of cyber-physical attacks, where a security breach in cyberspace has an adverse effect in physical space [Loukas, 2015], specifically on the vehicle's physical functions and potentially in the environment where it operates. Examples may include causing a vehicle to stop, a UAV to land prematurely [Jennings, 2014], or its controls to be hijacked, as in the case of Maldrone [Sasi, 2015], an experimental backdoor server for quad-copter drones, which kills the autopilot, takes control and can spread to other drones. Earlier examples include the work of Checkoway et al. [Checkoway et al., 2011] on a malware-infected audio file that was able to provide the researchers with remote control of an automobile's engine control and braking system, and the work of Kerns et al. [Kerns et al., 2014] in altering a vehicle's trajectory via global positioning system (GPS) spoofing. Beyond these organised research efforts, there are also hacking competitions with time-restricted challenges for hijacking vehicles, such as the Tesla S sports car hijacking contest reported in [Griffiths, 2014].

While vehicles may differ enormously in terms of their type, size, operation and how

safety-critical they are, most tend to share the following characteristics which make them challenging to secure: (a) Any cyber security functions on them are resource-constrained, either because of lack of processing power or because minimising energy consumption has priority; (b) most cyber-physical operations that an attacker would target are time-critical, especially if they affect mobility; and (c) unlike cyber threats to conventional computer systems, which have been meticulously observed and statistically analysed for decades, threats here are largely unknown, and consequently there are no meaningful datasets to be used for benchmarks.

1.1 Motivation

According to the Institute of Electrical and Electronics Engineers (IEEE) Computer Society, Cyber-Physical Systems (CPS) is amongst the Top 9 technology trends in 2016 [Society, 2016]. CPS are smart systems that have three main components: computation, communication, and control [Lee, 2008]. Within CPS, the cyber technologies in both hardware and software integrating with physical components in sensing and actuation lead to actual impact in physical space. CPSs range from smart vehicles, smart robots to smart grids and air traffic control. In all of these examples, the operation requires a high level of reliability, safety, and availability. As these systems are more and more applicable in daily life, their security aspects are becoming increasingly important.

Security is defined as a degree of resistance to, or protection, from harm [Gasser, 1988]. Cyber security attacks are often grouped into three categories which represent the CIA triad of Confidentiality, Integrity, and Availability.

Traditionally, confidentiality has involved protecting customer and corporate information from disclosure. Within CPS systems it is about safeguarding the communications channels from eavesdropping. These communication channels are found between actuators, sensors and controllers and are the links between the physical and cyber components of these systems [Lu et al., 2015, Wang et al., 2010, Gamage et al., 2011].

Integrity within cyber-physical systems concerns protecting these systems from cyber attacks that would cause a physical impact that may prevent the system from achieving its intended goals. This could be accomplished for example by preventing the injection of commands or by blocking DoS attacks [Lu et al., 2015]. Wang et al. defined integrity for CPS as preventing data or resources from being modified by malicious actors [Wang et al., 2010].

Cyber-physical systems also require a high level of availability in order to perform their designated function. Availability also can be affected by hardware failures and power outages, as well as malicious attacks such as DoS attacks [Lu et al., 2015].

As well as the CIA triad, Wang et al. (2010) also proposed authenticity as being essential for CPS security. Any process or communication within CPS should include the authentication of the communicating parties so that they can be validated. As well as this, reliability, robustness and trustworthiness were also included for the case of CPSs [Lu et al., 2015]. Robustness is important as this enables the CPS to withstand unforeseen disruptions which may or may not be malicious in nature [Wang et al., 2010].

Because CPS systems have specific characteristics identified by mobility, sensors, actuators and control, this brings challenges as well as opportunities for attack defence mechanisms. As both cyber and physical harm can be caused by cyber attacks to the system, it is beneficial to study the changing behaviour of both cyber and physical indicators that can be measured on the system. Understanding the pattern of change may potentially help to detect these malicious activities.

As they are CPSs, robotic vehicles, combine the capabilities of manual control as well as different automation based on input from the surrounding environment. The most common forms of automation application in vehicles include self-driving cars using GPS for navigation, unmanned ground and aerial vehicles used in exploration and military. With the advance of technology, robotic vehicles have been developed to aid humans ranging from daily activities to emergencies, from personal appliances to military missions, from underwater, ground to air-space. As these vehicles are mobile units with

capabilities of computation, communication and interaction with the physical world and humans, they share similar components, including hardware, software, network connectivity, power sources and sensors. These components are vulnerable to cyber attacks because they involve or rely on the transmission of data over some form of network. Therefore, developing protection systems to safeguard them from cyber attacks is important for their safe operation.

Attacks can cause damage by unauthorised access, injecting false data, making the vehicles malfunction or stop working. These attacks do not only fail the assigned tasks and harm the state of the mobile vehicles but also the users and the surrounding environment. Cyber security will provide technologies, processes and practises to protect vehicles from computing, network and data from cyber attacks.

The severity of cyber attacks is increasing and this means that this a prime topic in terms of research. The trend now is the increasing standardised network connectivity embedded in all appliances from households to national infrastructure and the increasing usage of automated vehicles in carrying out human tasks from the simplest to the most dangerous. In previous years, cyber crimes were often linked with breaches in banking systems, websites to deny services or steal sensitive information for financial gain. Nowadays, there occurs a shift to new physical targets for terrorism or political reasons, which is sometimes referred to as cyber warfare. An outstanding example is the Stuxnet worm, which was developed by a still unconfirmed team or group of teams of highly capable developers. Stuxnet selected targets, was able to mutate through different networks worldwide and exploited four zero-day vulnerabilities in Microsoft Windows and industrial control networks. This worm has been used to control physical machinery in various industries, which may have infected up to 100,000 computers in Iran, India, Indonesia and Pakistan [Chen, 2010]. One of them was the Natanz nuclear facility in Iran and its centrifuges. It is believed that this was indeed the main or only target of the attack [Langner, 2013]. This resulted in over 1000 centrifuges having to be replaced because of physical damage caused by the malware continuously changing their rotating speed. Another example which revealed holes in cyber security within the military is the incident of a virus found in a ground

control station of the US drone fleet, which was being used to fly missions in Afghanistan. It is not known how it got there or whether classified information was leaked but the virus could not be removed with common Kaspersky security guidance and kept coming back [Shachtman, 2011].

The nature of robotic vehicles comes in many different forms of usage with specific tasks and depends on the environment. With many variables that can affect the functioning of the vehicles, it is difficult to prevent from all possibilities and once there is an attack, it is difficult to detect which one is the cause.

These types of vehicles come with limited computing resources. The solution hence must be compact and efficient. Moreover, these vehicles have a direct impact on the physical world. Their defence system has to respond time-critically to any malfunction to minimise damage. What time-critical means depends on the type of vehicle and its physical functions. In this work, the focus is generally on minimising the overall detection latency, which is the time when a cyber security incident occurs to the time that it is detected. In this sense, detection accuracy may also influence detection latency, as false negatives (missing an attack) by definition increase the time it takes to detect the attack.

In carrying cyber physical security research, there are generally three potential approaches for evaluation of proposed solutions: Analytical (Mathematical modelling and analysis), Simulation (using dedicated software) and Experimental (using actual testbed implementations) [Loukas et al., 2013b]. To maximise the realism of evaluation and the applicability of the proposed solution in real environment, we have opted for the experimental testbed approach, using genuine cyber and physical components on a real robotic vehicle. We also used simulation scripts of malware attacks against the testbed and mathematical modelling for predicting the effect of different network configurations on the process of on-board and offloaded detection.

1.2 Research questions and objectives

We have identified the following research questions:

- Can monitoring physical characteristics improve detection of cyber attacks?
- Can light-weight machine learning be used for detection of cyber threats in robotic vehicles?
- Can deep learning be used for detection of cyber attacks in cyber-physical systems?

The objectives of this thesis are to:

- Conduct a literature review of existing security issues and challenges related to cyber-physical systems with an emphasis on vehicles.
- Develop software tools for capturing security-related data from a vehicle's sensing, actuation and control.
- Investigate the feasibility of different types of cyber attacks affecting the operation of a vehicle.
- Develop an intrusion detection system that is able to provide real-time detection against different cyber attacks using Machine Learning/Deep Learning techniques, using both local and offloading techniques.
- Design a mathematical model to evaluate the practicality of offloading detection.

The scope of this work focuses on a cyber-physical security detection system for a robotic vehicle.

1.3 Key contributions

The key contributions of this thesis are:

- A lightweight attack detection mechanism for robotic vehicles, which is based on decision trees. This includes the methodology for producing decision trees for four different types of attacks, and utilising the rules produced to detect these attacks based on both cyber and physical features.

- A highly accurate attack detection mechanism for robotic vehicles, which is based on a deep learning model taking detection decisions based on both cyber and physical features.

- A methodology for offloading a continuous intrusion detection process that requires heavy processing to a remote server in near real-time. This includes the security provisioning of the approach to ensure that an attacker will not easily disrupt the detection process' confidentiality or integrity, and a mathematical model that evaluates the practicality of offloading in terms of detection latency incurred for different network configurations.

1.4 Publications

These contributions have led to a number of peer-reviewed publications. The mapping of each publication to each chapter in this thesis is included.

- G. Loukas, D. Gan and T. Vuong. Taxonomy of cyber-attack and defence mechanisms for emergency management networks. Proceedings of International Conference on Pervasive Computing and Communications (IEEE PERCOM), IEEE, San Diego, CA, USA, 18-22 March 2013
  (Utilised in Chapter 2 in relation to the review of the related literature on securing vehicles)

- G. Loukas, D. Gan and T. Vuong. A Review of Cyber Threats and Defence Approaches in Emergency Management. Future Internet, MDPI, 5(2), pp. 205-236, 2013.

(Utilised in Chapters 1 and 2 in relation to the literature review and challenges in securing manned and unmanned vehicles)

- T. Vuong, A. Filippoupolitis, G. Loukas, D. Gan. Physical Indicators of Cyber Attacks against a Rescue Robot. Proceedings of International Conference on Pervasive Computing and Communications (IEEE PERCOM), IEEE, Budapest, Hungary, 24-28 March 2014.
  (Utilised in Chapter 3 in relation to the design of the testbed, experimental methodology and observations of physical impact caused by cyber attacks on the testbed)

- Tuan Vuong, George Loukas and Diane Gan. Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. 13th International Conference on Pervasive Intelligence and Computing (IEEE PICOM 2015), IEEE, Liverpool, UK, October 26-28, 2015.
  (Utilised in Chapters 3 and 4 in relation to the experimental methodology and the design, setup and experimental performance evaluation of the decision tree based algorithm for cyber-physical intrusion detection against different attack types)

- Tuan Vuong, George Loukas, Diane Gan, and Anatolij Bezemskij. Decision Tree-based Detection of Denial of Service and Command Injection attacks on Robotic Vehicles. 7th International Workshop on Information Forensics and Security (IEEE WIFS 2015), IEEE, Rome, Italy, November 16-19, 2015.
  (Utilised in 4 in relation to the experimental evaluation of the decision tree based algorithm for cyber-physical intrusion detection against different attack types)

- George Loukas, Yongpil Yoon, Georgia Sakellari, Tuan Vuong, Ryan Heartfield. Computation offloading of a vehicle's continuous intrusion detection workload for energy efficiency and performance. Simulation Modelling Practice and Theory, Elsevier, 2016.
  (Utilised in Chapter 5 in relation to the motivation for offloading the continuous task of deep learning based intrusion detection to a remote server)

1.5 Thesis Summary

The thesis is structured as follows. Following the current introduction (Chapter 1), Chapter 2 contains the literature review related to this thesis, including the general challenges in security cyber-physical systems, as well as more specifically the different types of cyber attacks on vehicles and corresponding defence mechanisms that have already been proposed. The focus is on intrusion detection, and especially machine learning-based approaches.

The technical development of the project began with the preparation of a testbed and a standard set of scenarios for the first set of experiments. This involves a small tracked robotic vehicle with embedded software modules that is remotely connected to a controller PC.

Chapter 3 introduces the design of the testbed, experimental methodology, setup and observations of the physical impact caused by cyber attacks. This includes a description of the decision tree algorithm for cyber-physical intrusion detection for different attack types. A set of monitoring modules has been added to the vehicles code, in C++, R, Python and shell scripting, to output different status values for its power usage, network traffic, disk and CPU usage and sensed movement characteristics. An automation of scenario experiments was coded using automated operator's keyboard and mouse action based on Jitbit automation tool. A series of attacks have been carried out on the testbed, with an emphasis on denial of service attacks, command injection and simulated malware against the network and the CPU which affected the physical behaviour of the robotic vehicle. These used both publicly available denial of service tools and ones that have been developed during this work. We use the monitoring tools that have been developed (above) to gather the cyber-physical impact data for the range of attack scenarios described and analyse the usefulness of physical features in detecting cyber attacks.

In Chapter 4, we present the design and development of an intrusion detection method appropriate for the characteristics of a robotic vehicle that has resource constrains in

terms of power consumption and processing capacity. A Machine learning Decision tree-based approach for generating simple detection rules. The program for this framework was written in R utilising C50 library. The experimental results of the experiments show the feasibility of using a light-weight detection system onboard the vehicle.

Chapter 5 presents a more accurate detection solution that can utilise the temporal characteristic of cyber-physical data collected during the attack scenarios by employing a deep learning based approach. The Deep Learning training and detection programs were implemented with Python neural network library Keras to run on top of the Theano Deep learning library. This solution has a better detection accuracy but is more processor demanding. To minimise the detection latency occurred due to the heavier processing, we employ computational offloading, where the vehicle offloads its intrusion detection task to an external cloud infrastructure. This chapter also provides a mathematical model for evaluating in which cases offloading is beneficial in terms of detection latency.

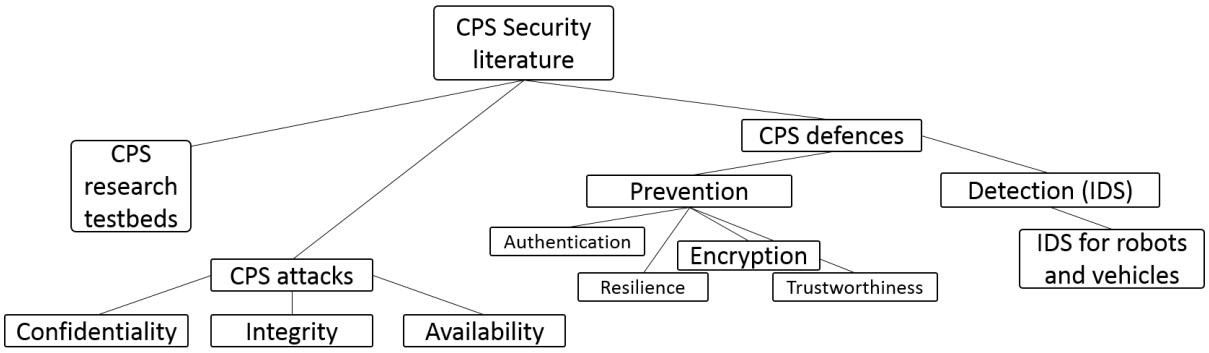
Chapter 6 provides the conclusion of the thesis, including a summary of this project's achievements and its wider applications, as well as a list of possible future research directions extending it, and some final remarks.

Chapter 2

Literature review

While physical damage has been traditionally caused by physical means and cyber damage by cyber means, our increasing dependence on highly automated and networked systems, from industrial control to robotic vehicles, has generated formidable cyber-physical vulnerabilities. These systems differ from traditional cyber targets because the CPS attack surface impacts on the physical components, the communications and/or the computing elements of the device and will have a physical effect. The operation of cyber-physical systems such as robotic vehicles often depends heavily on computer networks. A cyber attack against or through an associated network affects the movement of a vehicle in a manner that currently cannot be predicted or prevented. We start by articulating the cyber security of CPS and provide a taxonomy of ongoing research in term of testbeds, attacks and defence mechanisms (Figure 2.1). This chapter will also highlight the security challenges and threats in vehicular technologies, discuss the intrusion detection techniques for the robots and vehicles, especially with machine learning and deep learning algorithms.

Figure 2.1: Taxonomy in cyber-physical systems (CPS) security



2.1 CPS and mobile CPS security

CPS systems are becoming highly integrated in the modern world. Its application ranges from first responder situational awareness systems, pervasive health-care systems, smart grids and unmanned aircraft systems, as well as small robotic vehicles for use in emergency systems [Loukas et al., 2013a, Mitchell and Chen, 2014]. Having specific characteristics of size, sensor, actuators, control and network ability, these robots are able to benefit an emergency response procedure by searching for survivors, providing access to inaccessible areas and establishing an on-site communication network. This is different from the traditional CPS in which nodes are usually stationary. Mobile CPS indicates a CPS with a mobile capability integrated with a mobile device or hardware. Here, we consider networked (either manned or unmanned) ground, air or underwater vehicles with an interconnection of sensors, actuators and controllers to constitute Mobile CPS. A cyber attack against or through an associated network gives rise to a range of new security challenges. Such incidents have already been reported to have occurred both in the wild [Loukas, 2015], [Templeton, 2011], and in controlled experimental environments [Koscher et al., 2010], [Kerns et al., 2014], and competitions [Griffiths, 2014].

This paper [Vuong et al., 2014] investigates how a cyber attack on a robotic vehicle can adversely affect its operation and impair an emergency response operation. The focus is on identifying physical indicators of an ongoing cyber attack, which can help to design effective detection and defence mechanisms. A number of experiments have been

conducted on an Arduino based robot, under different cyber attack scenarios. The results show that the cyber attack effects have physical features that can be used in order to improve the robot's robustness against this type of threat.

A comprehensive review of cyber threats related to vehicular technologies, with an emphasis in emergency management, is given in [Loukas et al., 2013a], while a taxonomy providing a global view of the respective attack types and defence mechanisms is presented in [Loukas et al., 2013b]. In addition, the use of robotic robots for the establishment of a communication network between trapped civilians and an operation centre is presented in [Wasicek et al., 2014], [Timotheou and Loukas, 2009] and [Xue et al., 2014].

Research by [Koscher et al., 2010] has demonstrated that cyber-physical attacks can target production automobiles, since these vehicles incorporate various sensing and computing modules that can interact with each other in multiple ways. Initial attacks infected the vehicles electronic systems through the use of an audio file in the MP3 player device or through a smart-phone connected via Blue-tooth. One of the possible results of this attack is a change in the driving direction while the vehicle is in motion.

Apart from automobiles, another popular type of attack is related to unmanned aerial vehicles such as military UAVs. Iraqi militants used off-the-shelf software in order to intercept UAV video feeds. The original use of the software was satellite TV interception, however it could successfully apply to unencrypted military feeds as well [Gorman et al., 2009]. This incident resulted in military air-craft being retrofitted with video encryption modules. US military UAVs have also been the target of cyber-physical attacks. In 2011 numerous UAVs were infected by viruses, which resulted in the installation of key-logging software. The most probable motive for this attack was the creation of a mapping between the signals emitted by the pilots keystrokes and the corresponding vehicle parts that were operated. Moreover, Iranian television broadcasted images of a US UAV and claimed that it was hijacked and landed intact using electronic warfare. Since military vehicles have been targeted and compromised by cyber attacks, it is evident that civilian UAVs used by the police or by emergency services can also be hijacked and potentially

flown into a crowd with catastrophic results. Researchers from the University of Texas have also demonstrated this using a helicopter drone [Sabaliauskaite and Mathur, 2014]. Furthermore, researchers at Purdue University have investigated the autopilot mechanism of UAVs and have modelled numerous cyber-attacks that could exploit it [Skormin et al., 2014]. The authors of [Felix et al., 2014] have conducted a security assessment by performing cyber-attacks on a robot running the Robot Operating System and found that it is vulnerable to insider threats and to being physically compromised. Finally, an enhanced telesurgery protocol is presented in [Roesch et al., 1999] which aims at addressing the stringent requirements related to telesurgical robotics.

The aforementioned approaches focus on cyber attacks against pedestrian and aerial vehicles, including potential detection and defence mechanisms. They do not address directly, robotic vehicles. The aim of this work is to investigate cyber attacks targeted at robotic vehicles, in order to provide an effective detection mechanism based on physical and cyber indicators.

2.1.1 Security challenges

Cardenas et al. [Cárdenas et al., 2008] have discussed the fundamental challenges for control systems security. In their view, problems in control systems can be defined as computer-based systems that monitor and control the physical processes. The authors have listed multiple vulnerabilities and threats due to the problem of implementation flaws ("bugs") as in computing systems, networked issues in an ever-growing Internet and devices, complex system inherits of other exposed software and hardware components, open design where details are accessible to everyone, ever increasing scope and functionality of sensors and actuators, and organised cyber-crime from highly skilled groups. The challenge arises when the new security problem relates to an estimation and control algorithm, which affects the physical world. The authors also suggested understanding the consequence of an attack, designing a new detection algorithm and better attack-response algorithms and architectures.

In [Wells et al., 2014], other aspects of cyber-physical security challenges in manufacturing systems have been discussed, sharing similar security issues with the manufacturing industry where there are more and more networked devices. The attack vectors increase with the Internet of Things (IoT) that depends on Software as a Service and Cloud Computing. With a weak point in the chain of the system, the attacks can overcome the cryptography and perform intellectual property theft. There are distinctly different requirements for manufacturing where security research has not been developed. Quality Control in manufacturing ensures the process stability but has not yet considered cyber attacks as the root cause and manufacturing workforce need to be aware of cyber attack threats to their tools and systems.

Knightscope, a Silicon Valley start-up, developed a robot that was ready to take over a human security guard's job by patrolling an area and reporting to a remote security centre any suspicious trespassing and anomalous activities. The robot is a moving computer of nearly human-height (150 cm) and a weight of 135 kg of equipment and batteries. Its intelligence lies in the corporation of a map with multiple sensors and navigation tools. It takes in the surrounding variables such as high-definition images and weather, GPS location, laser ranging to understand different contexts and respond by sending information out or raising alarms for attention [Metz, 2014]. This event first demonstrates the use of robots in public service is getting closer to reality, which then denotes an urgency and challenge to find advanced protection against cyber and physical attacks in order to ensure the safety of the robots themselves, the humans in interaction and the assets involved. For this reason, in this thesis, a robotic vehicle is used as an example of an array of different CPS applications integrated in the physical world. The sensors in CPS systems provide availability of a case of big monitoring data along with fruitful challenges and opportunities differently from traditional IT systems [Sharma et al., 2014].

In [Pasqualetti et al., 2013], CPS is tackled from survivability matters using model-based analysis. The model of CPS as linear time-invariant descriptor systems helps to capture various real-worlds CPS (municipal water supply network and electrical power grid) and generate prototypical attacks.

An extended discussion of energy efficiency was studied for BAN (Body Area Network). [Venkatasubramanian et al., 2009] provides a solution on the energy footprint of a cyber-physical security solution that uses a physiological signal based Key Agreement (PKA). The energy requirement for PKA is computed for energy scavenging techniques such as body heat and ambulation.

When studying automotive embedded systems, [Anthony et al., 2008] and [Anthony et al., 2009] provide analysis on advanced dynamically middle-ware, self management for context-ware. A dynamic self-managing implementation of middle-ware and component architecture is provided to facilitate a flexible run-time configuration via the embedding of dynamically replaceable decision logic into software component. The application is illustrated in adjusting the heating-cooling in-car temperature to prove the credibility of this concept.

Figure 2.2: Brief summary of the security-related challenges encountered in cyber-physical systems

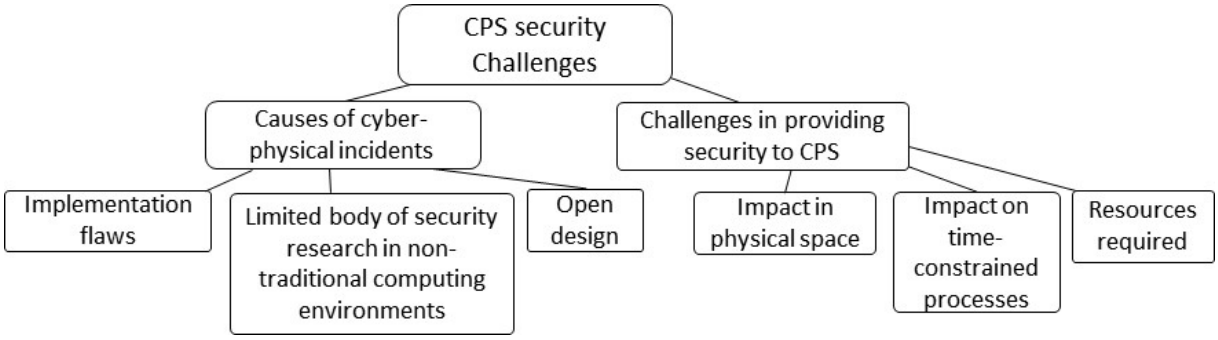


Figure 2.2 summarises the general challenges encountered in cyber-physical systems, as described above. In Section 2.2.2, we focus further on vehicular technologies and their more specific challenges within the CPS landscape.

2.1.2 CPS testbeds

CPS involves a large number of interdependent processes and technologies, which is difficult to replicate in a research environment. For this reason, researchers tend to use only small experimental implementations, often complemented by mathematical models and

software simulations. In the following, we discuss the three different testbed categories: Mathematical, Simulation and Experimental briefly, and identify the methods that this research employed.

Mathematical modelling

Mathematical modelling has traditionally been used in CPS for optimisation, decision support and risk analysis. The introduction of cyber threats has added additional complexities that need to be captured mathematically. Such an example is the impact dependency graph model presented in [Jakobson, 2011]. This paper [Mitchell and Chen, 2011b] also addresses the survivability matters of mobile CPS. Based on a mathematical model, mobile CPS energy exhaustion and security failure is assessed to propose a solution to balance energy conservation *Vs.* intrusion tolerance. Testing with dynamic voting-based intrusion is executed for identifying the optimal design setting.

Simulation Existing software simulators for CPS are often for UAV [Birnbaum et al., 2014]. The author designed a monitoring system for detecting security issues using flight data and real-time airframe and controller parameters from UAV simulating system. There is also RTAW-Sim, which has been designed specifically for simulating the traffic on controller area network (CAN) bus for an automobile [Navet, 2011]. Also, authors in [Kang, 2016] have proposed a real-time IDS based on the input features from simulated CAN bus data.

Experimental While mathematical modelling and simulation can often represent CPS sufficiently, an actual scaled testbed with real components naturally offers more realism. One such system is the U.S. Department of Homeland Security (DHS) Glanser, which is a collection of human-portable sensors and vehicle-mounted base stations. Mitchell and Chen have used the DHS Glanser system to experiment on detection mechanisms that would be applicable to cyber-physical systems [Mitchell and Chen, 2011a]. [Schumann et al., 2015] discussed on monitoring and diagnosis of security threats for UAV. Real-time, responsive, unobtrusive unit (R2U2) is a framework for run-time monitoring security threat properties. The monitored inputs are from GPS, ground control station, sensor

readings, actuator outputs, and flight software status. The design and implementation of R2U2 with statistical reasoning enables security threats on the NASA DragonEye UAS.

In general, in this thesis, the focus is on experimental evaluation based on a real testbed. We also use mathematical modelling for predicting the effect of different network configurations on the process of detection, which has been validated experimentally.

2.2 Attack Mechanisms

Cardenas et al. [Cardenas et al., 2008] provides an excellent investigation of security of Cyber-Physical Systems (CPS), which consists of computing and communication capabilities with monitoring and control with network agents including sensors, actuators, control processing units and communication devices. This fits in with the domain of semi-autonomous vehicles. Then, the authors use the approach of traditional security goals CIA (Confidentiality, Integrity and Availability) in information security to represent two threats, including deception attacks and Denial of Service (DoS) attacks. It also discusses related work in automatic control such as robust networked control systems, fault tolerant control, distributed estimation. After analysis of information security and automatic control, two definitions are pointed out regarding survivability and secure control with different claims regarding CPS. Finally, the authors suggest a set of challenges that can enhance the survivability of cyber-physical systems.

2.2.1 Cyber security triad

Cyber security attacks are often categorised as a triad of three principles: Confidentiality, Integrity, and Availability.

Confidentiality Confidentiality within CPS systems is about protecting the communications channels from eavesdropping. These communication channels are found between actuators, sensors and controllers and are the links between the physical and cyber com-

ponents of these systems [Lu et al., 2015, Wang et al., 2010, Gamage et al., 2011]. Confidentiality attacks against vehicles have gained notoriety since 2009, when militants in Iraq used cheap off-the-shelf software to intercept live video feeds from unmanned aerial vehicles (UAV) [Gorman et al., 2009].

Integrity Researchers have also shown that integrity attacks on a vehicle can have severe repercussions on its movement control, especially at speed. As a case study, they infected a common production automobile using an infected MP3 play-list and Blue-tooth connection with a smart-phone [Koscher et al., 2010].

Research dealing with cyber-physical attack detection has mainly focused on integrity attacks against industrial control systems. Attackers modify the payload of a network packet and manipulate a cyber-physical system into performing the wrong physical action [Amin et al., 2009]. Replay attacks are another type of cyber attack which is difficult to detect using generalist approaches. They target systems which are expected to be in steady time for a long period of time. The authors in [Mo et al., 2014] present a detection method for replay attacks on the sensors of supervisory control and data acquisition (SCADA) systems. Another approach for cyber attack detection involves measuring anomalies between physical and cyber properties of a cyber-physical system. These methods are inherent to the nature of a cyber-physical system but must overcome numerous challenges, such as timing. The work presented in [Gao et al., 2010, Reaves and Morris, 2009] detects integrity and availability attacks against a storage tank control system by using the water level measurements reported by the SCADA module. The approach is based on the fact that water levels can only change at rates related to pipe diameters and tank capacity. A similar approach based on semantic errors is presented in [Naess et al., 2005] and uses temperature readings outside a specific range to detect an intrusion. The authors state that by integrating the intrusion detection module in the middle-ware layer of an embedded system, they can achieve better results due to simultaneous access to application logic and communication streams among distributed components.

Availability Availability attacks, such as common Denial of Service attacks, affect ve-

hicles by delaying or preventing commands from reaching the movement control system. Researchers have addressed availability of cyber-physical attacks only in industrial control systems [Amin et al., 2009]. In [Loukas and Öke, 2009], a survey on protection against DoS attack is described. The authors provide a timeline of incidents, types and motives of DoS from the early 1980s to 2009. After identifying a defence mechanism, detection and response of a DoS attack and identifying the true source of an attack, based on a mathematical model, the paper identified the trends in DoS attack, weakness of protection approaches and also suggests several directions for future research on this topic to pursue. [Timotheou and Loukas, 2009] emphasises on the timely response for autonomous networked robots to operate in an emergency situation. In [Gelenbe et al., 2005], both analytical and simulation modelling and experiments on an autonomic routing protocol are carried for evaluating advantages and disadvantages in the imperfect detection of DDoS (Distributed Denial of Service) attack. [Gelenbe and Loukas, 2007] surveys classification and defence mechanisms, passive methods, mathematical models of evaluation of the benefits of DDoS defence, correct detection.

Such systems are inherently predictable as they perform routine operations and any deviation of the network traffic from an expected behaviour can be flagged as suspicious. Vehicles, on the other hand, are affected in a less predictable manner. Depending on the implementation approach and attack type, they might be forced to reset, continue moving blindly, jitter, delay changing direction etc.

This PhD analyses experimentally the impacts of availability and integrity attacks such as DoS, command injection, and malware attacks on the movement of a robotic vehicle, and utilises them to detect attacks based on cyber and physical features that can be collected in real-time.

2.2.2 Security threats in vehicular technologies

Most modern vehicles include sophisticated computational and sensing technologies for their controls, and may soon include ad hoc vehicular networks supporting their commu-

nications. At the same time, there is increased interest in unmanned vehicles, for example in emergency managements (EM) situations for reaching locations that are inaccessible to humans. We have summarised related cyber threats in Table 2.1.

2.2.2.1 Manned Vehicles

In the public transport sector, cyber attacks usually cause disruption in dispatching and signalling. In the 1990s they were related primarily to the lack of user authentication mechanisms, with hackers connecting via a dial-up modem to an airport network pretending to be the legitimate system administrator and altering critical information. Today, computer viruses and targeted cyber attacks affecting mass transportation are relatively common, especially in railways and airports [Turk et al., 2005]. Due to the increasing use of off-the-shelf computers running Microsoft Windows, a number of incidents in the transport sector were caused by common viruses and worms that spread via the Internet and infected computers indiscriminately, with one such virus disabling air traffic control systems in Alaska in 2006 [De Cerchio and Riley, 2012]. Yet, in most cases, there was no malicious intent and, more significantly, there was no damage beyond frustration and financial costs due to downtime. In 2008 though, a teenager managed to take control of the tram system in Lodz, Poland, and operated its track switches, eventually causing four trains to derail and 14 people to be injured [Storey, 2009].

The automotive industry is also increasingly showing interest in cyber threats, partly because of isolated incidents of cyber intrusions against specific car types and partly thanks to the pioneering work [Koscher et al., 2010]. In 2010, the latter demonstrated that it is possible to infect a car's networks via Bluetooth and other mechanisms and gain control of its locks, brakes and engine. A car interfered with in such a manner may be forced to veer towards one direction while driving at speed. An incident in Austin, Texas, the same year showed that large numbers of private cars can be simultaneously affected in an unexpected manner through a web-based vehicle immobilisation system. The specific system had been set up by car dealers to disable a car's ignition system

as a response to delinquent car payments, but it was exploited by a hacker who gained unauthorised access and issued rogue commands. Thus, it is particularly interesting that a website's security flaws had an indirect physical impact, causing 100 private cars to be simultaneously immobilised [Schoitsch, 2012].

2.2.2.2 Unmanned Vehicles

Unmanned Aerial Systems (UAS) have already started being used for civilian purposes, including law enforcement and emergency response, as they can provide aerial imagery of high resolution that enhances situational awareness and coordination. However, the security of unmanned aerial vehicles (UAVs) has been repeatedly breached in high-profile incidents in the past. In 2009, militants in Iraq used cheap off-the-shelf software to intercept live video feeds from US Predator drones and in 2011 a virus infected a number of US Predator and Reaper UAV drones, logging the keystrokes of the pilots who remotely controlled them [Shachtman, 2011]. The same year, Iranian TV showed a US RQ-170 Sentinel drone claiming that it had been electronically hijacked and landed by the Iranian army's electronic warfare unit [Cole, 2012]. According to [Ráček et al., 2012], this was achieved by spoofing the GPS signals to the drone and then tricking it into landing in Iran rather than Afghanistan. The drone reported that it was landing at its home base. It is important to note that as UAVs are sizeable objects, if hijacked and deliberately crashed into a crowd or other target, they could cause considerable physical damage themselves.

The elevated significance of UAV hijacking in the defence sector has sparked research in UAV cyber security. Most notably, a research team at Purdue University has created a simulation testbed that models UAV control systems and flight operations. Their current primary focus is on vulnerabilities of the autopilot systems. Up to now, they have confirmed that GPS spoofing can lead a UAV astray and in a manner that may not be detected by the legitimate operator. In addition, they have analysed a gain-scheduling attack affecting the vehicle's controls and stability through sensor spoofing, and a fuzzing attack where the attacker injects random inputs to the vehicle's actuators [Kim et al.,

2012]. In parallel, a less technical high-level analysis of the potential impact of cyber attacks on UAVs has been presented in [Javaid et al., 2012]. It is important to note that a recent congressional report on unmanned aircraft systems has stated that vulnerabilities in the command and control of UAS operations are a primary obstacle to their integration into the national airspace system [Dillingham, 2012].

Land vehicles that operate in an autonomous or semi-autonomous remotely-controlled manner are also increasingly proposed and trialled for industrial maintenance, military and emergency management applications. Such vehicles can reach areas often inaccessible to human beings and even set up an ad hoc communication infrastructure [Loukas et al., 2008]. However, unmanned vehicles are typically not designed with information security in mind and are vulnerable to multiple types of attacks affecting the collection or communication of critical information. An excellent survey of these cyber threats has been presented in [Kohno, 2012].

Technology	**Security threat**	**Impact**	**Countermeasures**
Manned Vehicles	Malware infection of traffic control systems [De Cerchio and Riley, 2012]	Disabled air traffic control, signalling *etc.*	Web security literature [Jensen et al., 2009]
	Malware infection of onboard computers [Koscher et al., 2010]	Hijacked control of locks, brakes and engine	Malware detection [Koscher et al., 2010]
	GPS Spoofing [Tippenhauer et al., 2011, Kim et al., 2012]	Artificial traffic jam caused	Signal analysis [Warner and Johnston, 2003, Jafarnia-Jahromi et al., 2012, Zeng et al., 2012, Carson et al., 2016]
	Web-based immobilisation hijacked [Schoitsch, 2012]	Cars immobilised remotely and simultaneously	Web security literature [Jensen et al., 2009]
Unmanned Vehicles	GPS Spoofing [Tippenhauer et al., 2011, Kim et al., 2012]	Unmanned vehicle redirected [Kim et al., 2012, Javaid et al., 2012]	Signal analysis [Warner and Johnston, 2003, Jafarnia-Jahromi et al., 2012, Zeng et al., 2012, Carson et al., 2016]
	Gain-scheduling attack [Kim et al., 2012]	Control stability affected [Kim et al., 2012]	No known solutions
	Fuzzing attack [Kim et al., 2012]	Random inputs to vehicle's actuators [Kim et al., 2012]	No known solutions

Table 2.1: Vehicular security threats and proposed countermeasures.

