

UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

MD AL AMIN KHAN

Digital Twin-Based Deception Solution for Autonomous Vehicles Security

Master's Thesis (30 ECTS)

Supervisor(s): Naveed Muhammad, PhD
Mubashar Iqbal, PhD

Tartu 2025

Digital Twin-Based Deception Solution for Autonomous Vehicles Security

Abstract:

The autonomous vehicle industry is rapidly advancing toward fully autonomous vehicles, raising significant concerns about cybersecurity and physical safety. These vehicles depend heavily on interconnected digital components, which enhance convenience, but at the same time pose risks. Those interconnected components expose critical systems to malicious actors. Recent incidents have demonstrated that adversaries are employing sophisticated techniques to exploit these systems, posing a greater risk to human life, data security, and public safety. This reminds us that the security of autonomous vehicles needs to be addressed. This study addresses the problem of securing autonomous vehicles against cyber threats by employing a digital twin-based deception solution. Here we show that our proposed digital twin deception system can effectively mitigate unauthorized access attempts targeting autonomous vehicles, redirecting them to an isolated honeypot and collecting valuable intelligence while keeping the attacker busy on the honeypot. At the same time, the data is visualised on the Kibana dashboard. This method not only identifies intrusions but also generates actionable insights through integrated monitoring and incident response features. The proposed system can also be extended beyond autonomous vehicles; it can also be applied to other cyber-physical models where a digital twin offers significant security and operational advantages. These findings contribute to the larger effort of protecting next-generation transportation systems through intelligent, resilient, and adaptive security mechanisms.

Keywords:

Digital twin, Autonomous vehicle security, Digital twin-based Deception, IoV Security, Self-driving vehicle security, AVs Security, Security Incident Handling

CERCS: P170, Computer Science, Numerical Analysis, Systems, Control

Digitaalse kaksiku põhine eksitussüsteem autonoomsete sõidukite turvalisuse tagamiseks

Lühikokkuvõte:

Autonoomsete sõidukite tööstus areneb kiiresti täisautonoomsete sõidukite suunas, tekitades olulisi muresid küberjulgeoleku ja füüsilise ohutuse osas. Nende sõidukite töö tugineb omavahel ühendatud digitaalkomponentidel, mis küll suurendab mugavust, kuid samal ajal tekitab uusi riske. Need omavahel ühendatud komponendid muudavad kriitilise tähtsusega süsteemid haavatavaks pahatahtlikele tegijatele. Hiljutised juhtumid on näidanud, et ründajad kasutavad nende süsteemide ära kasutamiseks keerukaid meetodeid, mis kujutavad endast suuremat ohtu inimeludele ning andmete- ja avalikule turvalisusele. Sellest tuleneb, et autonoomsete sõidukite turvalisus vajab lahendamist. Käesolev töö käsitleb autonoomsete sõidukite kaitset küberohtude eest, rakendades digitaalse kaksiku põhiseksitustehnoloogiat. Tulemused näitavad, et pakutud digitaalse kaksiku põhine eksitussüsteem suudab tõhusalt ära hoida autonoomseid sõidukeid sihtivaid volitamata juurdepääsu katseid, suunates need isoleeritud peibutuskeskkonda ning kogudes samal ajal väärtuslikku infot, hoides ründajat peibutuskeskkonnas hõivatuna. Samal ajal visualiseeritakse kogutud andmed Kibana juhtpaneelil. See meetod ei piirdu sissetungide tuvastamisega, vaid loob integreeritud seire- ja intsidentidele reageerimise funktsioonide kaudu ka tegutsemiseks vajalikke teadmisi. Pakutud süsteemi saab hõlpsasti laiendada ka väljaspool autonoomseid sõidukeid ning seda võib rakendada ka teistes küberfüüsilistes mudelites, kus digitaalne kaksik pakub olulisi turvaja operatiivseid eeliseid, mis aitavad kaasa järgmise põlvkonna transpordisüsteemide kaitsele läbi intelligentsete, vastupidavate ja kohanemisvõimeliste turvamehhanismide.

Võtmesõnad:

Digitaalne kaksik, autonoomsete sõidukite turvalisus, digitaalse kaksiku põhine eksitustehnoloogia, sõidukite IoV turvalisus, isesõitvate sõidukite turvalisus, AV-de turvalisus, turvaintsidentide käsitlemine

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Problem Statement	8
1.3	Research Questions	8
1.4	Research Method	9
1.5	Contributions	9
1.6	Thesis Structure	9
2	Background	11
2.1	Autonomous Vehicles	11
2.1.1	Current State of Technology	12
2.2	Digital Twin Technology	12
2.2.1	Concept and Applications	12
2.2.2	Benefits and Challenges	12
2.3	Cybersecurity in Autonomous Vehicles	13
2.3.1	Importance of Cybersecurity	13
2.3.2	Common Threats and Vulnerabilities	14
2.4	Deception-Based Security Strategies	14
2.4.1	Overview and Importance	15
2.4.2	Applications in Cybersecurity	15
2.5	Summary	15
3	Systematic Literature Review	17
3.1	Literature Review Questions	17
3.2	Search Strings	17
3.3	Literature Sources and Selection	18
3.4	Data Extraction Strategy and results	19
3.5	Security Challenges	21
3.5.1	Security threats	21
3.5.2	Security tools	22
3.5.3	Gaps in different security tools	22
3.6	Summary	23
4	Proposed Design	25
4.1	Design and System Specifications	25
4.2	Proposed Digital Twin Framework	25
4.3	Deception-Based Strategy Process	26
4.4	Real-Time Security Monitoring	28
4.5	Use Cases	28

4.5.1	Use Case 1: Simulating Normal Driving Behaviour	30
4.5.2	Use Case 2: Simulating Attacker Driving Behaviour	31
4.5.3	Activity diagram of system	33
4.6	Summary	33
5	Implementation	36
5.1	Development of Digital Twin Framework	36
5.1.1	Backend server development	36
5.1.2	Authentication and status server	40
5.1.3	Donkey car and simulation setup	43
5.1.4	Frontend app development	44
5.1.5	UI control setup	44
5.1.6	Image preprocessing	45
5.1.7	Running application	49
5.2	Deceptive Environment Implementation	49
5.3	Security Monitoring and Intrusion Detection	51
5.4	Testing and Validation	51
5.5	Summary	53
6	Evaluation	54
6.1	Incident Response Procedure	54
6.2	Scenario Testing	56
6.3	Discussion of Results	57
6.4	Summary	58
7	Discussion	59
7.1	Answer to Research Questions	59
7.2	Limitations	60
7.3	Future Work	61
8	Conclusion	63
	References	67
	Acknowledgement	67
	Grammarly	67
	Visual Studio Code	67
	ChatGPT	68

Appendix **69**
I. Resources 69
II. Figures 69
III. Licence 71

1 Introduction

The rise of autonomous vehicles (AVs) enhances user safety, transportation systems, and drivers' convenience through the use of advanced sensors, artificial intelligence (AI), and interconnected systems. However, as AVs become more integrated with urban ecosystems and rely on various components for complex computations, they also expose a vulnerable cyber-physical architecture to cybersecurity threats. Malicious actors (e.g., attackers) can exploit weaknesses in AVs' perception systems, such as LiDAR spoofing, manipulating decision-making algorithms, or intercepting vehicle-to-everything (V2X) communications [1]. These cybersecurity threats can potentially lead to catastrophic outcomes [1]. Traditional security measures like encryption and intrusion detection systems often fall short in addressing the unique and dynamic attack surfaces of AVs. This highlights the need for innovative defensive strategies.

The digital twin (DT) technology creates a virtual counterpart of the physical vehicle system that synchronizes in real-time through continuous data simulation. This dynamic replication enables comprehensive system monitoring, threat detection, and predictive analytics capabilities. Additionally, a deception solution utilizing virtual replication based on DT can provide a solution to effectively address cybersecurity threats, collect data on them, build attacker profiles, and uncover the motivations behind the cybersecurity threats. By utilizing a high-fidelity DT of an AV, we can divert attackers to a decoy system, isolating them from the physical system. This strategy prevents attackers from receiving misleading responses generated by the decoy. Not only does this approach help neutralize the threat, but it also allows for the collection and analysis of the attacker's data, which can be valuable for future defence strategies.

This thesis investigates the process of creating a DT-based deception system and implementing it using the Donkey Car system. To achieve this, a DT of the Donkey Car system is developed. Additionally, the ELK stack is used to collect data, profile the attacker, and analyze the results. The Donkey Car system operates using two methods: one via a web interface and the other through a game controller.

1.1 Motivation

AVs simplify and enhance people's lives by minimizing human errors on the road and improving overall comfort. Studies reveal that thousands of lives are lost annually due to human error, which remains the leading cause of traffic accidents in regions such as the USA and Europe [?]. By significantly reducing the likelihood of such errors, AVs have the potential to lower accident rates and improve daily life for individuals [2].

However, there are significant security risks associated with a heavy reliance on AV systems. Various high-risk attacks on AVs can occur, such as sensor spoofing, GPS manipulation, and adversarial machine learning. Research has shown that LiDAR signals can sometimes mislead perception, causing vehicles to misinterpret their surroundings [3].

Additionally, V2X communication can be compromised by malicious actors, potentially leading to traffic chaos and, in the worst-case scenario, vehicle collisions. All of these vulnerabilities pose serious threats to the safety of vehicle users and the public, as well as to the trust in this technology [4].

1.2 Problem Statement

The full potential of AVs can only be realized after evaluating the cybersecurity threats inherent in their connected cyber-physical systems. AVs rely heavily on interconnected and complex networks. They are vulnerable to cyberthreats that can compromise the safety of human life, the functionality of the vehicle, and user trust in the AVs [5], discussed in Section 3 in detail. Traditional security measures often fail to address malicious actors' sophisticated tactics, particularly in real-time V2X communication [6]. This exposes AV systems to data breaches, system hijacks, traffic disruptions, and physical collisions. The urgent challenge is to develop innovative and proactive security solutions that defend against these cyberthreats, anticipate and neutralise them dynamically, and save vehicles, human lives, and assets.

1.3 Research Questions

To address the aforementioned challenges, this research seeks to establish a main research question that guides the study. The focus is on exploring strategies to enhance the cybersecurity of autonomous vehicles using DT technology. The main research question is: "**How can digital twin technology enhance the security of autonomous vehicles?**" This question encapsulates the core objective of the study, which is to develop and evaluate an innovative DT-based solution to strengthen self-driving vehicles' resilience against cyber threats. The main research question leads to the following sub-questions:

1. **RQ1:**How do digital twins shield autonomous vehicles from cyberattacks?
2. **RQ2:** What are the security risks in autonomous vehicles? Which model or framework can provide security for autonomous vehicles?
3. **RQ3:** What are the practical limitations and challenges of implementing a digital twin-based deception framework in autonomous vehicles?
4. **RQ4:** How does a digital twin framework compare to existing cybersecurity measures regarding resilience and adaptability to cyber-attacks?

1.4 Research Method

We use a design science methodology [7] to address how DT technology can improve autonomous vehicle security in a different way. This method combines quantitative security metrics and qualitative assessment to develop an effective DT model that enhances AV cybersecurity. Firstly, we conduct a detailed analysis of the current cybersecurity solutions for AVs and DT technologies. The design science methodology is identified as the most appropriate approach for addressing the research problem and achieving the study's objectives. Guidelines for applying design research methods and the model produced in this paper are utilized.

In this paper, we follow a specific structure and a rigorous process informed by the literature review on DTs. We have developed a comprehensive evaluation model, which is presented in Table 1.

1.5 Contributions

This paper focuses on several key research topics in the field of cybersecurity for AVs utilising DT technology. The main contributions of this work are:

1. Developing a high-fidelity DT of an AV's cyber-physical system with an isolated deception system for cyber threat protection.
2. Implementing anomaly detection and machine learning capabilities for real-time adaptive security measures.
3. Developing the solution by creating a test case, measuring the success rate, and efficiency.
4. Integrating the open-source Donkey Car project as a practical demonstration of DT technology in AV systems.
5. Securing the AV through developing proactive, deception-based cybersecurity strategies.

1.6 Thesis Structure

The thesis is outlined as follows. Chapter 2 provides background on the history and evolution of AV, DT technologies, and cybersecurity measures for AV using deception-based technologies related to DTs. Chapter 3 presents detailed discussion of AV security and review of the literature addressing research questions through the findings of various authors and researchers, with a comparison of different perspectives. Chapter 4 provides the development of a comprehensive architectural design for research, following the established strategy. Chapter 5 is the implementation of the architecture and creation

Table 1. Design Science Guidelines and Thesis Artefacts

Guideline	Description	Thesis Artefact
DG1. Design as an Artefact	Design-science research must produce a viable artefact to address cybersecurity threats in AVs.	DT-based deception framework integrated into autonomous vehicle security using deception technology.
DG2.Problem Relevance	Develop technology-driven solutions to mitigate critical security risks in AVs.	Systematic Literature Review (SLR) results highlighting vulnerabilities in AVs and different types of attacks
DG3.Design Evaluation	Rigorous evaluation of the DT deception framework's utility and efficacy.	Security risk analysis, penetration testing of DT models, empirical metrics, quality, and availability
DG4.Research Contributions	Contributions to design artefacts, methodologies, and security frameworks.	DT system architecture diagrams, SLR findings, and a deception framework with adaptive honeypots.
DG5.Research Rigor	Application of rigorous methods in artefact construction and validation.	Threat modelling, ERD diagrams, DT architecture validation, and IoV component analysis.
DG6.Design as a Search Process	Iterative refinement of the DT deception solution to balance constraints and security goals.	Implement deception layers for DTs using a case-driven approach on Docker containers or in the cloud.
DG7.Communication of Research	Clear presentation for technical and management-oriented audiences.	SLR tables, decision tree sequence diagrams, and workflows for the deception framework.

of a test model, including validation of the model's performance. Chapter 6 shows the extraction of data generated from the previous chapter, along with presentation and evaluation of the results. Chapters 7 and 8 outline the discussion and analysis of the extracted data, concluding with the thesis summary and recommendations for further research.

2 Background

This chapter focuses on the fundamentals of DTs, autonomous vehicles, and security concerns related to these vehicles. It covers the historical context, technological advancements, and current solutions. Additionally, it is important to understand the reasons behind these security concerns and identify areas for improvement. The aim is also to clarify the basic concepts discussed later in this paper.

2.1 Autonomous Vehicles

The level of the autonomous vehicle can be categorized based on the level of human intervention. The level of autonomy can be categorized into six levels [8]. Those levels are adopted by the USA Department of Transportation.

Table 2. Levels 0 to 5 of autonomous driving technology

Level	Description
Level 0	Vehicles with no automated features require the driver to control the vehicle completely.
Level 1	Vehicles equipped with one or more primary automated features, such as cruise control, require the driver to perform all other tasks.
Level 2	Vehicles are equipped with two or more primary features, such as adaptive cruise control and lane-keeping, that work together to relieve the driver from controlling those functions.
Level 3	Vehicles with features allow the driver to relinquish control of the vehicle's safety-critical functions depending on traffic and environmental conditions. Given the constraints of the automated features, the driver is expected to take control of the vehicle after an appropriately timed transition period.
Level 4	Vehicles are equipped with features that allow the driver to operate vehicles within a specific area with predefined conditions. The vehicle can perform all aspects of driving even if the driver does not respond to a request to intervene.
Level 5	Fully autonomous vehicles that monitor roadway conditions and perform safety-critical tasks throughout the duration of the trip with or without a driver being present. This level of autonomy is appropriate for occupied and unoccupied trips.

According to Table 2, vehicles classified as Level 4 and 5 can operate autonomously, meaning the driver does not need to continuously monitor the situation. For Level 3

vehicles, drivers should pay attention to their surroundings and be prepared to take control quickly [9].

2.1.1 Current State of Technology

AVs are currently at level four according to Table 2 in terms of technical development, according to the Society of Automotive Engineers (SAE) [10]. Companies such as Tesla, Waymo, and Cruise have made significant strides in this field. Most of these companies utilize modern cameras, radar, LiDAR, and various sensors, processing the data with artificial intelligence (AI) to make independent decisions. However, despite advances in camera sensors and other components that rely on AI-driven perception, there are still significant challenges in handling adverse weather conditions and unexpected scenarios [11].

2.2 Digital Twin Technology

DTs are virtual representations of physical objects or processes that can collect information from the real environment to represent, validate, and replicate the present and future behavior [12]. DT is used in different fields such as manufacturing, production, smart city, transportation, security, etc. DT can help productivity in different fields by allowing virtual tests, simulation, and verification before making actual products or in parallel to physical products, where experimentation or physical products are highly expensive or risky, allowing DTS to save time, money, and cut development time.

2.2.1 Concept and Applications

DTs mirror the real data in a virtual format that can be used for different purposes, such as simulation of behaviours, optimization of the operation, and monitoring the system's performance. Because of this multi-behaviour, the DT concept can be implemented in different sectors and industries, including the most popular industrial production and manufacturing industry. At the same time, DT can also be implemented in the security industry. The property of real-time data collection and analysis is essential from a security perspective. Real-time data allows simulation and testing with different conditions and allows monitoring, which can finally be used for optimization. Later in this paper, we implemented a DT system to protect our autonomous vehicle from attackers. Industries like Manufacturing, healthcare, and smart cities use DTs.

2.2.2 Benefits and Challenges

There are several benefits to using a DT. One of the main advantages is cost-effectiveness; utilizing a DT is often cheaper than testing a real system. Halting an ongoing process to

conduct a test can be prohibitively expensive, making the DT a valuable solution [13]. Additionally, some systems require testing before completing system development. In such cases, a small DT can be highly beneficial as it minimizes the risk of potential damage to the physical or real system while collecting sufficient data for analysis. Testing on real systems can be hazardous in specific industries, such as healthcare and finance. Exposing real data can compromise the entire industry's security. DT can help enhance security while also saving time and money. Furthermore, a DT can operate alongside the system, testing parameters or data flows in real-time. Scalability and flexibility are crucial advantages that DT can provide in addressing various challenges.

