Reinforcement Learning in VANET Penetration Testing

Phillip Cyril Garrad

MEng of Connected and Autonomous Vehicles

Atlantic Technological University, Sligo

Supervisor of Research: Dr. Saritha Unnikrishnan

Submitted to the Quality Qualifications Ireland (QQI)

May, 2022

ABSTRACT

The recent popularity of Connected and Autonomous Vehicles (CAV) corresponds with an increase in the risk of cyber-attacks. These cyber-attacks have been instigated by both researchers or white-coat hackers, and cyber-criminals. As Connected Vehicles move towards full autonomy the impact of these cyber-attacks also grows. The current research details challenges faced in cybersecurity testing of CAV, including access and the cost of representative test setup. Other challenges faced are lack of experts in the field. Possible solutions of how these challenges can be overcome are reviewed and discussed. From these findings a software simulated Vehicular Ad Hoc NETwork (VANET) is established as a cost-effective representative testbed. Penetration tests are then performed on this simulation, demonstrating a cyber-attack in CAV. Studies have shown Artificial Intelligence (AI) to improve runtime, increase efficiency and comprehensively cover all the typical test aspects, in penetration testing in other industries. In this research a similar AI Reinforcement Learning model, Q-Learning, is applied to the software simulation. The expectation from this implementation is to see similar improvements in runtime and efficiency for the VANET model. The results show this to be true and using AI in penetration testing for VANET to improve efficiency in most cases. Each case is reviewed in detail before discussing possible ways to improve the implementation and get a truer reflection of the real-world application.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Saritha Unnikrashnan, for her advice and guidance at our weekly meetings and via email throughout the duration of the project. Furthermore, I would like to also thank all the staff within the Connected and Autonomous Vehicles program for their teaching, help and support over the past three years of my master's programme at Atlantic Technological University, Sligo.

Lastly, I would like to thank my family and friends who looked after my wellbeing while I focused on late night coding and early morning research.

TABLE OF CONTENTS

LIST OF ACRONYMS

Acronym	Description
AI	Artificial Intelligence
BSM	Basic Safety Message
CAN	Controller Area Network
CAV	Connected and Autonomous Vehicles
CGW	Central Gateway
DoS	Denial of Service
DQN	Deep Q-Learning Network
DSRC	Dedicated Short Range Communications
ECU	Electronic Control Units
GPS	Global Positioning System
HVAC	Heating, Ventilation and Air Conditioning
I2C	Inter-Integrated Circuit
ICT	Information and Communications Technology
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
ITS	Intelligent Transportation System
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
MDP	Markov Decision Process
MITRE	Massachusetts Institute of Technology Research & Engineering
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OEM	Original Engine Manufacturer
OMNET+	Objective Modular Network Testbed in C++
PASTA	Portable Automotive Security Testbed with Adaptability
RAM	Random Access Memory
RL	Reinforcement Learning
ROS	Robotic Operating System
RSU	Roadside Unit
SAE	Society of Automotive Engineers
SFA	Single-Factor Authentication
SQL	Structured Query Language
SUMO	Simulation of Urban Mobility
TraCI	Traffic Control Interface
V2I	Vehicle-to-Infrastructure

V2P	Vehicle-to-Person
V2S	Vehicle-to-Satellite
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANET	Vehicular Ad Hoc Network
VEINS	Vehicles in Network Simulation

LIST OF FIGURES

LIST OF TABLES

STUDENT DECLARATION

By inserting my name below, I declare that this material, which I now submit for assessment, is my own work and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. To the best of my knowledge and belief, all sources have been properly acknowledged, and the assessment task contains no plagiarism.
I understand that plagiarism, collusion, and/or copying is a grave and serious offence in the Institute and am aware that penalties could include a zero mark for this module, suspension or expulsion from the Institute.

I acknowledge that this assessment submission may be transferred and stored in a database for the purposes of data-matching to help detect plagiarism. I declare that this document was prepared by me for the purpose of partial fulfilment of requirements for the Degree Programme I am registered on. I also declare that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study either at Atlantic Technological University, Sligo or another college.

Signed: Phillip Garrad Date: 08/05/22

Ethical Considerations (where applicable)

I will follow the conditions of the Ethical Conduct for research involving humans. If required, Ethical approval for this research will be obtained from the Atlantic Technological University Ethics Committee

1. INTRODUCTION

1.1 Problem Statement

The use of connected vehicles has risen in recent years and is projected to continue growing well into the future. This is due to the improved safety and efficiency in connected and autonomous vehicles [1]. However, with these benefits there is also some risk. As a connected vehicle needs to communicate with vehicles, infrastructure and other road users constantly, each of these connections have to be secure and any threats must be detected and blocked. Connected vehicles have already been exploited many times by hackers using direct or wireless connection. Typical hacks have allowed the infiltrator access to personal data about the vehicle owner and control of minor functions such as radio or peripheral controls. Sometimes the infiltrator was able to gain full access over the main vehicle components including acceleration, braking, and steering [2]. These hacks can cause fatal crashes when they are not detected and stopped. For this reason, Original Engine Manufacturers (OEMs) perform penetration testing to detect any flaws in the communications infrastructure before the vehicle is released to the public. Penetration testing is a simulated cyber-attack on an application, network, or computer system to check for exploitable vulnerabilities. To perform a penetration test a lot of resources are required, including a highly skilled engineer to design and implement the test, the appropriate tools and time to setup and execute the test.

When analysing connected vehicles there are two layers to be considered – internal communications such as CAN, I2C, Ethernet communication buses within the vehicle and external channels such as LiDAR, cameras, and vehicle-to-everything (V2X) wireless communications. The latter leverages a network named Vehicular Ad-hoc NETwork (VANET), a mode of Intelligent Transportation System (ITS). Each node in the VANET sends and receives messages between each other and any available roadside units. A typical case in penetration testing would be to introduce an attacker vehicle, or an attacker node, and send erroneous or intentionally incorrect messages to divert, disrupt or corrupt another vehicle [3] [4]. As part of the penetration test exploration the tester would send different messages in a simulated scenario and track which message would most benefit the test. The time and resources taken to do this will be addressed in this research. In many industries it is common practice to perform this level of testing using an automatic system with a built-in

Reinforcement Learning algorithm. However, this is not common practice in developing Connected and Autonomous Vehicles (CAV), as discussed in the literature review section.

1.2 Scope of Research

This research will look to address the problem statement by applying an artificial Intelligence (AI) solution to automatically create penetration tests for a VANET. This will be achieved by first creating a VANET testbed that reflects the true implementation that is used in vehicles today. Once this testbed is created a bridge will be created that will allow an external AI agent to control the attacker node within the VANET simulation. The AI agent will then test each possible message that can be sent to the network and measure the state of the system by using a Reinforcement Learning technique.

1.3 Summary of Research

Chapter 2 will review how cybersecurity is handled within CAV compared to other industries and how machine learning techniques can be integrated into penetration testing. This will first include a review of penetration testing and how it is used before looking at integration with AI. Chapter 3 will detail the methodology and the implementation of the simulated environment. It will give the reader an understanding of how the AI model communicates with the simulated environment and how to understand the outputs of the model. Chapter 4 will present the results from test and experimentation. Chapter 5 will discuss and evaluate these results further before drawing conclusions. Chapter 6 and Chapter 7 will present the works cited and appendices respectively.

2. Literature Review

This chapter reviews the findings of the literature review. Section 2.1 will detail cybersecurity in general and how penetration testing is performed. It will then detail how AI is applied to improve the efficiency of penetration testing. This will be followed by section 2.2 which will detail the cybersecurity measures, common vulnerabilities a penetration testing in CAV. This will be followed by reviewing current uses of AI in CAV before finally finishing the chapter with some conclusions.