2.3 Defence Mechanisms

2.3.1 Preventative methods

Due to standardisation and connectivity to the Internet, many systems are sharing the similar threats caused by cyber attacks. [Zhu and Sastry, 2010] and [Zhu et al., 2011] provide a comprehensive survey and taxonomy of intrusion detection and prevention systems for Supervisory Control and Data Acquisition(SCADA)-specific systems. SCADA systems are generally deployed in large critical infrastructure systems such as electrical power grids, petroleum and gas pipelines, water and waste-water systems. SCADA is a hard real-time system with deadline deterministically, limited computing capabilities and memory resource, with a direct impact in the physical world which might cause safety issues. In [Zhu et al., 2011], cyber attacks on SCADA start off from an Internet connection, to layers of control networks, which can be grouped into: hoax input to the controller transferred from compromised sensors and/or exploited network link between controller and sensors. These cyber attacks can happen in the hardware, software or communication stack.

Authentication Authentication of users and network traffic can prevent cyber attacks against a CPS. An example is the access control scheme presented in [Wu et al., 2011] that proactively and dynamically modifies permissions during an emergency without explicit access requests.

Encryption Another common approach is to strengthen the encryption of the messages sent. This is typically used in relation to authentication, such as to lock/unlock an automobile [Latka, 1994] or when exchanging messages in a vehicle-to-infrastructure (V2I) scheme [Chuang and Lee, 2011].

Resilience A resilient CPS is one that ensures an acceptable level of operation in the presence of cyber threats. Resilience can be improved by ensuring that the first line of defence, such as firewalls and other security components, are patched and updated

properly [Walker, 2011], but can also be engineered into the systems design. Another common approach for achieving resilience is redundancy, as in a CPS context where communications may be affected by physical damage too, one needs to ensure that redundant network links or nodes are not in the same physical location and cannot all be taken out by a single physical event [Sterbenz et al., 2013].

Trustworthiness Another approach is to integrate both validation and trustworthiness in CPS. Validation in different components might all be correct but the system overall may not be dependable [Eze et al., 2012]. Adding trustworthiness at the architectural level would increase the overall security of the autonomic system.

2.3.2 Reactive methods

In contrast to preventative methods, reactive methods operate while an attack is occurring or afterwards.

2.3.2.1 Detection

Detection mechanisms aim to limit an attack's impact by identifying its existence and often its type. Their success depends largely on the input features that they use. In cyber-physical systems, for access control, early warning, sensing, and physical control, where computational, communication and physical processes have a direct impact on each other, we classify detection mechanisms based on the cyber or physical nature of their input features:

- Cyber input. Naturally, most detection mechanisms proposed to defend against network attacks use computational and communication data, such as packet source, data rate and protocol-specific characteristics, as their input features.
- Physical input. An example of physical input would be the monitoring of the GPS signal strength. Warner et al. have observed that GPS spoofing systems use signals

of much greater strength than legitimate GPS signals [Warner and Johnston, 2003].

- Combined cyber and physical input. In principle, this approach makes the best use of the dual nature of cyber-physical systems. For example, by linking cyber detection with physical monitoring, such as video surveillance and a central security room to monitor and report incidents, one may facilitate detection of suspicious cyber-physical behaviour [Rajamäki et al., 2012]. Chen et al. have proposed to use fuzzy logic to combine real-time network data and physical input features, including the differences between the values reported by neighbouring sensors [Chen et al., 2011]

Mitchell and Chen provided a discussion of intrusion detection techniques for CPS (including vehicles) which comprises two classification groups: detection techniques and audit materials. Detection techniques looked for the misbehaviour of the CPS while audit material identified whether the collected data was host-based or network-based [Mitchell and Chen, 2014].

In general, detection techniques can be categorised as behaviour-based, typically independent of the type of attack, e.g. based on Petri Nets [Skormin et al., 2014], or based on the knowledge of what impact a particular attack has on a particular robot.

Knowledge-based

Aspect-oriented modelling (AOM) for accessing security of Cyber-Physical Systems is introduced in [Wasicek et al., 2014]. AOM was inspired by Aspect-Oriented Programming (AOP). Incorporating AOM in the process design was claimed to retain the development method for CPS. The authors selected an adaptive cruise control system as an automotive case study, and applied the AOM to analyse four different cyber-attacks such as man-in-the-middle attack, fuzz attack, interruption attack and a replay attack. Then the control behaviour was analysed and evaluated for detection of different attacks.

Another security model is proposed in [Xue et al., 2014] for the dynamics of robotic vehicle networks. The authors applied a canonical double-integrator-network model to provide

security level analysis in both the noise-free case and the noise case. In both cases, security levels varied in terms of a graph matrix, graph topology and its spectrum. This proved the importance of the inter-vehicle communication topology and control design in pursuing security.

The authors in [Sabaliauskaite and Mathur, 2014] declared the usage of an Intelligent Checker (IC) and Cross-correlator (CC) for improving the security and safety of the cyber-physical system. ICs were smart sensors that were independent from the CPS both behaviourally and structurally. The ICs consisted of IC sensors which collected parameters from controlled processes and decision logic to verify the condition restriction. The ICs were successful in detecting cyber and physical attacks and sending an alarm to the operators. The CC approach was to detect False Data Injection Attacks that changed sensor reading values in Linear Time Invariant. The method depended on the analysis window size of control loops sensors and trusted sensor measurement and outputs. Dynamically the window size could enhance the efficiency of this method. Using these two security countermeasures, the detection probability reached 90% with a window size of 50 readings. Even with a critical failure of sensors, when no measurements were given to the CC, the IC could help to lead the vehicle to safety.

Behaviour-based

A behaviour-based approach was proposed in [Skormin et al., 2014] for the diagnostics of CPS for timely detection, identification and prediction of impact on computer control systems. The impact could have been caused by cyber attacks or a misused procedure. The behavioural model approach used connecting systems calls in an object access mega graph which consists of functions mapping for system calls and data links. Hence, the graph can be represented by Colored Petri nets to provide the basis for anomaly detection. In the case of diagnosis of failure caused by a cyber attack, a visual graph of component calls which differed from the normal profile could assist the expert analysis of detection and identification of cyber attacks. Hardware failures, which might be related to cyber attacks, could rely on the Recursive Least Squares algorithm for anomaly detection according to

the design specification.

2.3.2.2 Response

After a cyber attack against CPS is detected, a number of actions may be taken as a response, triggered automatically by the detection mechanism or manually by a human administrator that has been alerted. We have identified the following families of applicable response mechanisms:

- Network traffic control. Network traffic that fails authorisation or appears suspicious may be rate-limited, filtered, relegated to lower priority or simply dropped altogether [Gelenbe et al., 2012].
- Network reconfiguration. Recovery can also be achieved by changing the configuration of the network, such as its logical or physical topology, the routing criteria, policies etc. A generic approach for an agent-based infrastructure that achieves self-healing through reconfiguration of an overlay network has been proposed in [Sheldon et al., 2005].
- Shut-down. On some occasions, it may be preferable to shut down specific services or parts of the network, or the whole system, especially if a detected cyber attack is affecting the confidentiality or integrity of ongoing communication channel in CPS.

2.4 Intrusion detection for robots and vehicles

Intrusion detection in cyber-physical systems is a relatively new area of research [Mitchell and Chen, 2014], which tends to focus on industrial control systems, such as programmable logic controllers and supervisory control and data acquisition (SCADA) systems. The range of related research that is applicable to robotic vehicles is still rather limited. One approach that has been proposed is to base the detection on whether sensor values collected and utilised by a robot are realistic. For instance, the intrusion detection system

of the iRobot Create robotic platform needs to first determine what sensor values are "normal" and from these try to determine whether network traffic is suspicious [Uluagac et al., 2014]. One approach for this is to include an onboard intrusion detection module, which would be trained offline to learn simple rules in relation to its own normal behaviour [Bezemskij et al., 2016b, Bezemskij et al., 2016a] or to the signatures of different types of known attacks based on different monitored features. When in actual operation, the vehicle would then monitor these features and apply the simple rules, which would have a relatively low processing load. This approach can work well for simple and previously seen attacks. The authors in [Birnbaum et al., 2014] present a detection approach using flight data and real-time airframe and controller parameters with statistical method Recursive Least Squares technique. In [Birnbaum, 2015], the author extended this work with a behavioural profiling technique for anomaly detection. Behaviour monitoring for unmanned autonomous systems (UAS) has been suggested to require a systematic method to monitor at both lower level operation and high level mission execution [Felix et al., 2014]. By having the mission profile implementation checked from both levels, different cyber attacks could be detected by intelligent and machine-learning algorithms. The lower level algorithm read mission domain data to compare with function I/O data for rule evaluation. Mission Monitor ensured the validity of the energy consumption from subsystems, navigation and control analyser and behaviour coherence, otherwise raising alarm flags and taking the vehicle to a safety.

In most cases, some form of network monitoring for detecting attempts for control hijack or modification is necessary, whether host-based as in the robotic surgery arm discussed in [Bonaci et al., 2015a], or a hybrid host-based and network-based, as in [Shetty et al., 2014]. In [Fagiolini et al., 2014], detection is conducted both locally and collaboratively within a group of autonomous robots, by taking into account reputation, behaviour scores and distance. For distributed systems, which include multiple mobile elements, as in emergency response or battlefield situations, one can use a voting-based system, such as the one proposed in [Mitchell and Chen, 2013b].

Depending on its architecture and application, a robotic vehicle may be able to benefit

Table 2.2: Intrusion detection for robotic and mobile cyber-physical systems

Ref.	Type	Comms	Location	Attack Types	Input Features	Detection approach
Mitchell, Chen [Mitchell III, 2013]	*Mobile CPS*	*Wireless*	*Host Based, Network Based*	*Command Injection, Node Hijack*	*Position, Battery Exhaustion Nodes Compromised*	*Dynamic IDS Voting, Positional change, Enviroconsistency*
Fagiolini et al. [Fagiolini et al., 2008] [Fagiolini et al., 2009]	*Multi-Robot System*	*Wireless*	*Host Based, Decentralized*	*Misbehaviour*	*Node Reputation, Behaviour score, Distance Estimation*	*Clustered Monitoring, Voting*
Bezemskij et al. [Bezemskij et al., 2016b] [Bezemskij et al., 2016a]	*Robotic-Vehicle*	*Wireless*	*Host Based*	*Replay packet, Rogue node, Compass exploit*	*Sensors, Networks, Processing data*	*Behaviour-based anomaly detection*
Birnbaum et al. [Birnbaum et al., 2014] [Birnbaum, 2015]	*Virtual UAV Platform*	*Wireless*	*Host Based*	*Cyber attack, Sensor spoofing, Structural failure*	*Flight data, State data, Servo response*	*Behaviour-based detection*
Felix et al. [Felix et al., 2014]	*Conceptual UAS*	*Wireless*	*Host Based*	*Cyber attacks*	*Mission domain, I/O data*	*Behaviour coherence*
Bonaci et al. [Bonaci et al., 2015b]	*Robotic Surgery System*	*Wired*	*Host Based, Network Based*	*Intent Modification, Control Hijack*	*Motor Performance, Network Performance*	*Recommendations for Network Monitoring*
Shetty et al. [Shetty et al., 2014]	*Multi-Robot System*	*Wireless*	*Host Based, Network Based Decentralized*	*Denial Of Service*	*Lack of Connectivity*	*Network Monitoring*
Vuong et al. [Vuong et al., 2014] [Vuong et al., 2015a] [Vuong et al., 2015b]	*Remote-controlled Robot*	*Wired, Wireless*	*Host Based*	*Denial Of Service, Command injection, Malwares*	*Motor, Vibration, Power, Network, CPU, and Disk data*	*Rule-based, Machine learning*
Zeng et al. [Zeng and Chow, 2014] Fagiolini et al. [Fagiolini et al., 2014] Bicchi et al. [Bicchi et al., 2008]	*Multi-Robot System*	*Wireless*	*Host Based, Role Based Network Based Decentralized*	*Node Failure, Node Misbehaviour*	*Network Performance, Behaviour Score, Node Reputation, Neighbour State, Neighbour Actions, System Configuration, Agent Position*	*Reputation Based, Consensus Based, Set-Valued Consensus*

from communication with other agents (multi-agent) or may need to rely solely on its own sensing capabilities and monitoring processes (single-agent). Multi-agent approaches focus on the coordination between different agents (e.g, between different driverless vehicles or robots [Shetty et al., 2014]) in an effort to detect one agent's suspicious actions, reports, configuration or location. The detection criteria may include consistency with the laws of physics (e.g. velocity measurements that are physically possible, or a location that is consistent with the velocity measured [Loukas, 2015]), consistency with the sensor measurements reported by neighbouring agents [Mitchell and Chen, 2011a], voting [Mitchell and Chen, 2013a], reputation scores etc.

Most of these approaches and detection criteria become impractical in single-agent systems, such as the single remote-controlled robotic system used here. Without the opportunity to coordinate with multiple other robots, the focus has to shift to the identification of relevant characteristics that can be measured by its own onboard systems.

In addition, most discussed approaches do not take advantage of the temporal elements of the data features in a cyber attack against a robotic vehicle. For example, a cyber attack such as command injection attack may make the vehicle to make a sudden turn. Then, the impact of this attack causes a change in the vehicle wheel speed, which expands the power utilisation over a short period of time. This alteration happens one after the other in a specific sequence. Such sequentially-related behaviours represent a time series dataset which can be desirable for detection algorithm that can recognise these patterns.

2.4.1 Machine learning & deep learning based detection

Machine learning techniques have matured over the last couple of decades [Dietterich, 2002]. A comprehensive study by Delgado et al. [Fernndez-Delgado et al., 2014], which put to the test 179 different machine learning classifiers from 17 families (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalised linear models, nearest neighbours, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods) across 121 different datasets to compare different classifiers. The evaluation shows that tree-based algorithms such as random forest and decision tree C5.0 are among the best overall performing techniques. However, that study by Delgado et al. does not cover the evaluation of deep learning and its recent architecture techniques. Machine learning is common in intrusion detection systems for conventional computer systems [Sravani and Srinivasu, 2014] (but still not for vehicles or other mobile cyber-physical systems).

The application of machine learning, and in particular deep learning, in CPS security is a fairly new area of research. Machine learning is a branch of artificial intelligence where algorithms use datasets to identify patterns. Different from statistical learning, traditionally, machine learning is focusing on concrete algorithmic and mathematical problems rather than probability or statistics. The algorithm will then be able to identify patterns

in data that deviate from the norm. Machine learning performance relies on data-intensive areas such as search, social network analysis, recommendation algorithm, computer vision and voice recognition for both classification and regression problems. A machine learning algorithm can provide high accuracy when used in an intrusion detection system for traditional networked systems [Lane and Brodley, 1997, Hui and Cao, 2010, Hasan et al., 2014, Sravani and Srinivasu, 2014, Kotsiantis et al., 2007]. The objective is to reduce the number of false positives while increasing the number of true positives to identify cyber attacks when applied to robotic vehicles.

Junejo and Goh [Junejo and Goh, 2016] used behaviour-based machine learning which built on the NSL-KDD99 dataset. They used supervised machine learning for their IDS. The system learns the normal behaviour of the system using historical data. Their experiments were conducted on a CPS testbed and were based on the behaviour of the system to determine cyber attacks. They used nine machine learning classifiers for their experiments using data from ten different types of attacks. The results showed low FPs with high detection rates.

Machine learning algorithms, especially supervised technique ones, can fit well for intrusion detection in CPS systems because of their characteristics in high general accuracy and speed, and also high performance for correlated, unstructured and noise data from CPS systems and robotic vehicles particularly [Kotsiantis et al., 2007].

Over the past couple of years, the growing maturity in deep learning algorithms has led to wider use outside of its traditional applications in image and natural language processing. The deep learning technique refers to a branch of machine learning using artificial neural networks that compose many layers. In 2012, when [Hinton et al., 2012] beat the well-known datasets: ImageNet, MNIST, TIMIT, CIFAR and Reuters on speech, text and image classification, deep learning became ever more popular and influenced the research community. As the deep learning architecture allows a variety of architectures of neural networks, it provides general applicability in most areas, including cyber security.

In [Hardy et al., 2016], the authors applied a specific deep learning architecture called

stacked AutoEncoders for detecting malware. The input data were captured from Windows Application Programming Interface (API) from portable executable files. The author claimed this is the first work using the stacked AutoEncoders model with Windows API calls in malware detection which had promising experimental results. In [Dong et al., 2016], deep neural networks were utilised for detecting rogue certificates from trusted certificate authorities. The produced classification proved to work both in simulation and even with a limited number of collected rogue certificates in the case India Controller of Certifying Authorities (CCA) compromise incident.

Up to now, most deep learning based IDS proposals have addressed attacks against traditional computer networks, showing the superiority of the approach on the old Data Mining and Knowledge Discovery (KDD) Cup 99 [Kim et al., 2016] and the refined NSL-KDD [Javaid et al., 2016] datasets. Kim et al. constructed an IDS model with deep learning approach KDD Cup 99. The authors tested out the hyper-parameters including the learning rate (0.1, 0.01, 0.001 and 0.0001) and hidden layer size (between 10 and 90) and achieved better detection rates and accuracy rates than other classifier algorithms. Javaid et al applied a two-stage process of self-taught learning including unsupervised feature learning on unlabelled data and classification on labelled data. The deep learning approach was based on a sparse auto-encoder and a soft-max regression. The result of this technique is on a par with the best results in various other work on this NSL-KDD dataset.

An exception is a recent work by Kang et al. [Kang, 2016], which is geared towards the automotive industry and specifically vehicles that rely on the controller area network (CAN) bus. While a promising start, the particular work is limited to a generic command injection attack, which is detected by monitoring a single data source (the network packets on the CAN bus) and using a generic deep neural network architecture, which does not account for the temporal aspects or the overall state of the vehicle. Also, it has been evaluated only in simulation.

Here, it is important to note that technology used for detecting intrusion in resource-

constrained systems such as Internet of Things (IoT) devices may be partially applicable for CPSs. However, IoT systems such as smart home devices have very limited actuation capacities as the focus is on sensing much more than interactions with the physical environment [Blasch et al., 2017]. As the result, the impact of cyber attack against them typically have both limited severity and offer limited opportunity to detect based on it .

2.5 Conclusion

This literature review chapter has included the general challenges in the security of cyber-physical systems. It focused on the different types of cyber attacks on vehicles and the corresponding defence mechanisms that have already been proposed. It discussed the intrusion detection techniques for robots and vehicles, especially ones based on machine learning and deep learning algorithms. While conducting the literature review, it became clear that the range of related research that is applicable to robotic vehicles is still rather limited. This research carries out experimental evaluations based on a real testbed and mathematical modelling for predicting the effect of different network configurations. The thesis aims to provide efficient and accurate detection mechanisms based on physical and cyber indicators.

The next chapter discusses the experimental methodology, including the design of the testbed, the data that was collected to identify the ground truth as a starting point for the experiments, a discussion of the attack scenarios and the results.

Chapter 3

Experimental process

3.1 Introduction

This chapter presents the experimental methodology utilised for the development of detection mechanisms for vehicles. It includes a description of the robotic vehicle, which serves as the centrepiece of the testbed used, the attacks performed against the specific vehicle, as well as the data collected in relation to them. It also includes empirical results showing the impact of these attacks in physical space.

3.2 Testbed design

The testbed for the development and experimentation is a four-wheel-drive robotic vehicle (Figure 3.1) controlled via an onboard Intel Atom computer running the Linux operating system and an Arduino micro-controller for driving the motors. The vehicle also carries a camera with pan and tilt functionality for situational awareness and remote navigation. The vehicle and its camera can be controlled remotely via an Ethernet cable or Wi-Fi, by relaying commands over a Transmission Control Protocol (TCP) socket on the vehicles control board. Standard magnetic encoders fitted to the two rear wheel motors provide

information on the angular position of each wheel. The difference between two consecutive measurements is representative of its speed.

The detailed content of the testbed are presented in the following subsection, starting from the individual component (Section 3.2.1), and moving on to the remote operation (Section 3.2.2) and the vehicle setup (Section 3.2.3).

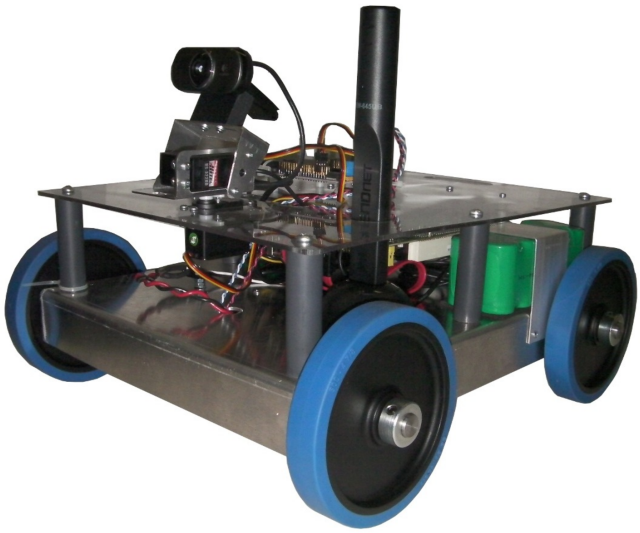


Figure 3.1: The vehicle used in the experiments

3.2.1 Testbed components

The robotic vehicle has three main component units including control, sensing and navigation, and actuation, as shown in Figure 3.2.

- Control. The vehicle uses a standard Intel Atom 1.8 GHz, which is representative of a large number of cyber-physical systems using commodity processing units rather than bespoke embedded systems. The control computer is connected to a simple micro-controller (ARC32) to relay commands to the vehicle's motor controller and

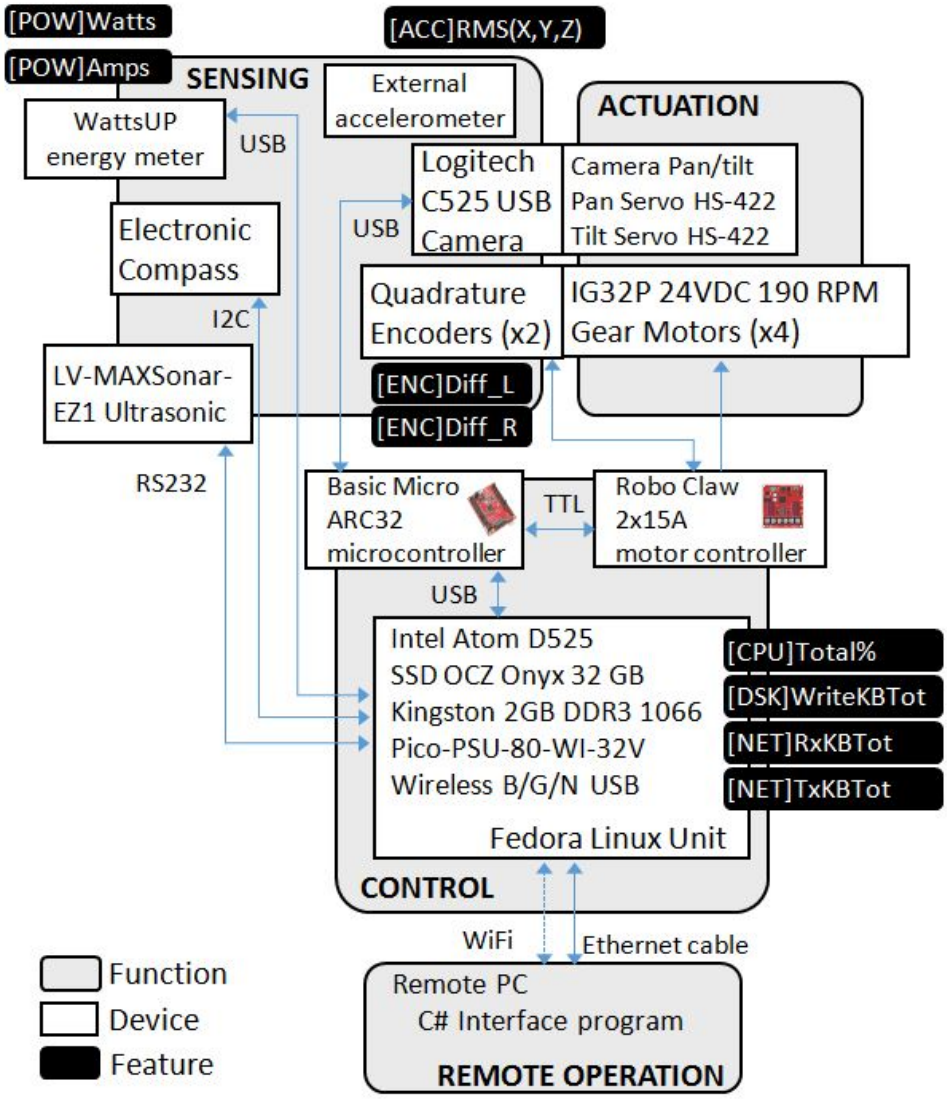


Figure 3.2: Detailed diagram of the testbed. The input features used for the detection are shown in black background

to collect data from sensors. In detail, the control unit consists of the following components:

- Intel BOXD525MW Intel Atom D525 1.8GHz (Dual Core)
- Solid State Drive (SSD) OCZ Onyx OCZSSD2-1ON32G 2.5" 32GB SATA II MLC. A standard hard-drive for storing the operating system, the detection code developed, as well as collected data
- Kingston ValueRAM 2GB 204-Pin DDR3 SO-DIMM DDR3 1066 (PC3 8500). A standard memory chipset.
- Wireless B/G/N USB Adapter. This allows remote connection via a standard

WiFi router.

  - Micro Controller - ARC32 - MCU-046-032. A micro-controller interfacing between the control computer and the sensing and actuation components.
  - RoboClaw 2x15A - TE-152-215. A motor controller dedicated for the motor control, as well as to collect the data from the motor encoders. It is connected to the ARC32 micro-controller via a Transistor-Transistor Logic (TTL) system.

- Sensing and navigation support. The vehicle includes a small set of sensing technologies, which is representative of what is typically used in the industry.
  - Logitech C525 USB Webcam. A USB Logitech webcam was attached to the vehicle for the common robotic task of real-time transmission of video, image and audio to its operators.
  - Electronic Compass Kit. It is connected to the ARC32 micro-controller via an inter-integrated circuit (I2C) for navigation based on a bearing.
  - LV-MaxSonar-EZ1 Ultrasonic Range Finder. Allows for collision avoidance and indoor path finding applications.
  - WattsUP energy meter. This is an externally connected device, which is used to monitor energy consumption during the experiments.
  - External accelerometer. This is an externally attached device, which was added to monitor levels of vibration on the chassis. This was deemed necessary only after the initial experiments which indicated this type of physical effect of some of the cyber attacks.

- Actuation. Naturally, the primary physical function of the vehicle is its mobility.
  - Camera pan and tilt system - standard - WC-003-200. The particular system was utilised only for facilitating navigation.
  - IG32P 24VDC 190 RPM Gear Motor with Encoder - TD-055-190 (one for front left and one for front right wheels). The two encoders measure angular speed through quadrature counting.

  - IG32P 24VDC 190 RPM Gear Motor - TD-015-190 (one for the rear left and one for rear right). The rear motors do not feature encoders.

- Power supply & Physical parts. The battery set to provide power for all components of the robot. Two 12V 4500 mAHr NiMH battery packs are wired in parallel to create a 24V power bus.
  - 80 Watt PicoPSU-80-WI-32V power supply unit
  - 12V 4500 mAHr NiMH battery pack - TE-099-210 (Qty 2)
  - Smart Charger for 9.6V - 18V NiMH and NiCad (Qty 2)
  - Electric power hookup kit
  - Aluminum base for 32mm motors
  - Custom 5" wheels and shafts

Figure 3.2 summarises the components of the robotic vehicle in terms of their architectural layout. It also shows the input features collected for the purpose of intrusion detection (Chapter 4), identifying also the components from where they are collected. Figure 3.4 shows a top side view of the vehicle. As shown, the vehicle is small enough for indoor experiments with limited space. The dimensions are 39.4 cm long, 35.6 cm wide and 30.5 cm inches high (with the compass mast). All together its weight is nearly 6.8 kg. The vehicle moves using four wheels, which gives 3.8 cm of ground clearance.

3.2.2 Remote operation

The testbed is a four-wheel-drive vehicle controlled via an onboard Intel Atom computer running the Linux operating system. The remote control of the vehicle can be via Ethernet cable or Wi-Fi, by relaying commands received over a TCP socket to the robotic vehicle control boards. In our experiments, the operator's machine and the vehicle have been on the same local area network. With the right ports opening on the vehicle to receive

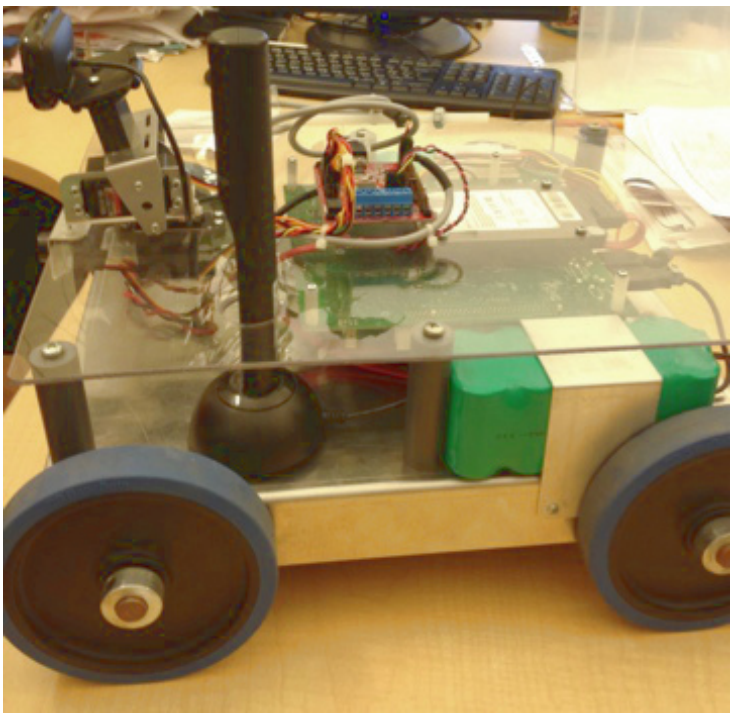


Figure 3.3: Top side view of the robotic vehicle

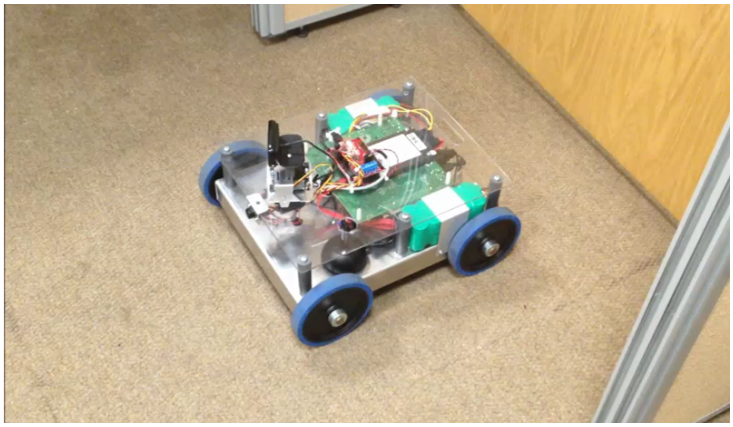


Figure 3.4: Top view of the robotic vehicle

incoming commands, the vehicle can be also controlled through the Internet as in Figure 3.5.

This operation is done through a C# Interface program. The graphical user interface (GUI) is depicted in Figure 3.6. This program allows the operator to set adjustments in the two actuation systems: the vehicle wheel and camera pan tilt. For changing the speed and direction of the vehicle movement, an operator can click on the control pad which would convert automatically to Right Motor and Left Motor value as in Figure 3.6. These values range between 0 and 127. At the value of 64, there is no wheel rotation,

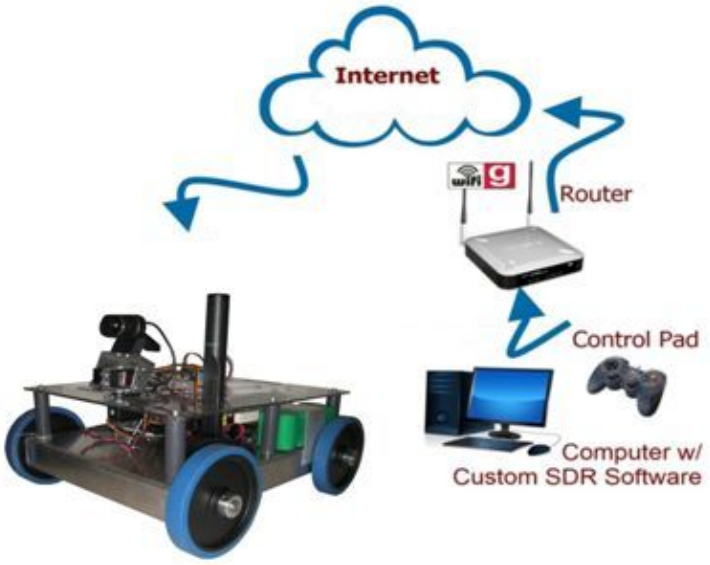


Figure 3.5: Remote control of the robotic vehicle.

while 127 and 0 are the fastest forward and backwards rotation. The different speed of left and right wheels will determine the direction of the vehicle movement (just like in a tank movement). In Figure 3.6, the left motor angular speed is set at 82, while the right one is set at 72. Values 82 and 72 are arbitrarily chosen and they are fixed for repeatability. Angular speeds 82 and 72 are both greater than 64, and 82 is greater than 72 so the robotic vehicle would go forward to the right slightly. For changing the direction of the webcam, an operator of the robotic vehicle can control the pan tilt system to face up, down or left and right around on an x and y-axis. The video stream from the webcam can provide the real-time view point from the robot.

The server program is written in C++ running on Linux Fedora OS. The program runs the Fedora demon to interact with the Interface Program on the operator's computer as well as the micro-controller ARC32, which controls the movement of pan tilt, the compass and the motor controller RoboClaw. The motor controller drives the vehicle motors with their encoders through electrical speed control. The encoders are also responsible for detecting the revolution/angular speed of left and right motors, which allows the movement direction and speed to be detected. A C++ method was written and embedded in this server program to provide real-time angular speed for the robotic vehicle left and right motors.

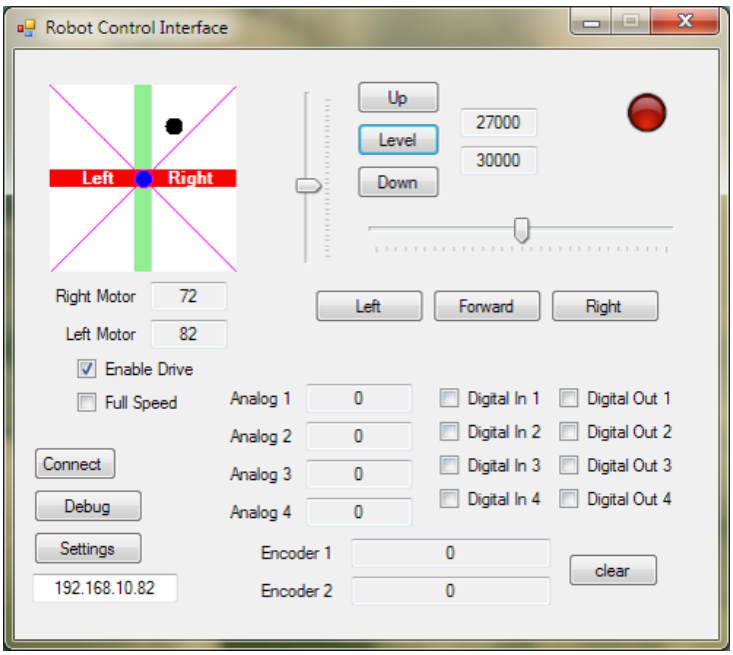


Figure 3.6: The GUI of the control interface on the operator's computer

For repeatability and reproducibility purposes, the human element was eliminated by automating the process of controlling the vehicle with a set of predefined scenarios. Each scenario includes a series of operation commands triggered at specific times. These automated actions are programmed using the JitBit macro software [Jitbit, 2016] to stimulate operators' keyboard and mouse instructions. The automation is especially useful when there is a need to have the cyber attack event in the training scenario, because the training needs to be repeated several times. The ground truth (labels used in the learning phase) is defined when the cyber attack starts and ends from the settings. The automation can define the precise time-stamp for the attack start and end points. Different commands in Jitbit such as SWITCH TO WINDOW, DELAY, and MOUSE Click, as depicted in 3.7, enable repeatability and reproducibility of the experimental scenarios.

3.2.3 Robotic vehicle setup

In order to better observe the physical effects caused by different cyber attacks and to eliminate any side factors which could potentially interfere with this goal, we have made some modifications to our testbed, as follows:

Figure 3.7: The GUI of the Jitbit automation tool

Friction: While the vehicle moves on the floor, the variable friction between the floor surface and the wheels can interfere with speed and steering control. To avoid this, the vehicle was positioned on a stand, effectively lifting the vehicle from the floor and providing consistent friction to all four wheels. This setup also overcame the space limitations of the research lab and of conducting experiments involving long movement paths, as seen in NASA Mars Rovers' set-up [NASA, 2010].

Network Interface: While launching a cyber attack against the robotic vehicle, it is possible that parameters affecting the wireless communication between the controller and the vehicle (such as signal strength) interfere with the functionality of the robot. Since it was important to focus on the physical effects of the attack, a wired Ethernet connection for communication was also used for reliable signal strength.

Power Supply: Under normal operation conditions, the vehicle is powered by a 24V battery pack. After running lengthy experiments, it was observed that the battery depletion rate was significantly affecting the motor's power, which had a direct impact on the measurements regarding the vehicle's speed. To remove this factor from affecting the results, the battery pack was replaced by a desktop DC power supply. This also removed the battery life limitation when conducting lengthy experiments. For example, some of the machine learning training sessions reported in Chapter 5, have taken over 50 hours, which is considerably longer than the vehicle's battery life would allow.

3.3 Data collection

When running experiments, the aim is to capture accurate and relevant data regarding the robotic vehicle's behaviour during normal operation and under cyber attacks. The data variables should reflect the characteristics of the cyber and physical performance of the vehicle which includes its physical mobility, energy consumption and other central processing unit (CPU), network and disk usage related to operating system activities.