There are some disadvantages also, for example, realism is one of the most important issues here. DT can be very effective; however, it will not fully behave like a real system. In the production industry, the DT testing outcomes differ from real outcomes in actual systems. For some systems, running a DT is unnecessary and expensive. Creating a DT like the actual system sometimes needs too much work and time. In this case, many workforce and resources must be allocated for DT rather than utilized in the system. For security purposes, the use of DT can also be vulnerable. Attackers can use this DT as an entry point and collect the complete system design or the actual system.

2.3 Cybersecurity in Autonomous Vehicles

AVs are complex systems that depend on complex software, AI, sensors, and connectivity. Because of this complexity, AVs are vulnerable to safety concerns. Discussing the cybersecurity of AVs is a more prominent topic we need to consider about the General Data Protection Regulation (GDPR) and use technical and organizational measures to secure the infrastructure, network, and data from unauthorized access or malicious activity [14]. Cybersecurity is critical to ensure users' safety, privacy, and reliability, where hacking could lead to catastrophic outcomes.

2.3.1 Importance of Cybersecurity

There are several reasons why cybersecurity is important for AVs. First of all, safety risk: without proper security, AVs can be hijacked or taken under control during operation, such as steering or sudden braking, which might be a fatality or major accident. Also, compromising physical sensors will generate wrong data, which misleads AVs' decision-making, which is also dangerous. Personal data is also crucial while using AVs. Personal data, including user location, driving type, or vehicle maintenance, is very sensitive and could hamper the user's daily life. More importantly, after developing AVs over time, the public is reluctant to adopt new technology if those security issues are not solved.

2.3.2 Common Threats and Vulnerabilities

AVs can be vulnerable in different ways. Physical components like GPS and LiDAR/radar manipulation can be one of them. GPS holds notable significance and provides precise geolocation, speed, and temporal data regardless of vehicle speed and condition [15]. However, GPS is also not immune to security threats. By spoofing GPS signals, an attacker can mislead AVs. Additionally, jamming can cause vehicles to become lost. Additionally, GPS signals can be intercepted through a meaconing attack, which involves the deliberate delay of retransmission. A replay attack is also a widespread GPS attack where the signal is retransmitted at an alternative time and location.

LiDAR system is an essential system for autonomous vehicles. The primary function is range-finding sensors emitting light three-dimensionally 100 times per second. Capturing the return pulse of reflection creates a surrounding environment by system [15]. However, this essential component could be the target of different types of attacks, such as spoofing attacks, where an attacker can create a virtual object environment in front of the LiDAR system, which misguides the outcome of LiDAR. The attack on sensors, vehicles might slow down or cause dangerous accidents that could be fatal. Another type of attack on LiDAR is a Cyber-Level attack where the attacker disturbs sensor data and Electromagnetic interference, where LiDAR can misclassify objects and perceive non-existent obstacles that are dangerous.

Remote exploitation and gaining unauthorized access to the AV over the cellular network or WIFI connection can be another threat. The attackers can gain complete control of the AV, implement malicious software, install bugs in the system, and the ECU could turn off the system, or ask for ransom.

AV cameras serve as the vehicle's eye, which collects photos and live videos of vehicles and the world to analyze the recognition of traffic lights, signs, lanes, front cars, and others. Target of cameras using light interference, which exploits the sensitivity to camera light and disturbs the camera's standard functionality. In this case, another system that depends on the camera system is disturbed.

2.4 Deception-Based Security Strategies

Deception-based security strategies are proactive defense mechanisms that involve creating decoys, traps, or false information to mislead attackers, detect intrusions, and gather intelligence about their tactics [16]. These strategies are designed to actively engage adversaries, diverting them from real assets and providing defenders with valuable insights into their methods and objectives [17]. Animals and insects have used deception for thousands of years for defense and survival. Recently, deception has become popular among humans in the defence industry and digital technology to protect valuable assets. Computer defence technology deception has been used for several decades. There are many popular defensive technologies, and honeypot is one of the most important and is

used to incorporate deception [18].

2.4.1 Overview and Importance

In deception, base technology intentionally creates mock data, fake systems, and networks to lure hackers into a trap and expose themselves. However, those attackers are also intelligent and aware of this type of technology, so before implementing a deception system, it created carefully so that attackers do not easily detect it.

Honeypots create a fake system or service designed to attract attackers. As soon as the attacker gains access, the attacker finds some mock data inside. Honeytoken generates fake credentials, data, and files while notifying system administrators about access to attacker-passed log data. This process allows a decoy system to create a mimic network segment that resembles a real environment [19].

The importance of deception systems is significant. Early threat detection is one of them. For example, deception strategies can help to find the attacker and process their intrusion into the system before an attack on critical infrastructure. It is also possible to collect attacker behaviours and working processes, methods, techniques, steps, etc. In some cases, deception systems are a cost-effective defence mechanism that actively engages attackers and consumes their time and resources while protecting the actual system. This can also check whether authentic users fall into the decoy system. If they fall into the decoy system, we will know there is a false positive in the system, which is possible to eliminate.

2.4.2 Applications in Cybersecurity

Different ways critical infrastructure protection provides. Active protection and passive protection are part of them. Network security of the application is one of the ways that is part of active protection. Protect the real application network, create a fake network, and mislead the attacker to Honeypots, which can previously set server, database, and IoT devices and collect data about the attacker's behavior. Create fake files inside the honeypot, like API keys, passwords, text, etc. When the attacker enters into honeypot, an alert is triggered. Big data can also be a proper way of wasting time by exploring those. In a cloud environment, fake resources like virtual machines, storage, or API connections can also identify unauthorized access attempts. Adding decoy devices like fake sensors and systems running on the central system is also possible for some IoT applications. When an attacker exploits those fake systems, those systems can trigger an attack.

2.5 Summary

In conclusion, the attempt at the invasion of autonomous vehicles has not started very recently. Many changes and innovations have brought this current stage. However,

new technology also makes autonomous vehicles more vulnerable. The most common technology used by autonomous vehicles now exposes different threats and vulnerabilities that attackers might exploit easily. However, there is a potential way to protect against common attacks using a deception security system.

3 Systematic Literature Review

This section conducts a systematic literature review to find contemporary advancements in DT-based deception solutions for autonomous vehicle security. This section is part of the **DG4, DG5, and DG6** design-science guidelines from Table 1. From the perspective of autonomous vehicle security, we explore existing research on cybersecurity in autonomous vehicles while focusing on deception-based strategies, threats, and techniques, and how they can mitigate those. Besides those, this section also discusses research questions, search strategies, selection criteria, and data execution that need to ensure a systematic and comprehensive review.

3.1 Literature Review Questions

The focus of the review is to answer the research question **RQ1: How do digital twins shield autonomous vehicles from cyber-attacks via deception-based security?** This research question is large, so it is divided into two subsections to understand two aspects of autonomous vehicle security.

RQ1.1 How can digital twins be used for autonomous vehicles?

RQ1.2 Which strategy can solve the security of autonomous vehicles?

In this first sub-question, we find that the DT is used in the context of the security of AVs. To limit this question, we focus on sensor data and unauthorized access to those data types. In the second sub-question, we focus on finding the deception-based strategy in DT frameworks and the most popular strategy or technology used recently.

3.2 Search Strings

In the search section, we selected the search string very carefully. We have defined a search string as follows: “*((Autonomous vehicles OR IoV OR Self-driving vehicle) AND (cybersecurity OR Security OR Threats OR Vulnerability) AND (Digital Twin OR DT OR Deception))*”. A Digital twin, a synonymous phrase of DT and Autonomous Vehicle (AVs) can be used to enhance the search. We did not find much difference in the search and found that using the full name gives a more comprehensive result. A large string “*Digital twin as a deception-based system for an autonomous vehicle to prevent a cyber-attack*”. We found that some keywords for this search were present in the abstract and some sections of the thesis; however, this search does not yield better results than what we used before.

3.3 Literature Sources and Selection

We consider several popular journals and databases to search for relevant papers. We explore the IEEE Digital Library, ScienceDirect, ACM Digital Library, and SpringerLink. These databases provide more recent work, and the quality of the publications is very high. We selected high-profile authors with strong citations from the best sources. The topic we are working on is relatively new, and the mentioned databases provide relevant results based on our inclusion and exclusion criteria. For our selection, we look for the papers published between 2020 and 2025 that are related to DT-based deception security in autonomous vehicles. Our primary focus is on DTs and the security aspects of autonomous vehicles.

Inclusion Criteria

- IC1: Literature related to DT-based deception.
- IC2: Literature discussion on Autonomous vehicle security.
- IC3: Literature discussed deception-based security.

Exclusion Criteria

- EC1: Literature that was published before 2020.
- EC2: Literature that has not been published in journals, magazines, or conference papers.
- EC3: Literature that does not mention DT-based OR Autonomous Vehicles security.
- EC4: Literature that needs early access, is not fully accessible, or is not in the English language.

The initial search results for digital libraries were expected to be productive, but many of the results were irrelevant to the query. Initially, there were 2,933 papers. After applying EC1, the number of relevant documents decreased to 1,224. Following the application of EC2, we obtained 295 results. When EC3 was applied, we narrowed it down to 247 results. After using EC4, we ended up with 89 relevant results. In total, 89 papers were retained. We then manually aligned our research interests with these papers using the inclusion criteria for analysis. Ultimately, we identified 13 documents for study. Furthermore, snowballing and references from previously included literature yielded two articles, which were manually checked for relevance and included in the scope of the review. Different phases of selection are shown in the table 3.

Table 3. Selection process of research papers and different phases

Phase	Description	Count
1st Phase	Search string result on Digital library	1224
2nd Phase	Filtering journals, Magazines or conference papers	295
3rd Phase	Paper with explicit mention of DT-based Autonomous Vehicles	247
4th Phase	Inclusion of paper with manual analysis and abstracts	13
5th Phase	Inclusion from cross reference and snowballing	15

3.4 Data Extraction Strategy and results

Fifteen papers were studied to gather data about the research questions. We collected and noted data from those papers for further analysis that might answer our research questions. The Table 4 lists the research work, applications, objectives, contributions, security threats, deception techniques, and threat detection and mitigation strategies. Those columns chose according to RQs. Most importantly, we look for threats of autonomous vehicles and mitigate those threats using deception and decoys or other techniques. The table 4 shows more details about this strategy.

Table 4. Data Extraction Form

Data Item	Value
Research work , year	Paper and Year of Publication.
Application	Area of Research.
Objective and contribution	Objective and contribution of the literature.
Threats of autonomous vehicles	threats discussed in the literature.
Threats detection and mitigation	Threats mitigation using DT or decoy system.

The review outcome is presented in Table 5. Besides, the main motivation of this review is to find new and existing research and different approaches that are used or suggested to secure autonomous vehicles. According to the findings, most researchers found that a DT-based solution might be a potential solution for secure AVs. Alongside deception, technology with DT might be more interesting.

Among the selected papers, most talk about cybersecurity and data security. The eight

Table 5. Application of Research

Application	Studies	Methods
Digital Twin	[20] [21] [13] [1] [22]	Framework design, simulations, ML-based detection.
Deception and Decoy	[1][23][24][19]	Deception frameworks, honeypots, blockchain integration.
Honeypot	[19]	Honeypot configuration, attack simulation.
Cybersecurity (VANETs)	[22]	DT-boosted ML, anomaly detection.
Blockchain	[1]	Blockchain integration with DT and deception.

literature studies propose DT-based AV security, three papers mention deception-based security. Some papers have also discussed blockchain for data and resource transfer and protection. Six papers propose frameworks, and three papers were surveys. Some of those papers generated their data in labs or in the field, whereas some used existing data.

RQ1.1: How can digital twins be used for autonomous vehicles?

Campole et al [20] proposed an edge-based DT framework for connected and autonomous vehicles that can optimize security, latency, and resource management. Besides, Hadraoui et al. [21] propose a design for distributing the DT architecture that could solve the problem of diagnosing and predicting failure in autonomous vehicles, which are powered by electricity. On-demand data synchronization for Vehicle DT can prevent attacks and ensure real-time security with accuracy by Hui et al. [13]. Kabir and Ray et al. [25] also explore security options using DTs. Yigit et al. [22] DTs can also be used to detect attacks in vehicular ad-hoc networks (VANETs).

RQ1.2: Which strategy can solve the security of autonomous vehicles?

Different researchers have solved the security of autonomous vehicles differently. Yigit et al. [22] use machine learning alongside a DT in real-time anomaly detection in VANETs. Honeycar was developed by Panda et al. [19] to detect and analyze attacks on autonomous vehicles, which can also be configured accordingly. Iqbal et al. [1] and Suhail et al. [23] proposed blockchain and deception-based DTs to secure against and mislead attackers from autonomous vehicles and cyber-physical systems.

3.5 Security Challenges

The SLR helps us identify several security issues. For example, remote hacking, where unauthorized access to vehicles happens through the wireless connection, which leads to the total takeover of the vehicle or turning off some function that might be a security risk [26]. Sensors like LiDAR, cameras, or radar populated with wrong data generate false positive results in vehicle collisions. Data breaches can also happen through unauthorized access. DoS Attacks are very common in cyberspace. This same attack can be done on autonomous vehicle systems to keep busy physical resources and disturb normal operations, degrade the vehicle's performance, and cause connectivity loss and other problems.

3.5.1 Security threats

Most researchers experimented with the simulator and its setup in the labs. Some of them only discuss theory without any experiments. Some have also done case studies of historical or well-known attacks on vehicles. The experiment with DTs is also mentioned. According to recent work and analysis, the biggest threats to AVs are bus spoofing attacks, man-in-the-middle attacks, and firmware reprogramming. More details about the finding are presented in the table 6.

AVs may get attacked by spoofing, DoS & DDoS, Model poisoning, etc. Attackers target the vehicles to make them unstable, dangerous, and not provide actual values that should provide, which can lead to fatal accidents. A spoofing attack is defined when the attacker pretends to be legitimate and, by doing this attacker can harm or mislead the actual work done by targeted components. Thus, IoV and autonomous vehicles are interconnected models and also connected to the internet or smart city networks. Attackers can use methods of bus spoofing attack, where attackers fake CAN bus messages and manipulate vehicle control. As well as methods, like DoS and DDoS, an attack on GPS by sending false GPS signals. Using this method also possible to take control by GPS spoofing. The same method is also applicable to Rader and LiDAR spoofing. DoS attackers are also used to degrade ECU performance.

In the case of, man-in-the-middle attack, the attacker manipulates communication with interconnected components, such as the message from the ECU and the V2X network. In some cases, DT might not be a solution; the attacker manipulates the digital-twin industrial Control system (DT-ICS) communication, making vehicle response slower. Figure 1 is presented to help illustrate and enhance understanding more clearly about the categories of different types of attacks on autonomous vehicles.

Transporting fake data to vehicles also makes them vulnerable because that hampers vehicles' everyday decision-making. The same happens in the case of model poisoning. Attackers modify the firmware of the hardware ECU to allow them to activate malicious control.

Table 6. Common attacks on autonomous vehicles

Event No	Attacking event and target	Impact and Succession	Methods Used
1	Jeep Cherokee Hack (2015) [27]	Remote exploitation of vulnerabilities to hijack vehicle controls.	Bus Spoofing Attack, Man in the middle, Firmware Reprogramming Attack.
2	CAN bus attacks disable breaking [19]	Simulated attacks based on real-world in-vehicle network vulnerabilities.	Manupulated ECU.
3	GPS spoofing in logistics tracking disruptions [22]	VANET vulnerabilities mimicking real GPS spoofing incidents.	DDoS , GPS spoofing.
4	Stuxnet (2010) [28]	Cyber-physical system (CPS) attacks disrupting industrial systems.	Dos , False data injection.
5	Industrial control system (ICS) breaches [29]	Attacks on critical infrastructure linked to DT vulnerabilities.	Model poisoning, False data Injection, Exploitation of DT-ICS synchronization gaps.
6	Tesla Autopilot vulnerabilities breaches [25]	Sensor spoofing risks in commercial AVs.	LiDAR Spoofing, Camera Adversarial, GPS spoofing, Radar spoofing .

3.5.2 Security tools

Different types of security tools have been presented by researchers to enhance cyber-physical systems and autonomous vehicles. Some researchers combine digital security techniques with blockchain technology, while others integrate machine learning methods, such as BiLSTM networks and GANs. Each approach targets specific attack vectors, including Sybil attacks, DoS attacks, jamming, and protocol exploits.

However, these presented solutions face significant limitations, such as high computational demands, limited adaptability to real-time scenarios, and challenges in scalability or deployment across dynamic environments. A comprehensive overview of these tools, their detection methods, targeted types of attacks, and associated constraints is presented in the table 7.

3.5.3 Gaps in different security tools

Various methods and technologies are employed to mitigate attacks on autonomous vehicles. Every tool often addresses specific areas rather than all aspects. The standard

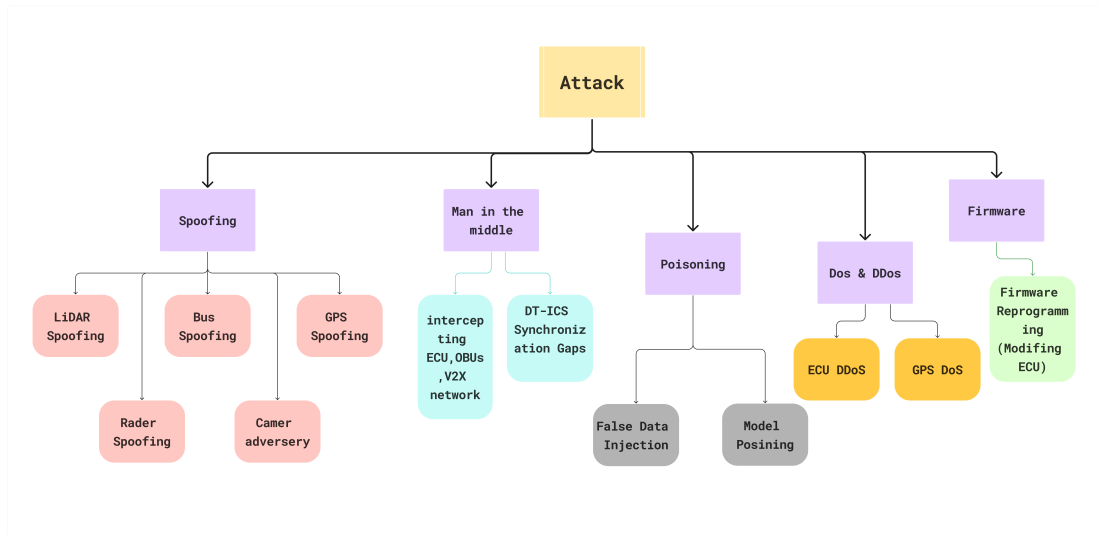


Figure 1. Common attacks organized by category

methods are DT and Machine learning models to prevent or divert attacks at different places or stops. However, those methods only work when attack patterns are known and not novel, and the model is trained based on a vast dataset. Still, there are limitations to scalability issues and enormous resources needed to protect against those types of attacks. Besides, the machine learning model sometimes raises false positive results, which might also interrupt the regular activity of vehicles. Cybertwin and HoneyCar, those two models try to solve Sybil, DDoS, and malware-related attacks along with the machine learning model. However, the researchers did not try DTs with deception and decoy systems for active security of autonomous vehicles. Although DECEPTWIN and Deception-Driven DT use DTs with blockchain, they are in different domains and are not related to autonomous vehicles. DT as a deception decoy system might be an opportunity to develop a new protection model.