2.1 Cybersecurity

Cybersecurity is a continuously growing field as society continues to advance in the Information and Communications Technology (ICT) industry. With the growth of the Internet of Things Technology (IoT), connected devices and systems are becoming more popular in public areas (Mobile phones, Smart City devices), homes (Google Home, Home Surveillance) and on the road (Connected Vehicles). The increase in connected technology corresponds to a greater need for cybersecurity techniques to safeguard our systems from any kind of information disclosure. Cybersecurity should be a consideration when developing any computer or network related product [5]. There are several main types of cyber-attacks as detailed below [6]:

- **Denial-of-Service (DoS) -** This type of attack over utilises the resources on the system making it inaccessible to others. This is achieved by sending many requests to the target server, flooding the network server with traffic. These requests are illegitimate and have false return addresses which mislead the server when it attempts to authenticate the requestor [7].
- **Malware –** The term “Malware” is derived from “Malicious Software”, meaning it is software that causes damage, disruption, or unauthorized access to a computer system. The attacker deploys the malware software on the victim’s computer typically to steal, damage or destroy data. The intention may also be to compromise computers on the further connected network. Malware includes many subgroups including virus, worms, trojan virus, spyware, adware, ransomware. Each of these subgroups reflect the various intentions [8].
- **Phishing –** Phishing leverages social engineering to collect sensitive information about an organisation or an individual system/user. This is done by opening communications by

social media, email, web pages, cell phone, pop-up messages, instant messages among others to collect data from users. The attacker typically pretends to be a trusted company to build the victim's trust. The attacker then asks for some private credentials or may ask the victim to download a file, posing as a software update but in fact it is malware [9].

- **SQL Injection Attack –** This type of attack is specific to databases. It is where the attacker inputs a SQL statement which can lead to unauthorized access and manipulation of the database. This can expose sensitive information or lead to data loss [6].
- **Session Hijacking and Man-in-the-Middle Attacks –** Man-in-the-Middle attacks are a type of eavesdropping attack. This is where an unauthorized third party secretly gains control of the communication channel between multiple endpoints, observing the data being shared, or corrupting the data for the attacker's advantage [10].

Over the years there has been a wide variety of methods, or defence strategies, used to enhance cybersecurity of computers and networks. While Table 1 represents a few of these defence strategies, there are many more mechanisms available [11]. In time, the best practices around these defence strategies can change as cybercriminals learn and develop their attack approach. For example, in many systems a single password is used for user authentication. A study showed that people commonly use simple passwords or use the same password for multiple systems and so their user login is not very secure [12]. These Single-Factor Authentication (SFA) could be broken through dictionary attacks, or social engineering attacks. Dictionary attacks is where a script attempts to try all words. Two, or multi-factor authentication, is a good defence strategy for securing user login details. By using two or more passwords, or security tokens, the account has a much smaller chance of being hacked successfully. The main factors of user credentials are, Knowledge, Ownership and Biometric. These factors are inputted respectively by secret password, smartphone token or card, and biometric data such a fingerprint [13]. Multi-factor is becoming the norm with high-risk accounts such as an individual's bank account.

Table 1: Defence Strategies to de-risk cyber-attacks

Defence Strategy	**Description**	**References**
Fuzzing	Allows detection of software safety errors introduced by Man-in-the-Middle attacks.	[14]

Encryption	Process of encoding information, converting original plaintext into a ciphertext with authorized bodies reverting the cipher.	[15]
Obfuscation	Obfuscation is used to obscure the meaning of the message by making it difficult to understand.	[16]
Anti-Malware	These systems monitor and scan for malware software and remove it.	[17]
Firewall	Monitoring incoming and outgoing network traffic and blocks untrusted traffic based on restrictions.	[18]
Access Control	This involves user authentication and Multi-factor authentication as detailed in the previous.	[19]

2.1.1 Ethical Hacking

Penetration testing, also known as ethical hacking, has been around since the late 90s. Automotive penetration testing is a controlled attack on automotive software to find any vulnerabilities and access potential damage that can be caused by an attack [20].

There are a few steps to performing a penetration test [21].

1. First the hacker must find an entry point. There are several ways to achieve this; the hacker may have login credentials, or use a brute force attack, or pretend to be from a trusted IP, among other methods.
2. Once the hacker has infiltrated the device or network, they can start the penetration test. At this point the hacker can start to target other connected segments of the device or network. The hacker could perform a man-in-the-middle/eavesdrop attack to launch an advanced persistent threat test or spoof the system to gain further privilege at this point.
3. Exploit – the hacker builds on the knowledge they have gained and can either disconnect and complete the hack or use their findings to exploit the network further. For example, if they have gained elevated privileges from spoofing, they can now access more data.
4. Performing an advanced persistent threat is the ability to access a device or network, maintain it and access valuable data without being detected. This is the most dangerous attack of them all [21].
5. The last step is exfiltration, or to “vanish without a trace”. This involves disconnecting while masking or removing any trace of being there.

Execution of the five steps describes a very successful penetration test, however being able to find a vulnerability as in step one can be difficult. There are many software tools to help with this, including Kali Linux, Nmap, Nessus and Wireshark.

- **Kali Linux:** This is a Debian-based Linux operating system that allows pen testers to perform the same malevolent attacks as hackers with relative ease. [22]
- **Nmap:** Performs scans of networks to discover everything that is currently in the network and information about each connected port. It also details the versions of each port service. This information may be used to exploit a known vulnerability. [23]
- **Nessus:** Nessus is a vulnerability scanner that scans a computer and reports any vulnerabilities in computers connected to the shared network. [24]
- **Wireshark:** Wireshark is a network protocol analyser, meaning it is used to review all messages in and out of your network with a very high level of detail. [25]

2.1.2 Artificial Intelligence

Traditional penetration testing methods are becoming less favourable in recent years due to resource consumption and the variance between systems. To ensure equal or better testing, AI has become a popular alternative method [26]. Artificial Intelligence (AI) has been used on several occasions to enhance cybersecurity defences [27], [28]. In 2019, McKinnel et al. [29] completed a systematic literature review on the increase of artificial intelligence in penetration testing and vulnerability assessment was complete. This review gave many examples of how AI is an effective tool in both vulnerability assessment and penetration testing. Some of the recommendations given were to use a specific system rather than being overly abstract. An area of suggested research that was presented is to investigate an AI-based approach to identify vulnerabilities that are exploitable, and to what degree.

[29] gives an overview on some common AI models used. The detail of the independent variables and AI models used across the 31 relevant papers analysed. In total there were 10 independent variables – these were problem size, number of hosts in exposure, genetic generation, training epoch, network state, number of objectives, action model, AI engine, connectivity, and vulnerabilities. Similarly, there are 10 different AI models used across these papers. The majority had a common approach that encompassed some degree of attack planning via attack graph generation, or attack tree modelling, or another form. Markov Decision Process (MDP) was used for attack graph generation approaches. From the attack

graph the AI model could then be determined. The most popular approach was Partially Observed MDP as nine of the papers analysed utilised it. The second choice, with four papers leveraging it, was the fast-forward model with some using contingent fast-forward model to enhance the results. A Fast-Forward model is one that starts at an initial state and expands a search tree until it reaches its goal state. The meta-analysis concluded this was the highest performing group of models for generating attack plans. Other less commonly used techniques were Multiple Value, NuSMV (Model Checker), genetic evolution, Reinforcement Learning. Some papers used a combination of models. Since McKinnel et al. [29] review paper was published in 2019, there has been further papers published supporting the use of Reinforcement Learning as an AI engine for penetration testing. Reinforcement Learning is discussed in further detail in section 172.2.6. In 2020, Hu et al. [30] suggest using Deep Reinforcement Learning for penetration testing. This deep reinforcement learning technique leverages the Deep Q-Learning Network (DQN). Using an attack tree methodology, a reward system is constructed and used to train the DQN. This case achieved an accuracy rate of 86% for selection the correct attack route from the attack plan.