The data variables were collected from three different devices including the Robot Linux operating system, the Robot micro-controller and a Samsung Android phone and a Wattsup device. Each source of data can have different timing and may require different techniques for monitoring and capturing measurement. Table 3.1 shows that traditional cyber data can be captured from the robot's operating system performance stats, while physical data such as wheel speed, vibration and energy consumption are drawn from devices such as the onboard micro-controller, and an external smart-phone accelerometer, as well as an external power measurement device.

Table 3.1: Cyber (C) and physical (P) data sources

Source	Description	Type (C/P)
Operating System	Network usage	C
	CPU usage	C
	Disk usage	C
Micro-controller	Wheel speed	P
Smart Phone	Vehicle vibration	P
Watts-up device	Energy consumption	P

3.3.1 Cyber data: network, CPU and disk usage

The values of the cyber features were written to text files using Collectl [Collectl, 2015], which is an open source tool for CPU, network and disk data collection. Collectl is an appropriate solution because it runs on all major Linux distributions and depends on Perl, which is typically installed by default in Linux. This tool was chosen due to the ability to capture a wide variety of system performance stats at once compared to other tools [Linux

performance tools, 2014]. At the same time, its light-weight characteristics fit well in a resource-limited device. Figure 3.8 is a snapshot of a default sample output which shows network, CPU and disk measurements in real-time. Each line in the figure represents one second of sampling data. In this option, the utility shows four measurements for each category: CPU, disks and network. For CPU usage, it shows the percentage of time that the CPU was busy, (CPU utilisation), percentage of time the CPU was executing in system mode, the number of interrupts/sec, the number of context switches/sec. The disk utilisation shows KB read/sec, number of reads/sec, KB written/sec, and number of writes/sec. Network traffic shows KB received/sec, packets received/sec, KB sent/sec, packets sent/sec.

```
waiting for 1 second sample...
#<--------CPU--------><----------Disks----------><----------Network---------->
#cpu sys inter  ctxsw KBRead  Reads KBWrit Writes   KBIn  PktIn  KBOut  PktOut
   7   0   157   1256      0      0      0      0      0      0      0       0
   7   0   151   1223      0      0      0      0      0      0      0       0
   5   0   124    835      0      0      0      0      0      0      0       0
   6   0   144   1228      0      0      0      0      0      0      0       0
   6   0   136   1194      0      0      0      0      0      0      0       0
   5   0   136    777      0      0     12      2      0      0      0       0
   7   1   164   1228      0      0      0      0      0      0      0       0
   7   0   145   1251      0      0      0      0      0      0      0       0
   5   0   130    783      0      0      0      0      0      0      0       0
   6   0   144   1215      0      0      0      0      0      0      0       0
   5   0   129   1004      0      0      0      0      0      0      0       0
   4   0   115    809      0      0      0      0      0      0      0       0
   7   0   147   1081      0      0      0      0      0      0      0       0
   5   0   121    937      0      0      0      0      0      0      0       0
   5   0   235    830      0      0   3688    198      0      0      0       0
   7   0   161   1310      0      0      0      0      0      0      0       0
   5   0   117    865      0      0      0      0      0      0      0       0
   7   0   137   1105      0      0      0      0      0      0      0       0
   7   0   147   1224      0      0      0      0      0      0      0       0
   5   0   115    788      0      0     40      3      0      0      0       0
   7   0   165   1416      0      0      4      1      0      0      0       0
```

Figure 3.8: collectl sample data

3.3.2 Physical data: wheel speed

Encoder values were collected from monitoring programme embedded with the robotic vehicle control unit. The programme was written in C++ to receive analogue values from micro-controller ARC32 responding the position of the wheel during operation. The C++ code can be found in the appendix of this thesis.

The data received from the micro controller is a stream of values in bytes (0..255). Each

packet contains 21 values as listed in Table 3.2. In this array of values, there are two important numbers from the encoders. Bytes 4-7 and 8-11 contain values of right and left encoder rotation numbers accordingly. These values represent the current number of wheel rotations that the encoders have counted. Together with the time-stamp in the monitoring script, the angular speed could be calculated by dividing the change in the number of rotations per executed time period. The default interval is 30ms, or about 33 times in 1 second.

Table 3.2: The structure of transmit packet from the micro-controller, which includes the speed in the form of encoder values

Bytes	**Description**
0-3	Reserved values
4-7	Analog values
8-9	Compass value
10	Digital value
11-14	Right Encoder value
15-18	Left Encoder value
19-20	Check sum value

3.3.3 Physical data: robotic vehicle vibration

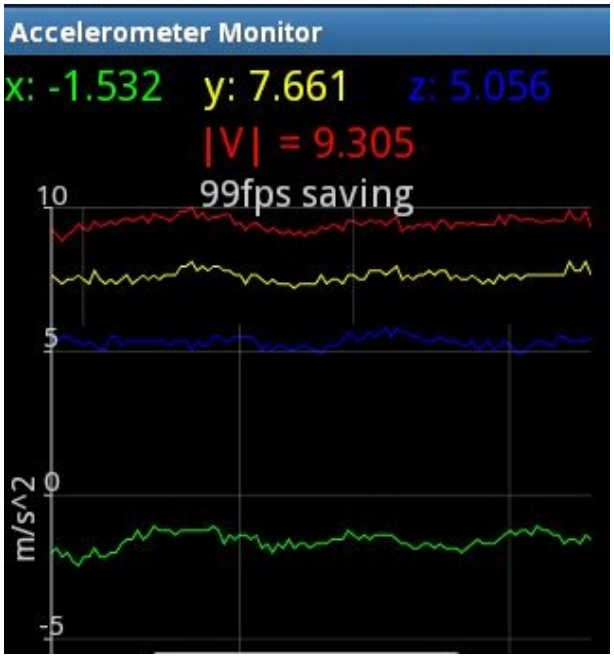


Figure 3.9: The GUI of Accelerometer Monitor tool used for collecting vehicle vibration

Through empirical observation, it has become apparent that vibration of the vehicle chassis is possible during a cyber attack. For this reason, external accelerometers have been added to measure this vibration.

Acceleration is measured on three axes (x, y, z) for representing the vector of different directions. For example, a positive value in x might mean left acceleration while a negative is for right. Similarly, y and z values can be for forward or backward, and up or down direction. Here, by measuring acceleration shows the vibration of the vehicle chassis real-time responding to a cyber attack or not.

A smart-phone equipped with an accelerometer was used in the experiment. The phone, a Samsung Galaxy S3, was securely attached to the body of the robot. To capture the acceleration values (x, y, z), an open source software Accelerometer Monitor, as depicted in 3.9 was installed on the phone. The phone accelerometer now can save the recorded x, y, z into a text file in real-time. The data values are captured as in Figure 3.10.

```
# X value, Y value, Z value, time diff in ms
0.067 0.172 9.136 20
0.076 0.162 9.155 20
0.047 0.162 9.241 21
0.019 0.153 9.203 20
0.028 0.172 9.251 20
0.047 0.172 9.232 20
0.028 0.181 9.155 21
0.019 0.143 9.165 19
0.028 0.114 9.136 20
0.0 0.105 9.136 20
0.019 0.095 9.136 20
0.019 0.124 9.145 21
0.028 0.134 9.212 20
0.047 0.162 9.174 20
0.057 0.153 9.184 20
0.028 0.134 9.212 20
0.038 0.153 9.184 20
0.009 0.162 9.184 20
0.028 0.191 9.184 20
0.038 0.162 9.184 20
0.038 0.172 9.174 20
0.038 0.162 9.136 20
```

Figure 3.10: Acceleration values (x, y, z) and its collection interval

3.3.4 Physical data: energy consumption

Watts and amps were collected externally using the WattsUp energy meter device, which can measure the energy consumption of the robotic vehicle. In order to capture the data, the device was also connected to a Linux computer through a USB cable to record live stream values of watts and amps. A Python program was written to read values from the meter, which were saved in /dev/tty.usbserial. The output data were then extracted and written to a comma separated values (CSV) file for later processing. The interval for meter reading is set at 1 second. Figure 3.11 shows an example from the CSV file that includes a variety of values regarding the energy meter from the robotic vehicle during the experiments.

```
ttyUSB1, 2016-06-20 13:50:33.927117, 62.5, 221.0, 0.394
ttyUSB1, 2016-06-20 13:50:34.684762, 62.6, 220.9, 0.394
ttyUSB1, 2016-06-20 13:50:35.687272, 62.7, 220.9, 0.396
ttyUSB1, 2016-06-20 13:50:36.695685, 62.8, 220.9, 0.393
ttyUSB1, 2016-06-20 13:50:37.686044, 62.9, 221.1, 0.393
ttyUSB1, 2016-06-20 13:50:38.683753, 62.9, 221.0, 0.387
ttyUSB1, 2016-06-20 13:50:39.687233, 63.0, 221.0, 0.395
ttyUSB1, 2016-06-20 13:50:40.692304, 63.0, 221.0, 0.393
ttyUSB1, 2016-06-20 13:50:41.681673, 63.1, 221.0, 0.393
ttyUSB1, 2016-06-20 13:50:42.689015, 63.0, 221.0, 0.398
ttyUSB1, 2016-06-20 13:50:43.688289, 63.2, 221.2, 0.398
ttyUSB1, 2016-06-20 13:50:44.679815, 63.3, 221.3, 0.401
ttyUSB1, 2016-06-20 13:50:45.685767, 63.3, 221.2, 0.402
ttyUSB1, 2016-06-20 13:50:46.684110, 63.4, 221.3, 0.399
```

Figure 3.11: Energy data captured through wattsup including source, time, watts, voltage, amps

3.3.5 Ground truth

Here, the term ground truth is used to refer to the information regarding the experiments that are there by design, such as the time an attack actually starts and actually finishes, as opposed to the inference made by IDS design. Specifically, the ground truth refers to the flag of attack-on and attack-off that is designed for each cyber attack scenario. This is saved along with other cyber and physical data, which are later used for not only

training phase in machine learning and deep learning algorithm, but also for testing and validation of each algorithm. Here, the ground truth is considered to be 1 for the period that the robotic vehicle in under attack and 0 for the period with normal operation (no cyber attack).

3.3.6 Data pre-processing

The data for the cyber and physical input features are captured from different locations within the architecture (Figure 3.2) and at different points in time due to the lack of perfect synchronisation and because of different collection periods (intervals).

The encoder value is collected by monitoring scripts embedded within the robotic vehicle control unit, while Watts and Amps are measured with the WattsUp device [Watts up?, 2016]. Plus with a ground truth (attack flag) value coming from attacking machine, it is important to have the time-stamps set consistently to the same source. From observation during the experiment, the time setting of each system can be different. As a result, the clocks amongst the different devices are to be synchronised to a standard time server [Rinaldi et al., 2016], and the data needs to be processed with the awareness of the issue.

Also, linear interpolation is used to address the fact that different devices would collect data at different time intervals, as shown in Table 3.3. Each sensor/device collected data values at a different frequency. Within each sensor/device, the frequencies of data collection were not always precise for the same configuration setting. For example, if the frequency was set at 20ms/event, the data could have been collected after 19ms or 21ms of the previous event. Hence, we had to apply a linear interpolation technique to establish a consistent sampling rate of the data values from physical and cyber data [Gurulian et al., 2016].

Table 3.3: Data sources and configured collection rate

Source	Collected data	Configured collection rate
Operating system	Network, CPU, Disk usage	1.0 s
Micro-controller	Wheel speed	30 ms
Smart phone	Robotic vehicle vibration	20 ms
WattsUp device	Energy consumption	1.0s
Attacker's machine	Ground truth (attack flag)	Per event

3.4 Attack scenarios

A comprehensive review of cyber threats related to communication, sensing, information management and vehicular technologies used in emergency management is given in [Loukas et al., 2013a], while a taxonomy providing a global view of the respective attack types and defense mechanisms is presented in [Loukas et al., 2013b].

Recent research [Koscher et al., 2010] has demonstrated that cyber-physical attacks can target production automobiles, since these vehicles incorporate various sensing and computing modules that can interact with each other in multiple ways. Initial attacks infected the vehicle's electronic systems through the use of an audio file in the MP3 player device or through a smart-phone connected via Bluetooth. One of the possible results of this attack is a change in the driving direction while the vehicle is in motion.

As representative of a wide-range of possible attacks against a robotic vehicle, experiments have been conducted where the vehicle was under a denial of service (DoS) attack, a command injection attack and two kinds of malware attack, one targeting the CPU and the other targeting the network (NET), as shown in Table 3.4, which also summarises the primary physical impact observed during each attack. The attacks are intermittent. They appear in-between sections of normal operation. Under normal operations, without a cyber attack, the network communication and applications were running legitimately. They correspond to the camera feed transmitted to the operator, as well as the operator's legitimate commands to the robot.

Table 3.4: Attack intention

ID#	Type	Intent
1	DoS	Resource unavailable
2	Command Injection	Disrupting control
3	Malware (NET)	Disrupting communication
4	Malware (CPU)	Disrupting computation

3.4.1 Denial of Service attack (DoS)

According to Loukas and Oke, a Denial of Service (DoS) attack is "any intended attempt to prevent legitimate users from reaching a specific network resource" [Loukas and Öke, 2009]. In the context of our robotic vehicle, a denial of service attack is an attack that aims to prevent the operators from reaching the robotic vehicle and vice versa, so that it cannot be controlled remotely and it cannot transmit sensor or other data to its operators.

In the experiments, the attack is carried out by sending large volumes of TCP traffic to the robot's network interface. For this, the Low Orbit Ion Cannon (LOIC) was used, which is an open-source and multi-threaded denial of service attack tool, able to flood the robot's network interface with different types of traffic. This particular tool has become well-known since it was used by the hacktivist group "Anonymous" in "Operation Payback", in retaliation to the opponents of Internet piracy in 2010 [Sauter, 2013]. Figure 3.12 shows different parameters set for launching an attack from LOIC against the robotic vehicle such as:

- IP Address: 192.168.10.82
- Port: 7000
- Method: TCP
- Threads: 100

Once the flooding attack rate is set, the tool produces illegitimate TCP segments, which increase the bitrate received by the vehicle to values above 60 Mbps (See Table 3.5 for the

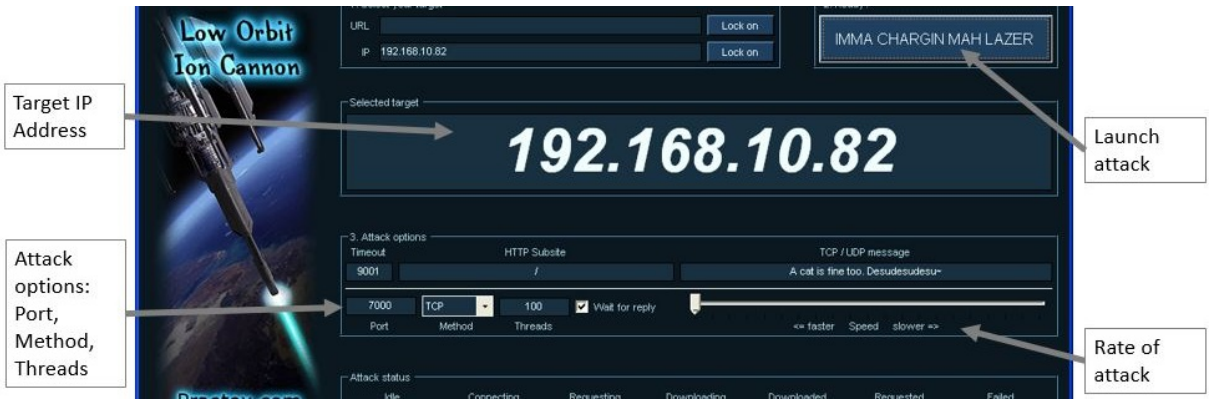


Figure 3.12: LOIC tool used for DoS attacks

average incoming and outgoing network traffic). As the capacity of the Ethernet interface used on the robotic vehicle is 100 Mbps, when the average link utilisation exceeds 60% (incoming rate 60 Mbps) and moves towards 70%, it starts causing connectivity problems. This is in line with the usual rule of thumb in the networking industry, which is that when the average utilisation gets close to 70%, the instantaneous utilisation reaches often 100% and produces high packet loss and delays. In these experiments, the average utilisation during a LOIC attack reached 62.86 Mbps for the incoming traffic. This was sufficient to cause intermittent connectivity issues.

Table 3.5: Average network traffic measured at robotic vehicle interface (Mbps)

	Attack-off	Attack-on
Outgoing	0.007	4.77
Incoming	0.005	62.86

Figures 3.13 and 3.14 illustrate the incoming and outgoing traffic rate at the robot's network interface under normal operation and under an ongoing DoS attack.

As denial of service attacks rely on high bit-rates of attack traffic, they involve a noticeable period of time where they are rapidly ramping up the attack rate. This effect is obviously noticeable on the incoming network traffic and processor use, as in conventional cyber attacks, but on cyber-physical targets such as this robotic vehicle, it also has an effect on the speed, jitter and response delay of the physical movement. Here, it was observed that the intermittent connectivity that is caused by the denial of service attack, leads to a situation where the vehicle is forced to trigger temporarily and very frequently its

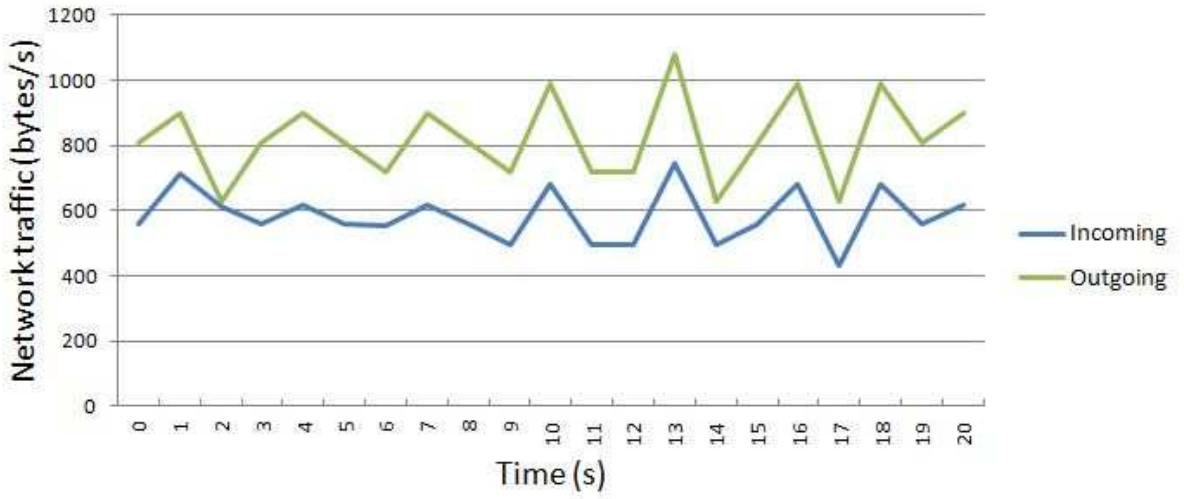


Figure 3.13: Robotic vehicle network interface traffic under normal operation

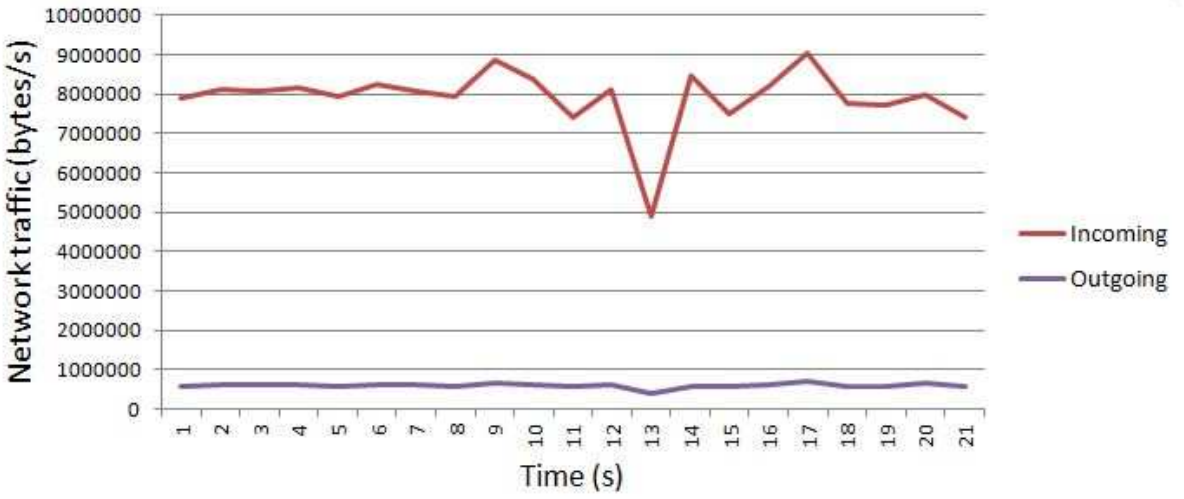


Figure 3.14: Robotic vehicle network interface traffic under DoS attack

fail-safe mechanism, which is simply to stop and it then resumes its movement. So, using physical indicators of the movement of a vehicle can potentially help recognise the impact of a DoS attack.

Below, two simple scenarios are discussed where the physical impact of a DoS attack was observed when the vehicle's speed is constant and when it is variable (e.g. when the vehicle accelerates/decelerates or turns).

3.4.1.1 Scenario 1: Constant Vehicle Speed

The first scenario involves the vehicle moving at a constant speed. The robot's speed is set by a remote software application and its value can range from 0 to 127. For this scenario a speed value of 94 is chosen arbitrarily, which results in an angular speed depicted in Figure 3.15 where the vehicle operates without the presence of a DoS attack. This graph is the result of sampling the wheel encoders with a period of 30ms, as mentioned in Section 3.2.

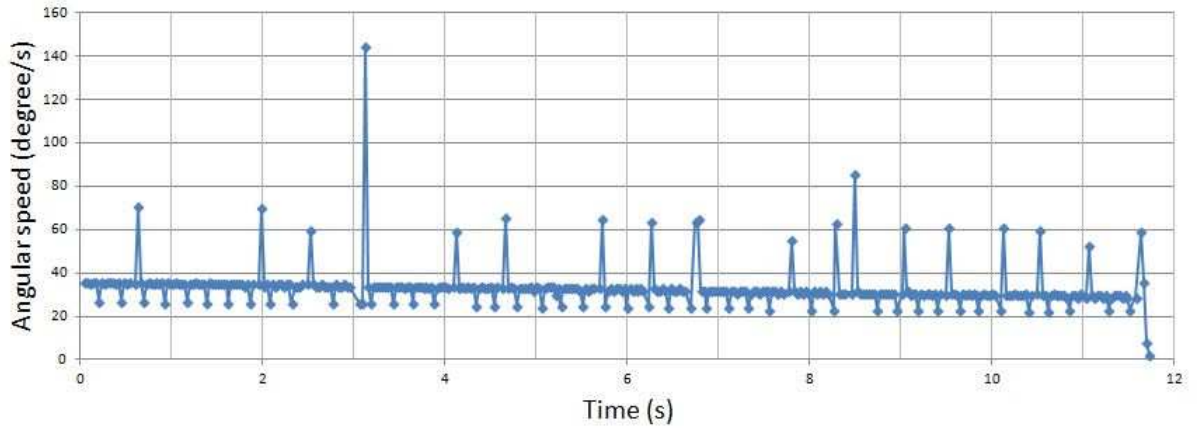


Figure 3.15: Scenario 1: Angular speed vs. time under normal operation

The next stage of this scenario involves the vehicle moving with the same selected speed while sustaining a DoS attack. The vehicle jitters during the DoS attack due to it coming to a full stop for a short period caused by the attack and then moving again. The result of this setting is depicted in Figure 3.16. The effect of the DoS attack impacts the vehicle at times 21.7s, 24.8s and 27.4s. The short period of 0 values for the encoder Figure 3.16 represent the halting of the vehicle. As the attack progresses the period that the robotic vehicle is affected increased and this is discussed further in section 3.4.1.3 Physical Impact. Note that the values reported by the magnetic encoders, as depicted in Figure 3.16, suffer from spikes in the angular speed figures. These are related to the data collection rate being too high rather than actual physical changes in speed.

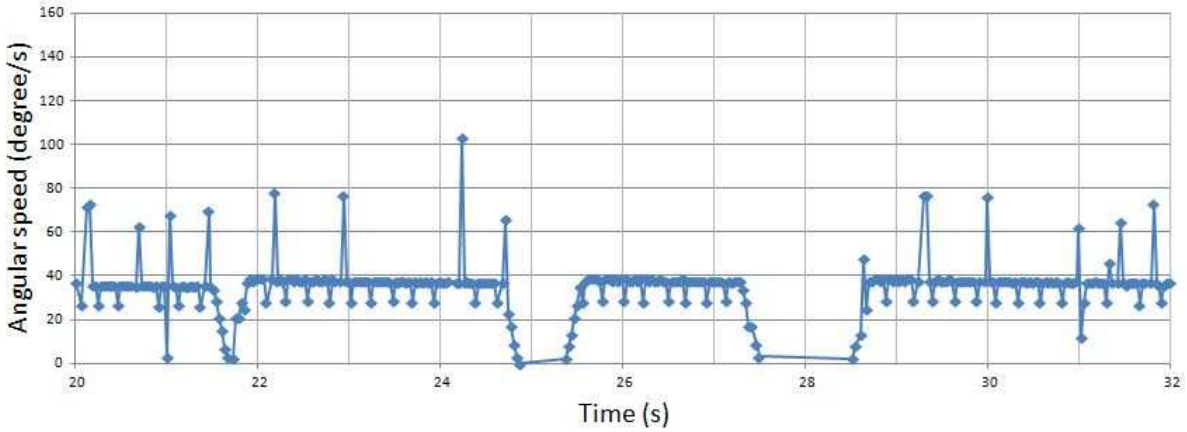


Figure 3.16: Scenario 1: Angular speed vs. time under DoS attack

Table 3.6: Variable speed timing

Speed	Speed Value	Duration(ms)	Start	End
Low	81	6000	0	6000
High	94	6000	6000	12000
Low	81	6000	12000	18000

3.4.1.2 Scenario 2: Variable Vehicle Speed

The second scenario involves the vehicle changing between two speeds (slow - high). The aim is to evaluate the effect of the DoS attack on the responsiveness of the vehicle with respect to navigation commands. For this, we arbitrarily chose a slow value (here, 81) and a high value (here, 94) that makes a noticeable difference in the vehicle's speed. The timing of the speed changes, illustrated in Table 3.6, is achieved by an automated script running on the remote controller computer and communicated to the robotic vehicle over a wired network.

There is a small delay of 0.4s for the vehicle in executing this scenario under normal operation, as shown in Figure 3.17. With the attack ongoing, the vehicle receives strong impacts at six different times 3s, 16s, 18.2s, 20.9s, 22.8s and 24.7s. The angular speed reaches 0 or near 0 values. Figure 3.18 represent the halting of the vehicle, as well as a significantly greater time in execution. This is the result of operation commands being queued due to interruption of the attack. The DoS caused the robot to delay operation from time to time, hence the total time is much larger during the on-going attack. Due

to the jitter and the delayed commands, the robot also moved a longer angular distance than under normal operation, which also costs more in terms of energy.

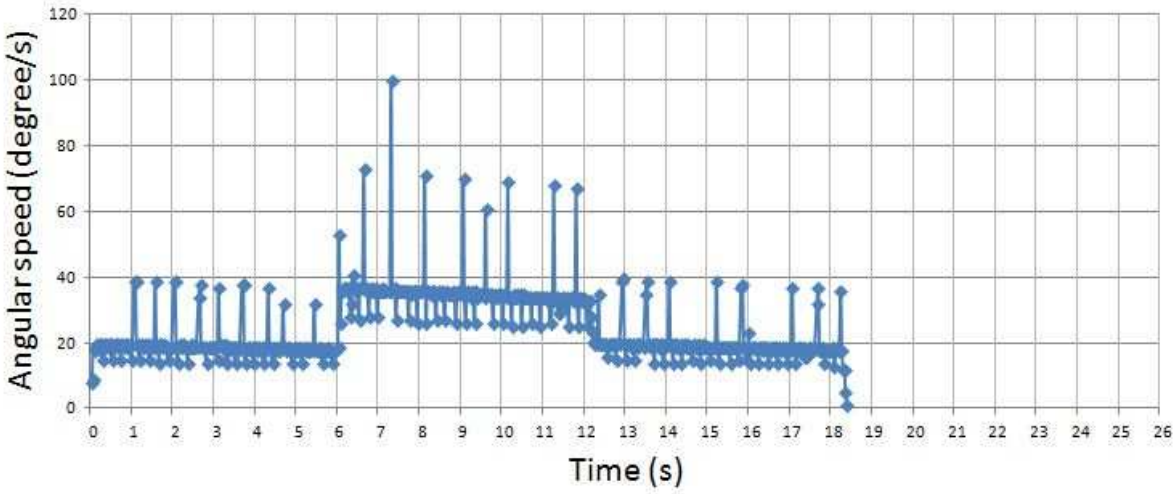


Figure 3.17: Scenario 2: Speed change response under normal operation

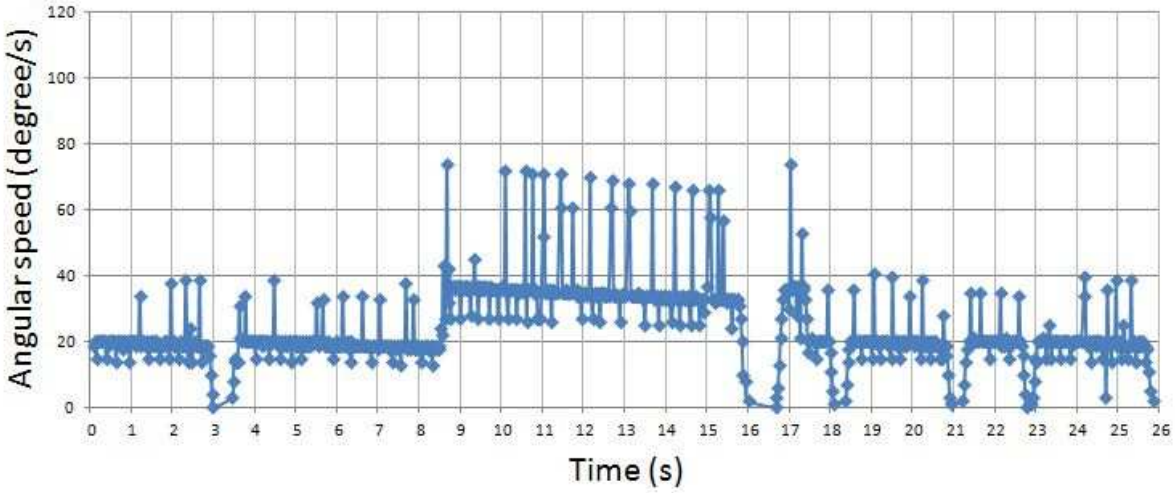


Figure 3.18: Scenario 2: Speed change response under DoS attack

3.4.1.3 Physical Impact

Figures 3.17 and 3.18 depict the angular speed of the vehicle's wheels versus time, without and with DoS attack respectively. The results clearly indicate that launching a cyber attack against a robotic vehicle is accompanied by physical effects which impair the vehicle's movement. More specifically, two significant physical indicators related to a cyber attack have been identified.

Vehicle Halting: The first physical indicator was identified by inspecting the experimental results and this is the robot's movement pattern. Figures 3.16 and 3.18 clearly show that when the vehicle is under a DoS attack, its movement becomes erratic. More specifically, in Figure 3.16 it was observed that the vehicle halted four times. Moreover, each of the halts has a different duration. A similar behaviour is shown in Figure 3.18, where the robot's speed varies throughout the course of the experiment. It was observed that the vehicle halted six times in this case.

Delay in Responding to Navigation Commands: A second significant physical feature that emerges from the vehicle's behaviour when it is under a DoS attack, is depicted in Figure 3.18. It can be clearly seen that there is a delay of 2.5s in transitioning from the low to the high speed setting. Moreover, the overall duration of the robot's movement is prolonged by 7s.

3.4.2 Command injection attack

The command injection attack aims to execute a command with malicious intent [Loukas, 2015]. The primary target of this attack is to interfere with the device that controls an actuator directly or indirectly. Command injection attacks can occur after an adversary gains access to the control system either by hacking or through the lack of authentication capability. The adversary must also understand the valid syntax to inject false commands.

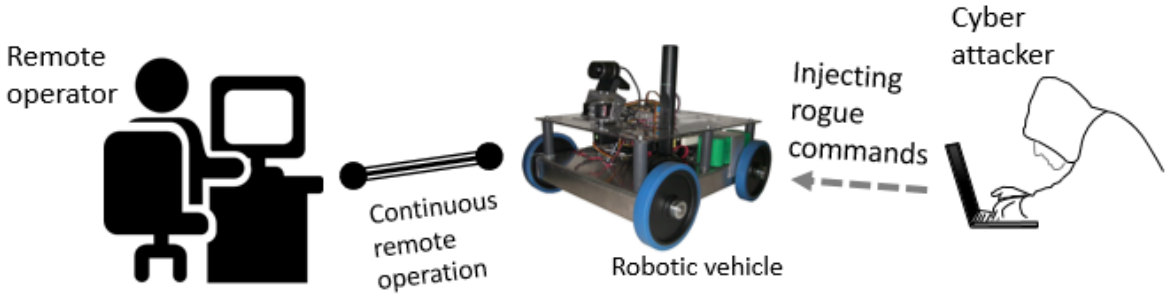


Figure 3.19: Command injection attack: cyber attacker sends rogue commands to the robotic vehicle during operation

With the testbed, the vehicle receives commands from its remote operator to move or

to turn. While receiving the continuous legitimate commands, another machine was set up to send rogue commands. The vehicle is then receiving contradicting commands from two different sources and trying to execute both of these incoming commands. While the remote controller might want the vehicle to move forward, the rogue command could be 'stop'. While the vehicle is meant to turn right in its mission, the false command could be 'turn left', sent by an attacker. The false command can also over-write legitimate ones.

The testing scenario was in two parts. During the first part, the vehicle was operating under normal conditions and then in the second part while under attack from a malicious machine. The robot's position can be decided either by its speed value, which ranges from the configured values 0 to 41, or the change in the position in one interval depicted in Figure 3.20. This graph is the result of the left wheel encoder value with an interval period of 30ms. The first half from 0s to 10.96s presents the normal operation while the second half presents the effect of command injection attack. The attack was achieved by injecting 'turn left' and 'stop' commands to the robotic vehicle control system.

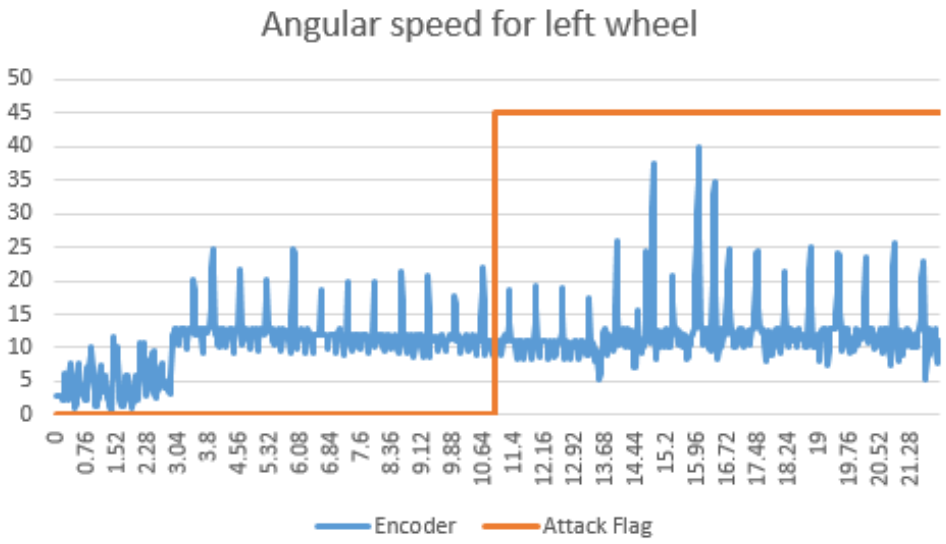


Figure 3.20: Angular speed change with and without command injection attack over time (s)

During the cyber attack, a physical impact was observed on the robot. The wheels show frequent consistent jitter in movement due to the conflicting operating commands. This physical indicator would help with detection of this type of cyber attack.

3.4.3 Malware attack

As the nature of malware has many types and forms, this work focuses on the following two types of general malware scripts. One is disruptive to the network communication by causing delays on the network and the other is destructive to the CPU which was performing heavy calculating tasks.

Malware can affect physical components such as Stuxnet on Programmable Logic Controllers. Stuxnet changes the value very slightly as there is no obvious physical impact. This is similar to this malware attack against the CPU as there was no change in the observable behaviour of the robotic vehicle. Whereas the other malware attack against the network, caused a clear physical impact, manifested by frequent and consistent stops.

3.4.3.1 Malware attack against network (Malware NET)

The malware is simulated by a Linux Shell script, including a number of commands to interfere with the network scheduler. The malware disrupts the network communication and causes a delay in the network. It utilises the Linux kernel's network scheduler to modify the network traffic control setting. As a result, the robot's movement becomes erratic with frequent stops during the attacks.

3.4.3.2 Malware attack against CPU (Malware CPU)

Similarly, another Linux Shell script was designed to disrupt the processing power of the vehicle. This piece of malware consumes processing through a resource-demanding calculation with the use of a continuous loop. Unlike with other attacks, there is no obvious change in the robot's physical behaviour observed.