3.6 Summary

In this systematic literature review, a search and analysis conduct and answer the research questions. We proceed with the paper based on the answers we found from the literature review. The papers we selected represent that most research talks about different ways of securing autonomous vehicles and IoV. Some papers also discuss the physical components of cars. It is evident that the amount of research on the security of autonomous vehicles based on a DT is reasonable. However, using a DT-based deception decoy systems has not been investigated in the literature to that extent, although this is not a totally new topic. Sabah Suhail [23] used a decoy system to mitigate the security threat of the

Table 7. Security tools used for attack detection and area of success

Security Tool	Detection method	Success area	Limitation
HoneyCar	Honeypot-based attack[19]	Malware DDoS attack.	Simulated environment , Manual configuration.
Cyber-twin	DT with ML model SVM, Random Forest [22]	VANET Sybil attacks.	Need to train data and false positive results.
Generative AI (GANs)	Synthetic attack generation for training[30]	Adversarial attack defense.	Computational overhead; Requires labeled datasets.
Edge-Based DT Framework	Anomaly detection via edge-computed DT models[20]	Jamming/MITM.	Resource constraints on edge devices ,Limited interoperability.
BiLSTM	Deep learning for lateral control attack detection[31]	False command injection.	Relies on historical data; Untested in real-time CPS.
On-Demand DT Data Sync	Blockchain , Hash-chain for data integrity [13]	Tampering prevention.	Synchronization delays, Trust Issue of nodes.
Deception-Driven DT	Dynamic decoys with DT mirroring [23, 24]	Simulated CPS .	Limited to novel attack patterns.
Distributed DT for Prognostics	Predictive maintenance via distributed DT models[21]	EV powertrains.	Only focus diagnostics not active protection.
DECEPTWIN	Deception , Blockchain [1]	Simulated IoV attacks.	Scalability issues in large networks, Blockchain latency.
Encryption , Intrusion Detection	AES-256 , Signature-based IDS[27]	Eavesdropping.	High overhead for lightweight devices; No proactive deception.
ML for CPS Attack Detection	Random Forest , K-means clustering[28]	Protocol exploits.	Simulated environment , Manual configuration.

cyber-physical infrastructure of the water pumps. This same concept can also be explored with autonomous vehicle security. In this thesis, we discuss more about the potential of working with this idea and further concept generation and illustration. Several gaps and challenges were found in those research, and after considering those and the complex cybersecurity of autonomous vehicles, we propose a model in this thesis.

4 Proposed Design

Ensuring the security of autonomous vehicles is crucial due to the various types of threats they may face from different driving scenarios. However, significant progress has not been made in addressing these security concerns so far. Our design is based on a DT-based deception solution for autonomous vehicle security that creates an exact virtual replica of the physical car system. This creates platform controls vehicles and provides a security deception mechanism to identify, analyse, and mitigate cyber threats to AVs. This section answers the question of **RQ2: Which model or framework can provide security for autonomous vehicles?** We present a design and go through the different use cases, scenarios, and strategies to see the effectiveness of the presented design.

4.1 Design and System Specifications

The design is divided into several parts for control and other advantages. It divides into frontend apps, backend server, database management, deception system (Honeypot), data analysis and presentation. Frontend apps is the gateways of the donkey car service, whereas basic authentication is checked and other steps are done on the back end. The DT version is also created like a frontend app to eliminate security threats and collect data. The DT operates by mirroring the vehicle's electronic systems and network communications. At the same time, establishing a baseline of regular operation while simultaneously deploying deceptive elements that can attract, identify, and safely contain potential attackers.

The framework requires different categories of hardware, virtual machines, and networking components, and an internet connection between them as illustrated in Table 11. Minimum resources needed to get a desirable output from our designed system. Also, secure communication channels between physical and virtual components using the Tailscale network¹. It is a lightweight software solution that is easy to implement, suitable for different operating systems and hardware constraints. we also use Donkey Car² for this. A small physical car that behaves like a real car is suitable for our research.

4.2 Proposed Digital Twin Framework

The DT framework comprises several parts. For example, frontend app is the main access point where users authenticate themselves by giving credentials. This credential check is in the backend Node.js server. If the user credentials match the registered credentials in the MongoDB database, then the user is authorised to log in and create a 1-hour session. Within this time, users are allowed to do all operations permitted. The

¹<https://tailscale.com/>

²<https://docs.donkeycar.com/>

figure 2 shows the working procedure for a normal user. Now the authorized user directly accesses the car environment and performs all operations. However, if the attacker tries to gain access and fails, the attacker might give up, and the system does not collect enough information about the attacker. So, the second option is to develop a system that collects all the information about the attacker for future use of the strategy.

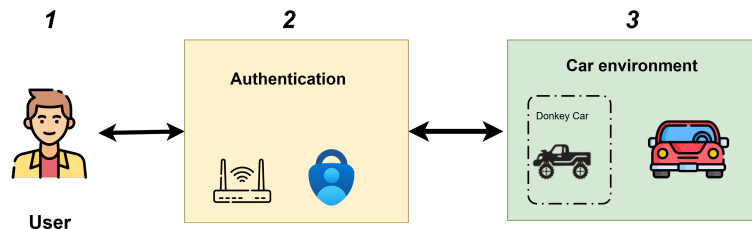


Figure 2. Working process for a normal user.

4.3 Deception-Based Strategy Process

A DT-based deception strategy model presented in Figure 3, when an attacker tries to gain access to the website to control the system remotely, the attacker enters the wrong credentials and presses the login button. Then the steps 2 of the design firewall with authentication activates and checks the input. The web application will not return this password is incorrect "or that wrong credentials have been used. Instead, the app secretly connects with the virtual machine as shown in steps 4 and 5, where the virtual donkey car is running as deception system. The design cleverly makes it seem like a legitimate system login. Upon the first login attempt, the system collects and analyzes all commands executed inside the VM, gathering data in step 6. At the same time find out the true intention of this unauthorized access and the command that run and time spent on that website. On the deception system , a virtual simulator setup and log generator are also included to make this seem more realistic. For the attacker, it will look like regular access. Besides, the video feed shown in the apps is pre-recorded and trained with captured images of the donkey car in advance, which effectively responds according to the control input from attackers.

The deception-based process is illustrated in Fig. 3, which shows several components. For instance, the attacker tries to gain access to the donkey car. This design combines several components, each responsible for different tasks. In Figure 3 step 2, all authentication and rerouting processes are carried out. In step 3, the donkey car operates. Steps 4 and 5 simulate interactions and create deceptions to mislead the attackers. In steps 6 and 7 (Figure 3), data monitoring and storage occur, which is essential for preserving information for future analysis.

We employ three different deception strategies. First, pre-recorded images playing as a video strategy (refer to Fig. 3 step-5), where pre-recorded videos are played; those videos are pre-trained with the recorder angle and other parameters. Categorized images from a different list of images record by Donkey Car, play those images as a video in the deception system.

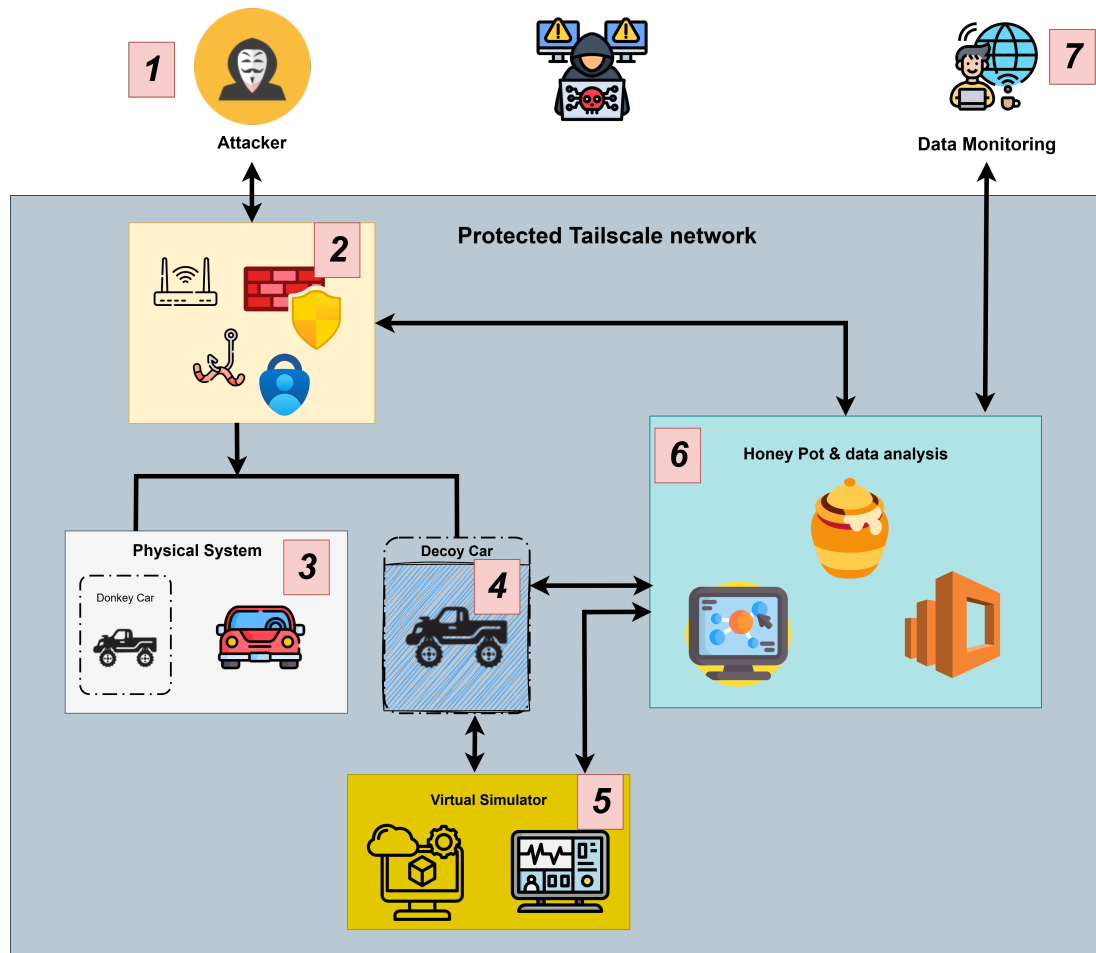


Figure 3. Deception-based strategy with DT

The second strategy is to control interaction (refer to Fig. 3 step-4 and 5) according to those categorized images. When video is playing simultaneously, if the attacker tries to control the car and presses any control key, the video will respond accordingly.

The third strategy is for the attacker to see the console logs simultaneously (refer to Fig. 3 step-5 and 6). These logs are generated in real-time and will give the attacker the impression that he controls the car. Those are the main deception strategies that are used primarily. At the same time, the web application console will capture and generate

events and logs.

4.4 Real-Time Security Monitoring

We use live monitoring and data analysis methods. For example, we leverage TPotce³ community edition open source multi-honeypot that has been designed for cyber threat detection and research. It comes with different honeypots, ELK stacks, and other technologies. Tpotce use by researchers for security analysis and monitoring, gathering intelligence and information. The advantage of using this is that logs are centralized, which can be accessed easily.

We use Kibana⁴ for visualizing logs which were collected in tpotce. Kibana visualizes an attack in different types of honeypots as well as actions. The input of password, username, or any logs input by the attacker is stored. Besides, it also tracks which IP addresses those requests came from and makes a category of alerts by type. At the same time, it is also possible to collect logs with ElasticVue⁵, where all connected nodes will send logs, which might be processed for further research or analysis. A figure 14 and a figure 17 represent the data collected by tpotce over the last month.

4.5 Use Cases

We develop a use case that clarifies various requirements, making the system design comprehensible. The given system modelling provides an overview of the workings of the donkey car system. It also shows the interaction of different components and the passing of data and information. We design a basic use case where different actors and components participate shown in figure 4. We discussed more details about the perspective, there are two different actors, the normal user and the attacker. Two other use case perspectives provide a clear understanding and ensure the design follows expectations. At the same time, features get ideas from the use case. Before creating the actual product, also help it to test and validate.

³<https://github.com/telekom-security/tpotce>

⁴<https://www.elastic.co/kibana>

⁵<https://elasticvue.com/>

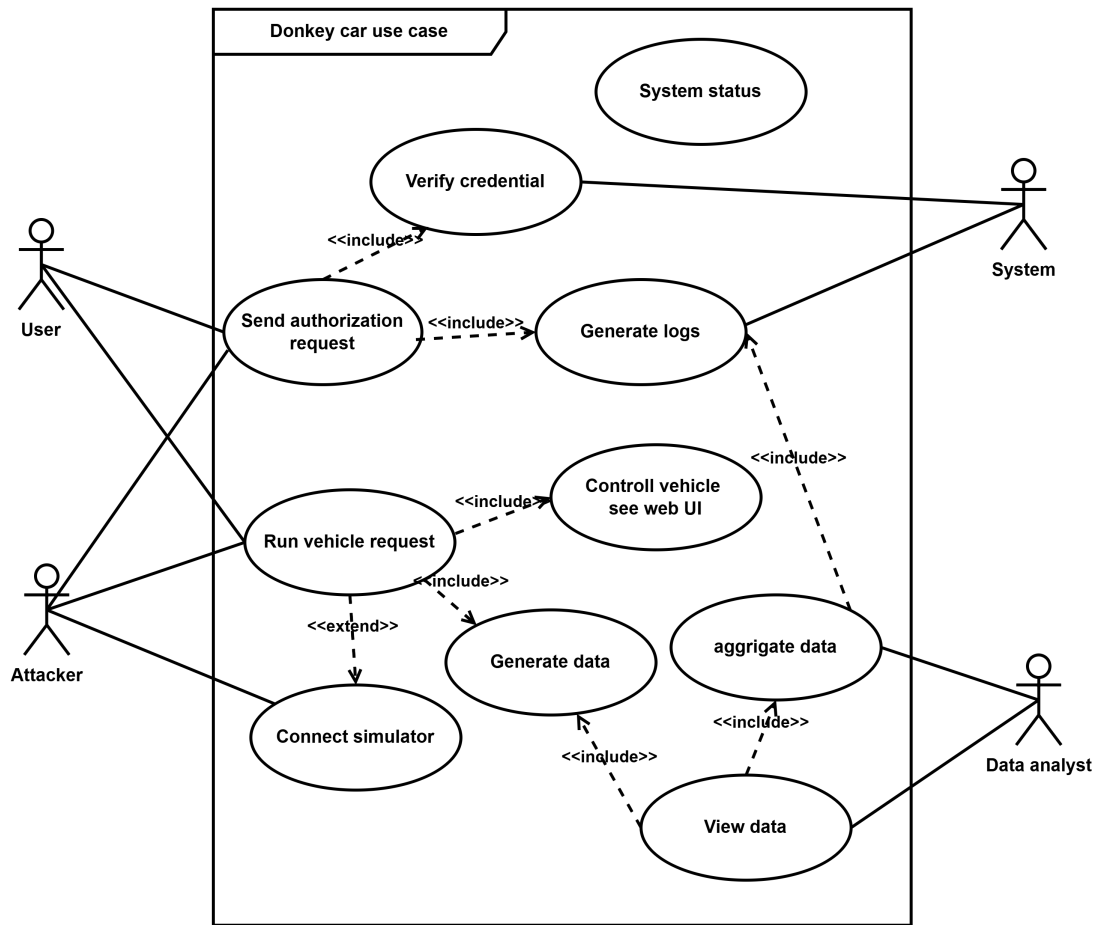


Figure 4. Use cases of Donkey Car

The UML diagram about the donkey car use case system consists of 4 actors. As shown in the figure 4, those actors are the user, attackers, data analyst, and system, where the system's boundary is defined of those actors. The diagram illustrates the security and operational workflow where:

1. A legitimate user can send authorisation requests, including credential verification, run vehicle requests, and interact with the system.
2. An attacker also attempts to interact with the vehicle request function and system.
3. The system actor manages verification of credentials and monitors system status.
4. The data analyst can view and aggregate data generated by the system and users.

The use cases diagram shows how these actors interact with various use cases within the system boundary in the Donkey Car system.

4.5.1 Use Case 1: Simulating Normal Driving Behaviour

We test a normal user condition, define the pre- and post-conditions, and how a user normally behaves during the connection and afterwards. To answer those questions, we have developed a use case Table 8 and a sequence diagram Figure 5, which explain every step. Sequence diagrams represent the system behaviour.

Table 8. Use Case: Run an Autonomous Vehicle Normal User Condition

Use Case ID:	UC_01
Use Case Name:	Run an autonomous vehicle
Created By:	Md Alamin Khan
Actors:	User, system, Data analyst
Description:	This case describes accessing the car and gaining control of it. Running a different command.
Trigger:	The user logs in to the system.
Precondition:	The user must have valid credentials and be connected to the internet.
Normal Flow:	1. The user sends a request for login with credentials. 2. Check the credentials; if verified, log in to the system. 3. Users use the system as usual and run all commands.
Postcondition:	The user must log out.
Alternative Flow:	User's credentials are not verified, and then the system triggers the send user condition.
Exception:	The user is not connected to the internet, or there is no connection.