2.2 Cybersecurity in CAV

Connected and Autonomous Vehicles (CAV) are becoming more popular in recent years resulting in a proportionate increase in the number of cyber-attacks [31]. There are multiple networks to be considered when discussing CAV networks:

- **Internal Network –** Internal or on-board Connected Vehicle technology includes an extensive cable network that connects to the powertrain, vehicle control, HVAC controls, on board diagnostics, multimedia system, on board sensors and telematics among others.
- **External Network –** Consists of all the external communications including Bluetooth for a phone or tyre pressure monitoring system, GPS signal, wireless communications dedicated short range communications (DSRC), 5G among others. The external network of each vehicle is part of a much larger network for this research, where each vehicle is an individual node. The larger network also may contain roadside units such as traffic signals, traffic cameras etc. External networks are able to perform

communications between vehicle-to-vehicle (V2V) or vehicle-to-everything (V2X). Figure 1 shows a typical larger external network [32].

Figure 1: Typical External Network in a city [32].

The external network is openly accessible to many public users and so there is more threats than an internal network. Figure 2 shows several potential eavesdropping attack modes including V2X and Vehicular Cloud Networks [33]. In the case of V2V, Vehicle-to-Person (V2P) and Vehicle-to-Infrastructure (V2I), users are actively transmitting data so any user within range can receive messages, including confidential information. When a satellite is involved, i.e., Vehicle-to-Satellite (V2S), the wrong recipient may receive the data if it is being used as a relay node as great distances are involved. Vehicular cloud networks communication shares a lot of data including weather updates, traffic updates, media data however it also contains user data which maybe overheard if the messages are decrypted.

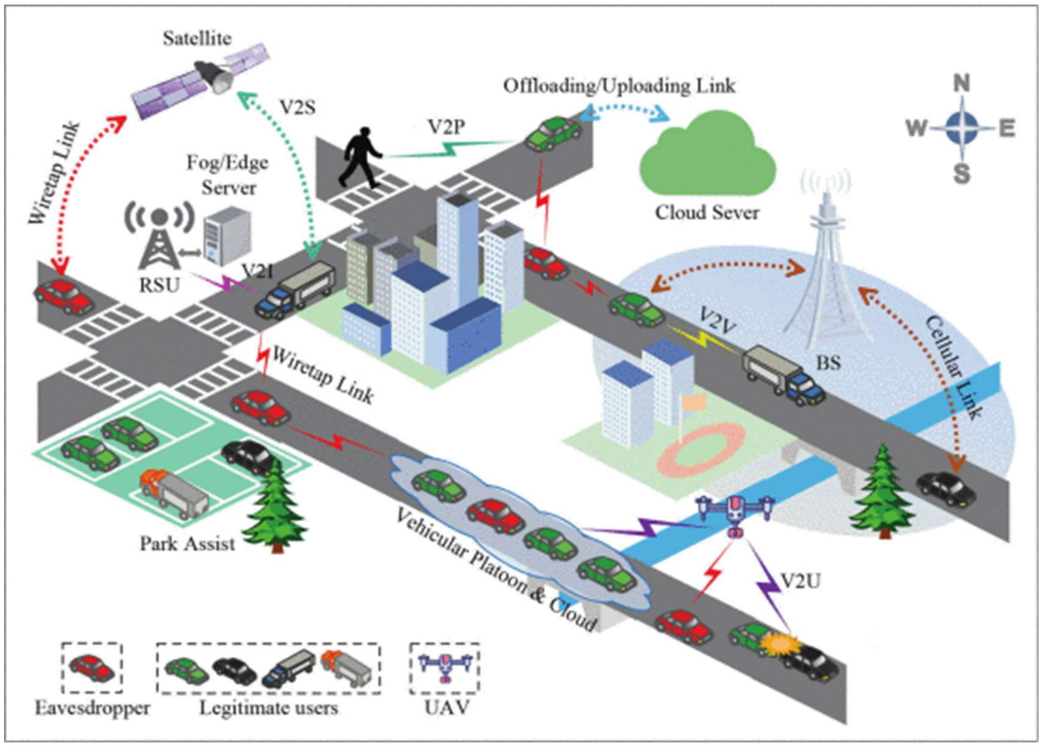


Figure 2: Major security threats in external CAV networks [33].

To mitigate these types of attacks discussed previously there has been several improvements made to cyber security for CAV. A few are listed in Table 2.

Table 2: Cybersecurity Improvements in recent years

Key Terms	**Description**	**References**
Fuzzing tools	Fuzz testing is an automated software testing technique which enables testing of various boundary test cases.	[34]
Lattice Model	Lattice Model network for V2X communication relies on continuous feedback to suppress cyber-attacks	[35]
Anomaly Detection	Use of anti-virus scanners to detect when a cyber-attack has occurred and flag or disconnect from source of attack.	[36]

Each of these approaches have several strengths and weaknesses. The anomaly detecting is a more repair method as it typically takes action after the attack has already started. In some cases, the damage has already been done and repairing the attack point will have little benefit to the vehicle. The fuzzing tools have been proven to work effectively on several projects however there is the risk that data may be lost or corrupted. The lattice model leverages redundancy to over communicate, that have proven to typically be an effective way to develop automotive systems. However it can cost more power and result in some noise.

2.2.1 Regulations

To protect organizations and to follow best practices there are regulations set out by International Organization for Standardization (ISO) to maintain high-level safety. One key element of the ISO guidelines for computer security covered under ISO15408 is that cybersecurity must be considered in the full organizational process [37]. There are ISO and Society of Automotive Engineers (SAE) regulatory documents that focus directly on cybersecurity in vehicles including ISO/SAE 21434, Road vehicles - Cybersecurity engineering [38] and SAE Guidelines, SAE J3061 [39]. These regulations must be met and the product certified for an automotive product to be released. Before 2021 there was no specific ISO for regulating cybersecurity in automotive as ISO26262 focused only on functional safety and SAE J3061 is only a cybersecurity guideline. This changed with the release of ISO21434. That means vehicles released before 2021 were developed with significantly less regulation for cyber security. The introduction of ISO 21434 has enabled the improvement of vehicle security by ensuring new technologies in the automotive industry meets security standards [40].

2.2.2 Current Vulnerabilities

In recent years there have been several vulnerabilities found and exploited in CAV technology by both cyber criminals and white hat hackers. Some of these have since been addressed by automotive manufacturers. These vulnerabilities are primarily focused on network related weaknesses as these are more relevant to the current research. The National Vulnerability Database (NVD) maintained by the National Institute of Standards and Technology (NIST) is an excellent source of tracking the vulnerabilities discovered. A summary table is shown in Table 3 focusing on the five largest auto manufacturers [41] by revenue. Tesla is included in the table due to its world-renowned connectivity and public interest.

Table 3: Vulnerabilities from the NIST's NVD [42]

Automaker	**Vulnerability type**	**Count**
Volkswagen*		**2**
	Session Hijacking	1
	Man In the middle attack	1
Toyota		**4**
	Denial of Service	1
	Non-Critical Session Hijacking	4
	Man In the middle attack	2
Daimler		**8**
	SQL Injection	1
	Session Hijacking	3
	Man In the middle/Eavesdropping attack	2
	Denial of service	2
Ford**		
General Motors		**3**
	Man In the middle attack	3
Tesla		**15**
	Man in the middle attack	1
	Session Hijacking	8
	Malware	1
	Denial of Service	5

*Audi is part of the Volkswagen group however due to the limits of the NVD filtering there were over 2000 hits when Audi was entered. Only a small percentage of the 2000 are relevant to this paper so none were considered

** Similarly, Ford is a generic term in the NVD and could not be filtered further

Table 3 offers insight into the vulnerabilities in the world and what is being reported. These attack types are described in more detail in section 2.1.1. In the case where malware is installed or Session Hijacking occurs, these are the high priority vulnerabilities for CAV. In the case of a functional behaviour change (i.e., drivetrain) there is risk to human life. If it is a denial-of-service attack it is unlikely to affect functional safety but may cause massive inconvenience. Some of these vulnerabilities are under dispute or have since been fixed but there are still some common trends. As shown in Figure 3 there is a trend of more vulnerabilities being found and reported in recent years. The fact that there are the same number of published vulnerabilities in this database for 2021 and 2018 shows that vehicles may have more vulnerabilities, or more are being discovered than before. As most of the vulnerabilities listed in Table 3 are related to new features, connected phones, over the air updates, remote key fob control etc, a correlation between new connected features and an increase in vulnerabilities is demonstrated.