3.5 Features

The focus here is on the types of data that can be extracted by a mobile cyber-physical system without a considerable overhead. Also, it is necessary to choose a small set of features that can represent the behaviours of both cyber and physical components of the robotic vehicles for quicker data processing. To make the selection of the features, the criterion was the relative important and the choice was made empirically through experiments, as well as through the literature review. Eight input features have been identified; four related to communication and processing, which are referred to as the cyber input features, and four related to the physical properties of the robot, which are referred to as the physical input features. The attack label is the ground truth for the scenario, which corresponds to whether an attack really is present or not at a specific point in time.

- **Network Incoming:** Received network traffic rates. This is one of the most common features used in network intrusion detection and especially for Denial of Service attacks [Loukas and Oke, 2007]. In the vast majority of such attacks (excluding the obscure category of low-rate ones [Kuzmanovic and Knightly, 2003]), the incoming traffic rate is expected to increase rapidly. In distributed attacks, there is also a ramp-up behaviour of this rate, as more and more sources start sending traffic to their target.

- **Network Outgoing:** Transmitted network traffic rates, which is also a key indicator to represent network performance during normal operation and cyber attacks. Together with Network Incoming rate, this feature is a good indicator of network bandwidth consumption which is helpful for indicating cyber attacks, especially for detection at the edge node of a network [Siaterlis and Maglaris, 2005].

- **CPU:** The total CPU utilisation. [Yampolskiy et al., 2012] assessed the applicability of Data Flow Diagram (DFD) techniques for CPS cyber attacks, which effects bandwidth utilisation and CPU consumption.

- **Disk Data:** The rate of data being written to the disk. [Paul, 2008] presented a malware detection method that monitors disk-level behaviours. The study claimed that this detection approach is resilient to many traditional obfuscation techniques, because it is based on the behaviour of the disk activities.

- **Encoder:** Represents the wheel speed. [Yampolskiy et al., 2013] provided a taxonomy of different cyber attacks, targets and effect on CPS. For attacks on UAV, it has been shown that the movement of the system will be altered. This effect is a key indicator for describing and categorising different attack types.

- **Accelerometer:** Represents the vibration of the chassis (using accelerometer measurements). Similar to wheel speed change, [Yampolskiy et al., 2013] identifies abnormal physical vibration as a malicious impact on the CPS victim. Classification of these insubstantial changes would link to known attacks on the CPS, which helps with detection.

- **Power:** Corresponds to power consumption. These measurements are more critical with Mobile CPS with limited power supply. With CPS smart grid system, [Marnerides et al., 2015] has created profiling power consumption measurements as part of an enhanced anomaly detection system for the CPS.

- **Current:** Corresponds to current. Similar to power consumption, these important values also reflect different physical impact of the CPS. With Mobile CPS, the movement of the system has a strong effect on the energy consumption, hence the current. These measurements can be helpful for indicating cyber attacks on the system.

- **Attack label:** This is the ground truth label, which states whether there is an attack or not at a particular point in time. This is used to train the intrusion detection system and also to evaluate its performance.

Table 3.7: Cyber (C) and physical (P) features and their source function

Feature name	**Description and Type (C/P)**		**Source Function**
Network Incoming	Network receive (KB)	C	Control
Network Outgoing	Network transmit (KB)	C	Control
CPU	Total CPU usage (%)	C	Control
Disk Data	Disk Write Data (KB)	C	Control
Encoder	Change in Left Encoder	P	Sensing
Accelerometer	Vibration of chassis	P	Sensing
Power	Power consumption (W)	P	Sensing
Current	Electric Current (A)	P	Sensing
Label	Attack Flag (1,0)		Ground truth

3.6 Contrasting attack impacts on the features

Figure 3.21 shows the effect of the denial of service attack on some of the features monitored on the vehicle. Naturally, for a denial of service attack based on volumes of network traffic, it is Network Incoming (in the figure, referred to as *RxKBTot*) that is the feature that is most obviously affected during an attack, but even physical features (e.g., RMS value) seem to be affected by the accompanied vibration.

In terms of the command injection attack, the conflicting commands cause consistent and frequent physical jittering, as the vehicle attempts to process and act on both commands within very small periods of time and continuously. This effect can be observed in Figure 3.22, especially in relation to the instantaneous speed value for each wheel, as represented by the encoder value (in the figure, referred to as *diff_encoder_l*), as well as by the very high RMS values, which are the result of the consistent physical jittering.

A piece of malware disrupts the network communication by causing a delay in the network. It utilises the Linux kernels network scheduler to modify the network traffic control setting. Figure 3.23 illustrates the effect of the attack on some of the features monitored on the vehicle, as captured experimentally.

The fact that cyber attacks on a cyber-physical system, such as a vehicle, can have physical manifestation is both a hazard and an opportunity. It is a hazard because they can cause physical damage, but it is also an opportunity because it can potentially enable

Figure 3.21: Denial of Service attack scenario.

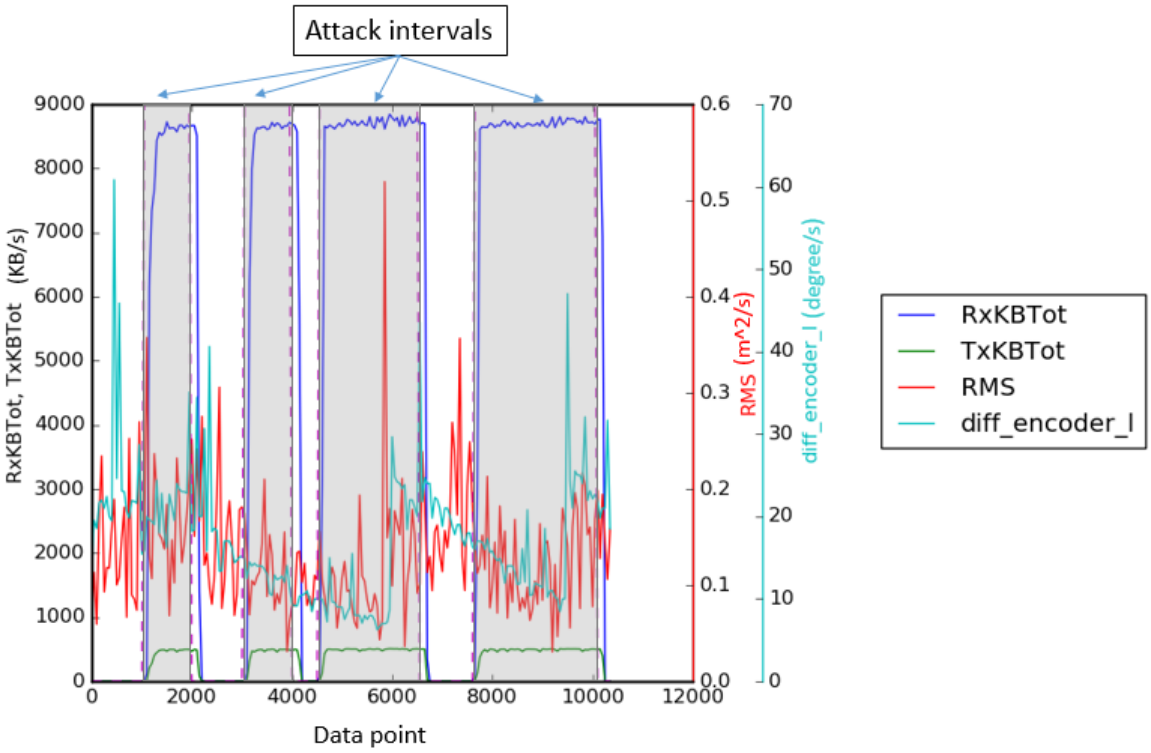


Figure 3.22: Command injection attack scenario.

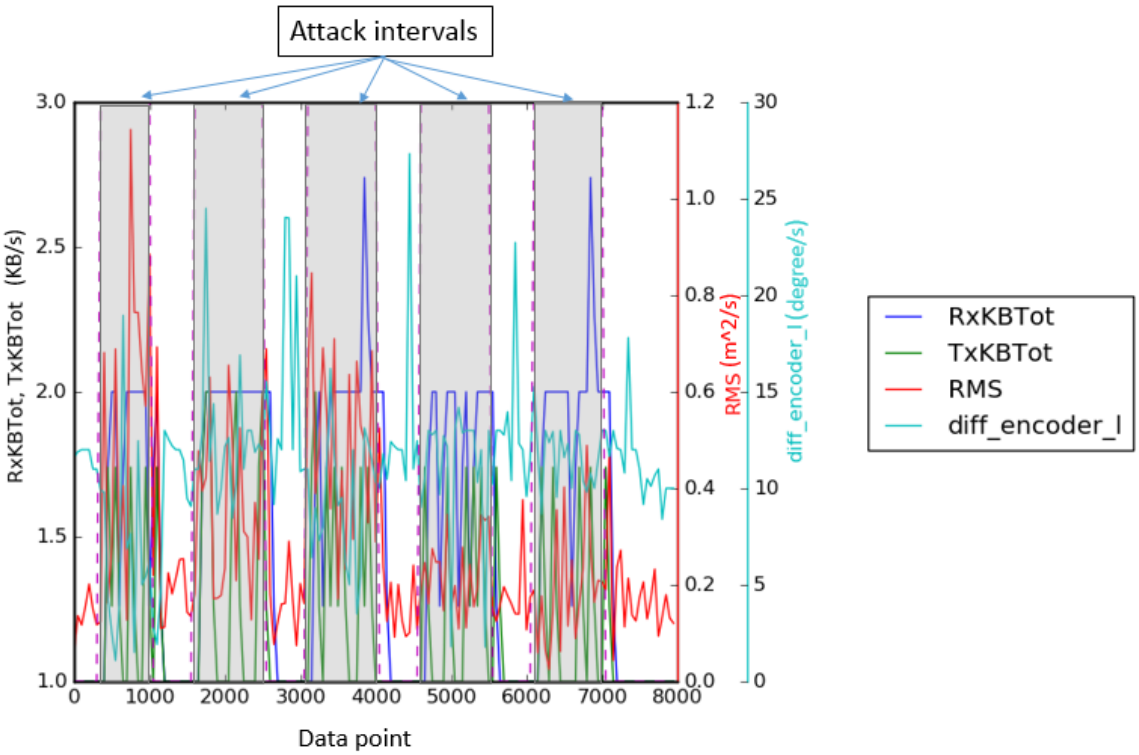
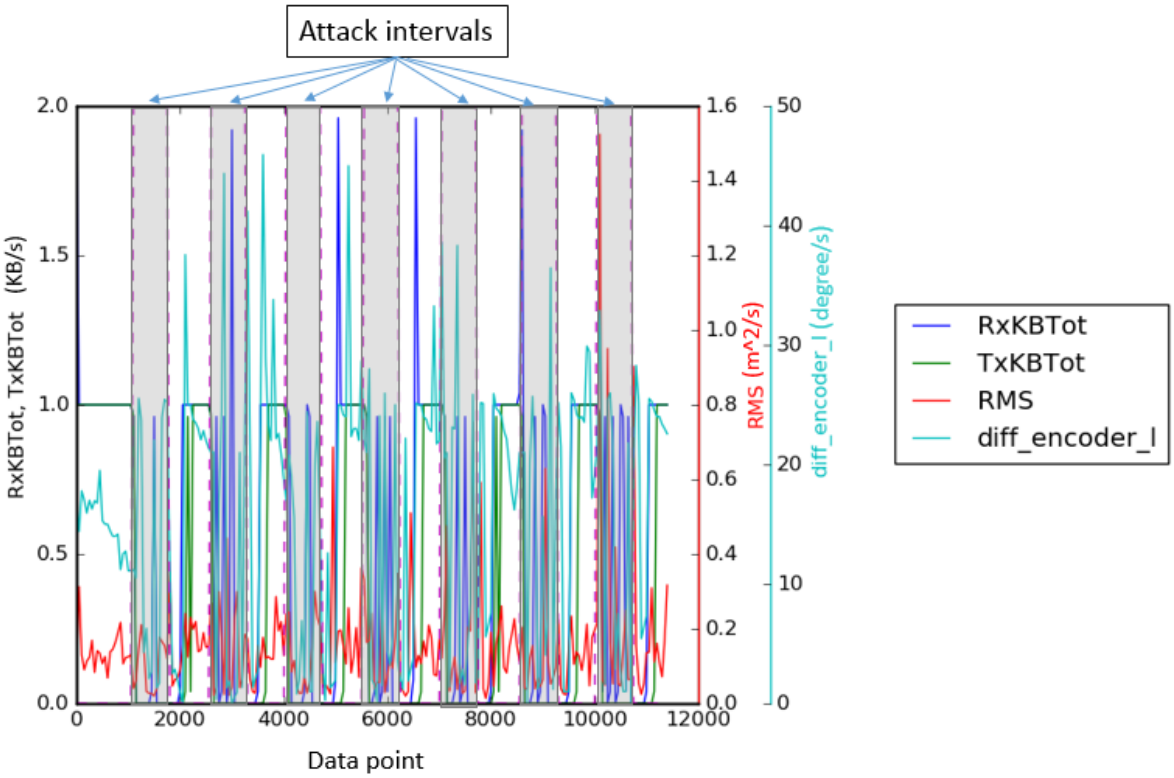


Figure 3.23: Malware attack scenario.



or contribute to detection.

After successfully capturing encoder values from a subroutine between the micro-controller and control server computer, the data file was analysed for a pattern to detect the physical impact of the attack. A monitoring process was developed on the robotic vehicle that measured the encode value difference. The monitor process output these values to a physical log file. The process raised a "Robot stopped" flag in the log file when the difference reaches 0 which means the robotic vehicle comes to a complete stop. Another flag "Robot is starting" was to alert when the vehicle changed from stopping state back to moving. These flags indicate the abnormal time length during each stops, hence they serve as Robot Halting physical indicators as discussed in the section 3.4.1.3 Physical Impact.

For an initial investigation of the hypothesis that physical features can be valuable in the context of detection, Snort [Roesch et al., 1999], which is a very widely adopted IDS technology, has been employed. Snort can collect packets from the network to search for

those that match a pre-specified pattern based on pre-defined rulesets.

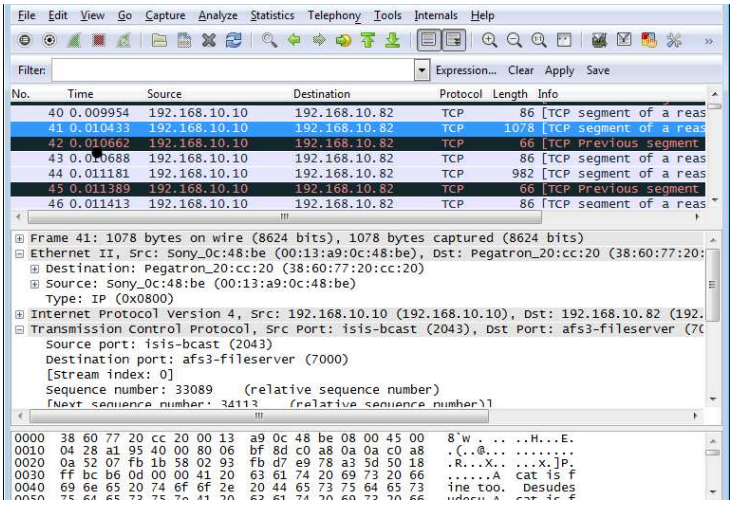


Figure 3.24: Wireshark packet analysis of DoS attack

Snort was set up to capture all network packets in these attack scenarios. Wireshark [Combs, 2007] was then used as a tool to analyse the similarity in the packet characteristics that belongs to certain attacks as in Figure 3.24.

For the particular simple attack, it was found that these packets shared the same values for source and destination IP address and ports, TCP flags PSH and ACK, as well as some common data in the content. Then, a simple set of Snort rules was developed to look for those similarities with a threshold that specified the maximum number of packets/s before Snort started to create a new alert. As this was a simple attack, it was easily detected and alerts were generated.

Then, we combined these results based on Snort and the sole use of traditional cyber (network) features with a flag showing when and whether the vehicle had experienced an unplanned stop, as shown in Figure 3.25.

Interestingly, for the first very short bursts of the attack (within the 4-9 s period), Snort did not pick up the attack because it had not yet matched its rules. Also, in the next attack sessions (from 13 s and on), the attack is indeed detected by Snort but with a delay in comparison to when the first physical stop was observed by the vehicle. This was a simple scenario and a simple attack, but it did demonstrate the possibility that taking

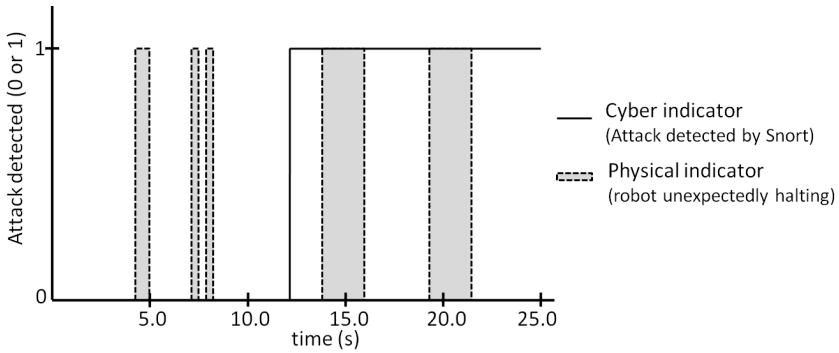


Figure 3.25: Physical and cyber flag combination

into account physical features can help in a process that traditionally has relied solely on cyber features. A more detailed investigation of this concept is presented in chapters 4 and 5.

Chapter 4

Detection with Decision Trees

4.1 Introduction

The goal here is (i) to provide a light-weight intrusion detection mechanism that can detect a cyber attack against a robotic vehicle using both cyber and physical input features and (ii) to compare the effectiveness of the intrusion detection against four different cyber attack types: DoS, Command injection, Malware against Network and Malware against CPU based on a performance metric including detection latency. As a lightweight approach, a decision tree learning algorithm has been used for automatically producing detection rules that will be used by the robotic vehicle.

A high-level overview of the system is illustrated in Figure 4.1. The framework design of the intrusion detection mechanism aims to run on the actual robotic vehicle based on its own monitoring processes and components without relying on any external system. We have opted for a rule-based approach, because it is light-weight at run-time. For the generation of the rules, a decision tree machine learning algorithm has been used for its speed and simple construction. Machine learning is common in intrusion detection systems for conventional computer systems [Sravani and Srinivasu, 2014] (but still not for vehicles or other mobile cyber-physical systems). Especially the C4.5 decision tree

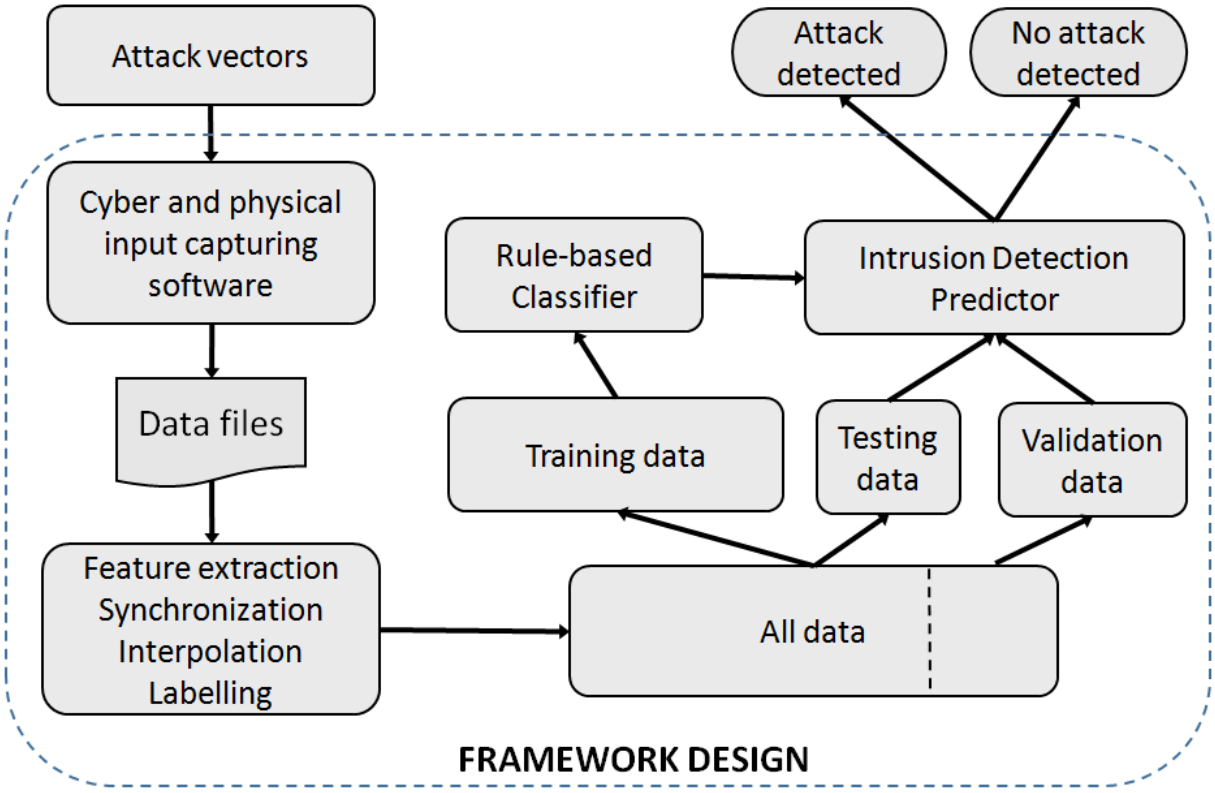


Figure 4.1: Intrusion detection framework

algorithm has been used extensively to detect denial of service and other attacks [Kim et al., 2014]. Here, the improved C5.0 algorithm [Patel and Rana, 2014, Noh et al., 2003, Filippoupolitis et al., 2014] is used, with live data collected from the robotic vehicle. Before applying the C5.0 detection mechanism, the data collection and preparation phase is discussed (Section 4.1.1).

4.1.1 Data preparation

As representative of a wide range of possible attacks against a robotic vehicle, experiments have been conducted where the vehicle is under a denial of service (DoS) attack, command injection (C.I.) attack, and two kinds of malware attack, one targeting the CPU and the other targeting the network (NET), as shown in Table 4.1, which also summarises the primary physical impact observed during each one. These attacks are intermittent between periods of normal operation.

Table 4.1: Experimental scenarios

S#	**Type**	**Impact observed**
S1	DoS	Inconsistent stops
S2	Command Injection	Frequent consistent jittering
S3	Malware (NET)	Frequent consistent stops
S4	Malware (CPU)	No clear physical effect
S5	Normal operation	No impact
S6	Combined DoS+C.I.	Mixed impact

In normal operation (S5), all network traffic and applications running are legitimate. They correspond to the camera feed transmitted to the operator, as well as the operator's legitimate commands to the robot.

S6 is a more complex scenario, where both denials of service and command injection are combined into the same timeline, sequentially, with normal operation and under different attack types and rates, while it is periodically moving and stopping, as shown in periods p1 and p4 (Table 4.2). For this part, we limit the experimentation to setting the vehicle to move in a straight line and at a constant speed and to halt repeatedly. The attack is a simple denial of service attack at a bit rate of approximately 8.7 MBit/s originating from an attacking machine. For a stronger DoS attack, a second PC was used to direct further illegitimate network traffic to the robot. The aim of the attack is to flood the robot's network interface with TCP traffic. Under this attack the vehicle receives commands to move forward from its legitimate operator, as well as stop commands from an attacker, as in period p2, or turn left commands from an attacker, as in period p3. This scenario is introduced to show a number of different realistic conditions that the vehicle might endure. It also helps to explore further whether the addition of physical input features can improve its effectiveness.

Table 4.2: Combined scenario (S6) with different attack types and time periods

Period	p1	p2	p3	p4	p5
Description	DoS	Data Injection "STOP"	Data Injection "LEFT"	Stronger DoS from two PCs	No attack. Normal traffic
Duration	307 s	173 s	79 s	29 s	221 s

The data for the cyber and physical input features are captured from different locations within the architecture (Figure 3.2) and at different points in time due to the lack of perfect synchronisation and different sample collection periods (T) (see Table 4.3). For example, the encoder value is collected by monitoring scripts embedded within the robotic vehicle control unit, while Watts and Amps are measured with the WattsUp device [Watts up?, 2016]. As a result, the data needs to be processed to address the synchronisation difference between the clocks of these different data collection devices, as previously discussed. Also, linear interpolation is used to address the fact that different devices would collect data at different time intervals. Figures 4.2, 4.3, 4.4, 4.5 and 4.6 show representative runs for each of the scenarios S1, S2, S3, S4 and S6 using the data after clock synchronisation and interpolation in R. We set the "ground truth" label to true when there is an attack and false when there is no attack. In total, the six scenarios present 92,669 data points for each feature.

Table 4.3: Cyber (C) and physical (P) features and their collection period

Feature name	**Description and Type (C/P)**		**Period (T)**
RxKBTot	Network receive (KB)	C	1.0 s
TxKBTot	Network transmit (KB)	C	1.0 s
CPU	Total CPU usage (%)	C	1.0 s
WriteKBTot	Disk Write Data (KB)	C	1.0 s
DiffEncoderL	Change in Left Encoder	P	30 ms
RMS	Vibration of chassis	P	20 ms
Watts	Power consumption (W)	P	1.0 s
Amps	Electric Current (A)	P	1.0 s
Label	Attack Flag (1,0)		1.0 s

4.1.2 Training, testing and validation data for each attack

As mentioned already, S5 corresponds to the normal operation of the robot. Each attack (S1-S4, S6) includes both the legitimate activities in the normal operation of S5 and the illegitimate activity introduced by the particular attack. For the purpose of training, testing and validation, we split the data (Figure 4.1) into two sets with sample size of:

- 80% for training and testing, of which the training data is chosen randomly from

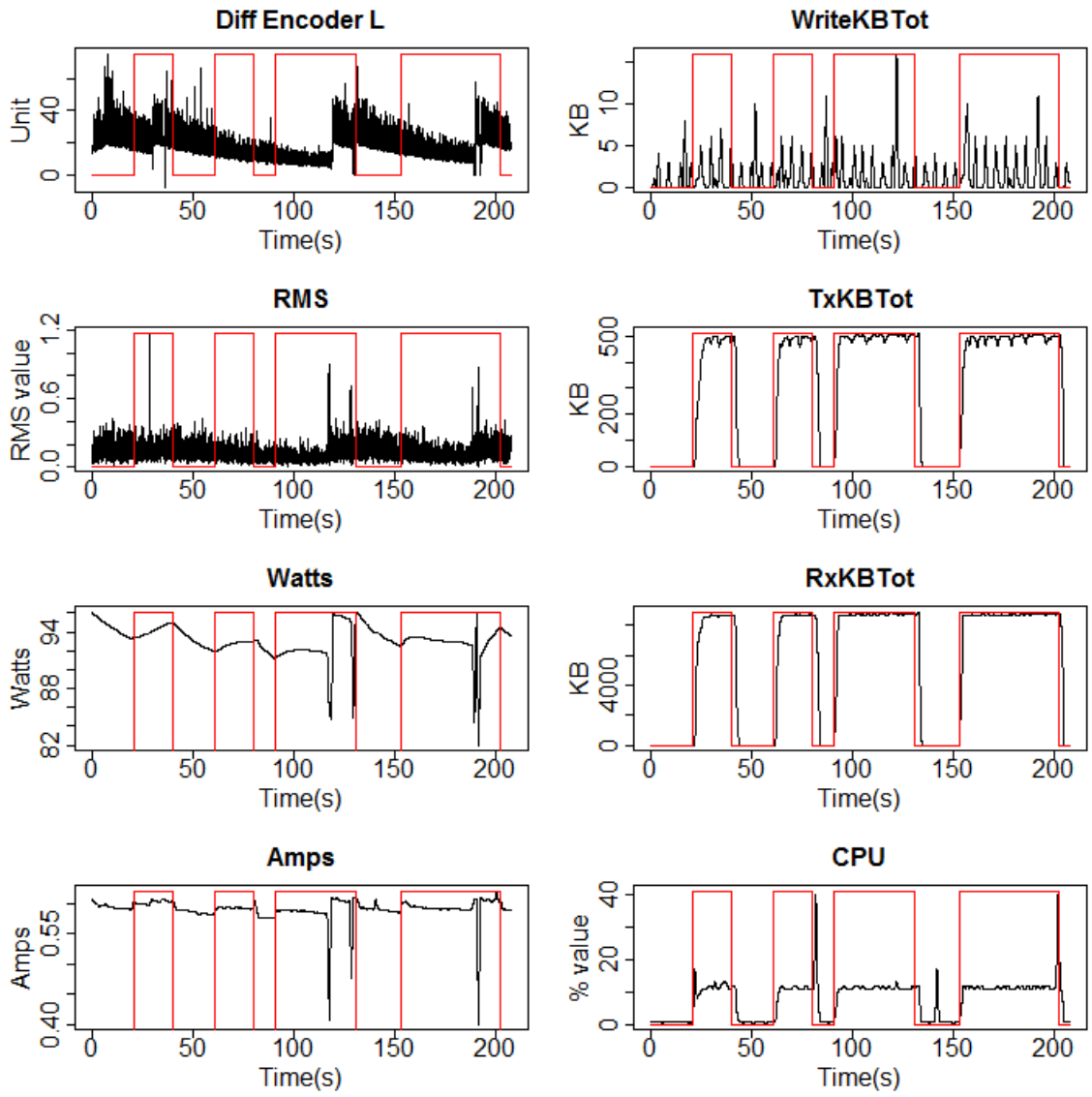


Figure 4.2: The data for cyber and physical features collected during the denial of service attack (S1). The overlaid frames denote the periods of time that the denial of service attack is on.

the 70% of this division, and testing is chosen from the remaining 30%.

- 20% for validation. This is holdout data that is not seen by the training models.

  In line with the first goal, we allow our learning algorithm to use all or subsets of the available input features to emphasise on the important of physical features:

  - Set 1: All eight cyber and physical features
  - Set 2: The four cyber features only
  - Set 3: The four physical features only

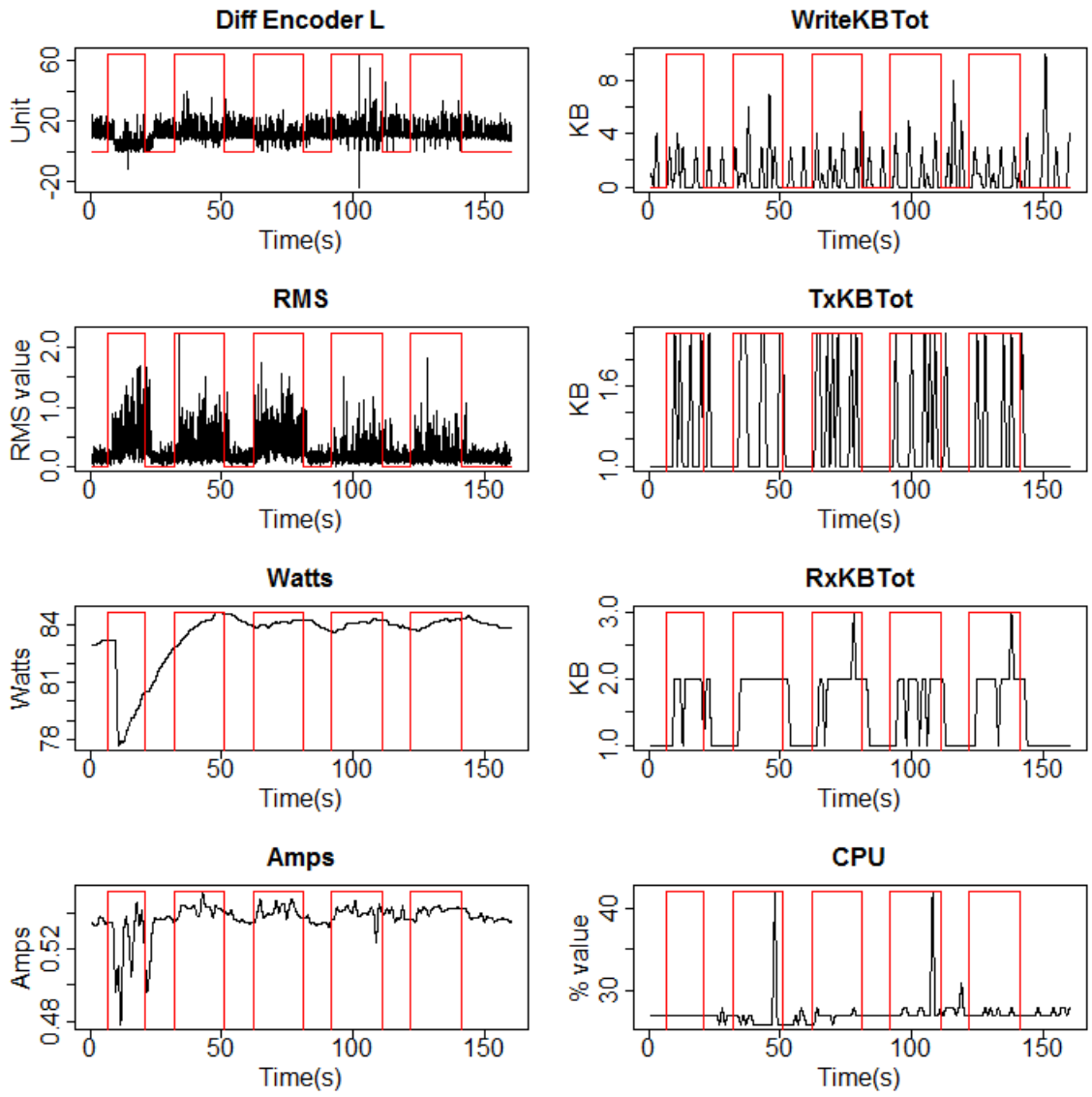


Figure 4.3: The data for cyber and physical features collected during the command injection attack (S2). The overlaid frames denote the periods of time that the command injection attack is on.

4.1.3 Detection method

As mentioned, we have identified the need for a robust classification method for behavioural characteristics of a robotic vehicle under attack using both physical and cyber input features. Towards this goal, the investigation was started using a lightweight knowledge-based approach. As an example of such an approach, a decision tree-based algorithm was used, as in [Filippoupolitis et al., 2014]. Decision tree machine learning is a common method for classifying data with high speed, strong learning ability and simple

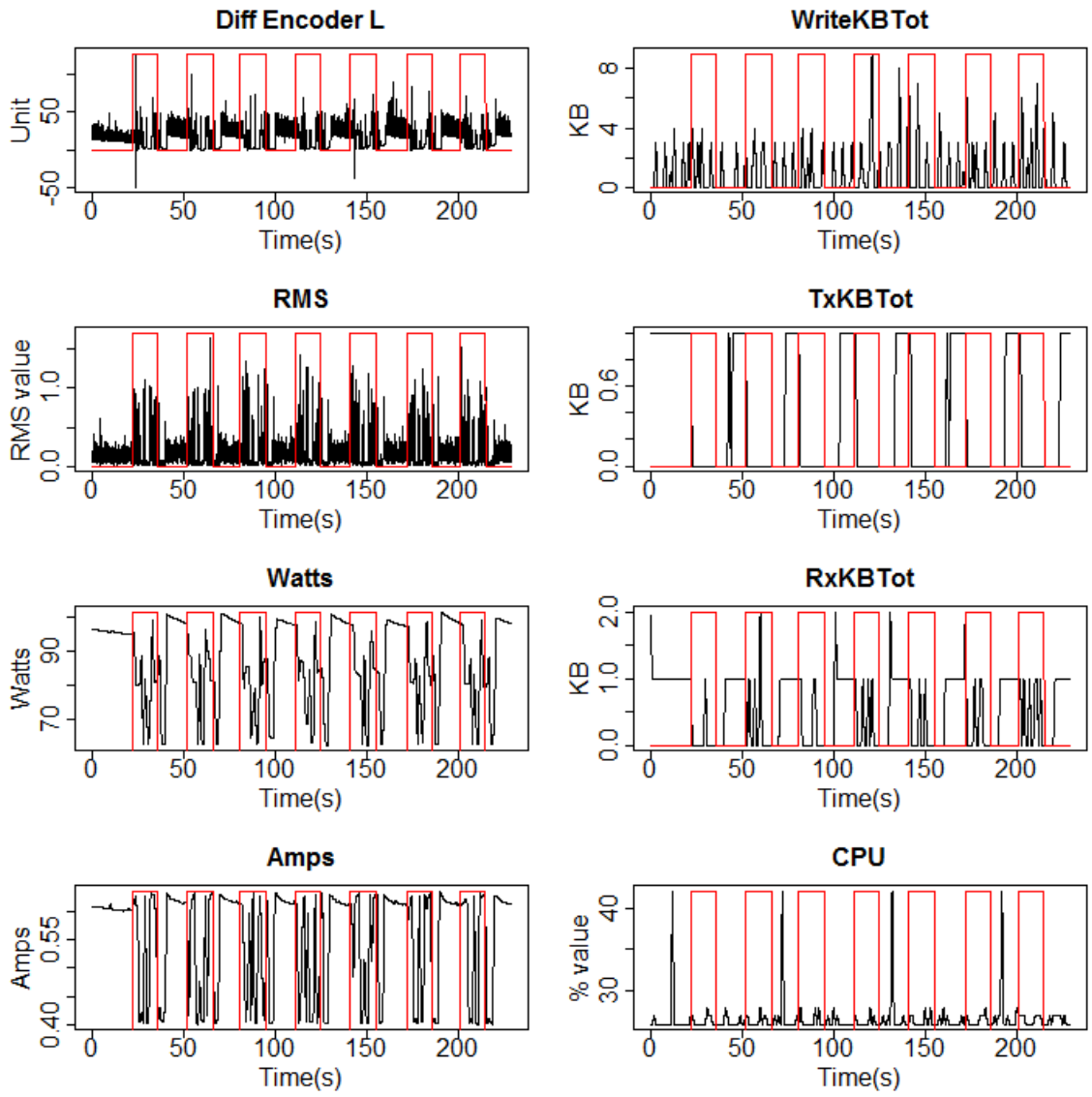


Figure 4.4: The data for cyber and physical features collected during the malware attack against network scenario (S3). The overlaid frames denote the periods of time that the network malware is active.

construction [Patel and Rana, 2014]. The decision tree C5.0 has been selected to define the rule set for detecting the physical impact as well as the cyber attack on the robotic vehicle testbed.