Figure 5 illustrates the authentication and operation flow of a system called Donkey APP, which controls a "donkey car" using a web application. Where web application is running on the remote virtual machine. The workflow shows:

1. A user starts the Donkey application and attempts to log in.
2. The authentication process uses JWT (JSON Web Token) authorization.
3. If authentication fails, the user is notified.
4. When authentication succeeds, the user can run the donkey car.
5. The "Run donkey car" command triggers a Python process that manages driving.
6. A remote controller opens ports for communication.
7. The system establishes web access and control ports for accepting input.

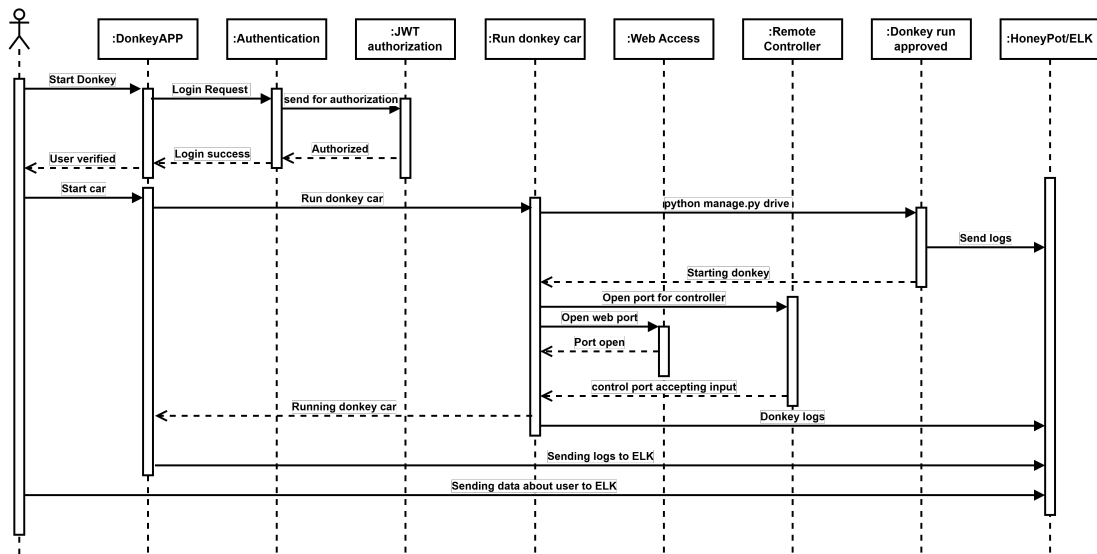


Figure 5. Normal user perspective sequence diagram

8. While running, the donkey car sends logs and user data to HoneyPot/ELK.

The sequence diagram demonstrates the security flow, operation sequence, and data collection process of this autonomous vehicle platform.

4.5.2 Use Case 2: Simulating Attacker Driving Behaviour

We now discuss use case two, similar to use case one. However, now we focus on the attacker's perspective and understand what happens when an attacker tries to gain access to the Donkey Car. Tables 9 and figure 6 explain this scenario better.

This sequence diagram Figure 6 illustrates the workflow of the Donkeyapp system, showing the interaction between a user and its various components. The key processes are as follows:

1. The user initiates by starting the Donkey application.
2. A login request is sent to the Authentication service.
3. The JWT authorization component verifies credentials.
4. If authorization fails, a "Login failed" message is returned to the user.
5. Upon unsuccessful authentication, the user is redirected to the donkey VM and starts the car.
6. The "Run donkey car" command activates a Python script that manages driving.

Table 9. Use Case: Attack Scenario on Autonomous Vehicle

Use Case ID:	UC_02
Use Case Name:	Run an autonomous vehicle
Created By:	Md Alamin Khan
Actors:	Attacker, system, Data analyst
Description:	This case describes accessing the car and gaining control of it. Running a different command.
Trigger:	The user tries to log in to the system with the wrong credentials.
Precondition:	The attacker is connected to the internet.
Normal Flow:	<ol style="list-style-type: none"> 1. The attacker sends a request for login with the wrong credentials. 2. Credentials are not verified, and then sent to the decoy system. 3. Attackers use a system that appears normal, but all commands are executed in the decoy environment.
Postcondition:	All data created by the attacker must be collected.
Alternative Flow:	The attacker cannot access the system due to a system error.
Exception:	The user is not connected to the internet, or there is no connection.

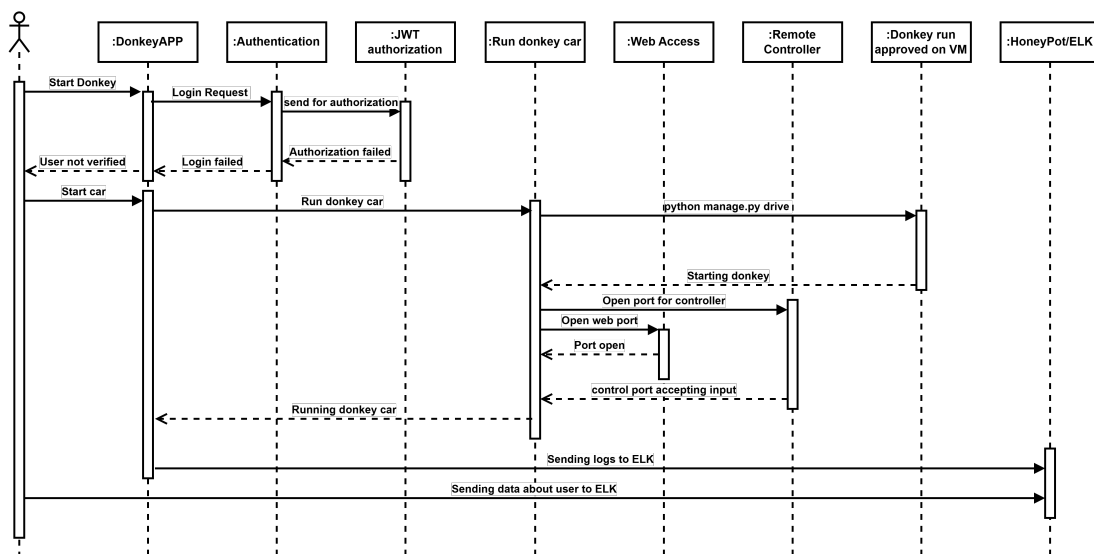


Figure 6. Attacking perspective sequence diagram

7. The remote Controller opens the necessary ports for communication.
8. Web access is established with open ports for control.
9. The system enables control port input for operating the vehicle.
10. Throughout the operation, the system sends logs and user data to HoneyPot/ELK for monitoring.

This sequence diagram 6 shows the error message flow when a failed authentication. Which explicitly states "Login failed" rather than "User not verified." and send to the deception environment, where the simulator Donkey Car is running with a pre-trained video deception system.

4.5.3 Activity diagram of system

Activity diagrams give a clear picture of the workflow of different components. The design of an activity diagram can help in the future to understand interactions when the project gets bigger, and help to quickly find out any vulnerabilities. Also, the diagram of donkey car design can be used further for security testing and improvement. For validating purposes use of an activity diagram is useful.

This activity diagram in Figure 7 shows the DT-based deception for an autonomous vehicle with five main components, each represented by a different purpose and working process. Every working process is presented in the Table 10 and its Activities. The diagram shows the authentication flow, command execution pathway, and how data flows between components. It also illustrates decision points where verification occurs and how the system responds to both successful and failed verification attempts.

4.6 Summary

Our proposed design effectively resolves the key challenges while specifically addressing the research questions identified in section 1.3. The RQ1 examines how digital twins can bolster cybersecurity measures for autonomous vehicles, while the RQ2 investigates which security frameworks are most suitable for protecting autonomous vehicle systems. Through this research, we generate several essential design artifacts that demonstrate our solutions. The system architecture and its security mechanisms are clearly illustrated through detailed component relationships and comprehensive use case diagrams, providing a complete visualization of our approach. This section answers the research questions and also delivers practical design elements that contribute to the implementation section.

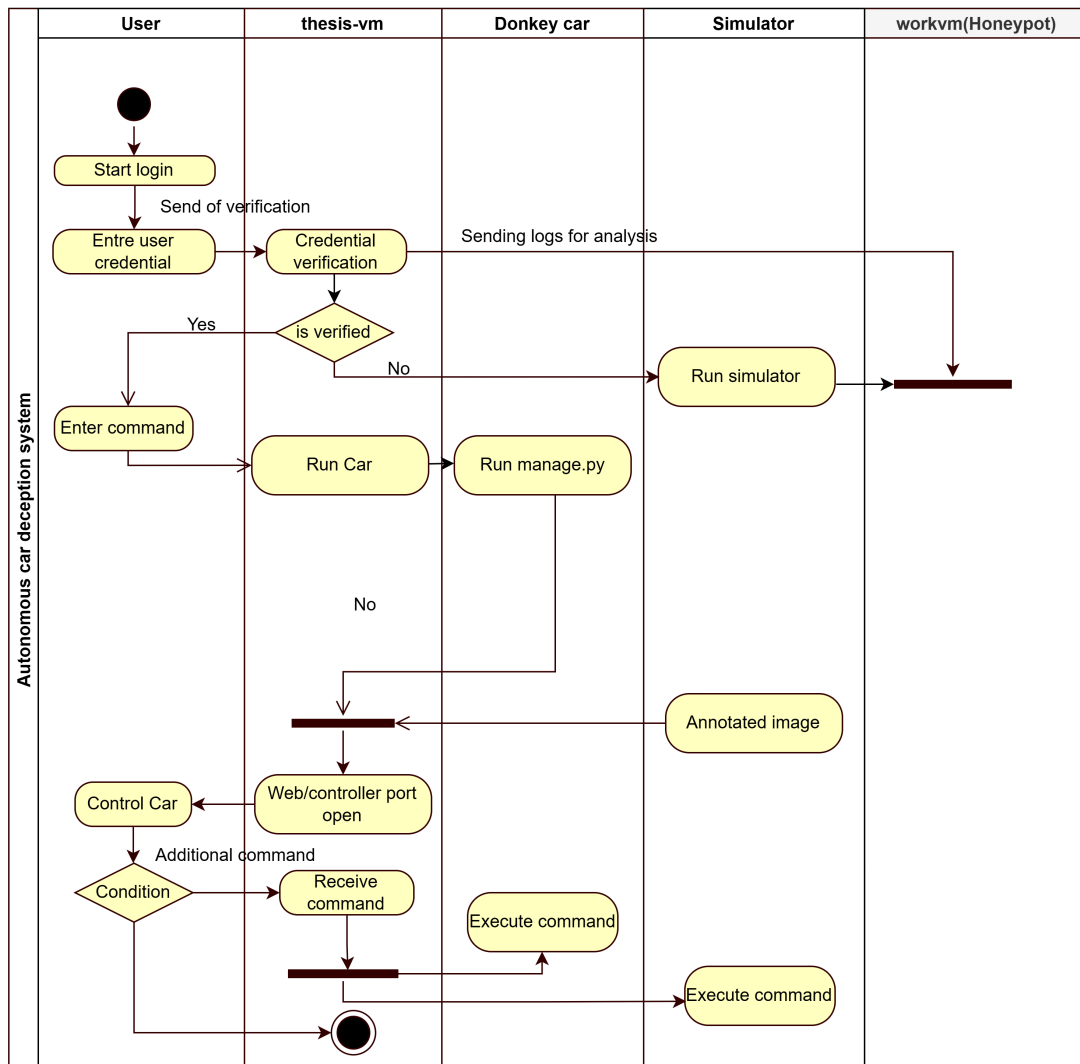


Figure 7. Summary of the Activity of a Digital Twin-Based Deception System

Table 10. System Components and Their Activity

Entity	Details
User	Initiates the process by starting the login
	Enters credentials
	If verified, enters commands to control the car
	Makes decisions based on conditions
Thesis-vm (Controller)	Handles credential verification
	Manages car operation through "Run Car"
	Opens web/controller ports
	Receives and processes user commands
Donkey Car	Sends logs for analysis
	Runs manage.py script
	Executes commands received from the user
Simulator	Runs simulations when needed
	Generates annotated images
	Can execute commands in parallel with the real car
Workvm (Honey-pot)	Receives logs and verification information
	Captures data throughout the system operation
	Also runs the Kibana dashboard that visualizes all gathered information

5 Implementation

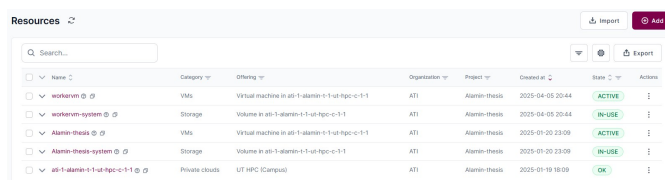
The implementation chapter delves into the details of the technology used to implement this solution. In the previous chapter, we discussed system design and theoretical aspects of the whole system's work. The implementation chapter explores all components and technologies used to make this solution. Frontend web application, several virtual machines, and a simulator work alongside the Donkey Car to create an active digital twin system. Additionally, a Raspberry Pi running Kali is also set up on the same network for testing. To solve different networking issues that hamper communication, we use Tailscale. All devices are connected to one network, allowing them to communicate without additional networking setup. We also perform some small tests about the action of DT in this chapter and carefully look at the response, which aims to bolster the resilience of the donkey car system against cyber threats during operations.

5.1 Development of Digital Twin Framework

Several technologies and configurations are used to create a DT of this Donkey car systems, as shown in Figure 8. Two virtual machines are used at the University of Tartu High Performance Computing Centre (HPC), as shown in Figure 9. Besides, one Raspberry Pi 400 with 4 GB RAM and 32 GB storage, and a laptop are used to make this system. Table 11 presents specifications in detail.



Figure 8. Donkey car



Category	Offering	Organization	Project	Created at	Status	Actions	
workitem	Virtual machine in ai-1-alamini-1-ut-hpc-c-1-1	ATI	Alamini-thesis	2025-04-05 20:44	ACTIVE		
workitem	Storage	ATI	Alamini-thesis	2025-04-05 20:44	IN-USE		
Alamini-thesis	Virtual machine in ai-1-alamini-1-ut-hpc-c-1-1	ATI	Alamini-thesis	2025-01-20 23:59	ACTIVE		
Alamini-thesis-system	Storage	ATI	Alamini-thesis	2025-01-20 23:59	IN-USE		
ai-1-alamini-1-ut-hpc-c-1-1	Private clouds	UT HPC Campus	ATI	Alamini-thesis	2025-01-10 18:09	OK	

Figure 9. ET AIS resources

5.1.1 Backend server development

The backend server is the essential component of the autonomous driving system for deception environments with the Donkey Car system. This system checks the configuration,

Table 11. System Specifications

Category	Component	Purpose of use
Virtual Machines	WorkVm for Tpotce (250Gb storage 8 vCPU, and 16 GB of RAM)	Running all honeypot.
	Alamin-thesis vm for Controller (8GB RAM , 200GB Storage)	System control and management.
Hardware	Kali Linux (Pi 400)	Security testing and penetration testing.
	Simulator with GPU	Running simulation environment.
	Personal laptop	Development environment.
	Donkey Car	Remote running and physical car with Donkey Car installed.
Networking	Tailscale	Creating secure private network between all components and hardware.
	Internet Connection	Make sure all the components and hardware are connected to the internet.

connects different devices, and allows them to run all operations.

The core functionality of SSH-based remote command execution is written using the paramiko library with the Python programming language. When a user sends a request for any command using frontend apps, it uses that command and parameter to establish a connection using SSH to request the device using a forwarding agent. It keeps track of connection establishment and others. The thread works in the background for non-blocking operations. Thus, several processes run in the background so that several threads are created accordingly, and every single thread keeps track of that command and other input data and keeps running the process. The main thing is that multiple commands run concurrently without blocking the main servers or each other. The logging system records detailed information about all operations, capturing timestamps, log levels, and relevant context for debugging and auditing. These logs are accessible through an API endpoint that returns the most recent entries for display in the web interface. The server implements proper error handling, catching exceptions, logging detailed error information, and returning appropriate HTTP status codes and error messages to clients. It also ensures adequate clean-up for resources like SSH connections when operations are complete or fail. Algorithm 1 presents a minimalistic approach to executing commands and monitoring processes on remote virtual machines.