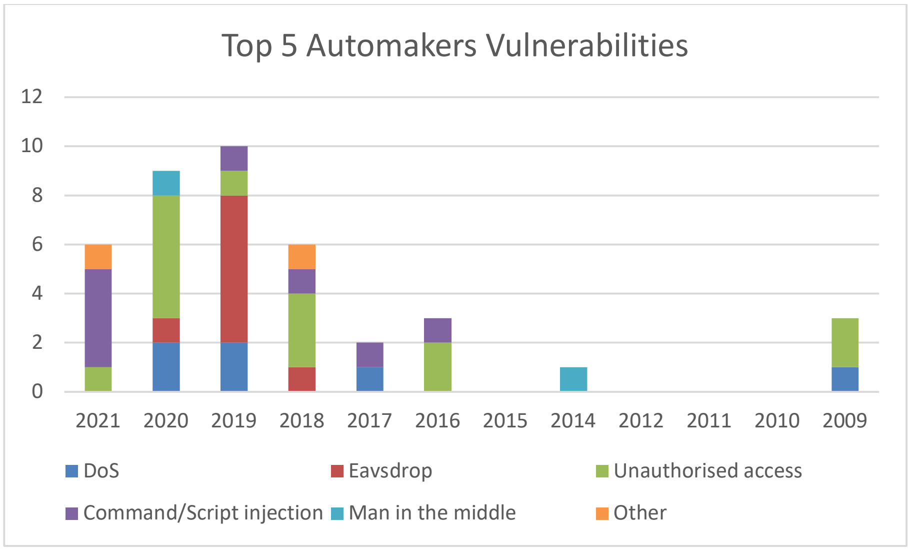


Figure 3: Top 5 Automakers Vulnerabilities in recent years [42]

The vulnerabilities listed in the NVD are documented and recorded by MITRE and other similar organizations. Meaning these vulnerabilities are found after-market where the vehicle is already publicly available. This highlights the need for more extensive testing by the automaker pre-market.

2.2.3 CAV Network Software Simulation

This section reviews research to find a suitable testbed simulation which can be used for completing the current research. Initially both software and hardware solutions were considered but a software solution would be preferable due to time and cost constraints. From findings an appropriate testbed would then be used for this study.

Fowler et al. [43] showed promising results that referred to a software simulation tool called CANoe, produced by Vector Informatik, a manufacturer of network debug and analyser tools. However very brief and did not detail how this simulated vehicle model was produced through CANoe or how the hardware was then connected to set it up as a useable testbed. Costley et al. [44] leveraged ROS to create a software and hardware solution. From the title this paper expresses a low cost open-source testbed to enable Automated vehicle research however on further reading the product is merely an interactive way of controlling a real vehicle. It is not a software simulation and requires full access to a vehicle for use and so is not an appropriate solution. From this a leaner search for software simulated solutions was taken.

Some auto manufacturers have also tried to introduce cost effective testbed solutions that do not require access to a vehicle. Toyota produces a Portable Automotive Security Testbed with Adaptability (PASTA) in 2018 [45]. PASTA consists of a hardware representation of a connected vehicle, involving 4 Embedded Electronic Control Units (ECUs); a Central Gateway (CGW), and 3 ECUs to control powertrain, body and chassis domains. These are then connected using CAN to inputs. The input control can use either the manual input or the inbuilt software. From the initial release of PASTA, this looked to be an effective testbed and further backed by Baar [46] and Higgins [47]. One drawback of PASTA as mentioned by Baar [46] and further discovered by some price comparisons online is that the testbed is ~$28,000 which leaves it out of reach for most individual researchers and hence is unsuitable for the current research. Despite PASTA being marketed towards researchers, the price positions it towards researchers within automotive companies. Baar [46] provided a prototyped alternative to PASTA at a much lower budget of approximately €398 which used raspberry pi for the ECUs and a CAN bus for streaming data between them. This was certainly a cheaper option, but it sacrificed the amount of data and control that was transferred within the vehicle, i.e., it was not setup for additional inputs such as drivers' inputs, nor was the ECU programmed to handle typical vehicle systems, such as powertrain, body or chassis control like PASTA.

As an appropriate vehicle testbed was not obtainable without further developing a solution the search parameters were re-evaluated. Instead of viewing messages on an inter vehicular level, viewing the vehicles on a nodular level provided significantly more hits [48], [49]. Simulation tools such as SUMO, OMNET+, VEINS and INET were used in these cases to build a working simulation of VANET. A VANET consists of groups of moving or stationary vehicles connected by a wireless network. VEINs break into the following software components [50]:

- **OMNET+** - "Objective Modular Network Testbed in C++" is a simulator for building network simulation. This can create nodes and communicate between them.
- **SUMO –** "Simulation of Urban Mobility" is a road traffic simulator which can build maps of roads and place vehicles on them, handle traffic, speeds etc.

- **TraCI –** "Traffic Control Interface" is a technique for interlinking traffic and road networks.
- **VEINS –** "Vehicles in Network Simulation" is an open-source framework which leverages OMNET+ and SUMO, with OMNET+ performing the network simulations and SUMO the traffic simulations. TraCI is added to VEINS as a module to understand the influence of network on traffic pattern.

Proceeding forward with VEINS as the simulation tool of choice identified other relevant studies in applying penetration testing using VEINS. Some of these are reviewed in the next section.

2.2.4 Penetration Testing

Penetration can be done by the automaker or by an outside team that specialises in penetration testing [20]. Using an outside team can be beneficial as they will have limited or no knowledge of the product and will have the same access as a cybercriminal. As more connected intelligence is added to vehicles, testing must also be amplified. However, there are several challenges faced in performing real-world penetration testing on connected vehicles.

- **Environment control -** Understandably the environment needs to be extremely controlled to guaranteed safety should the vehicle lose control. In performing a test, you don't want to interrupt other vehicles in the area by blasting repeating signals to generate noise. Some automakers overcome this by using remote test facilities or by using a large faraday cage [51].
- **Aggression and competition between automakers –** As the world's leading automakers are competing, there is insufficient collaboration when it comes to setting standards. To win this competition, automakers push for unrealistic deadlines when it comes to ensuring cyber resilience in their vehicles [52].
- **Skilled workers –** The skillset required to be an expert in cybersecurity, automotive and testing is specialised resulting in insufficient talent availability. To overcome this challenge some vehicle manufacturers such as Tesla have now started to encouraged academics and the public to attempt penetration testing on their vehicles to expose any vulnerabilities [53].

Penetration testing is similar to other test types and requires a clear framework when being applied. There are many different methods used on how to plan and record a test, but in general a framework like test process documented in [54] should be used. This framework is:

1. Define Item
2. Perform Risk and Threat Analysis
3. Define Security Concept (testing requirements)
4. Plan Test and Develop Scenarios
5. Select Test Scripts
6. Generate Test Cases
7. Perform Test
8. Generate Test Reports.