The decision tree C5.0 package [Kuhn et al., 2014] in R was used to generate the rule-based classifier. Each decision tree model is fit by Quinlan's C5.0 algorithm for each of the different training data as mentioned in Section 4.1.2. So, it creates one model (i.e. one ruleset) for each attack type. The decision tree C5.0 was chosen as the updated version of C4.5, which has been improved in terms of speed, memory and efficiency.

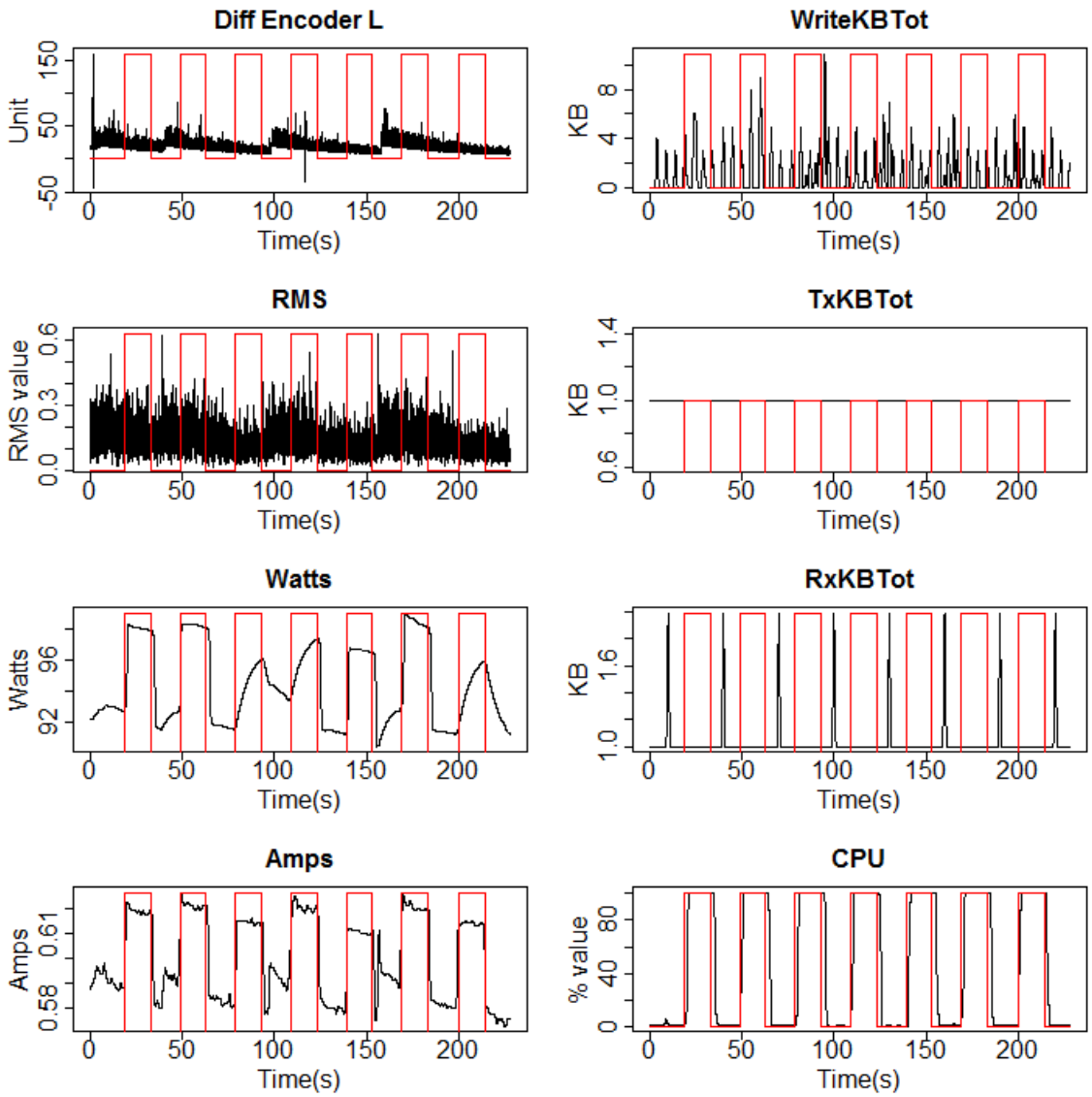


Figure 4.5: The data for cyber and physical features collected during the malware attack against CPU scenario (S4). The overlaid frames denote the periods of time that the CPU malware is active.

Then the intrusion detection classifier of each attack is evaluated against the testing data and the validation data separately. This evaluation is in terms of its ability to correctly recognise the existence or absence of an attack at each point in time. A sample section of the decision tree rules generated is shown in Figure 4.7.

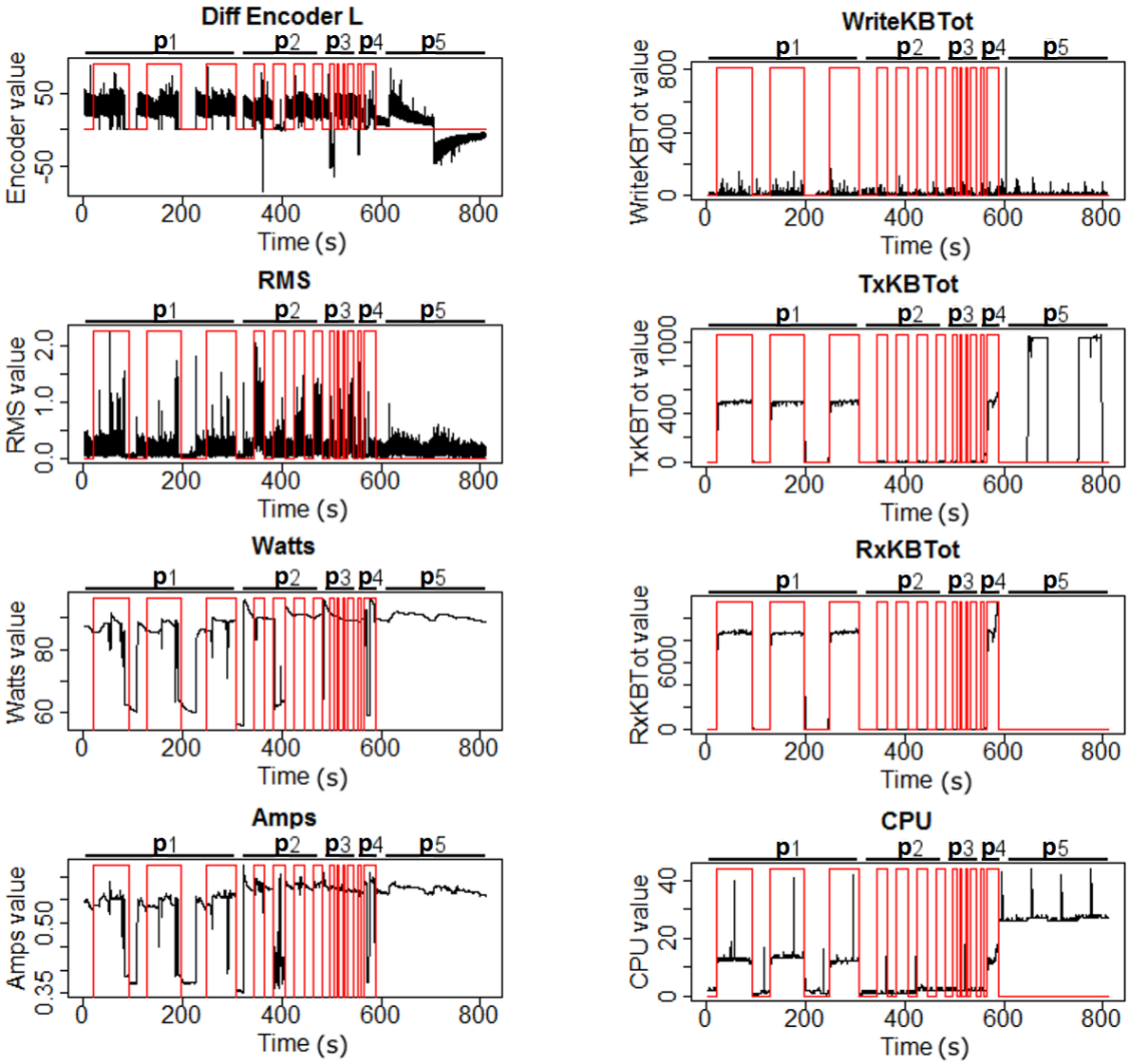


Figure 4.6: The data for physical and cyber features collected during S6 scenario with 5 periods (denoted as p1 - p5, and presented one after the other). The overlaid frames denote the periods of time that a cyber attack (denial of service or command injection) is on. Note that there is no attack in p5.

4.2 Evaluation

Choosing performance metrics for intrusion detection in cyber-physical systems is not trivial, because their priorities are different to those of conventional computer systems. Here, the confusion matrix, receiver operating characteristic (ROC) curves with its area under the curve (AUC) and detection latency are used.

```
Decision tree:

Amps <= 0.6098701:
:...Amps <= 0.5962737: 0 (9802/3)
:   Amps > 0.5962737:
:   :...Watts <= 92.19859: 1 (18)
:       Watts > 92.19859:
:       :...WriteKBTot <= 3.892: 0 (172)
:           WriteKBTot > 3.892:
:           :...CPU <= 2.032: 0 (4)
:               CPU > 2.032: 1 (8)
Amps > 0.6098701:
:...Amps <= 0.613997:
    :...Watts > 96.03431: 0 (35)
    :   Watts <= 96.03431:
    :   :...CPU <= 3.376004: 0 (9/2)
    :       CPU > 3.376004: 1 (155)
    Amps > 0.613997:
    :...Watts <= 97.85741: 1 (555)
        Watts > 97.85741:
        :...Watts > 98.1: 1 (545)
            Watts <= 98.1:
            :...Watts <= 97.9:
                :...WriteKBTot <= 0.01599979: 1 (42)
                :   WriteKBTot > 0.01599979: 0 (23)
```

Figure 4.7: An example of the decision tree rules generated

4.2.1 Confusion matrix

The confusion matrix relates to the number of errors in the outcome of the intrusion detection. The rate of false positives ($FPR = FP/(FP + TN)$) and false negatives ($FNR = FN/(FN + TP)$) with regards to the "ground truth" and the overall accuracy rate ($ACC = (TP + TN)/(TP + FP + TN + FN)$) are appropriate for the evaluation of the performance of the system, where TP, TN, FP and FN stands for true positive, true negative, false positive and false negative respectively. As cyber-physical systems are still not common targets of attacks, they are likely to be the target of non-standard and possibly zero-day attacks. This makes FNR important. In fact, FNR may also affect the detection latency.

As mentioned in Section 4.1.2, the experiment consisted of building four different detection models for the different attack types.

Tables 4.4 and 4.5 show high accuracy rates above 94% for testing data, which reflects the "local-optimal" feature of the decision tree C5.0 algorithm. This is because the training and testing data are sharing very similar characteristics as they are from the same set (but

Table 4.4: Detection results using only cyber input features

	Test	Validation		
Attack	**ACC%**	**FPR%**	**FNR%**	**ACC%**
DoS	99.45	15.77	7.26	90.47
Command inj.	97.58	31.79	22.34	72.80
Malware (NET)	94.99	21.42	18.99	79.70
Malware (CPU)	97.03	21.16	6.76	85.31

Table 4.5: Detection results using both cyber and physical input features

	Test	Validation		
Attack	**ACC%**	**FPR%**	**FNR%**	**ACC%**
DoS	99.84	10.76	41.44	66.70
Command inj.	99.53	29.60	5.74	81.99
Malware (NET)	99.20	25.70	11.31	80.92
Malware (CPU)	99.72	5.43	26.18	85.24

different samples). So, the testing data is the ideal condition where the attack observed is very similar to the attack the system has been trained on. The results using validation data correspond to a more realistic case, where the attack is of the same type but not identical. Detection results for the DoS attack are relatively poor (albeit better than the random guess). The other three attack detection models provide considerably better results with regards to ACC.

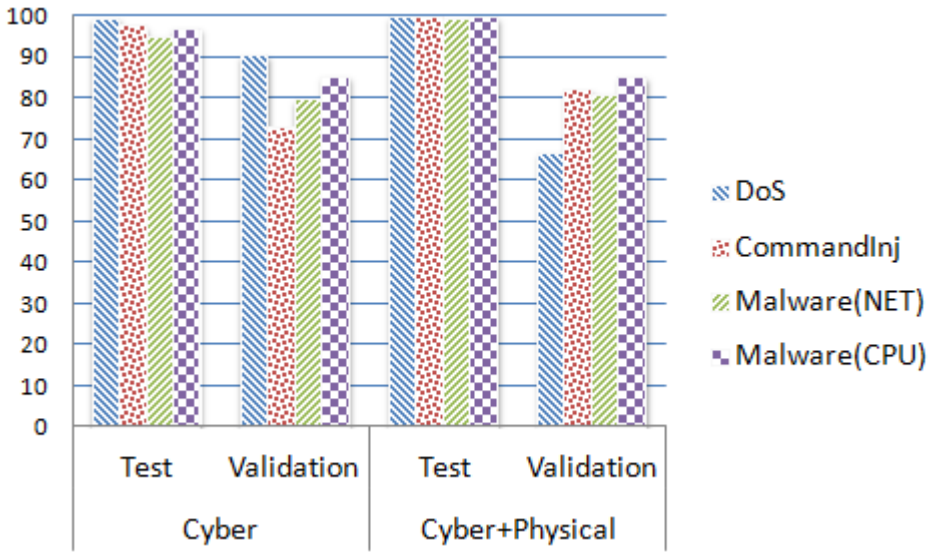


Figure 4.8: Accuracy chart for the four models on test and validation data using cyber only and cyber+physical features.

4.2.2 Receiver operating characteristic (ROC) curves

Table 4.6: Area under the curve (AUC) comparison using cyber only and both cyber and physical input features

	AUC	
Attack	**Cyber only**	**Cyber + Physical**
DoS	0.89	0.73
Command inj.	0.75	0.87
Malware (NET)	0.82	0.86
Malware (CPU)	0.91	0.97

As the system is effectively a binary classifier ("Yes, there is an attack" vs. "No, there is no attack"), the ROC curves are used to measure its performance. The ROC curve plots the true positive rate (TPR) against the FPR for different thresholds as shown in Figure 4.9. In an ideal detection result, the curve should go through the point (0,1) where the FPR is 0% and TPR is 100%. The area under the curve (AUC) of the ROC curves is a metric to compare the quality of different binary classifier models.

The performance of all four detection models is demonstrated in Figure 4.9. Using the ROCR package in R [Sing et al., 2005], it is possible to visualise how TPR and FNR change for each detection model with different probability thresholds. As can be observed, the AUC for the malware attack against the CPU (0.97) is considerably higher than other attack scenarios. The models for command injection and malware (NET) attack have similar AUC at 0.87 and 0.86 respectively. The DoS scenario has the lowest AUC model at 0.73.

Repeating the same experiments without taking into account the physical input features, the performance of the intrusion detection system drops noticeably for command injection and the malware attacks, but not for DoS (see table 4.6 for the detailed comparison).

4.2.3 Detection latency

A primary concern for cyber-physical systems is their real-time nature, which means that timeliness, and hence detection latency (DL) is particularly important. In fact, detection

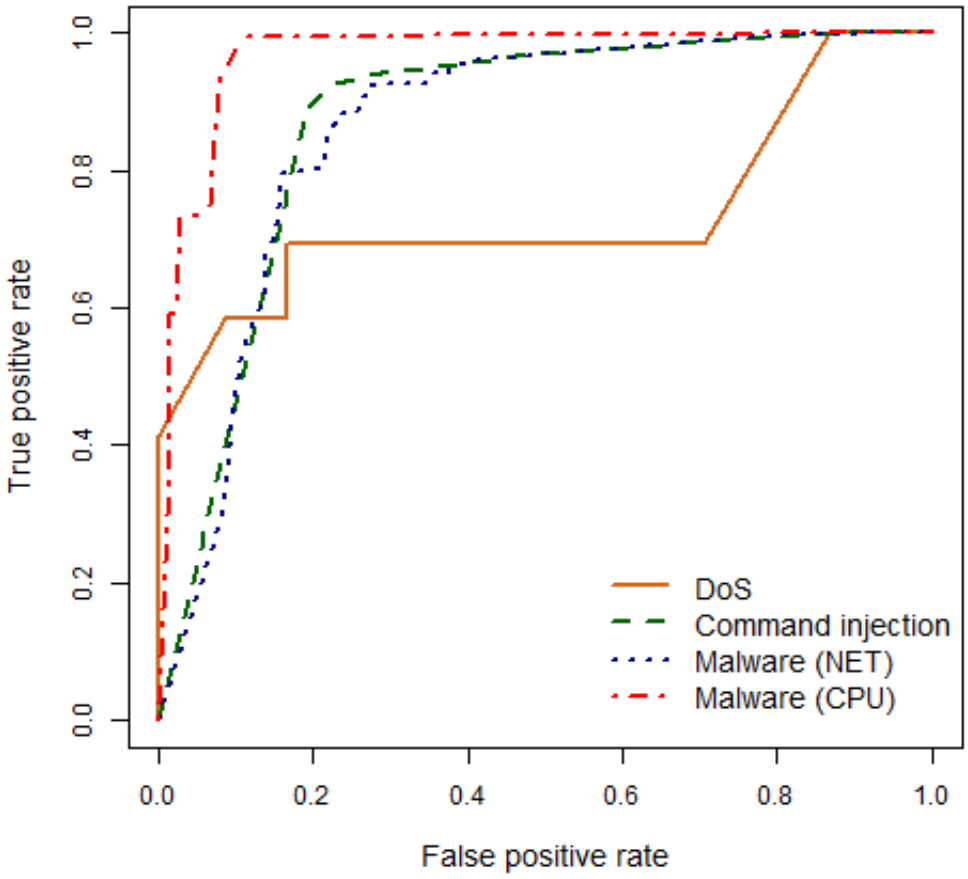


Figure 4.9: The ROC curves of the detection rules for the cyber attacks in the case where all eight cyber and physical input features are utilised.

latency may be potentially more important than the accuracy of the detection, as there is no point in detecting an event after it has caused permanent physical damage to a vehicle (e.g. by causing it to veer off the road and crash). A few researchers have included detection latency as a metric, but mostly in relation to mobile ad hoc networks and wireless sensor networks [Striki et al., 2009, Chin and Chuang, 2015]. It is proposed here that this is a significant metric for assessing the performance of our system. This has also been recognised in Mitchell and Chen's excellent survey in [Mitchell and Chen, 2014].

Based on the design of the intrusion detection framework (Figure 4.1), there are various factors that affect detection latency including the data collection time, the processing time and the actual detection accuracy of the mechanism.

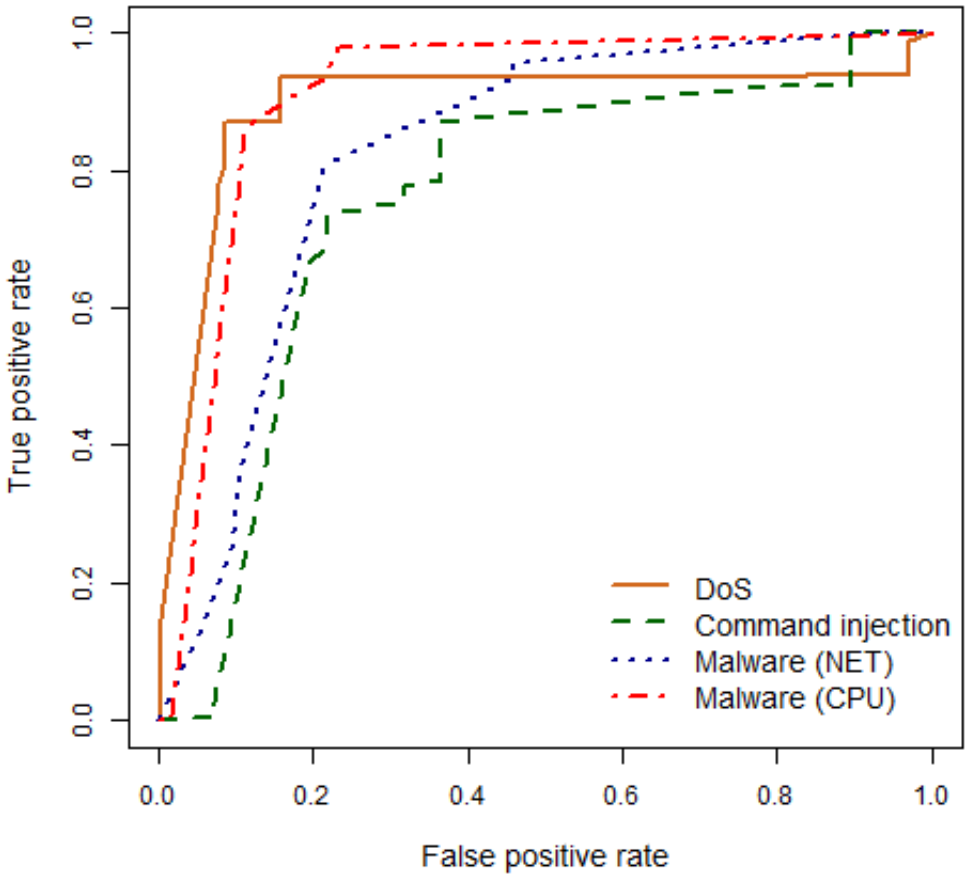


Figure 4.10: The ROC curves of the detection rules for the cyber attacks in the case where only the four cyber input features are utilised.

The data collection time is the time it takes for all data to reach the detection system. It depends on the collection period (T) of each feature, as in Table 4.3, the time taken for data preparation, including the interpolation period, and the time it takes the data to be sent for processing. Then, there is the processing time, which is the time it takes the algorithm to process the data and reach a detection decision.

The data collection and processing times remain largely the same across different detection scenarios (around 1 s). What does differ significantly is the delay in relation to the accuracy of the algorithm. A false negative outcome would delay the detection of an attack until the first true positive result is achieved for a given attack. So, a visual presentation of detection latency is depicted in Figure 4.11. Note that the YES Vs. NO

Table 4.7: Detection latency (ms) for different attack types (cyber only vs. cyber + physical)

	Attack block (s)			Detection latency	
Attack	Block	Start	End	C (ms)	C+P (ms)
DoS	B1	374.04	423.04	520	500
Command inj.	B2	312.32	331.32	1520	960
	B3	342.32	361.32	1840	540
Malware (NET)	B4	362.02	376.02	1520	1440
	B5	393.02	407.02	1020	500
	B6	422.02	436.02	1520	1520
Malware (CPU)	B7	360.06	374.04	1520	700
	B8	390.06	404.04	500	500
	B9	420.7	435.04	500	520

points correspond to the ground truth. So, an incorrect detection on a YES line is a false negative (FN), and on a NO line is a false positive. Detection latency is added by the time of the first FN block (detection latency block) occurring at the beginning of each YES (attack time) block (see Table 4.7).

4.2.4 The significance of physical features

To understand the role of physical features in detection, as mentioned in Section 4.1.2, there should be different detection models with different sets of features for scenario S6.

As shown in Table 4.8, detection based on physical features only is rather poor (albeit better than a random guess). Nevertheless, including physical features together with cyber features provides considerably better results than using only cyber features with regards to ACC and FPR, but is worse in terms of FNR. In terms of the features utilised the most by the decision tree approach, these are the network incoming traffic rate ([NET]RxKBTot) and the energy-related ones (Amps and Watts). While increased vibration of the chassis is visually observed during some of these attacks, this feature is utilised less by the decision tree algorithm. If we remove the physical features, then the algorithm relies almost exclusively on [NET]RxKBTot and the CPU utilisation.

The benefits of adding the physical input features are more clearly seen in Figure 4.12.

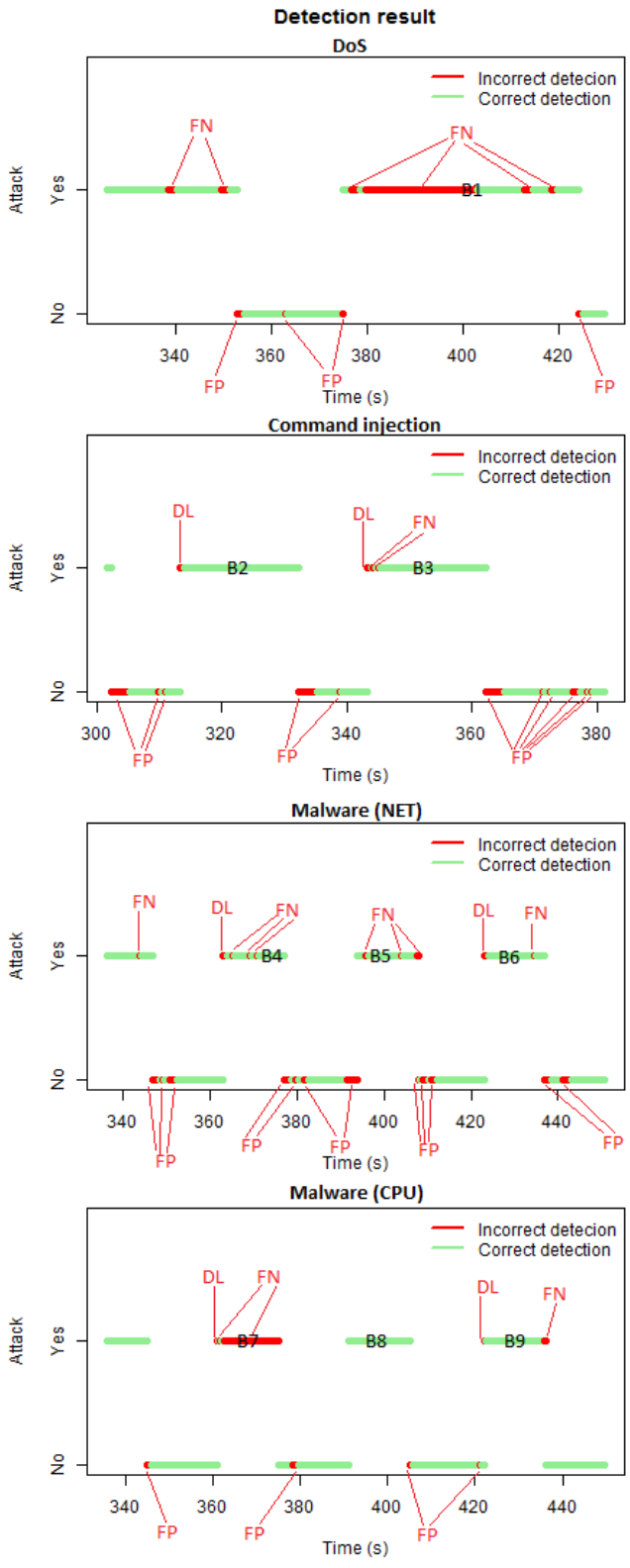


Figure 4.11: Detection result for representative attack scenarios

Using the ROCR package in R [Sing et al., 2005], the performance for the three sets of features are compared: cyber and physical; cyber only; and physical only. The three ROC

Table 4.8: Combined scenario (S6) result

Features	Attribute usage	FP%	FN%	ACC%
All Cyber & Physical (8)	100.00% RxKBTot 99.63% Amps 61.93% Watts 17.53% TxKBTot 5.15% CPU 3.03% WriteKBTot 0.88% DiffEncoderL	4.56	8.76	93.81
Cyber features only (4)	100.00% RxKBTot 60.93% CPU 5.31% WriteKBTot 0.50% TxKBTot	25.91	3.45	82.81
Physical features only (4)	100.00% Watts 85.57% Amps 85.26% DiffEncoderL 71.50% RMS	50.21	12.59	64.40

curves in Figure 4.12 show different TPR, and FNR for different probability thresholds. As can be observed, the area under the curve (AUC) is considerably higher when all eight features are utilised (96.79%) than when only the cyber features (82.38%) or only the physical features are utilised (69.31%). Figure 4.8 summarises the results for FPR, FNR, ACC and AUC in a single bar chart.

We consider these results to be promising. As expected, cyber features play the most dominant role in the detection rules identified by the decision tree algorithm. With different training data sets, the decision tree algorithm may provide different rulesets, but it is important to note that the inclusion of physical features has proven beneficial in all experiments conducted regardless of the particular choice of a ruleset. Importantly, using a simple ruleset-based approach allows for a very lightweight solution to the problem of intrusion detection. For illustration, the overhead in terms of processing time for the purpose of checking the rulesets continuously on the vehicle (e.g. every 1000 ms) is the number of sample points times the processing time per sample, which in these experiments was $50 * 0.014ms = 0.71ms$ of processing overhead (for every 1000 ms), which is insignificant for this robotic vehicle as well as the vast majority of vehicles.

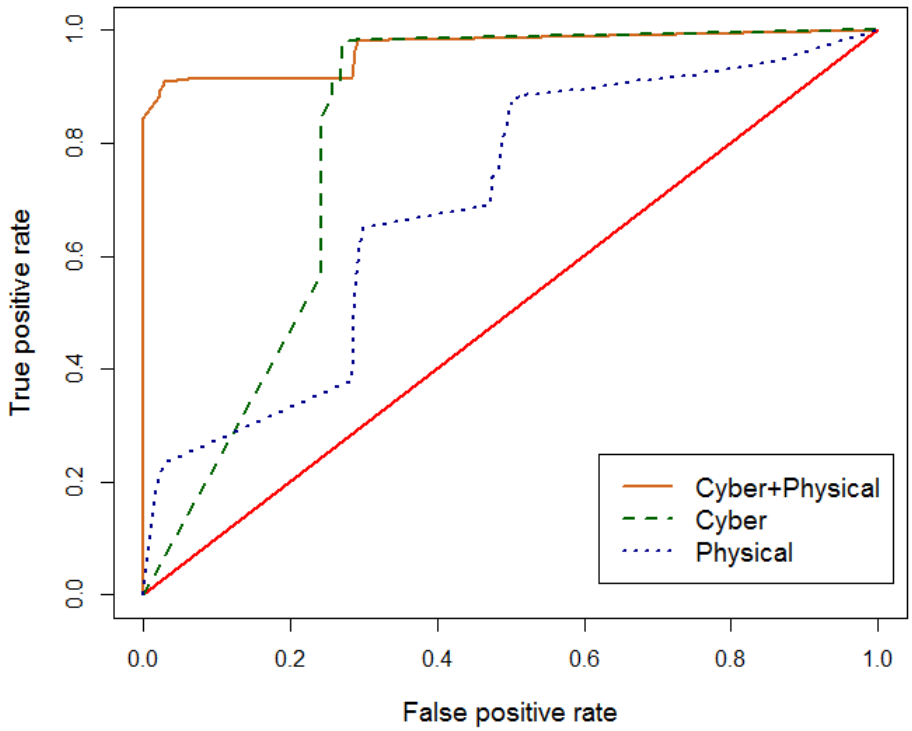


Figure 4.12: The ROC curves of the detection rules for the three sets of features: Both cyber and physical; cyber only; and physical only. The (0.0) to (1.1) line is the random guess line.

4.3 Conclusion

As real-world and experimental cyber-physical attacks are becoming more prevalent, on-board intrusion detection systems are expected to become particularly important, especially for standalone robotic vehicles, whether manned or unmanned [Loukas et al., 2013a]. It was hypothesised here that the existence of physical manifestations of cyber attacks on vehicles and other cyber-physical systems constitutes an opportunity for intrusion detection purposes. A series of experiments have been conducted with four different types of attack and different performance for each one has been observed. For example, it can be seen that utilising physical features in addition to cyber features can increase the false negatives dramatically, but this is not the case for command injection attacks. For these, the addition of physical features improves all detection performance metrics. Similarly, an improved performance is observed for the two types of malware attacks.

Looking beyond traditional performance metrics, what does appear to be a consistent

benefit is the noticeably lower detection latency. It is proposed here that for robotic vehicles this is important. Poor accuracy, such as high false positives, is expected in some cases, especially for highly dynamic systems that are influenced by their physical environment. Here, the detection latency can be more important, for example, if an unacceptably high delay means that a cyber-physical attack can cause the vehicle to crash and injure human beings before it is detected.

In the next chapter, the possibility of utilising the physical features to enhance detection accuracy are explored further, but using a more processing-heavy detection algorithm.

Chapter 5

Offloaded deep learning based intrusion detection

5.1 Introduction

The technique presented in the previous chapter is relatively lightweight because the vast majority of vehicles can afford only limited processing resources. The focus is on minimising the processing load, either by applying lightweight techniques from statistics or by predefining simple behavioural rules that are easy to monitor. This is because they are all limited by the onboard capabilities of the vehicle at hand. As a result, they usually cannot leverage modern classification techniques, such as those currently developed in the field of deep learning and also have a noticeable impact on the energy efficiency of the vehicle. In effect, the stronger the detection algorithm, the less attractive a solution is for a resource-constrained vehicle. To address this trade-off, we turn to the emerging field of cloud robotics [Hu et al., 2012]. The proposal is to offload the bulk of the processing required to run a deep learning algorithm for intrusion detection to a more powerful infrastructure (whether a single server, cloudlet or cloud). Computation offloading, refers to the process of executing certain computational tasks on more resourceful computers

which are not in the user's immediate computing environment. The concept has similarities with the online forensics techniques used for cloud-based detection of malware and tainted data on Android smartphones [Houmansadr et al., 2011, Portokalidis et al., 2010]. However, instead of crowd-sourcing detection, the focus here is on utilising computational offloading to enable deep learning without the processing and energy cost which would otherwise be prohibitive for a vehicle.

Computational offloading is a common approach for the similarly resource-constrained mobile devices. As the number of vehicles that require protection against cyber attacks is increasing, offloading the task of intrusion detection to remote servers, cloudlets or general purpose cloud infrastructures becomes a realistic option. The increased processing resources that are available in this manner can allow much more advanced detection approaches, e.g. based on deep learning. In this chapter, a proof of concept has been provided by the implementation of intrusion detection offloading for a small robotic vehicle, which uses a deep learning architecture to achieve noticeably better detection accuracy than standard statistical machine learning techniques. Data related to the cyber and physical processes of the vehicle are collected periodically and fed as time series data to a recurrent neural network architecture, benefiting from a long short-term memory hidden layer, which helps learn the time context of the impact of the cyber attacks on the robotic vehicle. The attacks include denial of service, command injection and malware exfiltrating large amounts of data. The practicality of offloading deep learning based intrusion detection periodically depends on the resources afforded onboard and on the remote infrastructure, as well as the reliability and performance of the means of communication between them. The relationship with a mathematical model is evaluated, which takes into account network failures and packet losses and validates it experimentally. Naturally, the more demanding the computation, e.g. when increasing the number of hidden neurons, and the more reliable the network connection, the more attractive the offloading option becomes.

At the same time, over the past couple of years, the growing maturity in deep learning algorithms has led to wider use outside of its traditional applications in image and natural

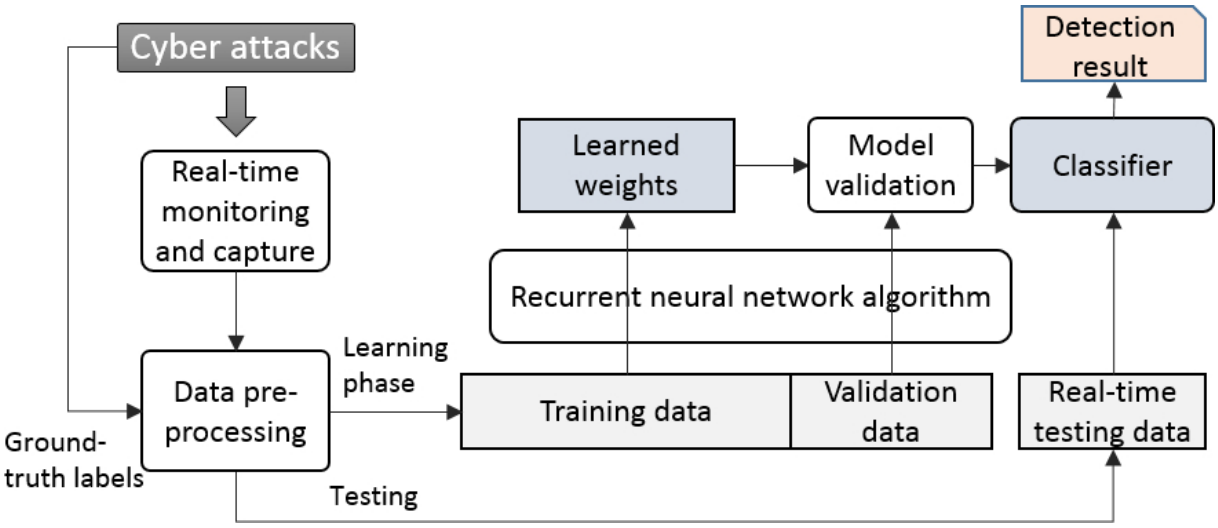
language processing. Up to now, most deep learning based IDS proposals have addressed attacks against traditional computer networks, showing the superiority of the approach on the old KDD Cup 99 [Kim et al., 2016] and the refined NSL-KDD [Javaid et al., 2016] datasets. An exception is a recent work by Kang et al. [Kang, 2016], which is geared towards the automotive industry and specifically vehicles that rely on the controller area network (CAN) bus. While a promising start, the particular work is limited to a generic command injection attack, which is detected by monitoring a single data source (the network packets on the CAN bus) and using a generic deep neural network architecture, which does not account for the temporal aspects or the overall state of the vehicle. Also, it has been evaluated only in simulation. Here, we progress further with the following key contributions:

- Design a recurrent neural network architecture that is appropriate for the real-time analysis of multiple sources of data collected periodically onboard the vehicle and representing both its cyber and physical processes
- Produce a prototype implementation of deep learning based intrusion detection for cyber-physical attacks on a real robotic vehicle, tested for three different types of attacks and compared against some of the most popular statistical machine learning techniques
- Evaluate both experimentally and using a mathematical model the practicality of a computational offloading configuration for providing resource-constrained cyber-physical systems with access to high-end intrusion detection

5.2 Cyber-physical intrusion detection using a recurrent neural network architecture

Figure 5.1 summarises the overall experimental approach taken for construction of the deep learning IDS. Three different types of cyber attack were launched against the robotic

Figure 5.1: Detection approach



vehicle and data was collected with regards to eight features, appending the ground truth labels based on the timings of the attacks (whether an attack was really in action at each point in time or not).