SSH server 1 provide remote command execution capability . This algorithm work

Algorithm 1 SSH process management Server

```
1: Import modules: os, subprocess, threading, flask, fabric, paramiko
2: Input: Vm Alias , command
3: Output: Running process, logs, API endpoint
4: Define  $\mathcal{C}_{vm}$  with host details and commands
5: Initialize global structures:
6:    $\mathcal{P} \leftarrow \{\}$  ▷ Process tracking map
7:    $\mathcal{L} \leftarrow \{\}$  ▷ Log storage
8:    $\lambda_{capture} \leftarrow false$  ▷ Output capture flag
9: Create Flask app with CORS support
10: function ADDLOG( $v, \tau, \epsilon$ ) ▷  $v$ : VM alias,  $\tau$ : text,  $\epsilon$ : error flag
11:   entry  $\leftarrow$  CreateTimestampedEntry( $\tau, \epsilon$ )
12:    $\mathcal{L}[v].append(entry)$  ▷ Maintain size limit
13:   Print(entry)
14:   return entry
15: end function
    LogStream
16: function INITIALIZE( $v, \epsilon$ )
17:   this.v  $\leftarrow v$  ▷ VM alias
18:   this. $\epsilon$   $\leftarrow \epsilon$  ▷ Error flag
19:   this.buffer  $\leftarrow ""$  ▷ For incomplete lines
20: end function
21: function WRITE(data)
22:   this.buffer += data
23:   lines  $\leftarrow$  SplitByNewline(this.buffer)
24:   for each complete line in lines do
25:     AddLog(this.v, line, this. $\epsilon \vee$  ContainsError(line))
26:   end for
27:   this.buffer  $\leftarrow$  lastIncompleteLine
28: end function
29: function FLUSH
30:   if this.buffer  $\neq \emptyset$  then
31:     AddLog(this.v, this.buffer, this. $\epsilon$ )
32:     this.buffer  $\leftarrow ""$ 
33:   end if
34: end function
```

Algorithm 2 SSH process management Server (continue)

```
1: function GETCONNECTION( $v$ )
2:   if  $v \notin \mathcal{C}_{vm}$  then
3:     return Error("Invalid VM alias")
4:   end if
5:    $conn \leftarrow \text{CreateSSHConnection}(\mathcal{C}_{vm}[v])$ 
6:   AddLog( $v$ , "Connection established", false)
7:   return  $conn$ 
8: end function
9: function RUNCOMMANDBACKGROUND( $v$ ,  $config$ )
10:   $\lambda_{capture} \leftarrow true$ 
11:   $conn \leftarrow \text{GetConnection}(v)$ 
12:   $cmd \leftarrow \text{BuildFullCommand}(config)$   $\triangleright$  Directory + Conda + Base command
13:   $\pi_{before} \leftarrow \text{GetRunningProcesses}(conn, v)$ 
14:   $stdout \leftarrow \text{new LogStream}(v, false)$ 
15:   $stderr \leftarrow \text{new LogStream}(v, true)$ 
16:   $execution \leftarrow conn.execute(cmd, stdout, stderr)$ 
17:   $\pi_{after} \leftarrow \text{GetRunningProcesses}(conn, v)$ 
18:   $\pi_{new} \leftarrow \text{FindNewProcesses}(\pi_{before}, \pi_{after})$ 
19:   $\mathcal{P}[v].extend(\pi_{new})$   $\triangleright$  Track new processes
20:  return  $execution$ 
21: end function
22: function GETRUNNINGPROCESSES( $conn$ ,  $v$ )
23:   $output \leftarrow conn.execute("ps -ef").stdout$ 
24:  return ParseProcessList( $output$ )
25: end function
26: function FINDNEWPROCESSES( $\pi_{before}$ ,  $\pi_{after}$ )
27:  return  $\pi_{after} \setminus \pi_{before}$   $\triangleright$  Set difference
28: end function
29: Configure API routes on app:
30: GET "/run/ $v$ "  $\rightarrow$  Start default command on VM
31: Start server on port 5001 with threading enabled
```

as follow :

- **AddLog**($v, ,$): Creates and stores a timestamped log entry for tracking system activities and errors, maintaining a historical record for each VM.
- **LogStream.Initialize**($v, ,$): Initializes a stream object with VM information and buffer storage to handle continuous output from remote command execution.
- **LogStream.Write(data)**: Processes incoming data streams by buffering content, identifying complete lines, detecting errors, and forwarding completed lines to the logging system.
- **LogStream.Flush()**: Ensures any remaining buffered content is properly logged when a stream operation completes, preventing data loss.
- **GetConnection**(v): : Establishes and validates a secure SSH connection to the specified virtual machine, confirming the VM exists in the configuration.
- **RunCommandBackground**(v, config): Executes commands asynchronously on remote VMs while capturing output streams and identifying all new processes spawned by the command.
- **GetRunningProcesses**(conn, v): Retrieves a comprehensive list of all processes currently running on the target virtual machine for tracking and monitoring.
- **FindNewProcesses**($\text{before}, \text{after}$): Identifies newly spawned processes by computing the set difference between process lists captured before and after command execution.

This algorithm enables remote system access while simultaneously keeping several processes open.

5.1.2 Authentication and status server

An authentication server written with Node.js technology for frontend application protection and controlling access. Node.js also uses Express.js, JSON Web Token (JWT), Mongoose, and others. This server connects to MongoDB database, where all user data and other information are stored. We already use Docker for this project, so MongoDB also pulls from Docker Hub in the same Docker network, which works perfectly without any problem. This server gives access to several APIs. One is pinging the entire list of devices and sending the status to the frontend, indicating which devices are alive and which are not. Additionally, with JWT, you can create a new user and log in with that user account. Whenever a user tries to log in, the backend server takes the user's

input, uses the JWT secret, and compares it with the MongoDB database's stored data. If the input passes, the server successfully sends a response, creates a session for 1 hour, and stores session data in the browser session for 1 hour. In Algorithm 3, we present a shorter version of the working procedure of the authentication server.

Algorithm 3 Express Server Configuration

```
1: Import modules: express, jwt, cors, dotenv, ping, mongoose, logger
2: Input: HTTP Requests, MongoDB URI, JWT Secret, Device List
3: Output: Authorization, Device health status, User activity
4:  $\mathcal{E} \leftarrow \text{LoadEnvironmentVariables}()$  ▷ Environment variables
5:  $\text{app} \leftarrow \text{InitializeExpressApp}()$ 
6: Configure middleware:
7:   SetupCORS() ▷ Allow headers
8:   EnableJSONParser() ▷ Json converter
9:  $\mathcal{DB} \leftarrow \text{ConnectToMongoDB}()$  ▷ connect
10: Define  $\mathcal{S}_{user}$  schema: ▷ User schema
11:  $\mathcal{M}_{user} \leftarrow \text{CreateModel}('User', \mathcal{S}_{user})$ 

12: function INITIALIZEADMIN
13:    $exists \leftarrow \mathcal{M}_{user}.findOne(\text{roles}='ADMIN')$ 
14:   if  $exists = \emptyset$  then
15:      $admin \leftarrow \text{new } \mathcal{M}_{user}(\mathcal{E}.adminDefaults)$ 
16:      $admin.save()$ 
17:     LogMessage("Admin user created")
18:   end if
19: end function

20:  $\sigma_{jwt} \leftarrow \mathcal{E}.jwtSecret \vee \text{"default-secret-key"}$  ▷ JWT secret
21:  $\mathcal{D}_{devices} \leftarrow \mathcal{E}.deviceConfig$  ▷ Device configurations

22: function VERIFYTOKEN(req, res, next)
23:    $\tau \leftarrow \text{ExtractAuthHeader}(req)$ 
24:   if  $\tau = \emptyset \vee !\text{IsValidFormat}(\tau)$  then
25:      $\text{res.status}(401).send(\text{"Unauthorized"})$ 
26:     return
27:   end if
28:    $decoded \leftarrow \text{JWT.verify}(\tau, \sigma_{jwt})$ 
29:   if  $decoded \neq \emptyset$  then
30:      $\text{req.user} \leftarrow decoded$ 
31:      $\text{next}()$ 
32:   else
33:      $\text{res.status}(401).send(\text{"Invalid token"})$ 
34:   end if
35: end function

36: Define API endpoints:
37:   GET  $\text{"}/api/database/all"$   $\rightarrow \text{FetchAllApi}()$ 
38:   GET  $\text{"}/public/ping-devices"$   $\rightarrow \text{PingConfiguredDevices}(\mathcal{D}_{devices})$ 
39: StartServer( $\text{port}=\mathcal{E}.port \vee 3000$ )
```

The algorithm 3 create a secure server for login and monitoring with authentication verify. Key features of the algorithm as shown:

1. **Server Initialization:** Loads environment variables, initializes an Express application, configures middleware (CORS, JSON parsing), and establishes a MongoDB connection.
2. **User authentication:** The system JWT (JSON Web Token) uses an environment provided secret key or a default for secure authentication. If the user is not present, create a user.
3. **Authorization Middleware:** Extract token `VerifyToken` from the header, check validity, if passed, then attach information about the validity, return response if not, send an error message.
4. **API Endpoints:** Open ended point for public and private access. Login open for everyone however, accessing other operations needs `VerifyToken`

This algorithm can monitor other devices' health and an authentication point for web applications.

5.1.3 Donkey car and simulation setup

Donkey car setup with the donkey car repository is a challenging task. However, the Donkey Car GitHub repository gives clear instructions about the setup [32]. Before installing the Donkey Car, we need to set up its environment based on the preferred operating system. In our case, we have set up on a Linux operating system, so we go through all the instructions on how to install on a Linux system. First step: set up Python 3.11 64-bit version. On the other hand, Miniconda3 is also a good option where all packages are in one place. We go with

```
wget https://repo.anaconda.com/miniconda/Miniconda3-py311_24.4.0-0-Linux-x86_64.sh
bash ./Miniconda3-py311_24.4.0-0-Linux-x86_64.sh
```

After that, there are two options for installing the donkeycar project: one for just installing as a user, and another is a developer install option, where developers can modify. We choose the developer option and create a project, clone the GitHub repo from there, and install it there [33]. As soon as the installation is done, we have to make a working path where all necessary code is copied and possibly modified accordingly in the future, so that we can create a path for mycar there. We are using a Linux system, so we need to give root user access for the future so it can run commands as a user without any problem. In this step, there is little difference between the real donkey car and the simulator. For the donkey car with a Raspberry Pi setup, a power management module is

used. Based on the instructions, need to run different commands and modify. however, in our case, virtual system do need to configure. If it is the first time installing Donkey Car on that Raspberry Pi, some additional steps need to be taken, like modifying the interface option and opening the camera input for images and videos. When all settings are correctly done, the car starts working.

For the simulator, there is another GitHub repository donekyGym [34]. A binary format of donkeyGym is available for Linux. After Download, unzip, make it executable, and run `donkey_sim.x86_64`. Before starting slight modification is needed. If the simulator is working on a remote server or a different pc, then both the server/pc have to be on the same network, and inside `config.py`, change `SIM_HOST = "127.0.0.1"` to the same machine or the remote IP address for the remote server. Also, the camera option needs to change `CAMERA TYPE = PICAM` or `MOCK`.

5.1.4 Frontend app development

We have used `Vue.js` for the development of frontend apps for single-page application features. This frontend app is protected by authentication as described in algorithm 3. This app has several routes for login, logout, and so on. Upon complete login, web apps take in the Home page, where several components render as a single web page. First, when entering the donkey car system, the user sees the device's status, which is essential for running the system. For example, the whole system would be useless if Donkey Car or a simulator were not online. After that, in the middle section, the user can see web control where real-time videos are streaming via `WebSocket`, which is dynamically connected to Donkey car or a simulator. The control options are active once the web control is connected, and the user can use them. Also, it is possible to record of images at the same time while driving. Users can control the Donkey car system in two ways: one uses a joystick/touch screen, and the other uses buttons/arrow keys. This web application main focus point is the live streaming of logs generated by the donkey car or deception system, which shows the bottom section of the system logs. The logs start streaming as soon as both web car control and control options connect to the backend system, and the authentication from the backend is complete. Another option is connecting to any VM where the used car runs the command. This option is mainly for attackers, a trap that collects information. The authentication will decide which button or system allows the user. If authentication fails, it redirects to the same app in a honeypot, where all information is record. Working web apps with logs shown in figure 10.

5.1.5 UI control setup

Establishing a connection between the frontend app, the back-end, and the donkey car/virtual simulator is challenging. Every time the frontend connects with one of the systems, define which system is trying to connect to when picking the system name. For

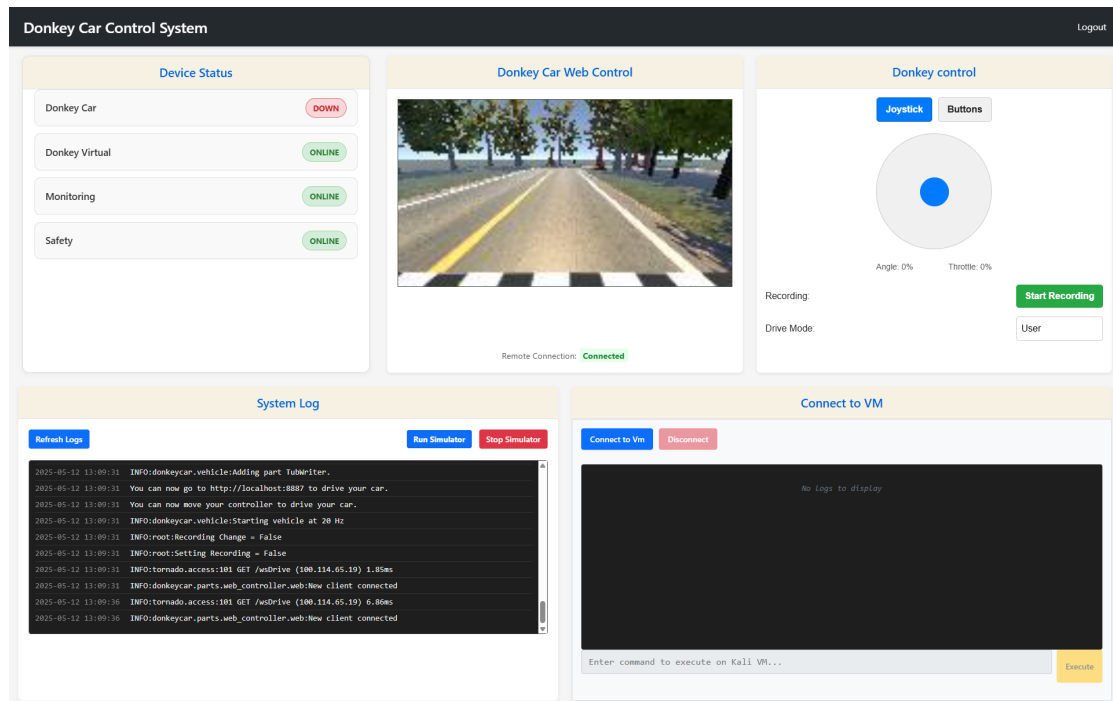


Figure 10. Running web application of Donkey car control

example, we choose the donkey car web control for a donkey car and find out where the vehicle is running. From the main component of web control, send this information to the event bus, where the event bus system shares the same event among all components of this frontend app. At the same time, try to connect to the WebSocket port opened by the donkey car manage.py system. For the video, the system connects with `http://${window.location.hostname}:8887/videos` This allows streaming videos. In the same way, joystick or arrow key connect with `http://${window.location.hostname}:8887/wsDrive` this option. Before this, some additional buttons and arrow keys need design, and a joystick design and calculation followed the main principle of the donkey car web control, which is written with Python and here with JavaScript. Figure 11 shows the donkey control connected with video and other options allowed by the system.

5.1.6 Image preprocessing

For image processing, we first need to record an image. For this purpose, we use the toy city (built for Donkey Cars) present on the second floor of Delta building (at University of Tartu). The donkey car captures a video as a small photo, around 4kb RGB images. Those images are fed through a tree convolution block, each consisting of a Conv2d layer followed by a MaxPooling2D layer. The first layer uses 32 filters, the second uses 64

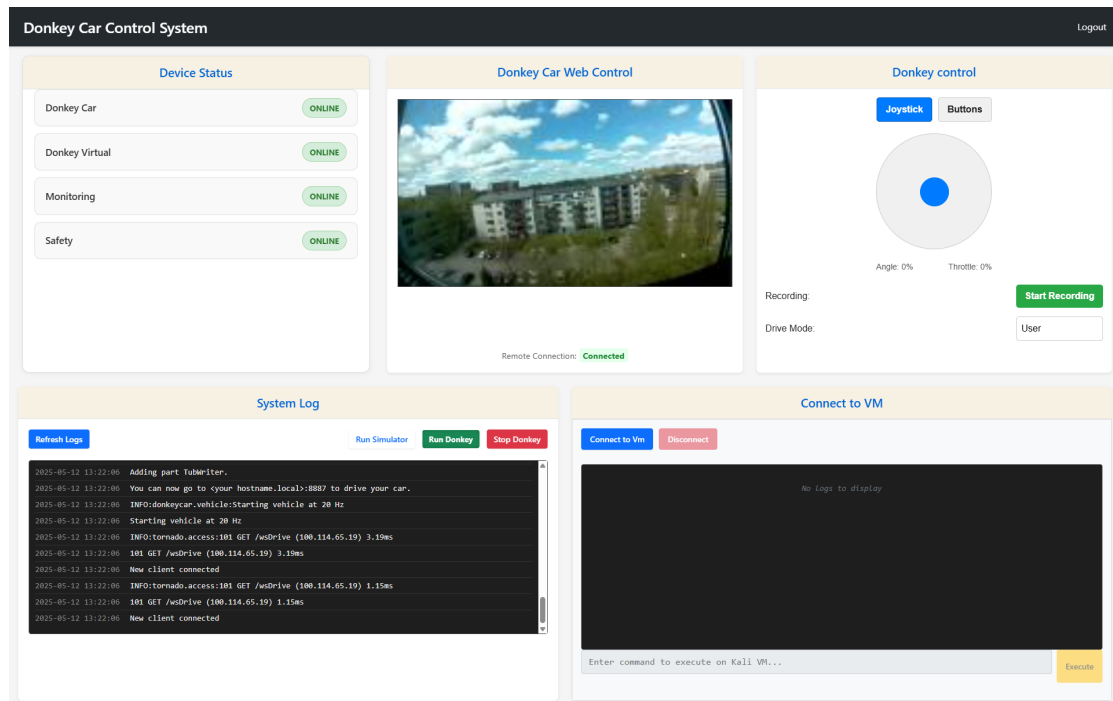


Figure 11. Donkey car connected in real-time with the controller

filters, and the third uses 128 filters - this progressive increase helps the network detect increasingly complex patterns in the images. After that convolution layer, the model converts 3D features into a 1D vector and feeds it into a fully connected Dense layer with 128 neurons. At the same time, a 0.5-layer rate dropout is used to prevent overfitting, which randomly deactivates half the neurons during training. The output layer uses softmax activation, which is used to classify the image into appropriate categories based on the number of classes defined. Adam optimizer, which uses the adaptive learning rate while training to get better convergence. The loss function is sparse categorical cross-entropy, which works well with integer-encoded class labels. This model also keeps track of performance metrics with accuracy. With this model, balance computational efficiency and sufficient depth of capture of relevant features of navigation data. Using this code, we annotated all images. We made a folder of annotated images with a JSON file, which keeps metadata of the annotated image, which we use later to play those images according to keyboard or joystick input to make a deception environment video playback. Simple image annotation and behaviour classification shown in Algorithm 4