This template is flexible to be used in various test environments and robustly record the relevant data. Similar processes were followed when performing penetration testing using VEINS in [55]. VEINS is used in a study of an Intelligent Transport System (ITS) to simulate a Misbehaviour Detection feature [55]. While performing the research, the team used VEINS to perform V2X attacks, which they would then later run against their Misbehaviour Detection algorithm. The V2X attack types simulated were DoS, Disruptive, Data Replay, Eventual Stop, Congestion Sybil, and Misbehaviour Authority Stress.

Table 4: Attack scenarios implemented as Open Source in a VEINS project [55]

Attack Type	Description
Denial of Service (DoS)	In the simulation the attacking vehicle increased the frequency it would send messages. It can also toggle between valid data and random data being transmitted.
Disruptive	The attacker attempts to flood the network by replaying old data received from a beacon. Simultaneously the attacker also increases the transmission frequency to maximize the negative effects on the network. As the data comes from a trusted beacon originally it may appear as valid.
Data Replay	The attacker chooses a target and replays data with a certain delay so an observer may think there are 2 vehicles following each other.
Eventual Stop	After a certain time delay the attacker stops sending signals to beacons and sets speed to zero giving the appearance of a sudden stop.
Congestion Sybil	The attacker introduces a ghost vehicle, the attacker transmitting data in place of the ghost vehicle.

Misbehaviour Authority Stress	The attacking vehicle targets the Misbehaviour Authority (managing server) by sending falsified reports, containing identities of neighbouring vehicles or random data to confuse it.

Kamel et al. [55] published their VEINS project and attack scenarios as opensource on GitHub. From their findings, the Disruptive attack type proved to be hardest to detect as an attack and so it is a strong penetration test, reusing trusted messages as noise. They also created a Python/C++ bridge to allow import of AI algorithms.

2.2.5 Artificial Intelligence

Building on from section 2.1.2 on how artificial intelligence is used in other industries, this section will focus only on AI in automotive cybersecurity. AI is not a brand-new concept to automotive cyber security. It has proven to be a useful method in cyber defence. A European project CARAMEL [56] uses advanced AI techniques to detect cyberthreats to the internal and external perception modules. Kyrkou et al. [56] shared little information as to the type of AI model used, which would have been useful information for the current research. Kamel et al. [55] use artificial intelligence for advanced misbehaviour detections but again does not detail the algorithm used.

Another application discussed in section 2.1.2 of AI and machine learning is automating the process of finding vulnerabilities in a system's network. In some cases, reinforcement learning, a machine learning algorithm that learns through trial and error of its environment, is used as an AI solution for penetration testing in general cybersecurity [57]. The focus of the current research is reusing these techniques for automotive. Mckinnell et al. [58] showed a wide variety of AI models are available to be used in penetration testing but the challenge is to find the best types to fit the simulation.

2.2.6 Reinforcement Learning and Q-Learning

Reinforcement Learning is a technique that enables the AI agent to interact with an environment and learn from trial and error of the received result [59]. Hanem et al. [26] states that the use of Reinforcement Learning provides better time efficiency, reliable outputs, accuracy, and covers more attack vectors when used for penetration in general, i.e. not CAV specific. Reinforcement is basically a Markov Decision Process where the agent performs an action based on the observations and feedback from the environment, which is illustrated in Figure 4. After each loop the outputs are stored in a learning table. The next time the same observation is made so that the agent can predict the feedback/reward for a given action. The principle of Reinforcement Learning is all actions are tested for all available observable states and the learning table is updated. Each run of the environment from start to finish is known as an episode.

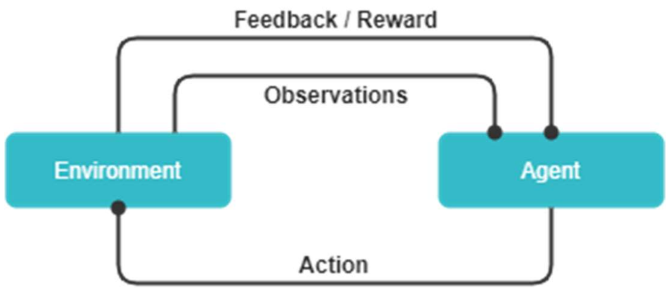


Figure 4: Markov Decision Process for Reinforcement Learning [60]

Q-Learning is a reinforcement learning algorithm that seeks to find the best action to take based on the observed state. The best action refers to the action that will return the highest reward. The learning table, or Q-Table, stores the best action based on the training so far. In each step of each episode the best-known action is selected from the Q-Table or, randomly, a different action is selected. If the random action receives a higher reward the Q-Table will be updated with this new action.

2.3 Conclusions from Literature Review

By applying the methodology learnt from this literature review, some of the challenges in the CAV cybersecurity area can be overcome. From reviewing the current literature on AI in cybersecurity and cybersecurity in automotive it is clear that there is a gap in using AI in automotive cybersecurity. When it comes to testing cybersecurity there is clear benefit set out in industries of leveraging AI models to improve runtime and to establish the best attack plans. From the meta-analysis review, it is clear AI models with an established attack plan had a better chance of success. An attack tree should also be designed to complete the process. Using a Markov Decision Process partially or fully observable was a common approach taken in other industries and proved to have positive results. Applying a Reinforcement Learning technique is hard due to the complexity of an automotive environment. However, creating a simulation environment to evaluate some selected scenarios would be the best approach to test the potential of Reinforcement Learning models in CAV security. This work could be expanded to other areas of CAV if proved beneficial.

Regarding the simulation, the open-source software VEINS is fit for purpose and previous projects using it have performed part of what the research aims to achieve. Implementing and creating open-source versions of simulated cyber-attack scenarios [55] gives this project an excellent start point for understanding the simulation environment and the python/C++ bridge allows AI models to be implemented in python and injected into the simulation.

3. Methodology

This chapter will initially review the general Design and workflow before taking a closer look at the technical details of the environment. Towards the end of this chapter implementation details of the simulation in both VEINs and OpenAI Gym will be presented. Lastly the integration of the Q-Learning Reinforcement model will be shared.

3.1 Research design

To apply any type of AI Algorithm to a penetration testing of a VANET system, the VANET network must first be simulated. From Chapter 2 VEINs is determined as a good candidate for building a VANET simulation. Once a VANET simulation is designed an attack vehicle is then added. The AI model would instruct the attack vehicle to perform actions based on the simulation observation. The simulation would then provide the current outputs to the AI model and the next action would be determined by the AI algorithm. The intended output from the simulation was the Basic Safety Message (BSM) coming from vehicles and Roadside Units (RSUs) within the simulation. Since the AI model action would include the vehicles response to the BSM, which has many possibilities that may affect the simulation. If the attack vehicle sends a BSM alert message about a crash on a particular route it would potentially divert other traffic as shown in in Figure 5.

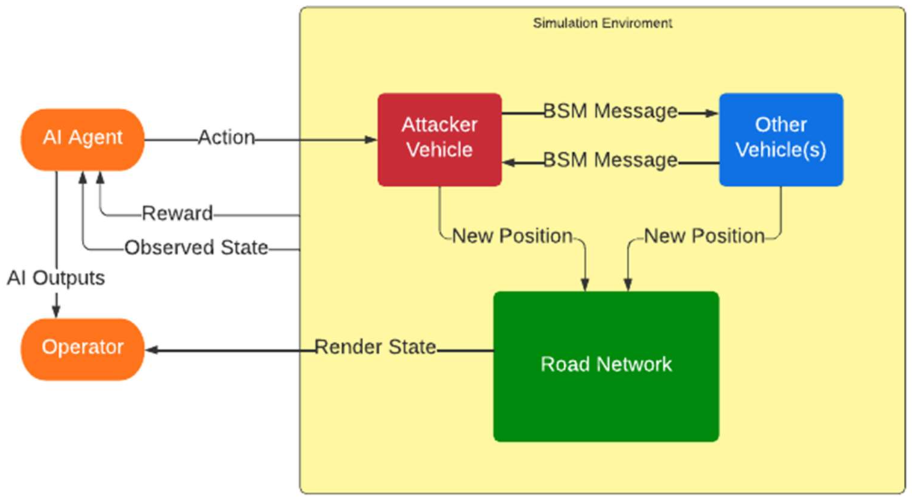


Figure 5: Implementation Design

As the simulation is designed to have a lot of back and forth with the AI model a reinforcement learning approach makes the most sense. The reasons are outlined in section

2.2.5 and the AI model can learn from the real-time observations as the simulation proceeds. The focus of the AI Model will be to send misleading messages to redirect other vehicles in the simulation. This is a type of disruptive attack. Automating and fine tuning this type of attack with an AI algorithm should create a more robust test than a manual implementation.