As the data from different features come at different times and at different sample rates, they were synchronised in a pre-processing phase. The output of pre-processing is a data stream with a data point sample interval of $\tau = 20ms$. In the learning phase, the data is split into a training set and a validation set (ratio 0.7:0.3). The recurrent neural network algorithm is applied on the training data to produce a detection model, as defined by the weights of the connections between neurons. The model is then validated using the validation set before producing the final classifier, which is evaluated experimentally using real-time testing data.

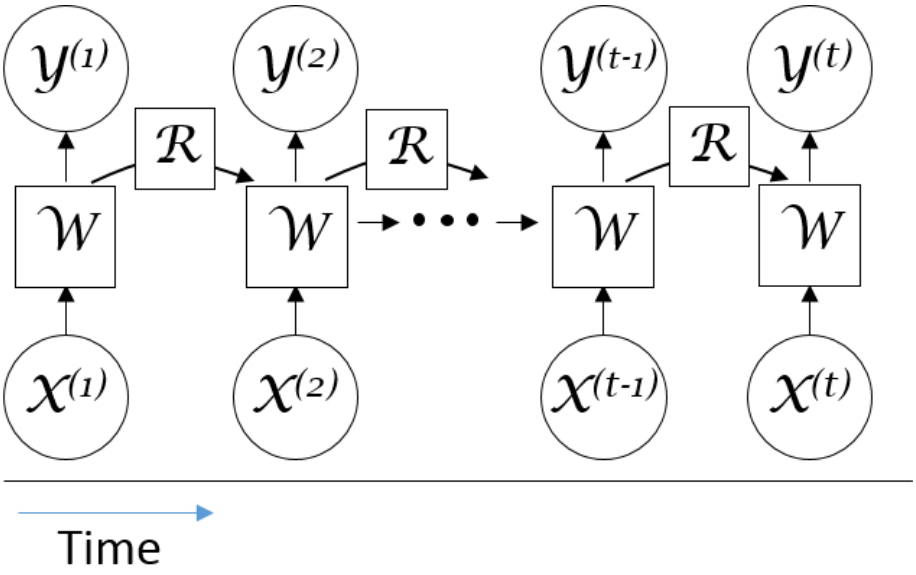
Cyber-physical attacks interacting between sensors, actuators and computational components often exhibit temporal correlations based on complex time dependencies of arbitrary length. For example, in a cyber-physical system such as a robotic vehicle, a rogue operator executing remote command injection may command the vehicle to accelerate forward. As a result, this may cause a spike in network traffic leading to a change in vehicle wheel speed, which increases power consumption and current. Here these feature interactions occur one after the other in a specific sequence. The result of such sequential temporally-related behaviours leads to the generation of time series datasets with the potential for

high-dimensional inputs that change over time. For feed-forward neural networks, this type of key temporal information can be lost due to the fact that classification is based on recognising patterns in unidirectional feature-space, where the output of an input layer does not affect the same layer. That is, a feed-forward neural network looks for occurrences of the same patterns in the feature-space based on current state, irrespective of the prior inputs patterns that came before. By comparison, recurrent neural networks exhibit dynamic temporal behaviour by using internal memory to process arbitrary sequences of inputs based on interconnected hidden layers from previous input states; feeding the hidden layers from the previous states as an additional input into the next state. In this manner, recurrent neural networks are trained based on historic and current inputs, where the likelihood of an attack occurring depends both on prior states of the features and the current states of the features at that point in time. In other words, a recurrent neural network can be seen as a feed-back neural network where input is bidirectional by introducing loops that feed prior input back into the network.

As cyber-physical attacks occur as a series of both cyber and physical events over time, a recurrent neural network approach has been chosen for the development of the cyber-physical IDS, which is proven to be highly appropriate for handling multivariate sequential time series data [Barbounis et al., 2006, Du et al., 2015]. Figure 5.2 shows the recurrent nature of the learning process, where $X^{(t)}$ is the vector of input features and $Y^{(t)}$ is the binary detection decision (0 if no attack, and 1 if attack) at time t, and $Y^{(t)} = WX^{(t)} + RX^{(t-1)}$, with W and R being the weight matrices in relation to X and the incorporation of the output of the previous step respectively.

In terms of the deep learning architecture designed for our intrusion detection methodology, in the input layer, $U^{(1)}$, $U^{(2)}$, ..., $U^{(n)}$ is the time series dataset in a period T, which corresponds to $n = \frac{T}{\tau}$ data points. We group k consecutive data points together into $X^{(k)}$, $X^{(k+1)}$, ..., $X^{(n)}$, as shown in Figure 5.3. The purpose of grouping is to help the algorithm have a picture of more than a single point in taking a detection decision, but without using the whole dataset (of n data points in one period T) either, which would increase considerably the detection latency. So, $1 \leq k \leq n$. The hidden layer includes

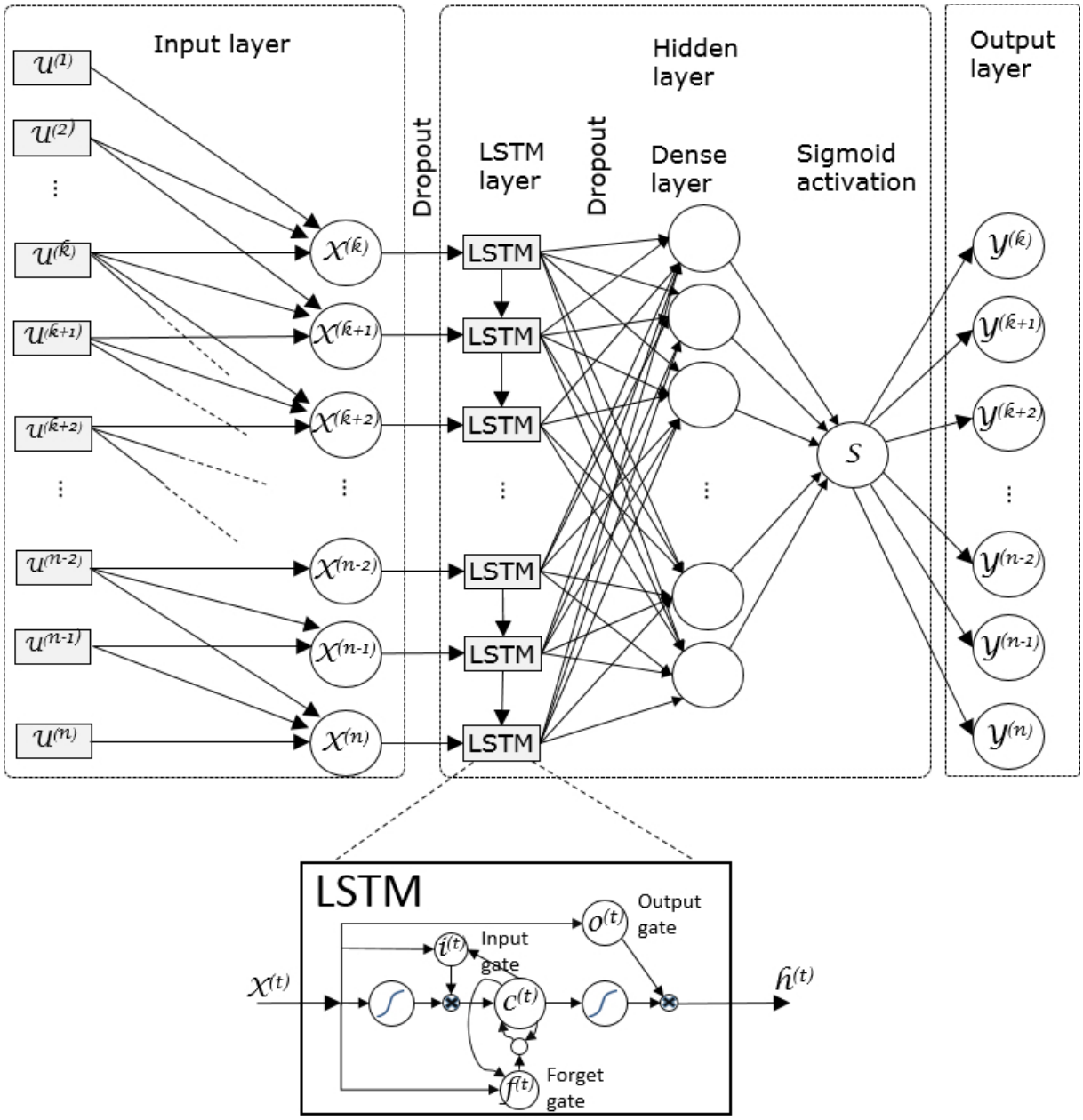
Figure 5.2: Learning process using recurrent neural network



a Long-Short Term Memory (LSTM) layer, a dense layer and Sigmoid activation. Conventional recurrent neural networks find it difficult to train with long step sizes due to the"vanishing gradient" problem in gradient-based activation functions (such as sigmoid or tanh). The vanishing gradient relates to the exponential decrease in the size of the gradient (from which the network learns changes in the input parameters which affect the expected output) by iteratively mapping large input regions into smaller output regions through sequential layers or long inputs sequences in the neural network [Bengio et al., 1994]. As a result, when the gradient reaches a value near zero, the recurrent nature of the neural network produces small outputs even for large changes in input.

LSTM helps solve this problem by employing a "gating" function (1 to remember the input and pass it to the next hidden node/layer or 0 to forget the input) that replaces the activation function. The network is trained on the combination of the gates in the network and as long as the gates are 1 along the input sequence or across all hidden layers in the network, the network can remember the values of early input to identify how it affects the expected output. Using this approach the neural network takes into account the time context of the impact of the cyber attacks on the robotic vehicle, by learning long-term dependencies on the input. We use a standard LSTM architecture, with each block containing gates that determine the significance of the input and whether it should

Figure 5.3: Deep learning architecture



continue to remember its value or forget it, and when it should output it. The LSTM layer is followed by a dense layer, where the number of hidden nodes serves as our main tuning parameter. Then, a sigmoid activation function converts the values produced by the dense layer into real values between 0 and 1. Finally, a binary detection decision (0 or 1) is taken based on a predefined threshold, which is our second tuning parameter.

For the deep learning implementation, we used Keras [Chollet, 2015] to run on top of TensorFlow/Theano library. Keras is a Python neural network library that supports easy and fast prototyping through modularity, minimalism and extensibility.

5.3 Experimental evaluation of deep learning based detection accuracy

For the deep learning intrusion detection model, the RNN architecture has been designed as shown in Figure 5.3, where Table 5.1 lists the configuration variables of the RNN which behave as a set of tuning parameters for the detection model. The cyber and physical input features are captured from different locations within the architecture (Figure 3.2) and with different sample collection periods/interval (see Table 4.3). Interval T is the shortest time to have all the features collected and single point sample interval τ is the fastest rate that the features were captured. The sample size n $(= T/\tau)$ per interval then is equal to 50 $(= 1s/0.02s)$.

The k grouping for combining a set of consecutive features into one is a type of recurrent sliding window technique in machine learning to improve the accuracy [Dietterich, 2002]. In the paper, applying the recurrent sliding window technique could help to increase the percentage of correct word-level pronunciation by 12%. [Trutschel et al., 2014] claimed that the classification errors are reduced by 8% on average using the application of feature combination. [Plahl et al., 2013] also found 5% relative better work error rate with fewer parameters using feature combination and stacking for both recurrent and non-

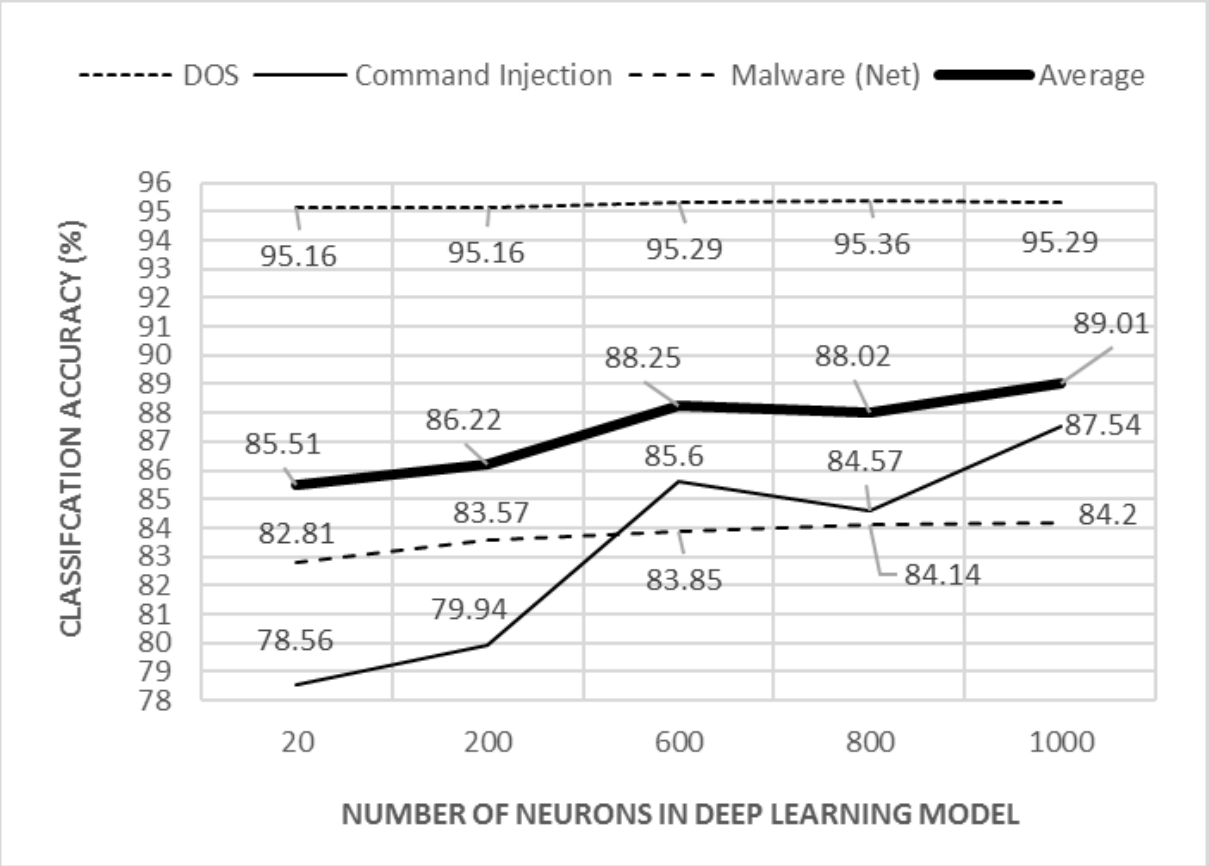
Table 5.1: Deep learning parameters

Parameter	**Value**
Interval T	1 s
Single point sample interval τ	0.02 s
Sample size n per interval	50
Grouping k	10
Number of hidden nodes	20, 200, 600, 800, 1000
Number of epochs	300
Dropout ratio	0.3
Batch size	16
Validation ratio	0.3
Loss function	Binary cross entropy
Optimiser	adam
Activation function	Sigmoid
Metric	Accuracy (ACC)

recurrent neural network. [Li et al., 2016] acknowledged the strength of sliding window design technique and proposed a multi-task end-to-end joint classification and regression recurrent neural network to capture the long-range temporal dynamics in a time series features. Here, we experimented with using groups of $k = 10$ and compared this case to not grouping at all ($k = 1$, no-grouping) and to using a single group for all the features in one interval ($k = 50$, max-grouping). It was found that grouping (i.e. $k = 10$) exhibited higher accuracy 87.54% compared to 86.81% and 86.03% for no-grouping and max-grouping cases in the experiments.

To evaluate the performance of the intrusion detection model, we have used a standard confusion matrix to determine the number of correct and incorrect detection results. This included TP (true positive - correct detection of attack), FP (false positive - incorrect detection of attack), TN (true negative - correct detection of non-attack), FN (false negative - incorrect detection of non-attack)) and calculated the overall accuracy rate ACC ($ACC = (TP + TN)/(TP + FP + TN + FN)$). ACC is selected as a consistent metric for comparing different RNN detection models performance on unseen test data. ACC was also used as the key performance criterion for comparing our deep learning model with other machine learning (ML) techniques. Figure 5.4 shows the overall accuracy of each RNN model, differentiated by a number of hidden nodes (neurons) in the model

Figure 5.4: Detection accuracy with deep learning models for different cyber attacks



configuration for different cyber attacks.

5.3.1 Deep learning vs. popular machine learning techniques

For deep learning to be practically useful in the long term, it is important that it performs better across a range of attacks against standard statistical machine learning algorithms. For comparison, algorithms have been chosen which are considered safe choices for classification across a range of domains. A comprehensive study by Delgado et al. [Fernndez-Delgado et al., 2014], which put to the test 179 different machine learning classifiers across 121 different datasets, showed that random forest was the best-performing technique, with an average accuracy of 94% and reporting over 90% accuracy across 84% of the datasets, followed closely by Support Vector Machines (SVM) with an average of 92% accuracy. For these reasons, random forest and SVM are commonly used in intrusion detection [Hasan et al., 2014]. In table 5.2, we compare our deep learning based approach with random

forest, SVM with a radial kernel, SVM with linear kernel, logistic regression and decision trees. Deep learning achieves the highest accuracy rate in two of the three attacks, as well as the highest overall average (89.03%).

Logistic Regression functions are a special case generalised linear model, employing Bernoulli distribution to estimate the probability of a binary response based on one or more independent features (e.g., robotic vehicles cyber-physical attributes) and their relationship with a categorical dependent variable (e.g. the attack label); the probability output is applied to a threshold (typically 0.5) as to whether the classification is 1 or 0. Support Vector Machines (SVM) employ the concept of a hyperplane in n-1 dimensional space that best separates two classes of datapoints with the maximum margin. In SVM, datapoints that support either side of the hyperplane are the "support vectors" and in cases where these data points are not linearly separable, are projected to a higher dimensional space where linear separation is possible. In the case where multiple classes are present (e.g. multiple feature variables and a dependent variable), a one versus many binary classification approach is taken. C5.0 and Random Forest are naturally non-linear machine learning algorithms, whereby C5.0 functions as a decision tree classifier employing Boolean logic in series of decision rules, inferred by the feature data, to determine which class the data belongs to. Random Forest is an ensemble tree classifier that trains n decision trees with different re-sampled versions of an original dataset, reducing the high variance inherent in a decision single tree and improving the generalisation of model performance by averaging the standard error of n trees across the ensemble in order to produce a final model with minimal variance.

In the experiments, the linear classifiers logistic regression and SVM comfortably outperformed their non-linear counterparts, with the exception of the deep learning classifier. In fact, compared to the popular machine learning techniques tested in the experiment our RNN-LSTM deep learning model achieves the highest detection accuracy rate in two of the three attacks, as well as the highest overall average (89.03%); practically matching the logistic regression classifier's performance for denial of service detection (95.36% compared to 95.92%, respectively). The experimental results give a good indication of

Table 5.2: Comparing the performance of the deep learning and other popular machine learning algorithms for cyber-physical IDS classification

Machine learning technique	Attack	ACC (%)	Average ACC(%)
Logistic Regression	Denial of Service	**95.92***	84.40
	Command Injection	82.51	
	Malware	74.77	
Decision Tree (C5.0)	Denial of Service	68.34	72.10
	Command Injection	63.56	
	Malware	81.91	
Random Forest	Denial of Service	75.43	77.51
	Command Injection	75.06	
	Malware	82.05	
SVM (Radial Kernel)	Denial of Service	79.45	78.61
	Command Injection	72.59	
	Malware	83.80	
SVM (Linear Kernel)	Denial of Service	95.18	86.69
	Command Injection	85.15	
	Malware	79.75	
Deep Learning (RNN)	Denial of Service	95.36	**89.03***
	Command Injection	**87.54***	
	Malware	**84.20***	

***Best performing technique**

the deep learning models general ability to perform accurate detection across a range of different attacks; supporting its applicability as a robust IDS classifier when compared to other standard machine learning techniques which were not as consistent across the three attacks tested.

5.4 The networking configuration of offloading

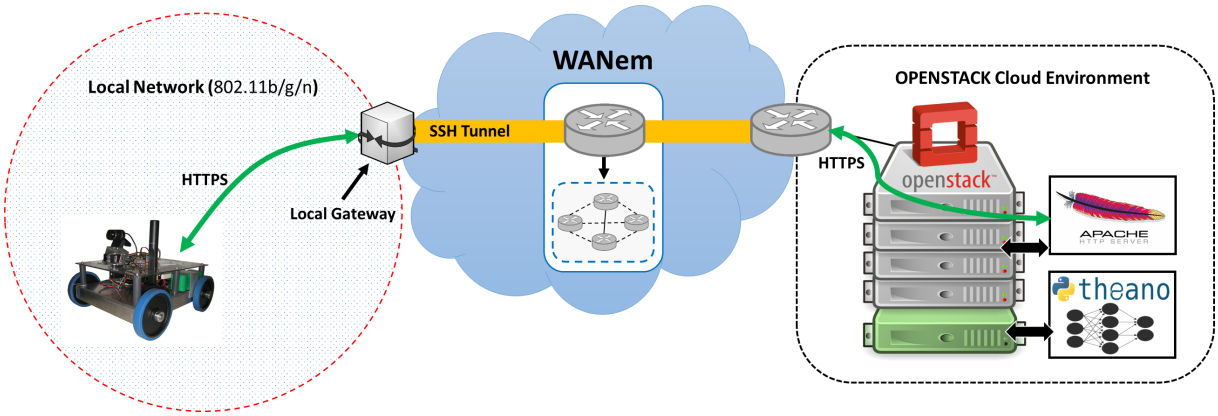
The network testbed consists of three discrete modules, an 802.11n wireless local area network (WLAN), a point to point wide area network (WAN) via the *WANem* wide area network emulator and a remote OpenStack cloud platform. The WLAN provides the vehicle with mobile connectivity to a local network gateway conducting port forwarding between the vehicle and the deep learning server for offloading, through an SSH tunnel over the WAN. Using the client-side URL transfer library *libcurl* [Stenberg, 2016] and *PyCURL* (Python Interface to libcurl), the vehicle offloads detection tasks by uploading

sensor data samples, at interval period T.

The offloading process has been designed to employ a lightweight mechanism that is both robust to different network conditions and suitably secure, enforcing data confidentiality, integrity and authenticity. For this the HTTPS (HTTP over Transport Layer Security 1.2 (TLS)) protocol was selected to perform network offloading to a web server (via *PyCURL*), using the traditional client-server model to transfer over a authenticated and encrypted communication channel. Certificate-based public key server authentication was employed to guarantee the identity of the cloud test-bed (and the trustworthiness of the detection results source). Aside from data confidentiality and integrity protection supplied by Transport Layer Security (TLS) 1.2, HTTP was selected as a robust network transport protocol due to its native reuse of existing persistent connections via keep-alive functionality; ensuring the TLS handshake is performed only during the initial connection. As a result, an HTTP request to offload data to the web server will reuse an existing HTTP connection as long as the detection offload period and transport latency is smaller than the HTTP keep-alive timeout configured. In this configuration, latency incurred by the TLS handshake is effectively avoided after the vehicle's initial offload (e.g. when the vehicle turns on or connects to a network), with subsequent encryption and decryption introducing negligible microsecond latency [Dierks, 2008], confirmed with the sample of data load in the empirical experiments. Moreover, HTTP provides a lightweight choice for data transport as it employs data transfer pipelining and automatic data compression which helps optimise TCP performance and packet transfer speed, reducing network load and symptomatically decreasing detection latency.

All data communication between the vehicle and remote web server is vehicle initiated, through the use of python scripts making calls to the libcurl library. The scripts operate as a set of continual loops. On initiation, a sensor sample generated at interval T is retrieved and transferred via an HTTP POST to the deep learning cloud for every detection period T_d. If the POST is successful, another loop is then spawned, continually polling the server with HTTP GET requests until a detection result is successfully retrieved. On receipt of the detection result, the next sensor sample is then collected when the next detection

Figure 5.5: Experimental testbed including vehicle and offloading infrastructure



period is reached and the HTTP data transfer process is repeated. Figure 5.5 shows a high level overview of the network topology used to remotely access the cloud.

5.5 Evaluating the practicality of offloading detection

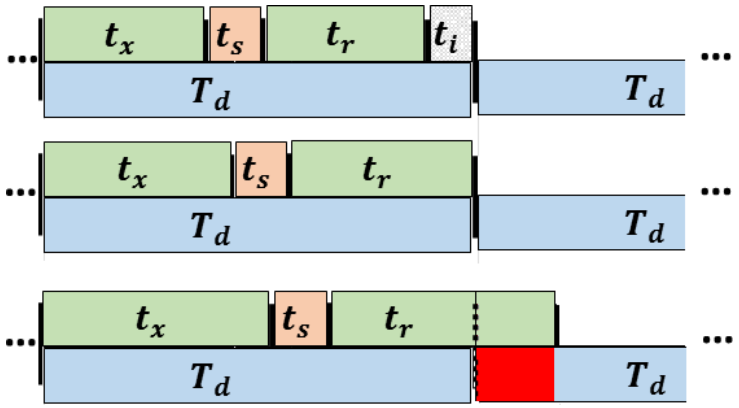
Recently, Canziani et al. conducted a study that compared the computational performance between multiple state-of-the-art neural network architectures in terms of classification accuracy, memory footprint, parameters, operations count, inference time and power consumption [Canziani et al., 2016]. The study showed that for a minor increase in classification accuracy, the computational cost (e.g., processing time) was also significantly increased. Therefore, given the resource constraints inherent in a power-limited vehicle, it would be more efficient to off-load this task to an external and likely more powerful system in order to reduce computational processing time and minimise detection latency as a result.

In this manner, realising the benefits of offloading detection across a distributed service to a server, cloud or cloudlet, the transportation of detection data requires network connectivity that is resilient to and practically useful over variable conditions; especially over the public Internet where no reliability or quality of service is guaranteed. For cyber-physical systems relying on fast and reliable attack detection, the problem is exacerbated by the

risk of dropping crucial data due to unreliable network connectivity. Therefore, a trade-off between onboard detection and remote offloading is largely determined by the available local resources and the quality of network conditions to the remote detection system.

We consider the task of offloaded periodic cyber-physical intrusion detection, which involves uploading the latest sample of data collected from the vehicle to a remote server in time t_x, processing the data on the remote server to produce a detection result in time t_s, and transmitting the result back to the vehicle in time t_r, followed by possible idle time t_i until the next iteration. So, the detection period T_d is $T_d = t_x + t_s + t_r + t_i$. Detection can be practical (and not cause an infinitely increasing queue of delayed results) only if $t_i \geq 0$, as presented visually in Figure 5.6.

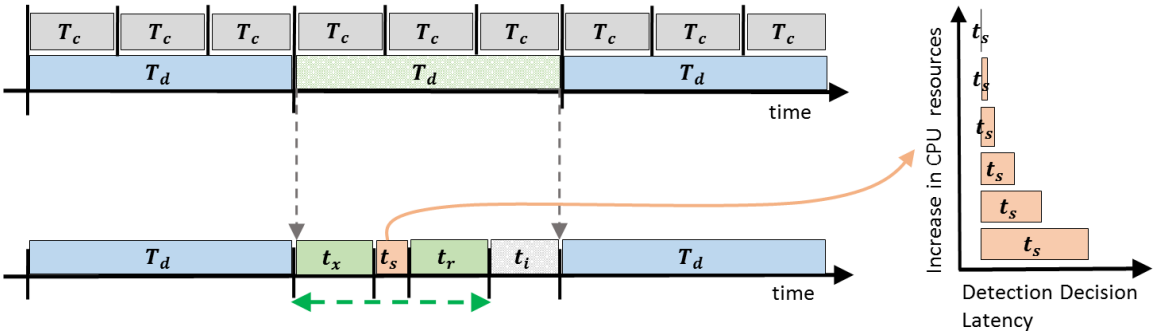
Figure 5.6: Example of variable offloading detection latency within the constraints of detection period T_d. The top and middle figure correspond to the practical cases, where $t_i > 0$ or $t_i = 0$ respectively, while the bottom figure corredsponds to the impractical case, where $t_i < 0$.



Assuming that detection is accurate and an attack has occurred at a random point in time within the previous interval T_d, then the detection latency t_l is the time between occurrence of attack and beginning of next detection cycle plus the time to upload the data, process them, and return the result. Assuming uniform distribution of the probability of the attack having occurred at a random time within T_d, then the mean corresponding delay is $\frac{T_d}{2}$, and overall the mean detection latency is:

$$\bar{t_l} = \frac{T_d}{2} + \bar{t_x} + \bar{t_s} + \bar{t_r} \tag{5.1}$$

Figure 5.7: Network offloading time sequence for offloaded IDS detection with sample collection period T_c and detection period T_d. The practicality of offloading depends largely on the time t_s needed to complete detection on the server, which in turn depends on the server's processing resources and the algorithm's complexity.



The time to complete the processing to produce the detection result depends on the algorithm chosen, implementation approach and the processing resources that are available, as illustrated in Figure 5.7. The aim is to evaluate the upper limits for t_s with CPU resources available in a typical public remote processing platform (remote server, cloudlet or cloud) that allows periodic offloaded detection to be practical, which translates into t_i not dropping below 0.

In ideal communication conditions, where there are no packet losses or network failures and in accordance with the standard practice in computation offloading modelling [Kumar and Lu, 2010,Loukas et al., 2016], the time to upload or receive data over a communication channel can be modelled in relation to the data size uploaded D_x and received D_r by the vehicle and the corresponding transmitting rate R_x and receiving rate R_r as:

$$\bar{t_x}^{(ideal)} = \frac{D_x}{R_x} \tag{5.2}$$

$$\bar{t_r}^{(ideal)} = \frac{D_r}{R_r} \tag{5.3}$$

Thus, the mean detection latency in ideal communication conditions can be represented as:

$$\bar{t_l}^{(ideal)} = \frac{T_d}{2} + \frac{D_x}{R_x} + \bar{t_s} + \frac{D_r}{R_r} \tag{5.4}$$

In non-ideal communication conditions, where we consider packet loss with a probability

p, we assume that, the delay in establishing that a packet is lost and retransmitting means that each bit lost incurs an increase in communication delay equivalent to the time it would take to transmit l bits, where $l \in \mathbb{R}^+$. Mean detection latency in the presence of packet loss becomes:

$$\bar{t_l}' = \frac{T_d}{2} + t_s + (1 + lp)(\bar{t_x}^{(ideal)} + \bar{t_r}^{(ideal)}) \tag{5.5}$$

$$= \frac{T_d}{2} + t_s + (1 + lp)(\frac{D_x}{R_x} + \frac{D_r}{R_r}) \tag{5.6}$$

Mean detection latency increases further if we also take into account the likelihood of a network failure occurring after random time t_θ since last failure and being repaired after random time t_ξ. We assume that failures occur independently and the number of failures occurring in a period T_d follow a Poisson distribution with constant mean $\frac{1}{\theta}$, where $\theta \in \mathbb{R}^+$ is the mean time between failures (MTBF). We assume that the time to repair after a failure follows a normal distribution with a mean time to repair (MTTR) ξ, and standard deviation σ_ξ for the latter.

Communication mechanisms used to transmit the data sample or receive the detection result may implement a form of "keep alive" functionality, which keeps a network session alive for up to T_K time after a failure has occurred. If this time elapses, the session needs to be re-established with handshakes, e.g., for SSL. We denote the delay incurred by the handshakes as t_h.

So, the extra delay incurred by one failure can be represented as:

$$\mathbb{1}[t_\xi < T_K]t_\xi + \mathbb{1}[t_\xi \geq T_K](t_\xi + t_h)$$

$$= t_\xi + \mathbb{1}[t_\xi \geq T_K]t_h$$

The mean extra delay due to one failure becomes:

$$\xi + (1 - P(t_\xi < T_K))\bar{t_h}$$

In a detection period T_d, due to the Poisson property, the expected number of failures is $\frac{1}{\theta}T_d$. So, the mean detection latency in the presence of both packet losses and network failures is:

$$\bar{t_l} = \bar{t_l}' + \frac{T_d}{\theta}(\xi + (1 - P(t_\xi < T_K))\bar{t_h}) \tag{5.7}$$

$$\begin{aligned}\bar{t_l} = \frac{T_d}{2} + \bar{t_s} + (1 + lp)(\frac{D_x}{R_x} + \frac{D_r}{R_r}) \\ + \frac{T_d}{\theta}(\xi + (1 - P(t_\xi < T_K))\bar{t_h})\end{aligned} \tag{5.8}$$

To evaluate the maximum mean time that processing should take to produce the detection result, we take the extreme case of no idle time between detection intervals, hence $t_i = 0$ and consequently $T_d = \bar{t}_{s,max} + \bar{t_x} + \bar{t_r}$. Equivalently:

$$\bar{t}_{s,max} = T_d - (\bar{t_x} + \bar{t_r}) \tag{5.9}$$

The mean latency introduced after the data sample has been collected on the vehicle is (based on (5.1)):

$$\bar{t_x} + \bar{t_s} + \bar{t_r} = \bar{t_l} - \frac{T_d}{2} \tag{5.10}$$

$$\begin{aligned}\bar{t_x} + \bar{t_r} = (1 + lp)(\frac{D_x}{R_x} + \frac{D_r}{R_r}) \\ + \frac{T_d}{\theta}(\xi + (1 - P(t_\xi < T_K))\bar{t_h})\end{aligned} \tag{5.11}$$

So, (5.9) becomes:

$$\begin{aligned}\bar{t}_{s,max} = T_d - (1 + lp)(\frac{D_x}{R_x} + \frac{D_r}{R_r}) \\ - \frac{T_d}{\theta}(\xi + (1 - P(t_\xi < T_K))\bar{t_h})\end{aligned} \tag{5.12}$$

Now, consider the general case where the detection period may be different to the data sample collection period T_c. So, $T_c = aT_d$, $a \in [0, 1]$. If $a = 1$, the detection mechanism runs often enough to ensure complete coverage of time, while $a < 1$ means that the mechanism covers a fraction of the time and an attack may be missed if it occurs outside

this fraction. Substituting T_d by $\frac{T_c}{a}$, (5.12) yields:

$$\begin{aligned} \bar{t}_{s,max} = \frac{T_c}{a} - (1+lp)(\frac{D_x}{R_x} + \frac{D_r}{R_r}) \\ -\frac{T_c}{a\theta}(\xi + (1 - P(t_\xi < T_K))\bar{t_h}) \end{aligned} \tag{5.13}$$

Four network scenarios have been chosen to utilise and validate the model: For network 1, we consider the ideal conditions experienced in a laboratory testbed (ours, in this case). Network 2 represents an ideal cloud service, profiled by measuring real cloud services via the *cloudharmony*[1] web service. The network latency results were used to determine a baseline average transmission delay from London (the experiment test-bed location) to *Google Compute Engine* cloud platform for the round-trip time (RTT) of an HTTPS GET request. For network 3, we have utilised the performance statistics from the 2014 Ofcom mobile broadband study in the UK [OFcom, 2014]; providing an accurate measurement of latency for a typical 4g/3g network service. Network 4 represents an unstable network with a high probability of packet loss, connectivity failures and latency. Table 5.3 provides a summary of the configuration parameters utilised in our evaluations.

Table 5.3: Network scenarios used in experiments and by mathematical model

Profile	Network 1	Network 2	Network 3	Network 4
Round-trip time	4 ms	12 ms	54 ms	200 ms
Packet Loss (p)	0	0.001	0.01	0.03
MTBF (θ)	160 s	60 s	20 s	10 s
MTTR (ξ)	4 s	4 s	5 s	5 s
R_x bitrate	174 Kbps	158 Kbps	116 Kbps	29 Kbps
R_r bitrate	82 Kbps	21 Kbps	5.62 Kbps	0.392 Kbps
Keep alive time (T_K)	5 s	5 s	5 s	5 s
Handshake time (t_h)	37 ms	52 ms	102 ms	416 ms

5.5.1 Network model validation against experiments

Figures 5.8-5.11 show the comparison between model and experiment for the four network configurations specified in terms of the detection latency (average of five runs). For

[1] https://cloudharmony.com/speedtest

Figure 5.8: Detection latency as measured experimentally and estimated mathematically for the case of network configuration 1. The black curve corresponds to the detection latency when the processing occurs on the vehicle itself without offloading via a network.