Algorithm 4 Image Processing Algorithm

```
1: Import modules: json, numpy, matplotlib, cv2, glob, shutil, collections
2: Input: Json, Image dir
3: Output: Categorized images , Trained model
4: function READCATALOGS( $\mathcal{P}$ )
5:    $\mathcal{R} \leftarrow \emptyset$  ▷ Records collection
6:   for  $f \in \text{Files}(\mathcal{P})$  do
7:     for  $\ell \in f$  do
8:       if IsValidJSON( $\ell$ ) then
9:          $\mathcal{R} \leftarrow \mathcal{R} \cup \text{ParseJSON}(\ell)$ 
10:      end if
11:    end for
12:  end for
13:  return  $\mathcal{R}$ 
14: end function
15: function CATEGORIZEDRIVING( $\theta, \tau, \mathcal{T}$ )
16:  if  $|\tau| < \mathcal{T}_{\text{stop}}$  then return “stop” ▷  $\tau$ : throttle,  $\theta$ : angle
17:  else if  $\tau < \mathcal{T}_{\text{reverse}}$  then return “reverse”
18:  else if  $\theta > \mathcal{T}_{\text{angle}}$  then return “right”
19:  else if  $\theta < -\mathcal{T}_{\text{angle}}$  then return “left”
20:  elsereturn “straight”
21:  end if
22: end function
23: function PROCESSDATASET( $\mathcal{P}, \mathcal{D}_{\text{img}}$ )
24:   $\mathcal{R} \leftarrow \text{ReadCatalogs}(\mathcal{P})$ 
25:   $\mathcal{C} \leftarrow \{\text{“left”}, \text{“right”}, \text{“straight”}, \text{“stop”}, \text{“reverse”}\}$ 
26:  CreateDirectories( $\mathcal{C}$ )
27:   $\mathcal{I}, \mathcal{S} \leftarrow \{\}, \{\}$  ▷  $\mathcal{I}$ : categorized images,  $\mathcal{S}$ : statistics
28:  for  $r \in \mathcal{R} | \text{HasRequiredFields}(r)$  do
29:     $c \leftarrow \text{CategorizeDriving}(r.\theta, r.\tau, \mathcal{T})$ 
30:     $\mathcal{S}[c] \leftarrow \mathcal{S}[c] \cup (r.\theta, r.\tau)$ 
31:     $\mathcal{I}[c] \leftarrow \mathcal{I}[c] \cup r.\text{path}$ 
32:    CopyImage( $\mathcal{D}_{\text{img}} \oplus r.\text{path}, c \oplus r.\text{path}$ ) ▷  $\oplus$ : path concatenation
33:  end for
34:  return  $\mathcal{I}, \mathcal{S}$ 
35: end function
```

Algorithm 5 Image Processing Algorithm (continue)

```
1: function CREATETRAININGDATASET( $\mathcal{I}, n$ )
2:    $\mathcal{X}, \mathcal{Y} \leftarrow \emptyset, \emptyset$  ▷ Features and labels
3:   for  $(c, \mathcal{P}) \in \mathcal{I}$  do
4:      $\mathcal{P}' \leftarrow \text{Sample}(\mathcal{P}, \min(|\mathcal{P}|, n))$ 
5:     for  $p \in \mathcal{P}'$  do
6:        $\text{img} \leftarrow \text{LoadAndResize}(p, 224 \times 224)$ 
7:        $\text{img} \leftarrow \text{img}/255$  ▷ Normalize to  $[0, 1]$ 
8:        $\mathcal{X} \leftarrow \mathcal{X} \cup \{\text{img}\}$ 
9:        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{c\}$ 
10:    end for
11:  end for
12:  return  $\mathcal{X}, \mathcal{Y}$ 
13: end function
14: function CREATEMODEL( $k$ )
15:    $\mathcal{M} \leftarrow \text{Sequential}()$  ▷  $k$ : number of classes
16:    $\mathcal{M}.\text{add}(\text{Conv2D}(32, 3 \times 3), \text{MaxPool}(2 \times 2))$ 
17:    $\mathcal{M}.\text{add}(\text{Conv2D}(64, 3 \times 3), \text{MaxPool}(2 \times 2))$ 
18:    $\mathcal{M}.\text{add}(\text{Conv2D}(128, 3 \times 3), \text{MaxPool}(2 \times 2))$ 
19:    $\mathcal{M}.\text{add}(\text{Dense}(128), \text{Dropout}(0.5), \text{Dense}(k))$ 
20:    $\mathcal{M}.\text{compile}(\text{optimizer}, \text{loss})$ 
21:  return  $\mathcal{M}$ 
22: end function
23: function VISUALIZESAMPLES( $\mathcal{D}, n$ )
24:    $\text{fig} \leftarrow \text{CreateFigure}(|\mathcal{C}| \times n)$ 
25:   for  $c \in \mathcal{C}$  do
26:      $\mathcal{P} \leftarrow \text{GetFirstNImages}(\mathcal{D} \oplus c, n)$ 
27:     for  $p \in \mathcal{P}$  do
28:        $\text{DisplayImage}(\text{fig}, \text{LoadImage}(p))$ 
29:     end for
30:   end for
31:    $\text{SaveFigure}(\text{fig}, \text{"samples.png"})$ 
32: end function
```

This algorithm 4 takes input of a dataset and annotates the data as described:

- **ReadCatalogs:** Parses JSON files from a specified path, collecting valid records of into a dataset.
- **CategorizeDriving:** Classifies driving actions into five categories (“stop”, “reverse”, “right”, “left”, “straight”) based on throttle (τ) and steering angle (θ)

thresholds.

- **ProcessDataset:** Orchestrates the workflow by reading catalogs, creating category directories, classifying each driving record, maintaining statistics of steering and throttle values per category, and organizing images into their respective category folders.
- **CreateTrainingDataset:** Constructs balanced training data by sampling up to n images from each driving category. Images are loaded, resized to 224×224 pixels, normalized to $[0, 1]$ range, and paired with their corresponding category labels.
- **CreateModel:** Implements a convolutional neural network with three convolutional layers (32, 64, and 128 filters respectively), each followed by max pooling. The architecture includes dense layers with dropout regularization and outputs predictions for k classes. The model structure follows a typical pattern for image classification tasks.
- **VisualizeSamples:** Creates a visualization grid displaying sample images from each driving category.

The algorithm effectively transforms an unclassified image to a classified image, which uses in deception system.

5.1.7 Running application

We need to ensure that the simulator and donkey car are running and connected to the same network. And then, start the web application with authentication, which takes the user to the main page if the authentication passes. The main page view has one button, start car, which allows the user to view the web and the controller options. Using those options, the user can use the donkey car system normally and see the logs. Figure 12 and Figure 13 represent the system-generated logs while web apps and the donkey car are running.

5.2 Deceptive Environment Implementation

Deception system with environment design, and implementation started with the Tpotce GitHub project [35]. This GitHub project has several honeypots, intelligence automation, attack maps, and CyberChef with very nice visualization. To install this topic, we have chosen another virtual machine, named WorkVm. The WorkVm runs with the Linux operating system with 250 GB storage, 8 vCPU, and 16 GB of RAM as described in Table 11. Following the instructions from GitHub⁶, we installed the tpotce and ran it

⁶<https://github.com/telekom-security/tpotce>

```

PS C:\Users\mdalamin\Pictures\Donkeycar\backend> node server.js
Server running on http://localhost:8089
Connected to MongoDB
{"level":"info","message":"[CLIENT] 2025-05-12T11:24:56.730Z - user login [object Object]","timestamp":"2025-05-12T11:24:56.749Z"}
{"level":"info","message":"Login attempt with body: {\"name\":\"alamin\",\"password\":\"alamin\"}","timestamp":"2025-05-12T11:24:56.752Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:24:56.777Z - user login","timestamp":"2025-05-12T11:24:56.792Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:03.268Z - EventBus URL change to http://100.109.137.11:8887","timestamp":"2025-05-12T11:25:03.289Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:03.270Z - Selected VM changed: Simulator","timestamp":"2025-05-12T11:25:03.293Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:03.270Z - EventBus URL change to http://100.109.137.11:8887","timestamp":"2025-05-12T11:25:03.295Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:07.675Z - WebSocket connection closed: [object CloseEvent]","timestamp":"2025-05-12T11:25:07.679Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:10.130Z - WebSocket connection closed: [object CloseEvent]","timestamp":"2025-05-12T11:25:10.138Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:19.042Z - WebSocket connection opened","timestamp":"2025-05-12T11:25:19.046Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:32.857Z - Sending command: angle=0.10, throttle=0.17, mode=user, action=none","timestamp":"2025-05-12T11:25:32.857Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:32.903Z - Sending command: angle=0.10, throttle=0.17, mode=user, action=none","timestamp":"2025-05-12T11:25:32.906Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:32.953Z - Sending command: angle=0.10, throttle=0.17, mode=user, action=none","timestamp":"2025-05-12T11:25:32.956Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.003Z - Sending command: angle=0.10, throttle=0.17, mode=user, action=none","timestamp":"2025-05-12T11:25:33.006Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.050Z - Sending command: angle=0.09, throttle=0.12, mode=user, action=none","timestamp":"2025-05-12T11:25:33.058Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.107Z - Sending command: angle=0.09, throttle=0.09, mode=user, action=none","timestamp":"2025-05-12T11:25:33.109Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.153Z - Sending command: angle=0.09, throttle=0.05, mode=user, action=none","timestamp":"2025-05-12T11:25:33.156Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.206Z - Sending command: angle=0.09, throttle=0.02, mode=user, action=none","timestamp":"2025-05-12T11:25:33.208Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.255Z - Sending command: angle=0.07, throttle=0.06, mode=user, action=none","timestamp":"2025-05-12T11:25:33.258Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.305Z - Sending command: angle=0.06, throttle=0.14, mode=user, action=none","timestamp":"2025-05-12T11:25:33.308Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.354Z - Sending command: angle=0.05, throttle=0.20, mode=user, action=none","timestamp":"2025-05-12T11:25:33.358Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.387Z - Joystick position: angle=0.04, throttle=0.22","timestamp":"2025-05-12T11:25:33.390Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.403Z - Joystick position: angle=0.04, throttle=0.24","timestamp":"2025-05-12T11:25:33.406Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.406Z - Sending command: angle=0.04, throttle=0.27, mode=user, action=none","timestamp":"2025-05-12T11:25:33.410Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.440Z - Joystick position: angle=0.04, throttle=0.27","timestamp":"2025-05-12T11:25:33.445Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.453Z - Sending command: angle=0.02, throttle=0.35, mode=user, action=none","timestamp":"2025-05-12T11:25:33.456Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.470Z - Joystick position: angle=0.02, throttle=0.35","timestamp":"2025-05-12T11:25:33.474Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.503Z - Sending command: angle=0.02, throttle=0.37, mode=user, action=none","timestamp":"2025-05-12T11:25:33.506Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.519Z - Joystick position: angle=0.02, throttle=0.37","timestamp":"2025-05-12T11:25:33.522Z"}
{"level":"info","message":"[CLIENT] 2025-05-12T11:25:33.555Z - Joystick position: angle=0.02, throttle=0.39","timestamp":"2025-05-12T11:25:33.557Z"}

```

Figure 12. User generated web application control logs while running

```

System Log

Refresh Logs Run Simulator Stop Simulator

2025-05-12 14:25:18 INFO:donkeycar.vehicle:Adding part TubWriter.
2025-05-12 14:25:18 You can now go to http://localhost:8887 to drive your car.
2025-05-12 14:25:18 You can now move your controller to drive your car.
2025-05-12 14:25:18 INFO:donkeycar.vehicle:Starting vehicle at 20 Hz
2025-05-12 14:25:18 INFO:root:Recording Change = False
2025-05-12 14:25:18 INFO:root:Setting Recording = False
2025-05-12 14:25:19 INFO:tornado.access:101 GET /wsDrive (100.114.65.19) 299.35ms
2025-05-12 14:25:19 INFO:donkeycar.parts.web_controller.web:New client connected
2025-05-12 14:25:23 INFO:tornado.access:101 GET /wsDrive (100.114.65.19) 2.14ms
2025-05-12 14:25:23 INFO:donkeycar.parts.web_controller.web:New client connected

```

Figure 13. donkey apps generated info logs while running

on workvm port 64297. This honeypot can give active protection and collect data at the same time. Now, when any attempt is made to log in to the web application or use SSH, iptables and a reverse proxy⁷ reroute those requests to Tpotce. Only the valid traffic will pass through this proxy and connect to a legitimate environment. Nginx reverse proxy is used to reroute all web application requests except the application's main entry point. If an attacker runs a port scan and finds an open port on the donkey car, it takes the attacker into the deception system when they try to access it.

5.3 Security Monitoring and Intrusion Detection

In this section, we discuss the use of Tpotce for monitoring. Tpotce ensures continued visibility against any potential threat and supports forensic analysis, monitoring tools and an interactive User interface with a summarized report. All data and logs are collected and stored in a database, creating a central control and analysis system. Several great open-source and other types of services bundled into one system, like ELK stack⁸ (Elasticsearch, Logstash and Kibana), which helps to collect, process and visualize logs from all active honeypots installed in the system, like Cowrie, Dionaea, Suricata, Elastic Filebeat and others. Using those honey pots, all collected logs are sent to one centralized database with a timestamp and are later visualized with a Kibana dashboard. In the same way as another virtual machine, sending data to that database and visualizing it in the Kibana dashboard is possible. In that case, the filebeat forwarding agent works with a minimum installation.

For the detection of anomalies, we can implement the rule in Kibana or another external rule in Suricata⁹ that raises an alert when activity is not within the defined rules or with some payload that might be malicious. Real-time events from different virtual machines and honeypots are entered into the central control node, which will change and update the policies and stages. It also tracks the behaviours of the attack, and next time an attack comes from the same source, it will send it directly to the sandbox environment, in this case, the deception donkey car environment. Figure 14 shows an overview of the Kibana dashboard, which displays data for the last 30 days, and it is also possible to see data for 1 minute to see the most updated data and logs.

5.4 Testing and Validation

Testing and validation have also been done to verify that the implemented function works well and effectively as an integrated system, with all functionalities as expected. We have set up one Virtual machine with Kali Linux for testing. We have run several commands

⁷<https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>

⁸<https://www.elastic.co/elastic-stack>

⁹<https://suricata.io/>

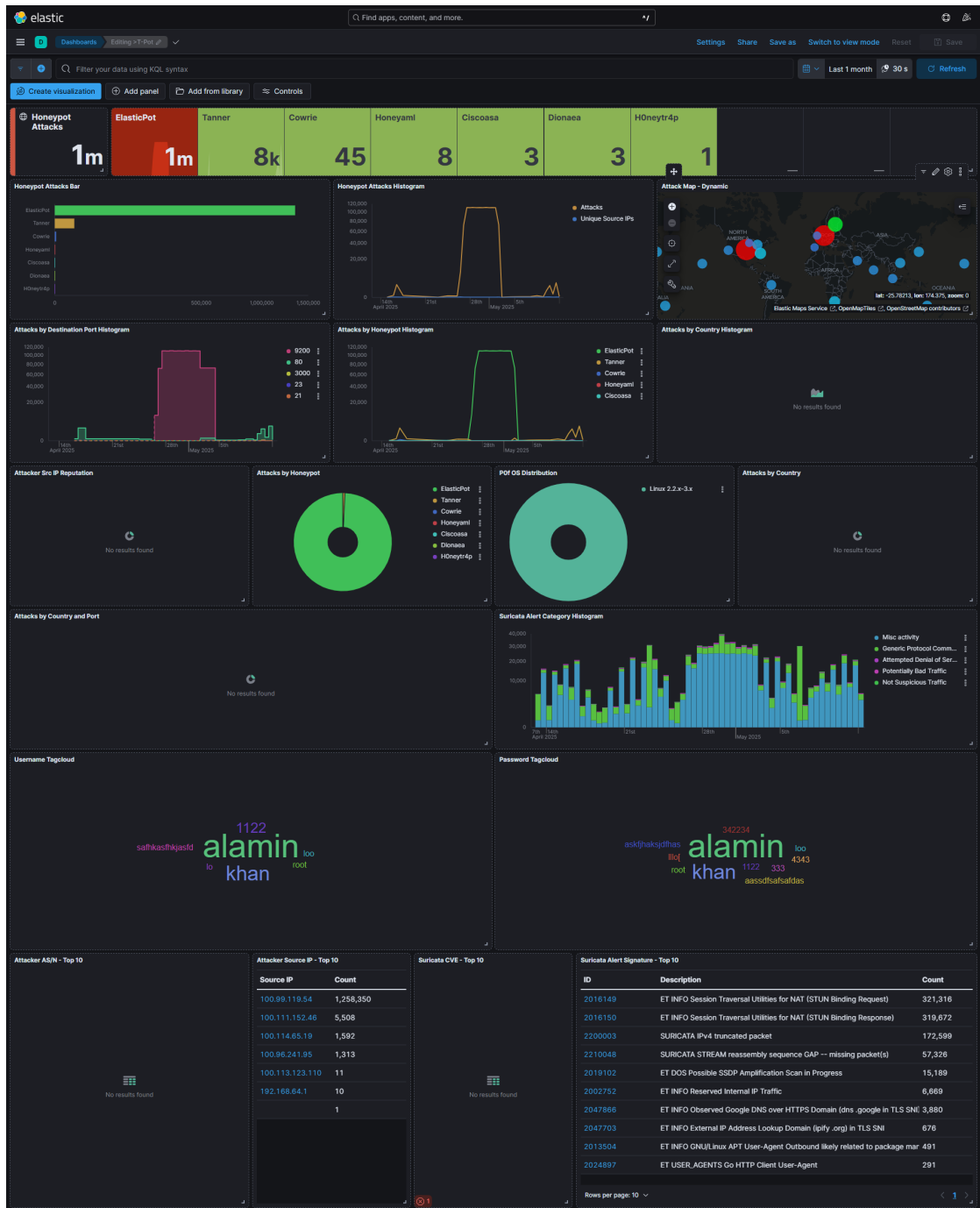


Figure 14. Kibana dashboard displaying logs and information from the past month.

and attacks from the Kali Linux system, like nmap, hydra and others, which logs are showing in Figure 14. According to our data in the Kibana dashboard, malicious traffic redirects to the honeypot. Also, Tpotce capture all SSH logging attempts with different credentials and download or upload malicious files from the internet using various methods. All the event records shows in the Kibana dashboard. Besides, the system withstands massive pressure and works well; latency and performance are excellent. For testing, verify the route, login functionality, and controller performance. Additionally, test the system using crafted network packets and analyze their flow. To enhance security, simulate various attacks on the donkey car and observe how the system responds. Finally, review all logs and visualize the results comprehensively.