The AI model is developed in Python due to tutorials [61] and skillsets available. The following sections will discuss how the simulation and AI components are developed, and the results of the implementation.

3.2 Procedure

To achieve a full end-to-end implementation of a VANET simulation and an integrated AI model there are several objectives that this research focuses on:

1. Create a full Vehicle-to-Vehicle simulation containing at least 2 vehicles and at least 2 routes.
2. Establish or leverage a messaging system for communication between all road users and roadside units. Identify the key message parameters that detail the current route and alert messaging that would cause the vehicle diversion.
3. Identify simulation variables that Q-Learning can control and learn from to perform an effective attack.
4. Integrate the Q-Learning model into the simulation so these actions can be performed in real-time, and the model learn from the observations.
5. Analyse results from simulation with and without reinforcement learning and draw conclusions from Q-Learning results.

3.3 Implementation

To achieve a functional end-to-end integrated VANET Simulation with an AI controller a few different techniques were tried and tested. First VEINs [50] was used as the simulation of choice as it was used in similar projects in the past [55], [62]. VEINs have several Open-Source projects, online video tutorials for project creation along with a Google Groups support community. However, it proved difficult to fit the desired implementation, which is reviewed in the section 3.4 in more detail. As Python was the preferred coding language for the project, OpenAI Gym toolkit [63] was also utilised. The biggest advantage of using the Gym toolkit for building a simulation when compared with VEINs was the ease of integration with a reinforcement learning algorithm. To achieve this, the Gym environment for a VANET simulation had to be built from scratch. The design and implementation of this is discussed in the follow section.

3.4 Test Environment

3.4.1 VEINs

The initial implementation VEIN's uses SUMO for constructing the foundation of the simulation, the road, the routes, and the vehicle attributes. SUMO is relatively easy to setup and utilise, especially with available tutorials [64]. The test environment design consisted of a crossroads with 2 vehicles following the same route. There were 3 route options, so either vehicle could go straight, turn left, or turn right. In Figure 6 you can see the two vehicles on the road. The vehicles indicated by red and blue triangles.

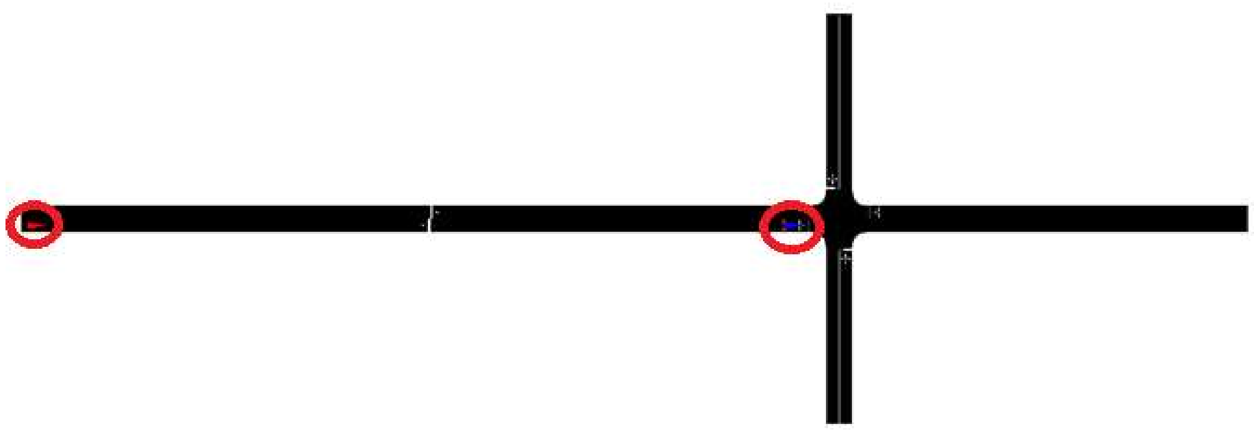


Figure 6: SUMO Crossroads Network (Vehicles circled)

Once the road network and traffic objects are created, the next step in the VEINs workflow is to import the road network into an OMNET++ Initialisation file. This initialisation file can configure all simulation parameters. Additionally, the TraCI library can be applied in the application layer. Utilising the TraCI library saves a lot of time in implementation as it already contains a function for doing vehicle communications including V2V and V2X. The default example in VEINs triggered V2V messages through RSUs when a crash occurred or when it was cleared, and the message data could be seen in real time moving between each of the road users. This application was applied to crossroads network as demonstrated in Figure 7. Here there are 2 vehicles present on the main road, they are labelled as node[0] and node[1] in the figure. The roadside unit is transferring an AirFrame, an over the air message consisting of Basic Safety Message data to each vehicle, or node. This Basic Safety Message consists of the vehicle position, speed, acceleration, orientation, and alert messages such as a collision, or the use of an emergency brake. Figure 7 shows a case where node[0] has identified a crash and sends an emergency BSM to come to a stop. The following vehicle then receives and handles this data.

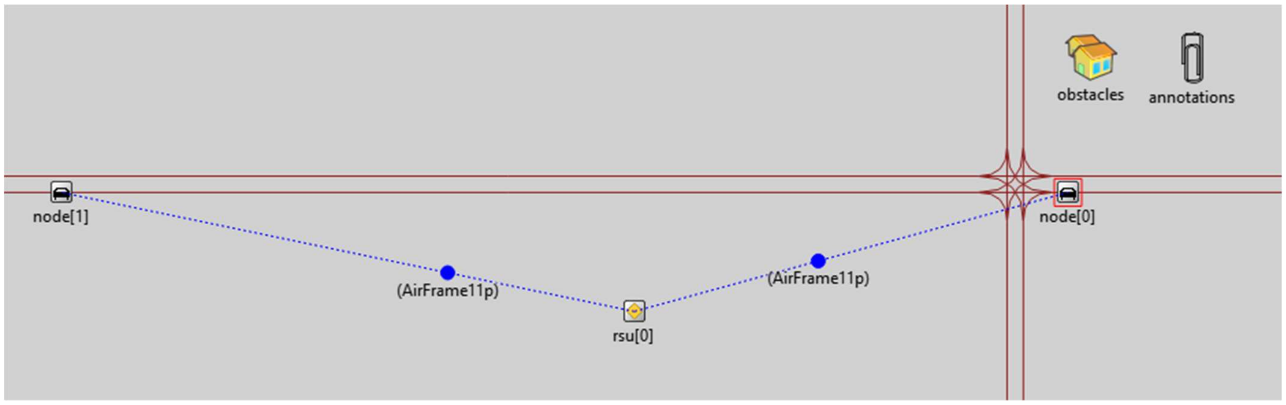


Figure 7: OMNET++ Crossroads with TraCI

The next step was to add a function to redirect traffic to an alternate route when a crash was detected. This was achieved by modifying the TraCI function to receive messages. This implementation helped complete many of the objectives as reviewed in more detail in the results section. However some limitations remained.