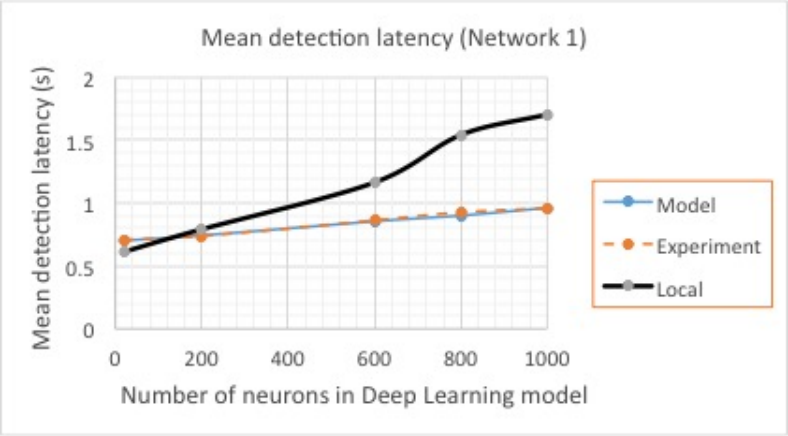
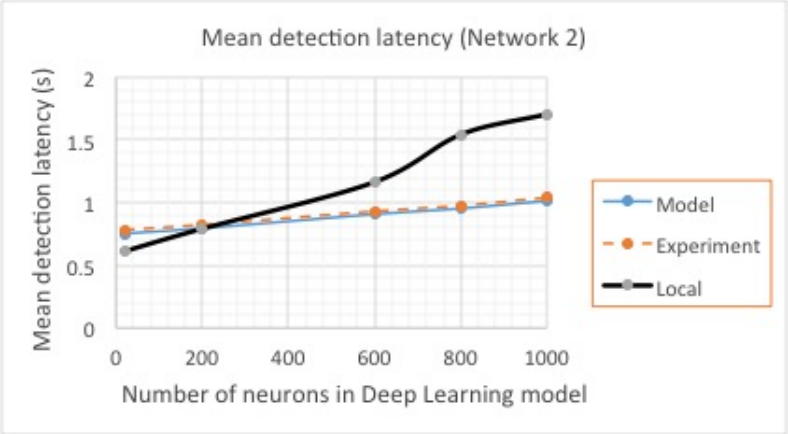


Figure 5.9: Detection latency as measured experimentally and estimated mathematically for the case of network configuration 2. The black curve corresponds to the detection latency when the processing occurs on the vehicle itself without offloading via a network.



network configurations 1-3, the model's estimation is very close to the actual detection latency values obtained via the experiments. In the case of the very unreliable network (network configuration 4), the model is less accurate (off by 15-33%), mainly because it assumes that a network returns to its healthy state immediately after recovery and handshakes. In practice, some residual delays may occur in highly congested networks. Nevertheless, we have found that the model's accuracy in reasonably reliable networks is excellent and can be used to take offloading decisions (whether detection should run onboard or offloaded).

Both model and experimental evaluation agree that, from the perspective of mean detec-

Figure 5.10: Detection latency as measured experimentally and estimated mathematically for the case of network configuration 3. The black curve corresponds to the detection latency when the processing occurs on the vehicle itself without offloading via a network.

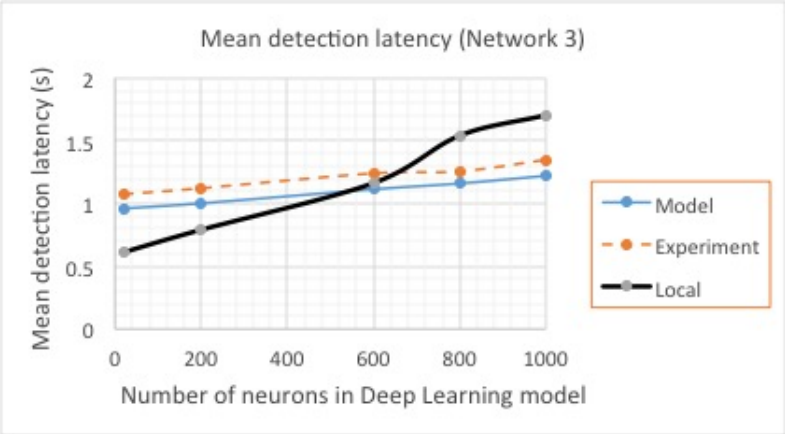


Figure 5.11: Detection latency as measured experimentally and estimated mathematically for the case of network configuration 4. The black curve corresponds to the detection latency when the processing occurs on the vehicle itself without offloading via a network.

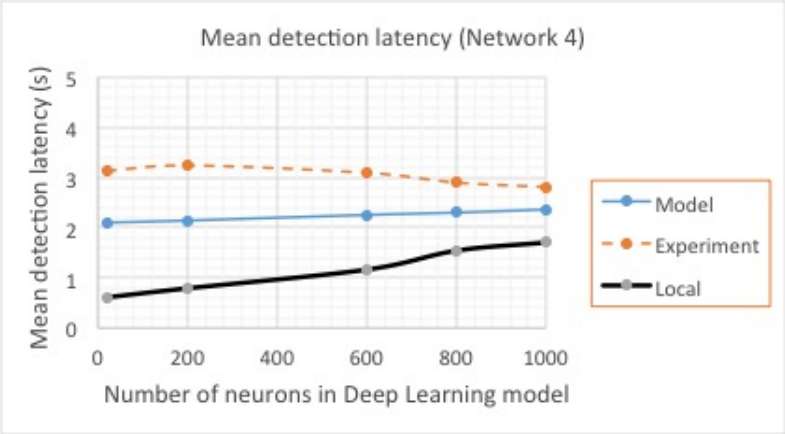
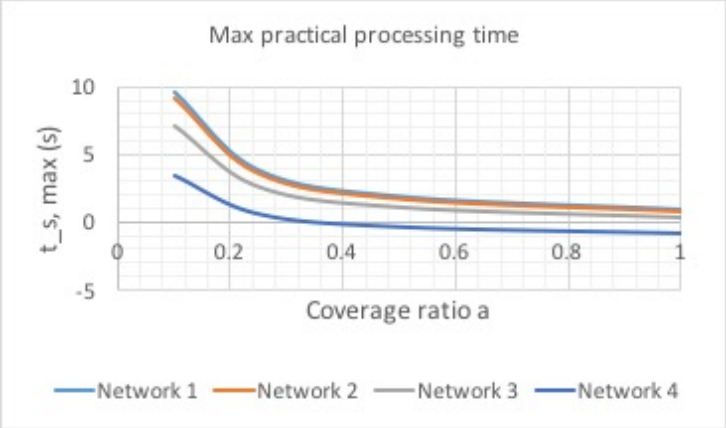


Figure 5.12: Max t_s against different values of a for the four network configurations



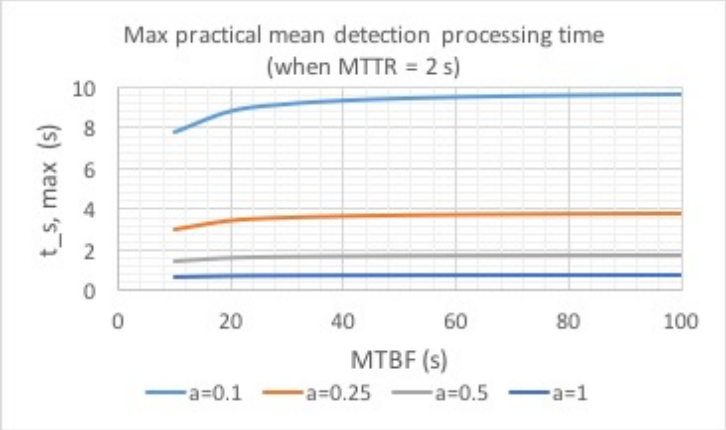
tion latency, offloading detection via networks 1 and 2 is preferable to running it onboard, if the deep learning architecture includes 200 neurons and above. This number increases to approximately 600 neurons for network 3. With the same criterion of reducing detection latency, it is never practical to offload detection via network 4 in any of the cases evaluated (between 20 and 1000 neurons).

5.5.2 Network model results

Having confirmed the relative accuracy of the model in Section 5.5.1, here, it is utilised to identify the maximum practical value for t_s in different conditions, as well as the estimated $\bar{t_l}$ in each network configuration specified in Table 5.3. Figure 5.12 summarises the effect on the maximum practical value for t_s of different values for a in the four network configurations.

We observe that out of the four network configurations utilised, only the fourth, which corresponds to the least reliable network, would be impractical for offloading the task of continuous deep learning based intrusion detection. This is because the maximum value for t_s would be negative, but t_s obviously needs to be a positive value, as it represents time. We also observe that, as expected, the lower the coverage ratio a, the greater the $t_{s,max}$. Of course, reducing the coverage ratio means that particularly short-duration attacks with no lasting cyber or physical impact may be missed by the detection process

Figure 5.13: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 2s$



altogether. Sections 5.5.2.1-5.5.2.4 provide the detailed results for each network.

5.5.2.1 Network 1 model results

Network 1 model represents the LAN-based server using the measurement from existing lab testbed. The detail configuration parameters are defined in Table 5.3. For each value of MTTR ξ = (2 s, 4 s, 6 s), we tested four different detection coverage ratios a = (0.1, 0.25, 0.5, 1). These results are depicted in Figures 5.13, 5.14, and 5.15. These line graphs compares the $t_{s,max}$ or Max t_s, measured in second, which are considered the tolerance level of processing time for detection system. Noticeably, the $t_{s,max}$ values are higher for a smaller value of coverage ratio (a) in all the cases.

In Figure 5.13, $t_{s,max}$ or Max t_s represents the maximum processing time that can be practical for the offloaded detection, it is effectively a measure of the minimum processing power that the setup can tolerate. Starting from Figure 5.13, it shows, as expected, that the lower the coverage ratio a the lower the processing power that is needed (as the higher the t_s that can be tolerated). Specifically, in the case of MTTR $\xi = 2$ s, $t_{s,max}$ starts at 0.63 s for $a = 1$ and reaches 7.8 s for $a = 0.1$, in the case of low MTBF. At MTBF $\theta = 100$, $t_{s,max}$ increases modestly to 0.81 s and 9.63 s for $a = 1$ and $a = 0.1$ respectively.

Figure 5.14 (MTTR $\xi = 4s$) shows a similar case to the previous Figure 5.13. As expected, with a low MTBF $\theta = 10$, the starting values of $t_{s,max}$ or the level of tolerance for

Figure 5.14: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 4s$

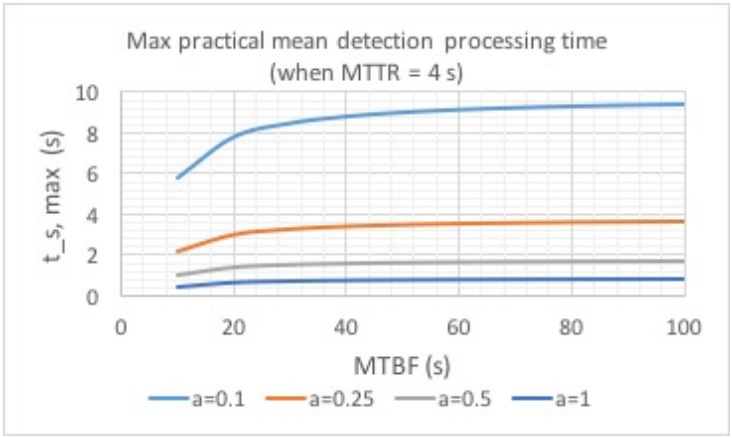
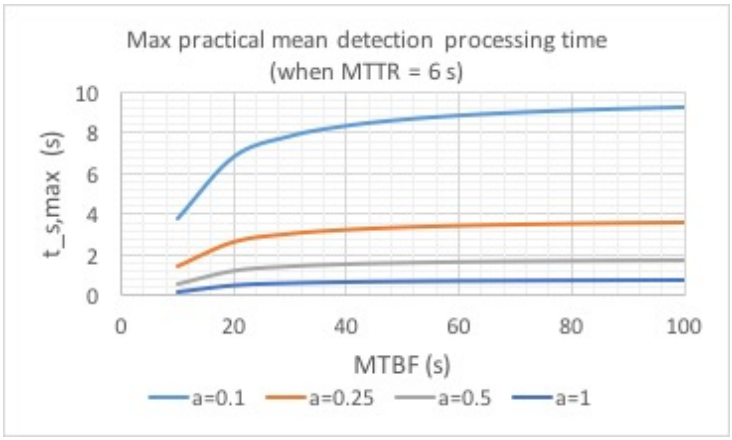


Figure 5.15: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 6s$

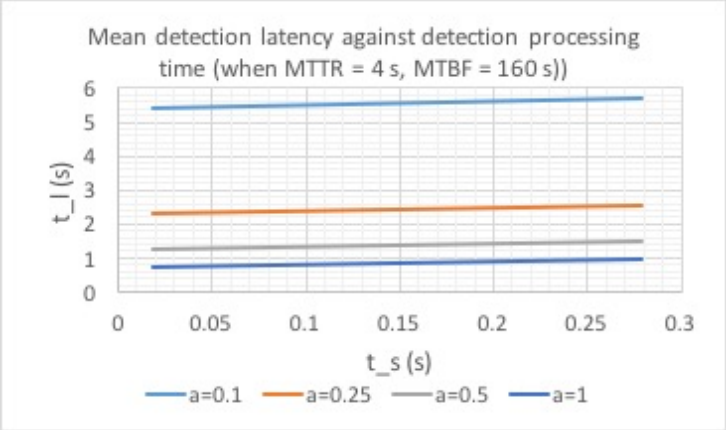


processing time is lower than previous case. $t_{s,max}$ starts at 0.42 s for $a = 1$ and 5.8 s for $a = 0.1$. $t_{s,max}$ in all four cases increase slightly quicker than the previous case. At MTBF ($\theta = 100$), $t_{s,max}$ are at 0.79 s and 9.42 s for $a = 1$ and $a = 0.1$ respectively.

Comparing Figure 5.15 to the two previous Figure 5.13 and 5.14, the $t_{s,max}$ lines are rising at a sharper rate. With MTTR $\xi = 6s$, $t_{s,max}$ starts out at at 0.22 s and 3.80 s for $a = 1$ and $a = 0.1$ accordingly. When MTBF θ is set 100, $t_{s,max}$ reaches 0.77 s and 9.22 s for the most and least coverage.

Figure 5.16 compares the mean detection latency $\bar{t_l}$, in second, against different values of detection processing time t_s in four detection coverage ratios a. The four lines shows a fairly similar trend of slightly steady linear increase. To begin, at $t_s = 0.0185$ which corresponds to the 20 hidden neuron setting, $\bar{t_l}$ is equal to 0.71 s for $a = 1$ and equal for 5.43 s for $a = 0.1$. With t_s reaches 0.2791 when the architecture has 1000 hidden neurons,

Figure 5.16: $\bar{t_l}$ against different values of t_s and a.



t_l becomes 0.97 s and 5.67 s for $a = 1$ and $a = 0.1$ accordingly.

5.5.2.2 Network 2 model results

Network 2 model represents an ideal cloud service where the parameters which are defined in Table 5.3. For each value of MTTR ξ = (2 s, 4 s, 6 s), we also modelled four different detection coverage ratios $a = (0.1, 0.25, 0.5, 1)$. These results are depicted in Figures 5.17, 5.18, and 5.19.

Since this is an ideal setting, the result patterns are similar to what we had for Network 1 in Section 5.5.2.1. These graphs estimate the $t_{s,max}$ against MTBF θ. Noticeably, the level tolerance $t_{s,max}$ are higher for a smaller value of coverage ratio (a) in the three MTTR cases.

In Figure 5.17, for the lowest MTBF value ($\theta = 10$), in the case of MTTR $\xi = 2$ s, $t_{s,max}$ is equal to at 0.59 s for $a = 1$ and reaches 8.0 s for $a = 0.1$. At the highest MTBF $\theta = 100$, $t_{s,max}$ rises modestly to 0.77 s and 9.59 s for $a = 1$ and $a = 0.1$ accordingly. So these $t_{s,max}$ values are slightly lower than our testbed's result in Section 5.5.2.1 Network 1 model results.

Figure 5.18 changes MTTR ξ to 4, it shows a similar case to the previous Figure 5.17. With a low MTBF $\theta = 10$, $t_{s,max}$ starts lower than MTTR $\xi = 2$ s case. $t_{s,max}$ starts at 0.39 s for $a = 1$ and 5.79 s for $a = 0.1$. At MTBF ($\theta = 100$), $t_{s,max}$ are at 0.76 s and 9.40

Figure 5.17: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 2s$

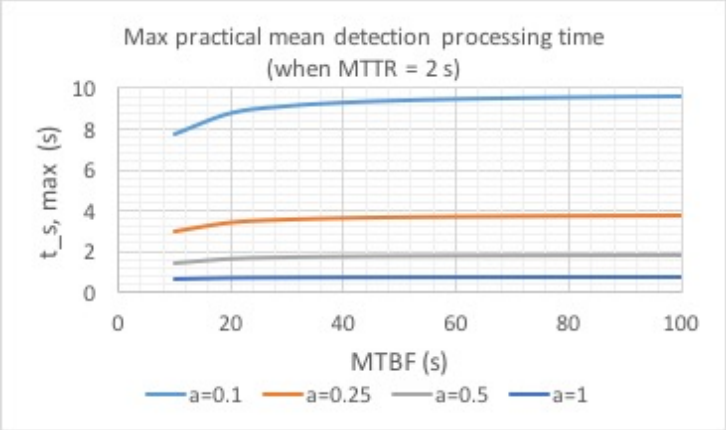
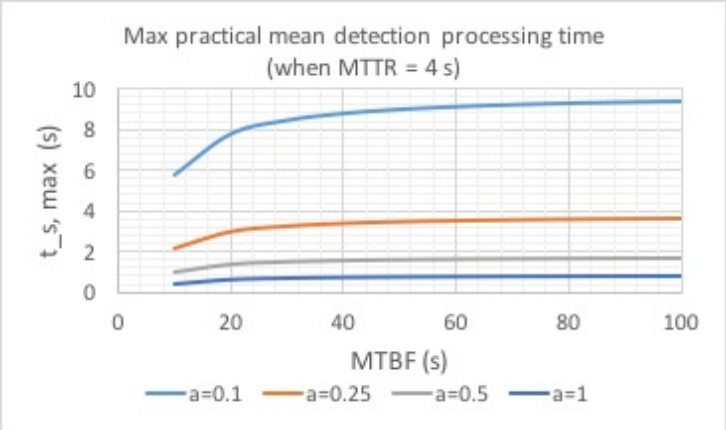


Figure 5.18: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 4s$



s for $a = 1$ and $a = 0.1$ accordingly. These values, again, are lower than our testbed's results.

Comparing Figure 5.19 to the two previous Figure 5.17 and 5.18, the $t_{s,max}$ values are increasing at a quicker rate. This case is configured with MTTR $\xi = 6s$, $t_{s,max}$ starts out at at 0.19 s and 3.75 s for $a = 1$ and $a = 0.1$ accordingly. When MTBF θ reaches 100, $t_{s,max}$ is then equal to 0.77 s and 9.22 s for the most and least coverage.

Figure 5.20 again compares the mean detection latency $\bar{t_l}$, in second, against different values of detection processing time t_s in four detection coverage ratios a. The other parameters MTTR and MTBF are set at 4 s and 60 s. The four lines show a similar increase of a linear line. To begin, at $t_s = 0.0185\ s$ which corresponds to 20 hidden neuron setting, $\bar{t_l}$ is equal to 0.78 s for $a = 1$ and equal for 5.88 s for $a = 0.1$. With t_s reaches 0.2791 s corresponding to 1000 hidden neurons, t_l becomes 1.04 s and 6.14 s for

Figure 5.19: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 6s$

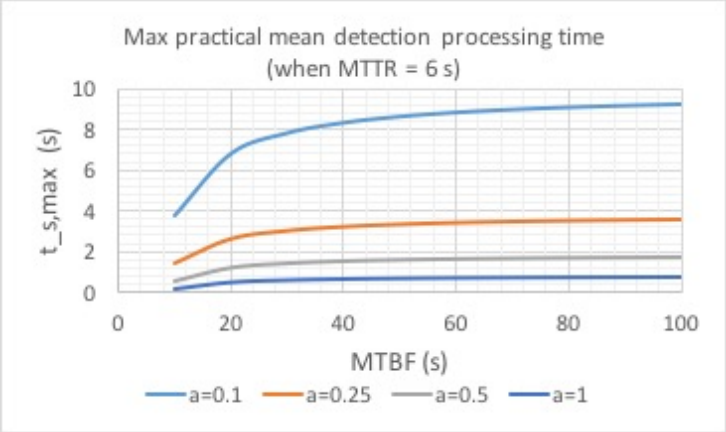
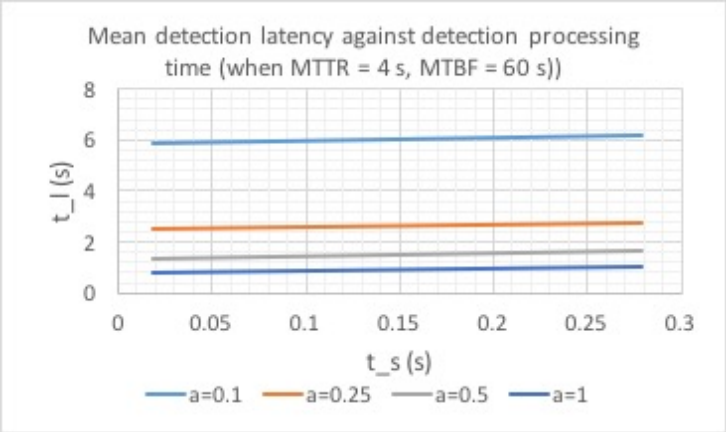


Figure 5.20: $\bar{t_l}$ against different values of t_s and a.



$a = 1$ and $a = 0.1$ respectively.

5.5.2.3 Network 3 model results

Network 3 model's parameters were configured to represents a typical 4g/3g network service, as in Table 5.3. We again run the model for each value of MTTR $\xi = (2\ s, 4\ s, 6\ s)$. With each MTTR, we modelled four different detection coverage scenarios with coverage ratio $a = (0.1,\ 0.25,\ 0.5,\ 1)$.

The tolerance level for processing time is a slightly worse (smaller) to what we had for Network 1 and 2 in Section 5.5.2.1 and 5.5.2.2. Here, the $t_{s,max}$, measured in second, is compared against MTBF θ in four coverage cases. The level tolerance $t_{s,max}$ are clearly showed higher for a smaller value of coverage ratio (a).

Figure 5.21: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 2s$

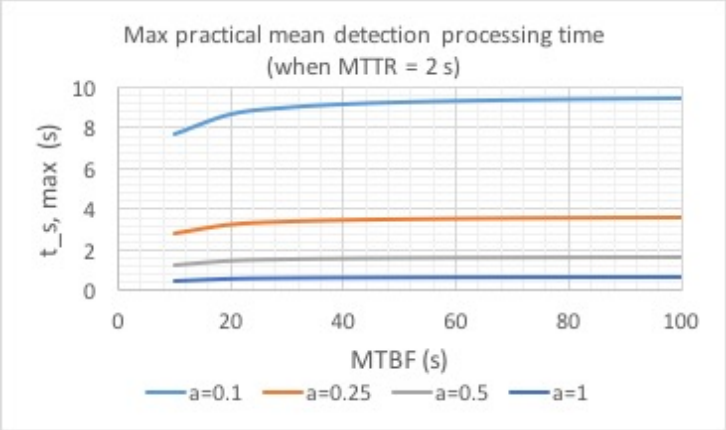
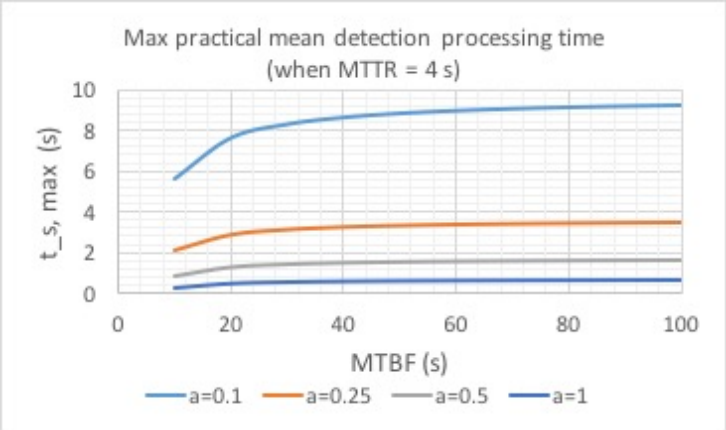


Figure 5.22: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 4s$



In Figure 5.21, the max practical mean detection processing time $t_{s,max}$ starts at 0.47 s, 1.27 s, 2.87 s and 7.67 s for $a = (1,\ 0.5,\ 0.25,\ 0.1)$ respectively, for the lowest MTBF value ($\theta = 10$), in the case of MTTR $\xi = 2\ s$. Then, the max practical value rises modestly to 0.65 s, 1.63s, 3.59 s and 9.47 s when MTBF θ reaches 100. These $t_{s,max}$ values are lower than the processing time when testing the algorithm in our testbed.

Figure 5.22 shows how $t_{s,max}$ changes with MTTR $\xi = 4\ s$. The graph looks similar to the previous Figure 5.21. When MTBF $\theta = 10$, $t_{s,max}$ starts with 0.27 s, 0.87 s, 2.07s and 5.65s, which are lower than MTTR $\xi = 2$ case. Then, $t_{s,max}$ rise more sharply to 0.63 s, 1.59 s, 3.51 s and 9.27 s for the according coverage ratios $a = (1, 0.5, 0.25, 0.1)$. Unsurprisingly, these $t_{s,max}$ values are smaller than Network 1 and 2 results.

When contrasting Figure 5.23 to the two previous Figure 5.21 and 5.22, the $t_{s,max}$ values are increasing at a quicker rate, when MTTR ξ is configured to be 6 s. The max practical

Figure 5.23: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 6s$

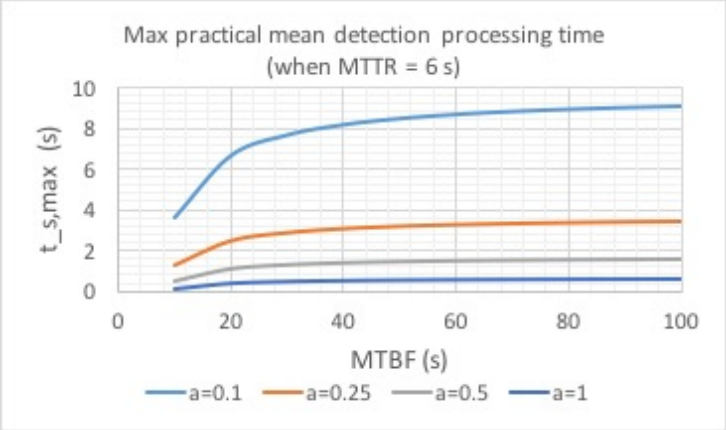
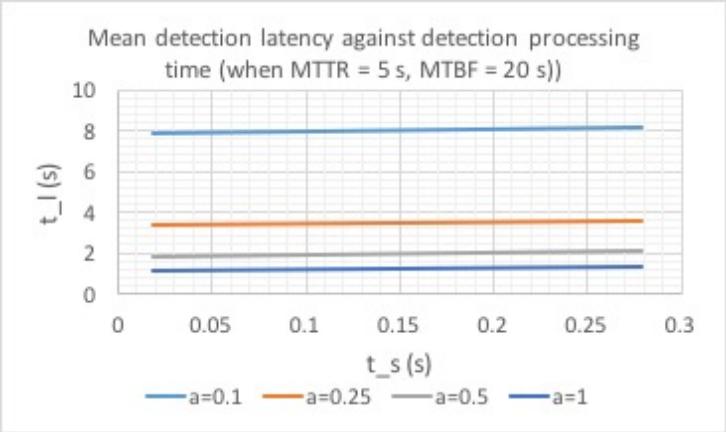


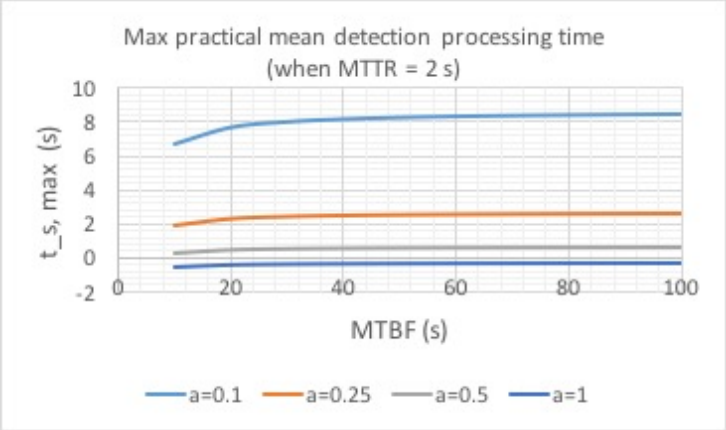
Figure 5.24: $\bar{t_l}$ against different values of t_s and a.



mean detection processing time $t_{s,max}$ starts out at at 0.06 s, 0.46 s, 1.24 s and 3.58 s, which mean the power processing is required to be stronger here. As MTBF $\theta = 100$, $t_{s,max}$ indicates a better tolerance at 0.63 s, 1.59 s, 3.51 s and 9.27 s for the four coverage cases.

Figure 5.24 illustrates the mean detection latency $\bar{t_l}$, in second, against values of detection processing time t_s according to number of hidden neurons in the detection architecture. $\bar{t_s}$ is calculated for four detection coverage ratios a, and when MTTR $\xi = 5\ s$ and MTBF $\theta = 20\ s$. The graph shows four linear-looking lines. With t_s starts 0.0185, $\bar{t_l}$ is equal to 1.10 s, 1.85 s, 3.36 s, and 7.87 s for $a = (1,\ 0.5,\ 0.25,\ 0.1)$ accordingly. When t_s reaches 0.2791 as for a 1000 hidden neuron architecture, t_l becomes 1.35s, 2.11 s, 3.61s, and 8.13s for the four coverage cases. These values are significantly higher than in Network 1 and 2.

Figure 5.25: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 2s$



5.5.2.4 Network 4 model results

Network 4 model's parameters represents a highly unstable network with high packet loss, frequent connectivity failures and latency, as configured in Table 5.3. Network 4 model was evaluated for MTTR $\xi = (2\ s, 4\ s, 6\ s)$. With each MTTR, we again calculated the max practical mean detection processing time in four different detection coverage scenarios $a = (0.1,\ 0.25,\ 0.5,\ 1)$.

Here, we see the tolerance of processing time dipping below 0 seconds which shows the impracticalities for such situations. In general, $t_{s,max}$ here is a much worse (smaller) to what we had the three previous network models, Section 5.5.2.1, 5.5.2.2 and 5.5.2.3. One last time, The $t_{s,max}$, measured in second, is compared against MTBF θ in four coverage cases.

In the case of MTTR $\xi = 2\ s$, Figure 5.25 shows that the max practical mean detection processing time $t_{s,max}$ line is below 0 entirely for a full detection coverage $a = 1$. A very strong processing power is required for half detection coverage $a = 0.5$ with $t_(s, max) = 0.27\ s$. $t_{s,max}$ are at 1.88 s and 6.68 s for $a = 0.25 and 0.1$ accordingly. With a higher MTBF value ($\theta = 100$), then, $t_{s,max}$ allows better tolerance at 0.63 s, 2.59 s, and 8.47 s for $a = (0.5, 0.25 and 0.1)$. It is noted that 100% detection coverage is impossible for this network model.

When MTTR $\xi = 4\ s$, Figure 5.26 shows an even worst result than Figure 5.25. The

Figure 5.26: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 4s$

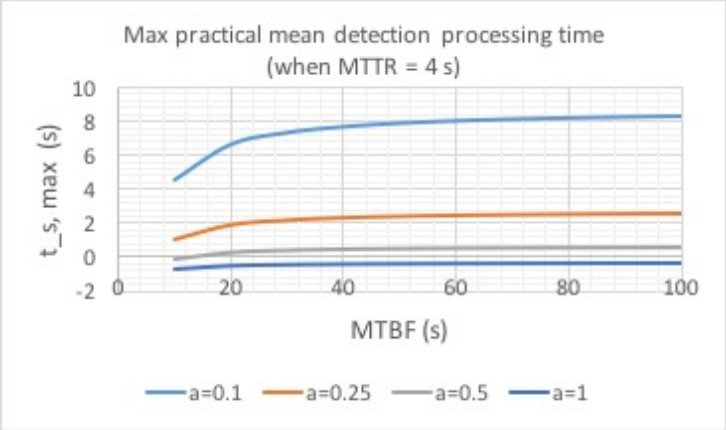
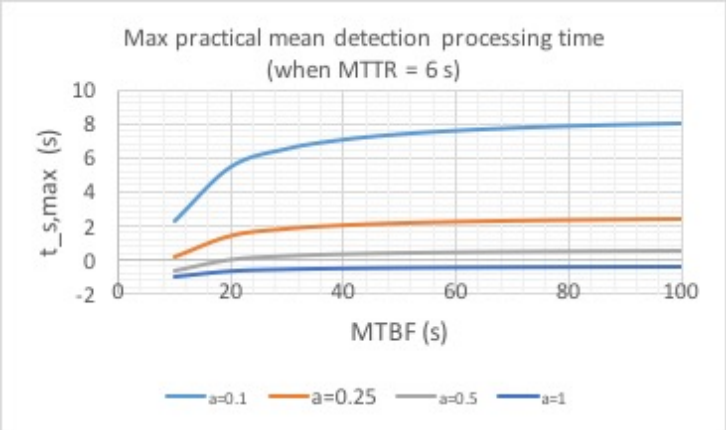


Figure 5.27: Max t_s against different values of MTBF (θ) and a for MTTR $\xi = 6s$



max practical mean detection processing time $t_{s,max}$ are nearly or below 0 for two out of four coverage cases at MTBF $\theta = 10$. A very strong processing power is required for half detection coverage $a = 0.5$ with $t_(s, max) = 0.59$ s at the best MTBF $\theta = 100$. Also with MTBF value $\theta = 100$, $t_{s,max}$ permits tolerance at 2.51 s, and 8.27 s for $a = 0.25,\ 0.1$. 100% detection coverage is impossible for this network model, and so is a part of 50% coverage.

When MTTR ξ reaches 6 s, Figure 5.27 shows the $t_{s,max}$ is nearly or below 0 for three out of four coverage cases at the first MTBF setting $\theta = 10$. A very strong processing power is required for half detection coverage $a = 0.5$ with $t_(s, max) = 0.55$ s at the last MTBF $\theta = 100$. When MTBF $\theta = 100$, the tolerance value $t_{s,max}$ is at 2.42 s, and 8.04 s for $a = 0.25,\ 0.1$ accordingly. Again, 100% detection coverage is impossible for this network model, and 50% detection coverage is also a challenge.

Figure 5.28: $\bar{t_l}$ against different values of t_s and a.

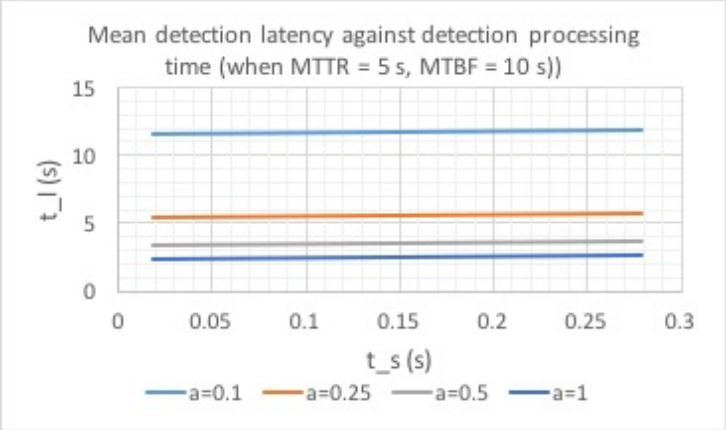


Figure 5.28 compares the mean detection latency $\bar{t_l}$, in second, against values of detection processing time t_s when MTTR $\xi = 5\ s$ and MTBF $\theta = 10\ s$. In general, these values are fairly high compared to the collection interval T. The graph show four linear lines. When t_s starts at 0.0185 s, $\bar{t_l}$ is equal to 2.36 s, 3.38 s, 5.42 s, and 11.55 s for $a = (1,\ 0.5,\ 0.25,\ 0.1)$ accordingly. When t_s is at 0.2791 s, t_l becomes 2.62 s, 3.64 s, 5.68 s, and 11.81 s for the four coverage cases. These values are significantly highers than what were shown in Network 1 and 2 models.

5.6 Conclusion

Deep learning training and detection phases with more intense multiple linear and non-linear computations might create a detection latency for limited resource systems. However, with the technological advances of deep learning research and enhanced capacity in processing systems, it becomes more and more practical to apply this technique for mobile cyber-physical systems. Offloading this computation task further strengthens the intrusion detection system. We have shown that our approach is already practical for different cyber attacks and different network configurations.

Chapter 6

Conclusion

6.1 Summary of thesis achievements

Mobile cyber-physical systems, such as automobiles, drones and robotic vehicles, are gradually becoming attractive targets for cyber attacks. Instead of assuming that a preventive measure, such as cryptography, would be sufficient for all attacks, we have considered the case that some will go through. However, intrusion detection systems built for conventional computer systems tend to be unsuitable. They can be too demanding for resource-restricted cyber-physical systems or too inaccurate due to the lack of real-world data on actual attack behaviours. The following is a brief summary of the key achievements towards addressing the problem of cyber security for mobile cyber-physical systems. These also answer the research questions raised in Chapter 1 on the importance of physical features and roles of machine learning and the solution for improving accuracy and latency in detecting cyber attacks against robotic vehicles.

- A testbed has been developed, including a small robotic vehicle, a laptop controlling it remotely and accompanying software for the collection of the relevant data onboard the vehicle. As the key characteristic that differentiates cyber-physical systems from conventional computer systems is their behaviour in physical space, a

series of cyber attacks have been conducted on the testbed and **different types of physical impact documented**, from change in speed and intermittent stops, to physical jittering and increased power consumption.

- A simple machine learning based system has been developed which was able to run on board the vehicle and which uses decision trees to detect denial of service, command injection and malware attacks. We used this approach to **demonstrate that taking into account physical input features in conjunction with traditional cyber input features can markedly reduce the false positive rate and increase the overall accuracy of the detection**.