5.5 Summary

To summarise this chapter, we configure a web application with Donkey car access. Backend solutions with frontend apps were fully developed by us and we also setup a honeypot by using TPOTCE. The ELK setup also helps to visualize all data. Several other important tools come with Tpot are shown in the Appendix section. Figure 16 data is collected every day, Figure 17 shows which type of data is collected, and a very useful tool, CyberChef, comes with Tpot, shown in Figure 18. Besides, we also answer the research question **RQ3: What are the practical limitations and challenges of implementing a DT-based deception framework in autonomous vehicles?** Answer to research question with with explanations is presented in Table 12.

Table 12. Answer to research question

Issue	Solution	Section
Connectivity between VMs and hardware	Tailscale mesh network	Implementation chapter 5
HW/SW version compatibility, Donkey car setup	Standardized specs and versions	Section - Donkey car and simulation setup 5.1.3
SSH access the VM	Remote run SSH command from web apps	Section -Backend and auth server 5.1.2 and 5.1.1
Web-socket connection	Robust connection among web-socket	Section -Frontend Apps development 5.1.4

6 Evaluation

This chapter goes through our research guidelines about evaluation, which we mentioned in (DG3: Design evaluation). It finds out the effectiveness of the project, the predefined rules used, and the project's outcome. This work implements two different procedures for solving the problem. One is a direct attack on the server, and the other is an attack on the web application. The actual purpose of this work is to gather as much as much information about the attacker as possible, create a profile of the attacker, update the security rules, and understand the techniques used by the attacker. The architecture has been implemented, now providing a solution to deception. This chapter discusses every step and how this proposed solution helps prevent massive damage to autonomous cars.

6.1 Incident Response Procedure

We illustrate the implementation of the Incident Response Plan (IRP¹⁰) based on the attack scenario utilizing the Deception Technology (DT). A flowchart is used to depict the IRP, as shown in Figure 15. The creation of the IRP follows the guidelines set forth by the Cybersecurity and Infrastructure Security Agency (CISA¹¹). The IRP figure is developed using the open-source tool Draw.io¹².

As shown in Figure 15, the IRP incorporates various components. The blocks in the flowchart are divided into two sections: the Deception System and Security Analyst. The arrows in the IRP are numbered from 1 to 18, indicating the steps participants take to contain the attack during the preliminary phase and effectively utilize the DT as an attack deception mechanism.

Color coding in the blocks represents the different phases of the IRP, following CISA guidelines, including Detection, Containment, Response, Remediation, Analysis, and Post-Incident Activity.

In the event of attacks on the process described in sub-section 6.2, it is important to engage the attacker in the DT to effectively collect data and protect the physical process from further damage or unauthorized modifications. The steps are explained as a real-time Incident response plan, taking an example attack scenario. All different steps are explained in Table 15. Those steps are listed below:

Step 1: First, start by choosing one of the two options.

Step 2: If any SSH request is made to the SSH server, it is directly redirected to a honeypot. In this project, we are using Tailscale, which uses a different SSH method, not port 22 or 23.

Step 3: When the attacker tries to use a web application version.

¹⁰<https://www.paloaltonetworks.com/cyberpedia/incident-response-plan>

¹¹<https://www.cisa.gov/>

¹²<https://www.app.diagrams.net>

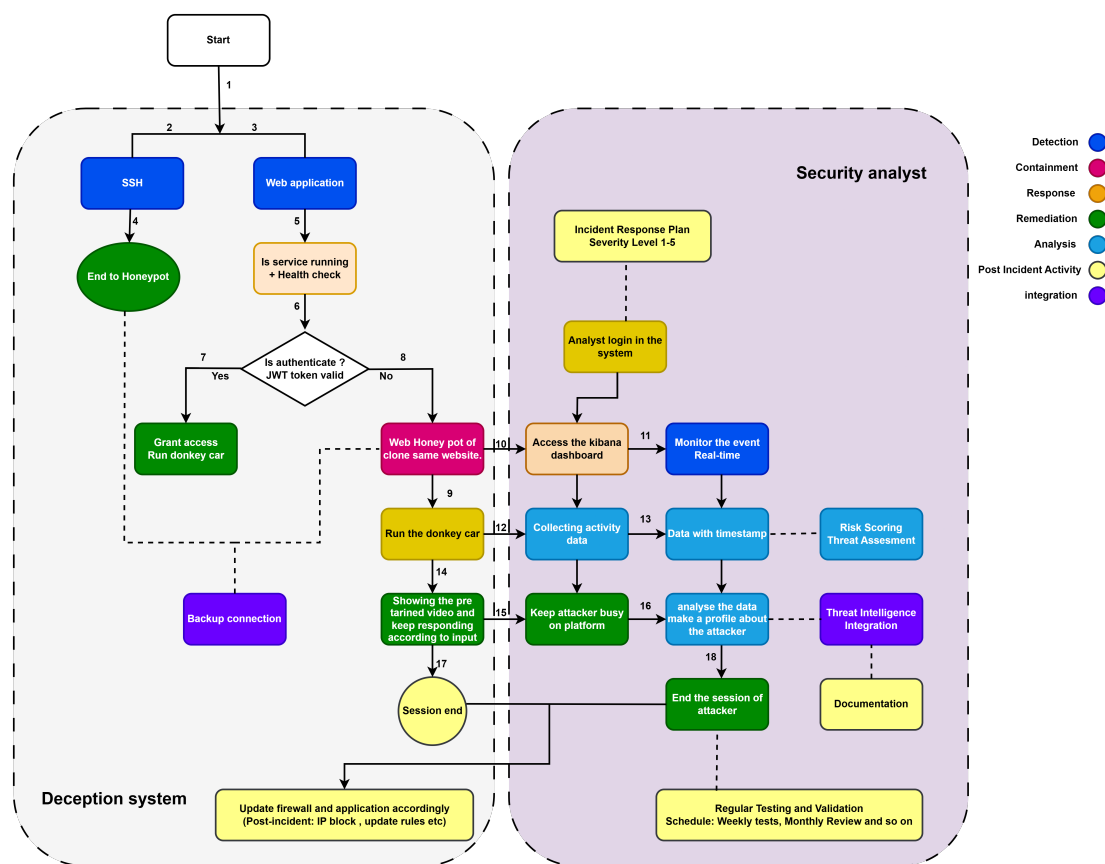


Figure 15. Incident response playbook with DT activated.

Step 4: The SSH server sends the attacker into the honeypot. This honeypot is populated with random data and logs.

Step 5: This section checks it's the service is running and the health of the service.

Step 6: Authentication is done by Node.js and JSON Web Token (JWT). This is done in the web application. There are two possibilities: one, logging in with a valid user ID or two, wrong credentials.

Step 7: With a valid credential, users have access to the real Donkey car system. It allows us to run and use all the commands. At the same time, it will open a port for streaming video and control remotely.

Step 8: On failure of authentication, it redirects to a web honeypot deception system, which is running on another virtual machine. It is a clone of the same website as the Donkey web application. However, there are some functionalities of this deception system that look alike, but under the hood, it's nothing real.

Step 9: In the web application, there is an option for running Donkey Car, which is a detection environment.

Step 10: Kibana authentication is used before using this system. In a successful login, it opens the web page where all the data is shown.

Step 11: In the Kibana dashboard, there is an option for real-time with a change setting of logs update option in real-time or 1 second.

Step 12: While Donkey Car is running, Attacker inputs some command, which is captured and displayed on the dashboard.

Step 13: All the data collected must be timestamp. Without a timestamp, that data is useless. Next time, when that data is analyzed, those timestamps are helpful.

Step 14: Pre-trained video is displayed and streamed with precision. To make a more reliable video, response of keyboard input added. If the attacker tries to go left or right, then the video responds according to that.

Step 15: Keep playing those videos, attackers believe that he/she is running the real car and try more commands that will keep them busy.

Step 16: The data collected is visualized and makes a profile of the attacker with information such as the attacker's identity, IP, intention and so on.

Step 17: At the attacker's end, the attacker can end the session, or the system disconnects automatically.

Step 18: At the end, the session can end with a kick out from the system or updating the firewall and application accordingly. Firewall rules might be updated or block the IP address.

There are some other possibilities in this project, such as Risk scoring and threat assessment, with the implementation of threat integration with third-party services. Also, the severity level of damage done by the attack can be categorized, and a plan can be set for a similar event in the future. All incidents can be documented by an analyst. Besides, regular testing and validation can be done on a weekly or monthly basis.

6.2 Scenario Testing

In this real-time scenario, an unauthorised user, as discussed in sub-section 4.5.2, possibly an attacker, takes control of the system. The attacker starts exploring the donkey car system from starting point. For this scenario testing, we assume the attacker starts with a web application.

1. **Initialized reconnaissance:** The attack started with scanning the network and discovering the web server running, which appears to be the donkey car running with a web application, which makes it possible to control the car using the dashboard of the donkey car application. (step 1, 3,5) Authentication Attempt: Attackers use login with common credentials like username-admin and password-admin, and some others. (steps 5 and 6)
2. **Honeypot engagement:** When the system detects wrong credentials and a failed login, instead of showing an error, it secretly redirects to a separate honeypot

system. The interface looks identical to what I expected the legitimate Donkey Car application to look like. (Step 8) Exploring the environment. Now, the attacker navigates to running the system and presses the run button. Deception starts the video feed and streaming logs. (Step 9) The video feed starts streaming pre-trained video when pressing run (step 14)

3. **Manipulation of the attacker:** After getting confirmation about running the application and video streaming, the attacker starts to control the car using the control environment, in this case, the keyboard or the stick; however, since this is running on a deception environment, the actual car is not running; it is responding with pre-trained videos. (steps 14, 15).
4. **Behind the scenes and an analyst:** The attacker is not informed about what is happening behind the scenes. All the logs are being recorded, and the passwords are used for logging into the donkey deception web application. All the information gathered about the attacker is being stored, including identity, IP address and so on. All the information gathered is sent to improve the system. (Step 11, 13, 16)

After interaction for a long time, the deception system collects lots of data about the attacker. Then the session ends or the attacker himself decides to leave. There are some steps that are possible to implement with third-party services, threat intelligence, risk scoring, and an incident playbook. However, this is based on a suggestion for a possible firewall update and an application modification (step 18).

6.3 Discussion of Results

This experiment of the deception system demonstrated the significant effectiveness of analyzing the attacker's behaviors and capturing logs. The designated web application successfully diverted unauthorized login attempts to web honeypot applications while trying to log in with a web application that looks like a legitimate honeypot application.

There are several findings here:

1. The authentication system successfully redirects unauthorized attempts to log in to identical honeypot applications.
2. The pre-trained video responds to keyboard input, convincing the attacker that he is interacting with a real Donkey car system.
3. Real-time data is visualized with the Kibana dashboard, which allows users to see all attack-related data.
4. Information comes with a timestamp, which is very important for any analysis to be performed later.

5. Collect and gather all the information about the attacker and other necessary information.

It is possible to improve the real-time data recorder and train it to show that as deception data. Using a third-party service or machine learning code, possible to predict threat intelligence and risk scoring. Additionally, it is possible to enhance the automatic response to such attacks.

6.4 Summary

This designed deception system demonstrates that it's able to protect the system from attack. At the same time, gather information about the attacker and make a profile of the attacker. Which is use to improve and upgrade the security system for future protection against the same type of attack. A multilayer design approach with a combination of different technologies and a simulation environment with monitoring makes the solution more effective. Although there are some improvements possible to make the existing system more affective. In this chapter, we also answer research questions from subsection 1.3-RQ4: **How does a digital twin framework compare to existing cybersecurity measures regarding resilience and adaptability to cyber-attacks?** We can assure that DT-based deception system works well to provide cybersecurity for donkey car and at the same time is also able to create an attacker profile. Where traditional security tools discussed in the literature review in Table 7 provide security specific segments.

7 Discussion

The discussion chapter delves into the multifaceted realm of cybersecurity deception systems. Especially in the context of providing critical autonomous vehicles protection. Instead of using a traditional safety detection system, we proposed a deception-based framework that works with our system seamlessly. Also integrates with several honeypots and real-time monitoring, and directs potential attackers to control the environment and feed it with pre-trained data.

Using a deception system with a redirect mechanism, all potential threats are redirected to a honeypot, analysed, and mitigated. That is how to bolster the security of autonomous vehicles against sophisticated cyberattacks. Using authentication with server-generated tokens creates an effective filter between legitimate users and potential attackers. During the login process, it efficiently filters and employs a strategic redirection approach. The redirection is executed with such seamlessness that adversaries are unlikely to detect the shift.

Using Tpotce offers several tools and technologies. The Kibana dashboard is one of them, providing real-time data in a great dashboard experience capable of different types of analysis. This tool selects one pick one specific IP address and views its details. Besides ElasticVue, where all other nodes are connected, it is also possible to see the specific node's Logstash data based on the date.

Running a deception-based system that behaves similarly to a donkey car with key input response demonstrates remarkable effectiveness, maintaining attacker engagement on the system. By responding to attacker input data, create a convincing illusion of control, keep the attacker busy long enough, and gather information that helps to build a comprehensive profile of attack methodologies, tools, and potential motivations.

This thesis has designed, implemented, and evaluated a comprehensive framework, making the work easier to understand.

7.1 Answer to Research Questions

This thesis starts with a generic question **How digital twins-based deception solutions provide autonomous vehicle security**. To investigate this question, we divide four sub-questions in Table 13. Each of the questions answer the respective section of this thesis.

RQ1 answer in the section SLR, we have found that DTs are used to provide security for autonomous vehicles. DT can be used alongside machine learning or blockchain. In subsection 3.5.1, we also address the security risks. Our findings reveal that attackers employ various methods, including physical attacks on the vehicle's instruments and remote attacks via the internet to take control. We also discuss various events and the technologies utilized by attackers. In the design and strategy section (4.3), we present

a framework that is effective against remote hacking. **RQ3** focuses on our proposed solution and its implementation.

Limitations of our presented design and implementation. Our implemented solution uses a control environment and a small donkey car. Control environment testing is not for everyone. We operate under the assumption that the attacker is using the web application; however, this may not always reflect the actual scenario.**RQ4** We discuss different types of cyberattacks and the distinct approaches attackers use. SLR, we found several solutions proposed by researchers; however, all of these proposed solutions are only about how to prevent attackers from sophisticated attacks, but do not understand the methods attackers use. The attacker intends to take over the car’s control. We have addressed those issues and found that our solution with digital-twin-based deception collects attacker data and makes a profile about the attacker, which is a different approach from other researchers.

Table 13. Research Questions and Corresponding Thesis Sections

RQ	Description	Section Addressing RQ
RQ1	How do digital twins shield autonomous vehicles from cyberattacks?	Chapter 3.4 and 6.2
RQ2	What are the security risks in autonomous vehicles? Which model or framework can provide security for autonomous vehicles?	Chapter 3.5.1 and 4.3
RQ3	What are the practical limitations and challenges of implementing a digital twin-based deception framework in autonomous vehicles?	Chapter 7.2
RQ4	How does a digital twin framework compare to existing cybersecurity measures regarding resilience and adaptability to cyberattacks?	Chapter 6 and 3.5.1

7.2 Limitations

Despite the system’s demonstrated effectiveness, several limitations must be acknowledged when considering the practical implementation of the proposed framework. In this section, we answer the research question **RQ3: What are the practical limitations and challenges of implementing a digital twin-based deception framework in autonomous vehicles?**

1. **Restricted Network Environment:** A significant limitation of this study is that we use tailscale to create a closed network environment and connect all devices

into one place using that closed network environment. Besides, we have used High performance computing (HPC) infrastructure from the university, which was restricted to use only from the university network, and they didn't allow us to make a honeypot public for their security risk. So the demonstration as well as the whole experimental setup was based on those restrictions and the experimentation was performed accordingly. In this case, more focus is on the internal threat and demonstration only.

2. **Resource requirements:** For this project, we have used resources from HPC, personal laptops, and Raspberry Pi for different purposes. However, only a single deception environment and donkey car instances are possible to create after using all of those. Load balancing or multiple instances running simultaneously is not possible for now. To run the simulator, you need a GPU, which is only possible for one running application. It may be possible to run two; however, the performance is unsatisfactory.
3. **Authentication bypassing system:** The current system assumes that a legitimate user has credentials and logs in with credentials smoothly; however, if a legitimate user puts the wrong credentials, they might fall into a honeypot. That system identifies a user as an attacker. The redirecting technique is that on a failed login in Donkey Car, it redirects to a honeypot.

7.3 Future Work

This work has revealed several promising directions for future research, addressing both the identified limitations and expanding the capabilities of the digital-twin-based deception solution for autonomous vehicles.

1. **Adaptive deception environments:** The system is more about gathering information about the attacker, which could extend intelligently in real-time to make decisions based on gathered data. A machine learning algorithm for real-time response and action. Multiple instances of the donkey car use make the system more reliable. Multiple instances boost handling multiple threats same time.
2. **Real car use:** This experiment is complete but has limited functionality, meaning it does not fully simulate a real car system or demonstrate how it operates. For further improvement, testing with a real car is necessary. However, this may introduce new challenges that we are not currently aware of.
3. **Fully automated system:** The current setup of this system is done manually on each device, which makes maintenance difficult. Implementing automated setup processes would simplify this task. Furthermore, developing advanced automated

remediation actions based on attack signatures would reduce the workload on security teams and improve response times. These actions include generating dynamic firewall rules and addressing vulnerabilities.

8 Conclusion

This thesis presents a novel approach to solving cyber threats on AVs by implementing a DT-based deception system. Using Donkey Car and TPotce with our web application. We create a robust security system for AV with backend servers and firewall rules. We have done an SLR about the DT-based AV security, autonomous vehicles' security, and so on. During SLR, we have selected several papers using defined methods, where we found the uniqueness of this work. Besides, we have collected extensive information about AV security, notable incidents, and the reasons behind the success of those attacks and damages.

To solve this problem, we have developed a web application with Vue.js, which runs on VM and connects with Donkey Car by completing an authentication on fail take to deception system. Besides, some basic status options are displayed on that web application. At the same time, images collected from the donkey car train are used to make a pre-recorded video playing option to keep the attacker on the platform. Kibana visualization is implemented to visualize all data logs and other information for a real-time view and profiling.