- Unable to trigger the attacker vehicle (the following vehicle, node[1]) to send false crash messages. Instead, they were sent from the lead vehicle on a time stamp.
- The simulation was contained with OMNET++ which made it difficult to integrate the Python AI model into it. An alternative solution was to capture the OMNET++ output data and post process it with the AI model. The AI model's decision could then be applied in the next iteration. Developing in the VEIN's workflow proved to be difficult as it requires a strong C++ skillset particularly when making changes to the TraCI functions.

3.4.2 OpenAI Gym

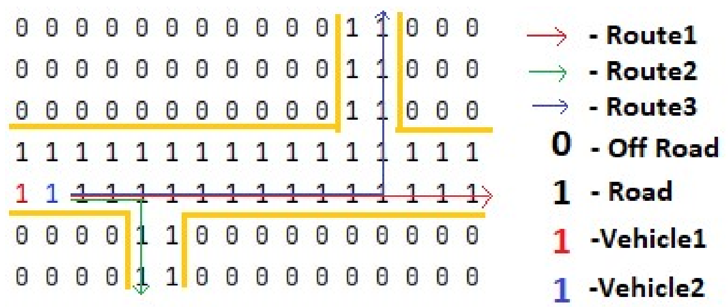
OpenAI has many pre-built environments available for test and comparison but none of them fit the purpose of this research. Instead, a custom environment must be built from scratch to represent the desired scenario. To create the custom environment, other premade environments were reviewed such as “Taxi-v3” and “MountainCar-v0”. These Gym environments proved very useful to understand what is expected for the final implementation.

Every Gym environment must consist of initialisation, step, reset and render functions [65]. Additional functions or submodules may be added after this as necessary. This framework is required for performing reinforcement learning with the environment. Each of the core functions are described further:

- **The initialisation function**: This function takes in a JSON configuration file containing vehicle and simulation parameters. This configuration JSON is passed as an input and sets the total number of gym environment observation states available, the actions, total number of states, the reset state, the default setup for the simulation map, routes, and the default state for each vehicle. This is discussed in more detail later.
- **The step function:** The main function for operating and controlling the simulation should move the simulation forward one timestep every time it is called. An additional Python class was created to handle all the vehicle related functions. It has attributes for the vehicle position, speed, route, sending and receiving communication messages. The class also has functions for acquiring available positions from the roadmap and moving the vehicle. An available position is any position that is within range of the vehicle within that timestep based on the vehicle speed. This available position must be a valid road and the vehicle must make a legal move to get there, i.e. no going through other vehicles or over non-road co-ordinates.
- **Reset Function:** This resets the simulation, the map, vehicles, and sets everything back to the default state, provided by the configuration JSON file.
- **Render Function:** This generates a display of the current state of the simulation. It will show the vehicles position, their current status, the road and whether an action occurred.
- **Other Functions:** There are other sub-functions used for calculating the current state of the simulation and calculating the reward based on the observation.

After creating these first 4 functions following the Gym Framework, the low-level logic could be developed. As the simulation was to be driven by actions, a Markov Decision Design was adapted. Much of the Markov Decision Design could be derived from the way VEINs functioned. The main assumptions and controls built into the OpenAI Gym custom environment are:

- Vehicles are constantly sending BSM messages within the network at every time interval
- Vehicles could not overtake one another, this obviously is not reflective of the real-world but as the communications channel is under test and not the vehicle manoeuvrability, this is a fair assumption to make.
- Vehicles cannot pass through each other – if the lead vehicle is moving slower than the following vehicle, it will move where it can but will not share space with the lead vehicle.
- Vehicles cannot move off-road and must follow a designated route preventing them from travel on the wrong side of the road, or off road. It also stops them from U-turns.
- The other vehicle always has multiple route options.



```
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
```

Figure 8: Scenario2 Road network of ~~a~~ staggered Crossroads

Figure 8 shows the initialised output of the Gym environment. This example road map contains vehicle 1, the attack vehicle, and vehicle 2, the other vehicle. Both vehicles are moving towards a common end point, labelled in Figure 8. Here the attacker is moving 2 units per time interval whereas vehicle 2 is only moving 1 unit per time interval. The reason behind applying Q-Learning in this case is to create a testcase where the attacker vehicle redirects vehicle 2 so vehicle 1 can reach the destination first. The simulation can perform one of four actions per time step to achieve this goal:

- 0: Send a message to vehicle 2 saying all routes are good.
- 1: Send a message to vehicle 2 indicating route1 is blocked by a collision.
- 2: Send a message to vehicle 2 indicating route2 is blocked by a collision.
- 3: Send a message to vehicle 2 indicating route3 is blocked by a collision.

When an action is passed into the step function both vehicles will update their route and positions with respect to their speed and available path. Based on the state change a reward is calculated. The reward table is shown below.

Table 5: Rewards Table for Simulated Environment

Area	**Check**	**Points if True**	**Points if False**
Running	For every time step is run.	-5	NA
Progress towards goal	Vehicle 1 is closer to its destination (multiplied by vehicle 1 speed)	5	-20 (-5 if same distance)
Divert Traffic	Vehicle 1 Route is different to Vehicle 2 Route	10	-5
Message	Vehicle 1 sends an error message instead of an “all clear”	-10	5

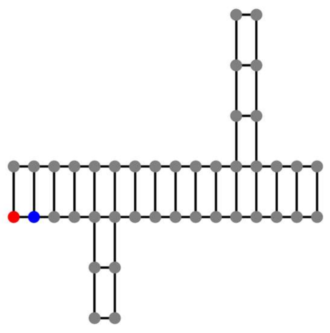
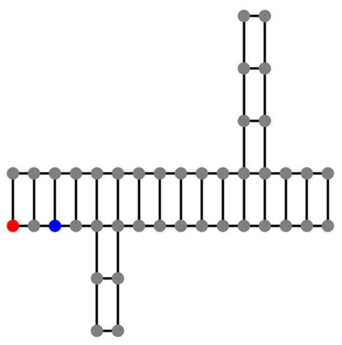
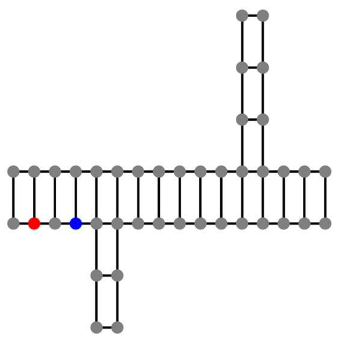
The reward table was influenced heavily from the results in chapter 2 and then further tested and tweaked to try and reflect a real-world case as much as possible. In previous VANET simulators misbehaviour detection was applied, and it detected cases where the same message was sent repeatedly [55]. To prevent misbehaviour detection the “Divert traffic Message” reward would encourage the AI agent to send error messages only when it provided the maximum benefit. “Running” reward check was received every time the step function is called. This is a constant negative reward when the goal has not been achieved and the simulation is still in progress. The “progress towards goal” reward check is key to ensuring the vehicle is moving to the destination quickly. This is achieved by redirecting a vehicle if necessary. When a step is complete the simulator returns to current simulation state, the reward, a done status, and an info field. These are then given back to the operator; in this case this is the AI Agent.