- A deep learning architecture has been developed based on recurrent neural networks and long short-term memory to produce more powerful intrusion detection than is currently possible with traditional statistical machine learning techniques. **The deep learning based prototype was able to achieve higher detection accuracy than the earlier decision trees approach, as well as five other popular machine learning techniques that it was compared against**.

- As the timeliness of detection of cyber-physical attacks is particularly important for vehicles and deep learning can be very resource-demanding, we have **evaluated the practicality of offloading it to a remote infrastructure in terms of the overall detection latency incurred**. For this, a mathematical model has been developed, which was validated experimentally. This has shown that offloaded deep learning becomes impractical only when the network that supports it is particularly unreliable, with high packet losses and extremely frequent failures.

6.2 Critical discussion

In relation to the research approach followed, the choice of having a real testbed was very successful because it enabled us to practically evaluate techniques within real world scenarios. However, it brought a considerable challenge in effort and time in collecting

real-time data in a synchronised manner. In many cases, it was not always clear if an error occurred because of hardware challenge or an algorithmic mistake or environmental conditions. This added time delays but helped me reach a better understanding of each component's characteristics.

It is also important to note that deep learning required up-front time to learn because it was a new paradigm where there was a lack of instruction and standard libraries available. It also required much greater processing power than standard machine learning techniques, which was appropriate with the robotic vehicle. It was not uncommon to need over 3 days to complete training for a model. On the upside, it brought huge benefits by increasing the accuracy of cyber attack detection significantly.

The final phase of the research was to resolve the issue of restricted resources on the testbed itself. This was achieved by offloading the heavy computation load to a cloud environment. This reduced the computing time but it strongly depended on the quality of the network connection between the testbed and the cloud. With an ideal network connection, offloading techniques could offer us both higher accuracy and a reasonable detection latency.

6.3 Applications

The aim of this work has always been to provide insights and tools that will be useful beyond the narrow scope of the particular robotic vehicle used as the testbed and perhaps beyond robotic and vehicular systems, to more generally in cyber-physical systems. For this reason, all the components chosen were commodity electronics and personal computing components that are easily accessible and utilised in a large-number of robotic and other cyber-physical and Internet of Things (IoT) systems. In addition, the scenarios experimented with were related to general families of cyber attacks typically encountered in cyber space (i.e., denial of service, command injection and malware attacks) rather than very specific attacks exploiting device-specific vulnerabilities. The intrusion detection

methods, both the lightweight and the resource-demanding ones, are feature-agnostic, in the sense that they can be used to incorporate any cyber or physical feature that is appropriate for a given device. In fact, the offloading mechanism presented in this thesis in detail can allow access to resource-demanding intrusion detection techniques, for instance using complex deep learning architectures, as long as there is access to a commodity remote server, private cloud or even public cloud, such as Amazon's. The mathematical model provided estimates whether and when this offloading would be practical.

With this emphasis on applicability beyond the systems experimented with in this thesis, it is reasonable to expect that the techniques developed and presented here can also be used to:

- The security of the resource-constrained Internet of Things devices, e.g., in the context of home automation or body area networks.
- Other security tasks that would benefit from offloading, such as network, application or user filtering.
- Other types of cyber-physical systems, where a physical impact is of high criticality, such as the smart grid and industrial control systems in general.
- Areas of application that make use of robotic vehicles, such as physical security surveillance and civilian and military emergency response situations, where the safe and secure operation of a remotely controlled or autonomous vehicle can be highly useful.

6.4 Future work

The work conducted towards achieving cyber-physical intrusion detection for vehicles has opened up several possibilities for future research. A brief summary of each identified possibility is described in the following subsections.

6.4.1 Extending the scope of this work through more attack scenarios, features and a behaviour-based detection approach

Intrusion detection for cyber-physical systems is a relatively new area of research. It has been explored to some extent for industrial control systems, but remains at an early stage for mobile cyber-physical systems such as robotic vehicles. Through experimentation, it has been observed that different attacks have different impacts on the operation of the robotic vehicle and this includes both its cyber (network, CPU, disk data) and physical behaviour (speed, vibration, power consumption). This presents an opportunity, as by leveraging both types of features, one can improve the accuracy of the detection of an attack. This has been validated for an intrusion detection approach based on machine learning and deep learning techniques.

The next step is to extend the scope of the experiments on different attack types [Lang et al., 2007], such as communication jamming, relay attacks, and sensor attack. We are also working towards testing the hypothesis that the addition of physical input features can improve the accuracy and detection latency of behaviour-based detection approaches, which are more suitable than knowledge-based approaches for zero-day cyber-physical attacks.

6.4.2 Response mechanism to accompany intrusion detection

Different detection mechanisms offer different types of output, which can trigger different types of automated responses. Response mechanism was not within the scope of this research project, but can easily be a natural extension. Depending on the confidence of the detection mechanism in the result or depending on a specific measure of likelihood of an ongoing attack, there could be different pre-defined responses triggered, such as stopping completely for likelihood of a serious attack having been detected or slowing down if an attack is detected but with low confidence, or even triggering no physical response at all but only reporting to a user if the attack suspected can have no physical

impact. So, determining a response mechanism is one more area where differentiating between cyber and physical feature monitoring can be very useful.

6.4.3 A cloud-based "security guard" for robotic vehicles

Deep learning introduces a detection latency that may make it less practical for protecting against some types of attack. However, deep learning is the most rapidly advancing area of machine learning, demonstrating impressive performance improvement every year. At the same time, introducing a remote computation element, such as a remote server or cloud, to the security of a vehicle means that the latter may inherit some of the vulnerabilities of the former. Again, cloud security is rapidly improving, both thanks to technological advances and to the business necessity of the matter. It has been shown that this approach is already practical for attacks of differing nature. By positioning it where distributed and cloud computing meets deep learning, it is expected that this will further benefit from the parallel ongoing research in these two fields. Furthermore, using a remote infrastructure acting as the "security guard" of multiple vehicles means that information about an attack collected on one vehicle can be re-used to protect other vehicles against the same attack in the future. This "security guard" may be located at a public cloud service, such as Amazon, and accessed via the Internet or may be local, located on one of the robots in a robotic swarm. Such a scenario would be particularly practical for robotic setups where a group of lightweight robots may belong to a single, more powerful mothership.

6.4.4 Distributed cyber-physical intrusion detection

In other types of cyber-physical systems, such as in geographically dispersed industrial control systems, it is common to use distributed intrusion detection, where both data collection and reasoning may be performed in a manner that is distributed. As discussed in [Loukas, 2015], some simple attacks may be relatively easy to detect using an intrusion detection system located on an individual node. An example may be a single smart meter

in the context of the smart grid. Attacks that are more complex may be more difficult to detect. They may need information that is aggregated from multiple smart meters [Zhang et al., 2011]. In this work, the focus has been on protecting a single vehicle. So, data collection for intrusion detection is run only on that one vehicle. However, vehicles often operate in groups and in environments that can be collaborative. The first example of collaborative intrusion detection for autonomous vehicles has been studied theoretically in [Fagiolini et al., 2008], where, if a vehicle misbehaves, it is other vehicles on the same road or in the same group that notice it and collectively reach a consensus on whether it is dangerous to them. This consensus-based approach has not been tested in actual vehicles or for a combination of cyber and physical features. This is an interesting problem to pursue, especially as autonomous vehicles are becoming a reality.

6.4.5 Evaluating the security and energy cost of offloaded intrusion detection

In addition, it is worth noting that as the overall purpose of this approach is to improve the security of the vehicle, one also needs to consider any additional security threats introduced by the cloud-vehicle interaction. The approach adopted in this thesis is to utilise HTTPS, which provides basic security of data sharing between the vehicle and the cloud, assuming that the latter is trusted. In the future, the scope can be extended to so-called "unfaithful" clouds. For these, one can implement one of the several excellent approaches for secure computation offloading that have been proposed in the literature [Gennaro et al., 2010, Chen et al., 2015], but it would also be important to evaluate their overhead in terms of detection latency and energy cost. A first investigation towards this direction [Loukas et al., 2016] has shown that the robotic vehicle saves energy in all practical configurations of offloading deep learning based processing. Future research may expand on these findings and may lead to more energy-aware intrusion detection. For example, the complexity of the deep learning architecture (e.g., the number of hidden neurons used) and the period of data collection on the vehicle may be reconfigured dynamically based

on the energy state of the vehicle at any point in time.

6.5 Final remarks

Mobile cyber-physical systems are expected to become increasingly important as the rising number of automobiles, drones and robotic vehicles, and their ubiquitous application from personal appliances to military missions, from underwater, on-ground, and in the airspace. Unfortunately, these systems and applications are generating new vulnerabilities in physical space, which allows cyber attacks against them. The purpose of the work presented in this thesis was to analyse, design and model methods and network setting to detect cyber attacks with a high accuracy and with an acceptable latency for these time-critical systems. Continuous advancements in terms of hardware and software for vehicles create opportunities for more advanced methods for protecting them against cyber attacks. With the lightweight statistical learning technique and the heavier deep learning based technique, this project has provided proof of concept solutions for both resource-restricted and more processing-powerful vehicles.

References

[Amin et al., 2009] Amin, S., Cárdenas, A. A., and Sastry, S. S. (2009). Safe and secure networked control systems under denial-of-service attacks. In *Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control*, HSCC '09, pages 31–45, Berlin, Heidelberg. Springer-Verlag.

[Anthony et al., 2009] Anthony, R., Chen, D., Törngren, M., Scholle, D., Sanfridson, M., Rettberg, A., Naseer, T., Persson, M., and Feng, L. (2009). Autonomic middleware for automotive embedded systems. In *Autonomic Communication*, pages 169–210. Springer.

[Anthony et al., 2008] Anthony, R., Ward, P., Chen, D., Hawthorne, J., Mariusz, P., Rettberg, A., and Törngren, M. (2008). A middleware approach to dynamically configurable automotive embedded systems. In *ISVCS 2008: The First Annual International ICST Symposium on Vehicular Computing Systems*. EUDL-European Union Digital Library.

[Barbounis et al., 2006] Barbounis, T. G., Theocharis, J. B., Alexiadis, M. C., and Dokopoulos, P. S. (2006). Long-term wind speed and power forecasting using local recurrent neural network models. *IEEE Transactions on Energy Conversion*, 21(1):273–284.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Bezemskij et al., 2016a] Bezemskij, A., Anthony, R. J., Loukas, G., and Gan, D. (2016a). Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. In *Eighth International Symposium on Cyberspace Safety and Security*.

[Bezemskij et al., 2016b] Bezemskij, A., Anthony, R. J., Loukas, G., and Gan, D. (2016b). Threat evaluation based on automatic sensor signal characterisation and anomaly detection. In *The Twelfth International Conference on Autonomic and Autonomous Systems (ICAS 2016)*. IARIA.

[Bicchi et al., 2008] Bicchi, A., Fagiolini, A., Dini, G., and Savino, I. M. (2008). Tolerating, malicious monitors in detecting misbehaving robots. In *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*, pages 109–114. IEEE.

[Birnbaum, 2015] Birnbaum, Z. (2015). *Behavior based analytics for securing cyber-physical systems*. State University of New York at Binghamton.

[Birnbaum et al., 2014] Birnbaum, Z., Dolgikh, A., Skormin, V., O'Brien, E., and Muller, D. (2014). Unmanned aerial vehicle security using recursive parameter estimation. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 692–702. IEEE.

[Blasch et al., 2017] Blasch, E., Kadar, I., Grewe, L. L., Brooks, R., Yu, W., Kwasinski, A., Thomopoulos, S., Salerno, J., and Qi, H. (2017). Panel summary of cyber-physical systems (cps) and internet of things (iot) opportunities with information fusion. In *SPIE Defense+ Security*, pages 102000O–102000O. International Society for Optics and Photonics.

[Bonaci et al., 2015a] Bonaci, T., Herron, J., Yusuf, T., Yan, J., Kohno, T., and Chizeck, H. J. (2015a). To make a robot secure: an experimental analysis of cyber security threats against teleoperated surgical robots. *arXiv preprint arXiv:1504.04339*, pages 1–11.

[Bonaci et al., 2015b] Bonaci, T., Herron, J., Yusuf, T., Yan, J., Kohno, T., and Chizeck, H. J. (2015b). To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots. *arXiv preprint arXiv:1504.04339*.

[Canziani et al., 2016] Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

[Cárdenas et al., 2008] Cárdenas, A. A., Amin, S., and Sastry, S. (2008). Research challenges for the security of control systems. In *HotSec*.

[Cardenas et al., 2008] Cardenas, A. A., Amin, S., and Sastry, S. (2008). Secure control: Towards survivable cyber-physical systems. *System*, 1(a2):a3.

[Carson et al., 2016] Carson, N., Martin, S. M., Starling, J., and Bevly, D. M. (2016). Gps spoofing detection and mitigation using cooperative adaptive cruise control system. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 1091–1096. IEEE.

[Checkoway et al., 2011] Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *Usenix Security Symposium*.

[Chen, 2010] Chen, T. M. (2010). Stuxnet, the real start of cyber warfare?[editor's note]. *IEEE Network*, 24(6):2–3.

[Chen et al., 2015] Chen, X., Huang, X., Li, J., Ma, J., Lou, W., and Wong, D. (2015). Rocking drones with intentional sound noise on gyroscopic sensors. *Transactions on Information Forensics and Security*, 10(1):69–78.

[Chen et al., 2011] Chen, Y.-J., Shih, J.-S., and Cheng, S.-T. (2011). A cyber-physical integrated security framework with fuzzy logic assessment for cultural heritages. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1843–1847. IEEE.

[Chin and Chuang, 2015] Chin, T.-L. and Chuang, W.-C. (2015). Latency of collaborative target detection for surveillance sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 26(2):467–477.

[Chollet, 2015] Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

[Chuang and Lee, 2011] Chuang, M.-C. and Lee, J.-F. (2011). Ppas: A privacy preservation authentication scheme for vehicle-to-infrastructure communication networks. In *International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 1509–1512. IEEE.

[Cole, 2012] Cole, C. (1 January 2012). The drone war briefing. [Online; accessed 18-Oct-2016].

[Collectl, 2015] Collectl (2015). Collectl. [Online; accessed 16-May-2015].

[Combs, 2007] Combs, G. e. a. (2007). Wireshark.

[De Cerchio and Riley, 2012] De Cerchio, R. and Riley, C. (2012). Aircraft systems cyber security. In *2012 Integrated Communications, Navigation and Surveillance Conference*.

[Dierks, 2008] Dierks, T. (2008). The transport layer security (tls) protocol version 1.2.

[Dietterich, 2002] Dietterich, T. G. (2002). Machine learning for sequential data: A review. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 15–30. Springer.

[Dillingham, 2012] Dillingham, G. L. (2012). Unmanned aircraft systems: Measuring progress and addressing potential privacy concerns would facilitate integration into the national airspace system. *US Government Accountability Office, Washington, DC, Rep. GAO-12-981*.

[Dong et al., 2016] Dong, Z., Kane, K., and Camp, L. (2016). Detection of rogue certificates from trusted certificate authorities using deep neural networks. *Transactions on Privacy and Security (TOPS)*, 19(2):5.

[Du et al., 2015] Du, Y., Wang, W., and Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1110–1118.

[Eze et al., 2012] Eze, T. O., Anthony, R. J., Walshaw, C., Soper, A., et al. (2012). A new architecture for trustworthy autonomic systems. *Curran Associates, Inc.*

[Fagiolini et al., 2009] Fagiolini, A., Babboni, F., and Bicchi, A. (2009). Dynamic distributed intrusion detection for secure multi-robot systems. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2723–2728. IEEE.

[Fagiolini et al., 2014] Fagiolini, A., Dini, G., and Bicchi, A. (2014). Distributed intrusion detection for the security of industrial cooperative robotic systems. In *World Congress*, volume 19(1), pages 7610–7615.

[Fagiolini et al., 2008] Fagiolini, A., Pellinacci, M., Valenti, G., Dini, G., and Bicchi, A. (2008). Consensus-based distributed intrusion detection for multi-robot systems. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 120–127. IEEE.

[Felix et al., 2014] Felix, R., Economou, J., and Knowles, K. (2014). Uas behaviour and consistency monitoring system for countering cyber security threats. Technical report, SAE Technical Paper.

[Fernndez-Delgado et al., 2014] Fernndez-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *J. Mach. Learn. Res*, 15(1):3133–3181.

[Filippoupolitis et al., 2014] Filippoupolitis, A., Loukas, G., and Kapetanakis, S. (2014). Towards real-time profiling of human attackers and bot detection. In *Proceedings of 7th International Conference on Cybercrime Forensics Education and Training (CFET)*.

[Gamage et al., 2011] Gamage, T. T., Roth, T. P., and McMillin, B. M. (2011). Confidentiality preserving security properties for cyber-physical systems. In *2011 IEEE 35th Annual Computer Software and Applications Conference*, pages 28–37. IEEE.

[Gao et al., 2010] Gao, W., Morris, T., Reaves, B., and Richey, D. (2010). On SCADA control system command and response injection and intrusion detection. In *eCrime Researchers Summit (eCrime), 2010*, pages 1–9.

[Gasser, 1988] Gasser, M. (1988). *Building a secure computer system*. Van Nostrand Reinhold Company New York.

[Gelenbe et al., 2005] Gelenbe, E., Gellman, M., and Loukas, G. (2005). An autonomic approach to denial of service defence. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pages 537–541. IEEE.

[Gelenbe et al., 2012] Gelenbe, E., Gorbil, G., and Wu, F.-J. (2012). Emergency cyber-physical-human systems. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–7. IEEE.

[Gelenbe and Loukas, 2007] Gelenbe, E. and Loukas, G. (2007). A self-aware approach to denial of service defence. *Comput. Netw.*, 51(5):1299–1314.

[Gennaro et al., 2010] Gennaro, R., Gentry, C., and Parno, B. (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in CryptologyCRYPTO 2010*, pages 465–482.

[Gorman et al., 2009] Gorman, S., Dreazen, Y. J., and Cole, A. (2009). Insurgents hack u.s. drones.

[Griffiths, 2014] Griffiths, J. (July 17, 2014). Zhejiang University team scoops 10,600 Yuan for hacking into Tesla Model S. South China Morning Post.

[Gurulian et al., 2016] Gurulian, I., Shepherd, C., Markantonakis, K., Akram, R. N., and Mayes, K. (2016). When theory and reality collide: Demystifying the effectiveness of ambient sensing for nfc-based proximity detection by applying relay attack data. *arXiv preprint arXiv:1605.00425*.

[Hardy et al., 2016] Hardy, W., Chen, L., Hou, S., Ye, Y., and Li, X. (2016). Dl4md: A deep learning framework for intelligent malware detection. In *Proceedings of the Inter-*

national Conference on Data Mining (DMIN), page 61. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[Hasan et al., 2014] Hasan, M. A. M., Nasser, M., Pal, B., and Ahmad, S. (2014). Support vector machine and random forest modeling for intrusion detection system (IDS). *Journal of Intelligent Learning Systems and Applications*, 6(1):45.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Houmansadr et al., 2011] Houmansadr, A., Zonouz, S., and Berthier, R. (2011). A cloud-based intrusion detection and response system for mobile phones. In *IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops*, pages 31–32. IEEE.

[Hu et al., 2012] Hu, G., Tay, W., and Wen, Y. (2012). Cloud robotics: architecture, challenges and applications. *Network*, 26(3):21–28.

[Hui and Cao, 2010] Hui, L. and Cao, Y. (2010). Research intrusion detection techniques from the perspective of machine learning. *2010 International Conference on MultiMedia and Information Technology, MMIT 2010*, 1:166–168.

[Jafarnia-Jahromi et al., 2012] Jafarnia-Jahromi, A., Lin, T., Broumandan, A., Nielsen, J., and Lachapelle, G. (2012). Detection and mitigation of spoofing attacks on a vector-based tracking gps receiver. *ION ITM*.

[Jakobson, 2011] Jakobson, G. (2011). Mission cyber security situation assessment using impact dependency graphs. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE.

[Javaid et al., 2016] Javaid, A. Y., Niyaz, Q., Sun, W., and Alam, M. (2016). A deep learning approach for network intrusion detection system. In *Proceedings of the 9th*

EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) on 9th, pages 21–26. ICST(Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Javaid et al., 2012] Javaid, A. Y., Sun, W., Devabhaktuni, V. K., and Alam, M. (2012). Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 585–590. IEEE.

[Jennings, 2014] Jennings, G. (2014). Iran claims to have flown reverse-engineered us stealth uav. [Online; accessed 11-July-2015] London - IHS Jane's Defence Weekly.

[Jensen et al., 2009] Jensen, M., Gruschka, N., and Herkenhöner, R. (2009). A survey of attacks on web services. *Computer Science-Research and Development*, 24(4):185–197.

[Jitbit, 2016] Jitbit (2016). Jitbit macro recorder.

[Junejo and Goh, 2016] Junejo, K. N. and Goh, J. (2016). Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 34–43. ACM.

[Kang, 2016] Kang, M. J.and Kang, J. W. (2016). Intrusion detection system using deep neural network for in-vehicle network security. *PloS One*, 11(6).

[Kerns et al., 2014] Kerns, A. J., Shepard, D. P., Bhatti, J. A., and Humphreys, T. E. (2014). Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31:617–636.

[Kim et al., 2012] Kim, A., Wampler, B., Goppert, J., Hwang, I., and Aldridge, H. (2012). Cyber attack vulnerabilities analysis for unmanned aerial vehicles. *Infotech@ Aerospace*.

[Kim et al., 2014] Kim, G., Lee, S., and Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700.

[Kim et al., 2016] Kim, J., Kim, J., Thu, H. L. T., and Kim, H. (2016). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE.

[Kohno, 2012] Kohno, T. (2012). Security for cyber-physical systems: case studies with medical devices, robots, and automobiles. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 99–100. ACM.

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

[Kotsiantis et al., 2007] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

[Kuhn et al., 2014] Kuhn, M., Steve, W., and Coulter, N. (2014). Package C50. [Online; accessed 16-May-2015].

[Kumar and Lu, 2010] Kumar, K. and Lu, Y. (2010). Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56.

[Kuzmanovic and Knightly, 2003] Kuzmanovic, A. and Knightly, E. W. (2003). Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 75–86. ACM.

[Lane and Brodley, 1997] Lane, T. and Brodley, C. (1997). An application of machine learning to anomaly detection. *Proceedings of the 20th National Information Systems Security Conf erence*, pages 366–377.

[Lang et al., 2007] Lang, A., Dittmann, J., Kiltz, S., and Hoppe, T. (2007). Future perspectives: The car and its ip-address–a potential safety and security risk assessment. In *Computer Safety, Reliability, and Security*, pages 40–53. Springer.

[Langner, 2013] Langner, R. (2013). To kill a centrifuge: A technical analysis of what stuxnets creators tried to achieve. *Online: http://www. langner. com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge. pdf*.

[Latka, 1994] Latka, D. S. (1994). Resynchronizing transmitters to receivers for secure vehicle entry using cryptography or rolling code. US Patent 5,369,706.

[Lee, 2008] Lee, E. (2008). Cyber physical systems: Design challenges. In *11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369.

[Li et al., 2016] Li, Y., Lan, C., Xing, J., Zeng, W., Yuan, C., and Liu, J. (2016). Online human action detection using joint classification-regression recurrent neural networks. *arXiv preprint arXiv:1604.05633*.

[Linux performance tools, 2014] Linux performance tools (2014). Best command line tools for linux performance monitoring. [Online; accessed 01-Sep-2016].

[Loukas, 2015] Loukas, G. (2015). *Cyber-Physical Attacks: A Growing Invisible Threat*. Butterworth-Heinemann (Elsevier).

[Loukas et al., 2013a] Loukas, G., Gan, D., and Vuong, T. (2013a). A review of cyber threats and defence approaches in emergency management. *Future Internet*, 5(2):205–236.

[Loukas et al., 2013b] Loukas, G., Gan, D., and Vuong, T. (2013b). A taxonomy of cyber attack and defence mechanisms for emergency management networks. In *Proceedings of the Third International Workshop on Pervasive Networks for Emergency Management (IEEE PerNem 2013), San Diego, CA, USA*, pages 18–22.

[Loukas and Oke, 2007] Loukas, G. and Oke, G. (2007). Likelihood ratios and recurrent random neural networks in detection of denial of service attacks. In *Proceedings of International Symposium of Computer and Telecommunication Systems, SPECTS*, volume 7. Citeseer.

[Loukas and Öke, 2009] Loukas, G. and Öke, G. (2009). Protection against denial of service attacks: A survey. *The Computer Journal*, page bxp078.

[Loukas et al., 2008] Loukas, G., Timotheou, S., and Gelenbe, E. (2008). Robotic wireless network connection of civilians for emergency response operations. In *Computer and Information Sciences, 2008. ISCIS'08. 23rd International Symposium on*, pages 1–6. IEEE.

[Loukas et al., 2016] Loukas, G., Yoon, Y., Sakellari, G., Vuong, T., and Heartfield, R. (2016). Computation offloading of a vehicles continuous intrusion detection workload for energy efficiency and performance. *Simulation Modelling Practice and Theory*.

[Lu et al., 2015] Lu, T., Zhao, J., Zhao, L., Li, Y., and Zhang, X. (2015). Towards a framework for assuring cyber physical system security. *Int. J. Security Appl.*, 9(3):25–40.

[Marnerides et al., 2015] Marnerides, A. K., Smith, P., Schaeffer-Filho, A., and Mauthe, A. (2015). Power consumption profiling using energy time-frequency distributions in smart grids. *IEEE Communications Letters*, 19(1):46–49.

[Metz, 2014] Metz, R. (2014). Rise of the robot security guards.

[Mitchell and Chen, 2011a] Mitchell, R. and Chen, I.-R. (2011a). A hierarchical performance model for intrusion detection in cyber-physical systems. In *Wireless Communications and Networking Conference*, pages 2095–2100. IEEE.

[Mitchell and Chen, 2013a] Mitchell, R. and Chen, I.-R. (2013a). Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles Using Behavior Rule Specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1.

[Mitchell and Chen, 2013b] Mitchell, R. and Chen, I.-R. (2013b). On survivability of mobile cyber physical systems with intrusion detection. *Wireless Personal Communications*, 68:1377–1391.

[Mitchell and Chen, 2014] Mitchell, R. and Chen, I.-R. (2014). A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 46(4):55.

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

[Mitchell III, 2013] Mitchell III, R. R. (2013). *Design and Analysis of Intrusion Detection Protocols in Cyber Physical Systems*. PhD thesis, Virginia Tech.

[Mo et al., 2014] Mo, Y., Chabukswar, R., and Sinopoli, B. (2014). Detecting integrity attacks on scada systems. *IEEE Transactions on Control Systems Technology*, 22(4):1396–1407.

[Naess et al., 2005] Naess, E., Frincke, D., McKinnon, A., and Bakken, D. (2005). Configurable middleware-level intrusion detection for embedded systems. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 144–151.

[NASA, 2010] NASA (16 Sep 2010). Nasa - five things about nasa's mars curiosity rover. [Online; accessed 18-Oct-2016].

[Navet, 2011] Navet, N. (2011). Automotive communication systems: from dependability to security. In *talk at the 1st Seminar on Vehicular Communications and Applications (VCA 2011), Luxembourg*.

[Noh et al., 2003] Noh, S., Lee, C., Choi, K., and Jung, G. (2003). Detecting distributed denial of service (ddos) attacks through inductive learning. *Intelligent Data Engineering and Automated Learning*, pages 286–295.

[OFcom, 2014] OFcom (2014). Measuring mobile broadband performance in the uk: 4g and 3g network performance.

[Pasqualetti et al., 2013] Pasqualetti, F., Dörfler, F., and Bullo, F. (2013). Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729.

[Patel and Rana, 2014] Patel, B. R. and Rana, K. K. (2014). A survey on decision tree algorithm for classification. *International Journal of Engineering Development and Research*, 2.

[Paul, 2008] Paul, N. R. (2008). *Disk-level behavioral malware detection*. PhD thesis, University of Virginia.

[Plahl et al., 2013] Plahl, C., Kozielski, M., Schlüter, R., and Ney, H. (2013). Feature combination and stacking of recurrent and non-recurrent neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6714–6718. IEEE.

[Portokalidis et al., 2010] Portokalidis, G., Homburg, P., Anagnostakis, K., and Bos, H., . (2010). Paranoid android: versatile protection for smartphones. In *26th Annual Computer Security Applications Conference*, pages 347–356. ACM.

[Ráček et al., 2012] Ráček, J., Ministr, J., et al. (2012). *ICT Support for Emergency Management*. Trauner Verlag.

[Rajamäki et al., 2012] Rajamäki, J., Rathod, P., Ahlgren, A., Aho, J., Takari, M., and Ahlgren, S. (2012). Resilience of cyber-physical system: a case study of safe school environment. In *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pages 285–285. IEEE.

[Reaves and Morris, 2009] Reaves, B. and Morris, T. (2009). Discovery, infiltration, and denial of service in a process control system wireless network. In *eCrime Researchers Summit, 2009. eCRIME '09.*, pages 1–9.

[Rinaldi et al., 2016] Rinaldi, S., Della Giustina, D., Ferrari, P., Flammini, A., and Sisinni, E. (2016). Time synchronization over heterogeneous network for smart grid application: Design and characterization of a real case. *Ad Hoc Networks*.

[Roesch et al., 1999] Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99(1), pages 229–238.

[Sabaliauskaite and Mathur, 2014] Sabaliauskaite, G. and Mathur, A. P. (2014). Countermeasures to enhance cyber-physical system security and safety. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 13–18. IEEE.

[Sasi, 2015] Sasi, R. (2015). Maldrone the first backdoor for drones. [Online; accessed 11-July-2015] Fb1h2s aka Rahul Sasi's Blog.

[Sauter, 2013] Sauter, M. (2013). loic will tear us apart the impact of tool design and media portrayals in the success of activist ddos attacks. *American Behavioral Scientist*, 57(7):983–1007.

[Schoitsch, 2012] Schoitsch, E. (2012). Cyber-physical systems - what can we learn from disasters with respect to assessment, evaluation and certification/qualification of systems-of-systems? In *Proceedings of 20th IDIMT Conference, Jindrichuv Hradec, Czech Republic*, pages 69–81.

[Schumann et al., 2015] Schumann, J., Moosbrugger, P., and Rozier, K. Y. (2015). R2u2: Monitoring and diagnosis of security threats for unmanned aerial systems. In *Runtime Verification*, pages 233–249. Springer.

[Shachtman, 2011] Shachtman, N. (7 October 2011). Computer virus hits u.s. drone fleet. [Online; accessed 18-Oct-2016].

[Sharma et al., 2014] Sharma, A. B., Ivančić, F., Niculescu-Mizil, A., Chen, H., and Jiang, G. (2014). Modeling and analytics for cyber-physical systems in the age of big data. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):74–77.

[Sheldon et al., 2005] Sheldon, F. T., Batsell, S. G., Prowell, S. J., and Langston, M. A. (2005). Position statement: Methodology to support dependable survivable cyber-secure infrastructures. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 310a–310a. IEEE.

[Shetty et al., 2014] Shetty, S., Adedokun, T., and Keel, L.-H. (2014). Cyberphyseclab: A testbed for modeling, detecting and responding to security attacks on cyber physical systems. *Academy of Science and Engineering (ASE), USA,© ASE 2014*.

[Siaterlis and Maglaris, 2005] Siaterlis, C. and Maglaris, V. (2005). Detecting incoming and outgoing ddos attacks at the edge using a single set of network characteristics. In *Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on*, pages 469–475. IEEE.

[Sing et al., 2005] Sing, T., Sander, O., Beerenwinkel, N., and Lengauer, T. (2005). Rocr: visualizing classifier performance in r. *Bioinformatics*, 21(20):3940–3941.

[Skormin et al., 2014] Skormin, V., Dolgikh, A., and Birnbaum, Z. (2014). The behavioral approach to diagnostics of cyber-physical systems. In *AUTOTESTCON, 2014 IEEE*, pages 26–30. IEEE.

[Society, 2016] Society, I. C. (2016). Ieee computer society predicts top 9 technology trends for 2016.

[Sravani and Srinivasu, 2014] Sravani, K. and Srinivasu, P. (2014). Comparative study of machine learning algorithm for intrusion detection system. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*, pages 189–196. Springer.

[Stenberg, 2016] Stenberg, D. (2016). curl.

[Sterbenz et al., 2013] Sterbenz, J. P., Çetinkaya, E. K., Hameed, M. A., Jabbar, A., Qian, S., and Rohrer, J. P. (2013). Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation. *Telecommunication systems*, 52(2):705–736.

[Storey, 2009] Storey, D. (2009). Securing process control networks. *Network Security*, 2009(10):10–13.

[Striki et al., 2009] Striki, M., Manousakis, K., Kindred, D., Sterne, D., Lawler, G., Ivanic, N., and Tran, G. (2009). Quantifying resiliency and detection latency of intrusion detection structures. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–8. IEEE.

[Templeton, 2011] Templeton, S. J. (May 25-27, 2011). Security aspects of cyber-physical device safety in assistive environments. In *4th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, pages 53:1–53:8. ACM.

[Timotheou and Loukas, 2009] Timotheou, S. and Loukas, G. (2009). Autonomous networked robots for the establishment of wireless communication in uncertain emergency response scenarios. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1171–1175. ACM.

[Tippenhauer et al., 2011] Tippenhauer, N. O., Pöpper, C., Rasmussen, K. B., and Capkun, S. (2011). On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 75–86. ACM.

[Trutschel et al., 2014] Trutschel, U., Heinze, C., Sommer, D., and Golz, M. (2014). The influence of feature combination for the discrimination results between two heart conditions. In *8th Conference of the European Study Group on Cardiovascular Oscillations, ESGCO 2014*.

[Turk et al., 2005] Turk, R. J. et al. (2005). *Cyber incidents involving control systems*. Idaho National Engineering and Environmental Laboratory.

[Uluagac et al., 2014] Uluagac, A. S., Subramanian, V., and Beyah, R. (2014). Sensory channel threats to cyber physical systems: A wake-up call. In *2014 IEEE Conference on Communications and Network Security (CNS)*, pages 301–309. IEEE.

[Venkatasubramanian et al., 2009] Venkatasubramanian, K. K., Banerjee, A., and Gupta, S. K. (2009). Green and sustainable cyber-physical security solutions for body area

networks. In *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pages 240–245. IEEE.

[Vuong et al., 2014] Vuong, T., Filippoupolitis, A., Loukas, G., and Gan, D. (2014). Physical indicators of cyber attacks against a rescue robot. In *IEEE International Conference on Pervasive Computing and Communications*, pages 338–343. IEEE.

[Vuong et al., 2015a] Vuong, T., Loukas, G., and Gan, D. (2015a). Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. In *Proceedings of 13th International Conference on Pervasive Intelligence and Computing (PICOM)*. IEEE.

[Vuong et al., 2015b] Vuong, T., Loukas, G., Gan, D., and Bezemskij, A. (2015b). Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. In *Proceedings of 7th International Workshop on Information Forensics and Security (WIFS)*. IEEE.

[Walker, 2011] Walker, J. J. (2011). Cyber security concerns for emergency management. *Edited by Burak Eksioglu*, page 39.

[Wang et al., 2010] Wang, E. K., Ye, Y., Xu, X., Yiu, S.-M., Hui, L. C. K., and Chow, K.-P. (2010). Security issues and challenges for cyber physical system. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 733–738. IEEE Computer Society.

[Warner and Johnston, 2003] Warner, J. S. and Johnston, R. G. (2003). Gps spoofing countermeasures. *Homeland Security Journal*, 25(2):19–27.

[Wasicek et al., 2014] Wasicek, A., Derler, P., and Lee, E. A. (2014). Aspect-oriented modeling of attacks in automotive cyber-physical systems. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6. IEEE.

[Watts up?, 2016] Watts up? (2016). Watts up? PRO meters. [Online; accessed 22-May-2016].

[Wells et al., 2014] Wells, L. J., Camelio, J. A., Williams, C. B., and White, J. (2014). Cyber-physical security challenges in manufacturing systems. *Manufacturing Letters*, 2(2):74–77.

[Wu et al., 2011] Wu, G., Lu, D., Xia, F., and Yao, L. (2011). A fault-tolerant emergency-aware access control scheme for cyber-physical systems. *arXiv preprint arXiv:1201.0205*.

[Xue et al., 2014] Xue, M., Wang, W., and Roy, S. (2014). Security concepts for the dynamics of autonomous vehicle networks. *Automatica*, 50(3):852–857.

[Yampolskiy et al., 2012] Yampolskiy, M., Horvath, P., Koutsoukos, X. D., Xue, Y., and Sztipanovits, J. (2012). Systematic analysis of cyber-attacks on cps-evaluating applicability of dfd-based approach. In *2012 5th International Symposium on Resilient Control Systems (ISRCS)*, pages 55–62. IEEE.

[Yampolskiy et al., 2013] Yampolskiy, M., Horvath, P., Koutsoukos, X. D., Xue, Y., and Sztipanovits, J. (2013). 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.

[Zeng et al., 2012] Zeng, Q., Li, H., and Qian, L. (2012). Gps spoofing attack on time synchronization in wireless networks and detection scheme design. In *MILCOM 2012-2012 IEEE Military Communications Conference*, pages 1–5. IEEE.

[Zeng and Chow, 2014] Zeng, W. and Chow, M. (2014). A reputation-based secure distributed control methodology in d-ncs. *IEEE Transactions on Industrial Electronics*, 61(11).

[Zhang et al., 2011] Zhang, Y., Wang, L., Sun, W., Green, R. C., , and Alam, M. (2011). Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE ransactions on Smart Grid*, 2:796–808.

[Zhu et al., 2011] Zhu, B., Joseph, A., and Sastry, S. (2011). A taxonomy of cyber attacks on scada systems. In *Internet of things (iThings/CPSCom), 2011 international*

conference on and 4th international conference on cyber, physical and social computing, pages 380–388. IEEE.

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