Our research also faced limitations and challenges, including limited experiments and scalability issues. The resources we used for this work are limited, and the research domain was not broad. Where we work, we define a problem and an area where real-life problems might not be the same as expected. Despite those challenges, significant progress has enhancing the security of autonomous vehicles.

In conclusion, our proposed design showcases the strong security potential for autonomous vehicles. The project was implemented on a narrow gap between theoretical and practical implementation. During the real-time action scenario and demo, it proved that the system would provide significant protection against any attacker and is capable of mitigating that. In this work, we focus more on the effectiveness of the project rather than making it complex.

References

- [1] Mubashar Iqbal, Sabah Suhail, and Raimundas Matulevicius. Deceptwin: Proactive security approach for iov by leveraging deception-based digital twins and blockchain. *ACM International Conference Proceeding Series*, 7 2024.
- [2] Sai Sneha Channamallu, Sharareh Kermanshachi, and Apurva Pamidimukkala. Impact of autonomous vehicles on traffic crashes in comparison with conventional vehicles. *International Conference on Transportation and Development 2023: Transportation Safety and Emerging Technologies - Selected Papers from the International Conference on Transportation and Development 2023*, 1:39–50, 2023.
- [3] Junae Kim and Amardeep Kaur. A survey on adversarial robustness of lidar-based machine learning perception in autonomous vehicles. 11 2024.
- [4] Monowar Hasan, Sibin Mohan, Takayuki Shimizu, and Hongsheng Lu. Securing vehicle-to-everything (v2x) communication platforms. *IEEE Transactions on Intelligent Vehicles*, 5:693–713, 12 2020.
- [5] Junaid M. Qurashi, Kamal Mansur Jambi, Fathy E. Eassa, Maher Khemakhem, Fawaz Alsolami, and Abdullah Ahmad Basuhail. Toward attack modeling technique addressing resilience in self-driving car. *IEEE Access*, 11:2652–2673, 2023.
- [6] Zsombor Pethő, Tamás Márton Kazár, Zsolt Szalay, and Árpád Török. Quantifying cyber risks: The impact of dos attacks on vehicle safety in v2x networks. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 8 2024.
- [7] Alan Hevner, Salvatore March, Jinsoo Park, and Sudha Ram. Design science in information systems research, 09 2013.
- [8] Choongwon Kang, Chungheon Lee, Xiangying Zhao, Daeho Lee, Jungwoo Shin, and Junmin Lee. Safety still matters: Unveiling the value propositions of augmented reality head-up displays in autonomous vehicles through conjoint analysis. *Travel Behaviour and Society*, 39:100915, 4 2025.
- [9] Xiaomeng Li, Ronald Schroeter, Andry Rakotonirainy, Jonny Kuo, and Michael G. Lenné. Get ready for take-overs: Using head-up display for drivers to engage in non-driving-related tasks in automated vehicles. *Human Factors*, 65(8):1759–1775, 2023. PMID: 34865560.
- [10] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

- [11] M. Nadeem Ahangar, Qasim Z. Ahmed, Fahd A. Khan, and Maryam Hafeez. A survey of autonomous vehicles: Enabling communication technologies and challenges. *Sensors*, 21(3), 2021.
- [12] S. M. Mostaq Hossain, Sohag Kumar Saha, Shampa Banik, and Trapa Banik. A new era of mobility: Exploring digital twin applications in autonomous vehicular systems. *2023 IEEE World AI IoT Congress, AIIoT 2023*, pages 493–499, 5 2023.
- [13] Yilong Hui, Yingmeng Li, Nan Cheng, Changle Li, Conghao Zhou, Zhou Su, and Rui Chen. Prevent deception: On-demand data synchronization for vehicle digital twins. *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [14] Na Liu, Alexandros Nikitas, and Simon Parkinson. Exploring expert perceptions about the cyber security and privacy of connected and autonomous vehicles: A thematic analysis approach. *Transportation Research Part F: Traffic Psychology and Behaviour*, 75:66–86, 2020.
- [15] Anastasios Giannaros, Aristeidis Karras, Leonidas Theodorakopoulos, Christos Karras, Panagiotis Kranias, Nikolaos Schizas, Gerasimos Kalogeratos, and Dimitrios Tsolis. Autonomous vehicles: Sophisticated attacks, safety issues, challenges, open topics, blockchain, and future directions. *Journal of Cybersecurity and Privacy 2023, Vol. 3, Pages 493-543*, 3:493–543, 8 2023.
- [16] Alireza Dehlaghi-Ghadim, Ali Balador, Mahshid Helali Moghadam, Hans Hansson, and Mauro Conti. Icssim — a framework for building industrial control systems security testbeds. *Computers in Industry*, 148:103906, 2023.
- [17] Mohammed H. Almeshekeh and Eugene H. Spafford. Planning and integrating deception into computer security defenses. In *Proceedings of the 2014 New Security Paradigms Workshop, NSPW '14*, page 127–138, New York, NY, USA, 2014. Association for Computing Machinery.
- [18] Wasim A. Ali, Maria Pia Fanti, Michele Roccotelli, and Luigi Ranieri. A review of digital twin technology for electric and autonomous vehicles. *Applied Sciences*, 13(10), 2023.
- [19] Sakshyam Panda, Stefan Rass, Sotiris Moschoyiannis, K. Liang, George Loukas, and Emmanouil Panaousis. Honeycar: A framework to configure honeypot vulnerabilities on the internet of vehicles. *IEEE Access*, 10:104671–104685, 2022.
- [20] Claudia Campolo, Giacomo Genovese, Antonella Molinaro, Bruno Pizzimenti, Giuseppe Ruggeri, and Domenico Mario Zappala. An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation. *IEEE Access*, 12:46290–46303, 2024.

- [21] Hicham El Hadraoui, Nada Ouahabi, Nabil El Bazi, Oussama Laayati, Mourad Zegrari, and Ahmed Chebak. Ieee vehicular technology society section toward an intelligent diagnosis and prognostic health management system for autonomous electric vehicle powertrains: A novel distributed intelligent digital twin-based architecture. 2024.
- [22] Yagmur Yigit, Ioannis Panitsas, Leandros Maglaras, Leandros Tassiulas, and Berk Canberk. Cyber-twin: Digital twin-boosted autonomous attack detection for vehicular ad-hoc networks. *IEEE International Conference on Communications*, pages 2167–2172, 2024.
- [23] Sabah Suhail, Mubashar Iqbal, and Kieran McLaughlin. Digital-twin-driven deception platform: Vision and way forward. *IEEE Internet Computing*, 28:40–47, 2024.
- [24] Sabah Suhail, Mubashar Iqbal, Kieran McLaughlin, Brian Lee, and Babar Imtiaz. A framework for enhancing cyber incident response with security-enhancing digital twins in cyber–physical systems. *Internet of Things*, 31:101547, 5 2025.
- [25] Md Rafiul Kabir and Sandip Ray. Vise: Digital twin exploration for automotive functional safety and cybersecurity. *Journal of Hardware and Systems Security 2024 8:2*, 8:133–144, 5 2024.
- [26] Irmina Durlik, Tymoteusz Miller, Ewelina Kostecka, Zenon Zwierzewicz, and Adrianna Łobodzińska. Cybersecurity in autonomous vehicles—are we ready for the challenge? *Electronics*, 13(13), 2024.
- [27] Abdullahi Chowdhury, Gour Karmakar, Joarder Kamruzzaman, Alireza Jolfaei, and Rajkumar Das. Attacks on self-driving cars and their countermeasures: A survey. *IEEE Access*, 8:207308–207342, 2020.
- [28] Zhi Lian, Peng Shi, and Mou Chen. A survey on cyber-attacks for cyber-physical systems: Modeling, defense and design. *IEEE Internet of Things Journal*, 2024.
- [29] Cristina Alcaraz and Javier Lopez. Digital twin: A comprehensive survey of security threats. *IEEE Communications Surveys and Tutorials*, 24:1475–1503, 9 2022.
- [30] Martin Andreoni, Willian Tessaro Lunardi, George Lawton, and Shreekanth Thakkar. Enhancing autonomous system security and resilience with generative ai: A comprehensive survey. *IEEE Access*, 12:109470–109493, 2024.
- [31] Chuanlin He, Xing Xu, Haobin Jiang, Jiachen Jiang, and Te Chen. Cyber-attack detection for lateral control system of cloud-based intelligent connected vehicle based on bilstm-attention network. *Measurement*, 247:116740, 4 2025.

- [32] Donkey Car Community. Setting up a host pc with ubuntu for donkey car. https://docs.donkeycar.com/guide/host_pc/setup_ubuntu/, 2025. Accessed: 2025-05-12.
- [33] Donkey Car Community. Creating an application for donkey car. https://docs.donkeycar.com/guide/create_application/, 2025. Accessed: 2025-05-12.
- [34] Donkey Car Community. Donkey car simulator installation guide. https://docs.donkeycar.com/guide/deep_learning/simulator/#installation-video, 2025. Accessed: 2025-05-12.
- [35] Deutsche Telekom Security GmbH. T-pot: The all in one honeypot platform. <https://github.com/telekom-security/tpotce>, 2025. Accessed: 2025-05-07.

Acknowledgement

I would like to thank both of my supervisors for patiently answering all my questions related to this work and being the kind guiding force needed for the project.

I would also like to thank my friend **Lauri Laud** for helping me to translate the abstract of this thesis.

Grammarly

During the writing of this thesis, I used a digital writing assistant called Grammarly¹³. Grammarly is an AI-powered tool designed to help users improve their writing skills by offering suggestions for grammar, spelling, punctuation, clarity, and style. I used to make immediate corrections and improve the overall quality of their writing. Spell and grammar check, Suggestions for proper punctuation, Analysis of readability and clarity. Enhancement of vocabulary.

Visual Studio Code

For developing the web application and server part of this thesis, I use Visual Studio Code¹⁴ as the primary integrated development environment (IDE). Additionally, GitHub Copilot, an AI-powered coding assistant by Microsoft, played a valuable role by providing intelligent code suggestions, generating boilerplate code, and offering real-time

¹³<https://app.grammarly.com/>

¹⁴<https://code.visualstudio.com/>

autocompletion based on context. These suggestions were critically reviewed, selectively accepted, and refined to suit the specific requirements of the project, contributing to improved coding efficiency and enhanced overall software quality.

ChatGPT

In the course of writing this thesis, we have used Chatgpt¹⁵ version V4.0. It is a type of An AI language model developed by Openai. Chatgpt was used solely as a brainstorming assistant and for formatting text into LaTeX. All core design and content decisions. were made independently. Here is an example of a prompt I have used:

"I want to run SSH commands on another computer using Python. What library should I use?"

Response: A good starting point is the **Paramiko** library, which is a widely used SSH client in Python. However, for a simpler and more high-level interface, you might want to try **Fabric**.

Fabric is built on top of Paramiko and allows you to easily connect to remote servers and run commands with minimal code.

Example usage of Fabric:

```
from fabric import Connection
```

```
c = Connection(host="user@remotehost", connect_kwargs={"password": "yourpassword"})  
c.run("ls -la")
```

To install Fabric:

```
pip install fabric
```

This auto-generated answer helps me to find out the suitable library and the most updated information I need to know. With this information, I can Google or go to YouTube on how to implement this in our code.

¹⁵<https://chatgpt.com/>

Appendix

I. Resources

Gitlab

<https://gitlab.ut.ee/md.al.amin.khan/digital-twin-based-deception-solution-for-autonomous-vehicles-security>

Demo video

https://tartuulikool-my.sharepoint.com/:f:/g/personal/mdalamin_ut_ee/Ej-Cu-hXYnpLmXOqznoaXYABk6fxtNyhqP9BoYDZhXwh_w?e=YfdR1b

II. Figures

Data collect and store in one node. Different other node also possible to add with same system shown in Figure 16.

The screenshot shows the Elasticvue 'Indices' page. At the top, there's a navigation bar with 'HOME', 'NODES', 'SHARDS', 'INDICES', 'SEARCH', 'REST', and 'SNAPSHOTS'. Below the navigation, the 'Indices' section is active, showing a table of indices. The table has columns: Name, Health, Status, UUID, Aliases, Shards, Segments, Docs, Storage, and Created. Each row represents an index, such as 'logstash-2025.05.12'. The 'Health' column shows 'green' for all indices, and 'Status' shows 'open'. The 'Created' column shows dates from 5/12/2025 to 4/13/2025. At the bottom left of the table, there is a 'BULK ACTION' dropdown menu. The footer of the page includes 'Elasticvue 1.1.0-stable | Changelog' and '©2025 - Carsten König'.

Name	Health	Status	UUID	Aliases	Shards	Segments	Docs	Storage	Created
logstash-2025.05.12	green	open	JhwERD1P2apQ1lyJd-Q	[]	1p 0r	30	36681	31.6 MB	5/12/2025, 3:00:13 AM
logstash-2025.05.11	green	open	ZKcJx--B53516x5S1kwaYg	[]	1p 0r	50	61383	54.2 MB	5/11/2025, 3:00:16 AM
logstash-2025.05.10	green	open	MW5u39PAS_KeqHFKR6gnnA	[]	1p 0r	43	75389	54.2 MB	5/10/2025, 3:00:11 AM
logstash-2025.05.09	green	open	C3ND0M7066H05F9eau-MA	[]	1p 0r	33	58482	48.2 MB	5/9/2025, 3:00:05 AM
logstash-2025.05.08	green	open	MC9H79EQvKNTeFXBoqbaQ	[]	1p 0r	35	52629	42.2 MB	5/8/2025, 3:00:06 AM
logstash-2025.05.07	green	open	N3f--sXpcTDmsqHMA9p1g	[]	1p 0r	21	32841	21.9 MB	5/7/2025, 3:00:06 AM
logstash-2025.05.06	green	open	JBFUUA4R8G5GgTfDQJ96A	[]	1p 0r	11	53277	16.4 MB	5/6/2025, 3:00:00 AM
logstash-2025.05.05	green	open	16vR0b6Tf5m7-L_90hP7A	[]	1p 0r	38	60157	39.7 MB	5/5/2025, 3:00:00 AM
logstash-2025.05.04	green	open	TzpzMLVATX6GInSr-TMfQ	[]	1p 0r	33	67873	47.3 MB	5/4/2025, 3:00:01 AM
logstash-2025.05.03	green	open	JxoxmJZ581DASEKJCl4ow	[]	1p 0r	36	63843	43.7 MB	5/3/2025, 3:00:00 AM
logstash-2025.05.02	green	open	LwRL5CZQ5-ahKDFzmDonIQ	[]	1p 0r	68	290423	151 MB	5/2/2025, 3:00:01 AM
logstash-2025.05.01	green	open	uPYPAeH-QPMXzueZhp35g	[]	1p 0r	26	365715	181 MB	5/1/2025, 3:00:01 AM
logstash-2025.04.30	green	open	YESWv37T56PhLzsqXDL-Q	[]	1p 0r	62	391801	190 MB	4/30/2025, 3:00:00 AM
logstash-2025.04.29	green	open	yoLwMy1QwMy85waaJ3VQ	[]	1p 0r	55	486648	285 MB	4/29/2025, 3:00:02 AM
logstash-2025.04.28	green	open	T0z280HqJ0W6k9H938rQ	[]	1p 0r	58	365392	187 MB	4/28/2025, 3:00:01 AM
logstash-2025.04.27	green	open	NPS4ya0hQNSZQkceC3y9	[]	1p 0r	7	358165	118 MB	4/27/2025, 4:13:06 AM
logstash-2025.04.26	green	open	g_Y35MG8B5CgTmCfKj9	[]	1p 0r	37	81385	52.1 MB	4/26/2025, 3:00:02 AM
logstash-2025.04.25	green	open	FK9vZKPS1e98z9p5_UspA	[]	1p 0r	14	23770	15 MB	4/25/2025, 3:00:03 AM
logstash-2025.04.24	green	open	8y8AZpe3yu0PMB8ZMSA	[]	1p 0r	52	41288	37.5 MB	4/24/2025, 3:00:43 AM
logstash-2025.04.23	green	open	6xexKASLSTCl61_p_cJ52v	[]	1p 0r	32	76645	35.2 MB	4/23/2025, 3:00:01 AM
logstash-2025.04.22	green	open	9q9hNIR-Q0exz7N4TyhesQ	[]	1p 0r	29	58582	44.3 MB	4/22/2025, 3:00:33 AM
logstash-2025.04.21	green	open	cJJCCv95xepFuf7JVCag	[]	1p 0r	34	58804	38.5 MB	4/21/2025, 3:00:22 AM
logstash-2025.04.20	green	open	xP0tH_05TLmpLzsqAKKsg	[]	1p 0r	48	42963	39.5 MB	4/20/2025, 3:00:24 AM
logstash-2025.04.19	green	open	KX8Mjx0FT42LQ08h5wv5Q	[]	1p 0r	15	29764	23.4 MB	4/19/2025, 3:00:01 AM
logstash-2025.04.18	green	open	WCLJesopTy-Kzmyiq451sg	[]	1p 0r	31	20911	14.6 MB	4/18/2025, 3:00:23 AM
logstash-2025.04.17	green	open	b06EV5w457-NzQLLNKumw	[]	1p 0r	16	28723	21.8 MB	4/17/2025, 3:00:00 AM
logstash-2025.04.16	green	open	av8K-ySHTRCN_Fv_ar1Vg	[]	1p 0r	27	63171	45.1 MB	4/16/2025, 3:01:53 AM
logstash-2025.04.15	green	open	S_hMfearT1st8te1fru34w	[]	1p 0r	23	55655	36.1 MB	4/15/2025, 3:01:36 AM
logstash-2025.04.14	green	open	Fkjbv78Vw3PPCLk7MBQ	[]	1p 0r	52	48172	41.5 MB	4/14/2025, 3:00:02 AM
logstash-2025.04.13	green	open	pndsyPUL758PELhARRCQ	[]	1p 0r	55	49168	41.7 MB	4/13/2025, 3:01:15 AM

Figure 16. Logs stored in Elasticvue per day

Every data detail screenshot how it looks like shown in Figure 17 . Cyber chef tool for encrypted data analysis comes with Tpotce shown in Figure 18.

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,
Digital Twin-Based Deception Solution for Autonomous Vehicles Security,
supervised by Naveed Muhammad, PhD and Mubashar Iqbal, PhD
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Md Alamin Khan
15/05/2025