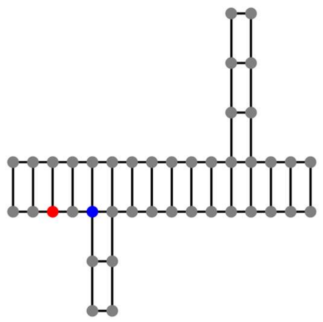
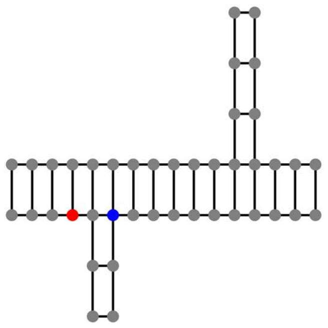
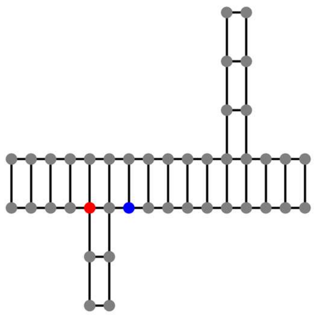
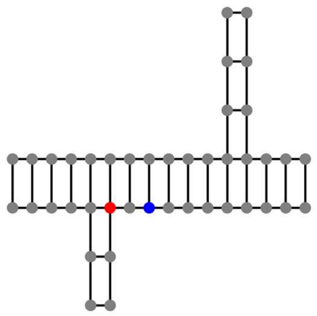
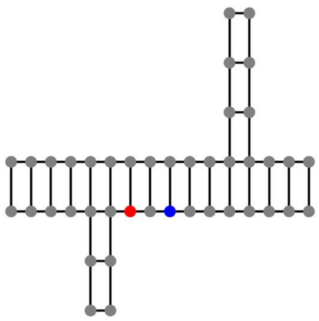
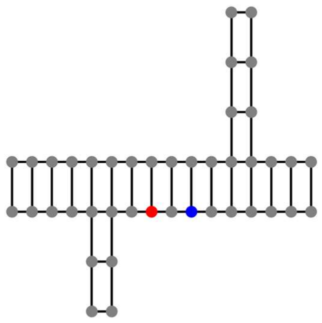
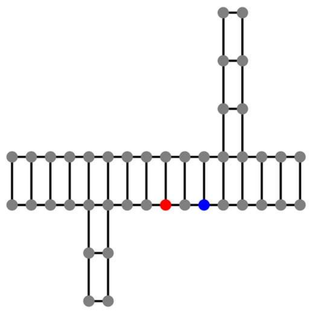
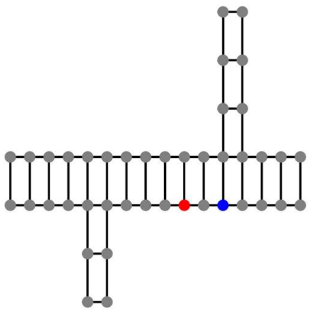
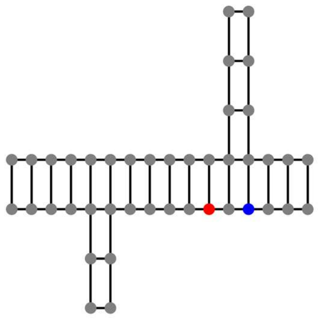
Previously the JSON configuration file was mentioned. It gets pulled in during environment initialisation. The configuration file comprises the following:

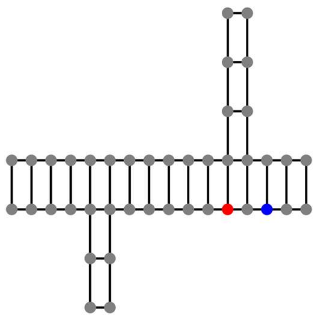
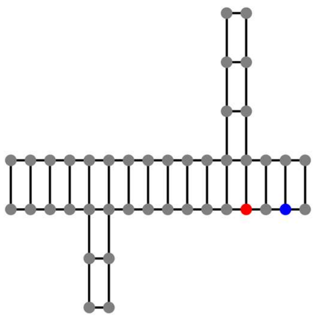
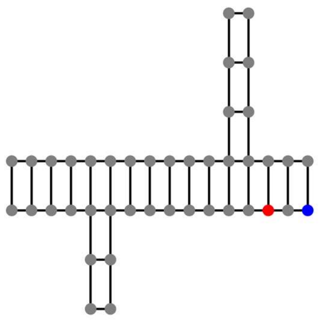
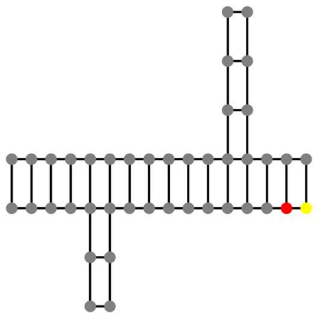
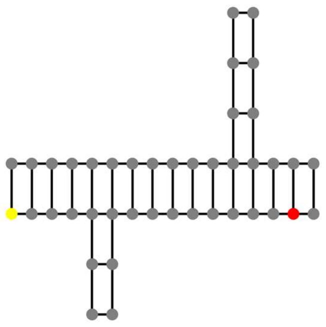
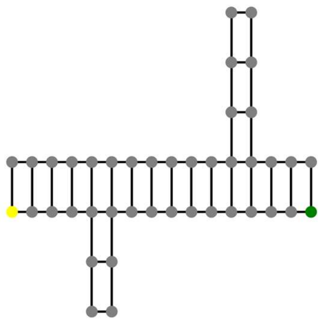
- Simulation Map
- Simulation starts state (also the reset state)
- Number of Simulation states available
- Simulation Route details
    - The path for each route.
- Vehicle Parameters for each vehicle
    - Speed
    - Route
    - Destination position
- Metadata
    - Total Rewards when action is 0
    - Runtime, to slow the runtime of the simulation to render in real time.

During development, several scenarios were tested with different routes and road configurations. Scenario 2 is shown in Table 6 for each timestep it runs. This scenario consists of 2 vehicles approaching ~~a~~ staggered crossroads, the rear vehicle being the attacker. It is the same scenario detailed in Figure 8. In Table 6 the full scenario is shown where vehicle 1 progresses using route 1 in 17 steps. The total reward achieved is 20, which can be taken as a baseline for Scenario 2 when it comes to comparing how the Q-Learning performs.

Table 6: Step by step simulation of Scenario2

Initial State	**Step #: 0**	**Step #: 1**
		
Simulation State: [22 24 0 0] Reward: 0	Simulation State: [22 24 0 0] Reward: 10	Simulation State: [23 25 0 0] Reward: 0

Step #: 2	**Step #: 3**	**Step #: 4**
		
Simulation State: [24 26 0 0] Reward: 0	Simulation State: [25 27 0 0] Reward: 0	Simulation State: [26 28 0 0] Reward: 0
Step #: 5	**Step #: 6**	**Step #: 7**
		
Simulation State: [27 29 0 0] Reward: 0	Simulation State: [28 30 0 0] Reward: 0	Simulation State: [29 31 0 0] Reward: 0
Step #: 8	**Step #: 9**	**Step #: 10**
		
Simulation State: [30 32 0 0] Reward: 0	Simulation State: [31 33 0 0] Reward: 0	Simulation State: [32 34 0 0] Reward: 0

Step #: 11	**Step #: 12**	**Step #: 13**
		
Simulation State: [33 35 0 0] Reward: 0	Simulation State: [34 36 0 0] Reward: 0	Simulation State: [35 37 0 0] Reward: 0
Step #: 14	**Step #: 15**	**Step #: 16**
		
Simulation State: [36 37 0 0] Reward: 0	Simulation State: [36 22 0 0] Reward: 10	Simulation State: [37 22 0 0] Reward: 0

Next a review of how Q-Learning is applied within the AI agent to further improve the score and reduce the step count.

3.4.3 Q-Learning

OpenAI Gym is published widely [65], [63] with articles on how to integrate reinforcement learning using a variety of Q-Learning techniques. OpenAI Gym commonly uses other techniques such as Random Walker to test a new simulated environment. In this case each new scenario ran through a single loop of the simulation, to ensure it worked as expected. Random Walker was then used to test the environment structure and root out any bugs in the implementation for unique actions. Q-Learning was applied once any bugs were resolved. The

Bellman equation [66] is leveraged to apply Q-Learning on a Markov Decision simulation. Using the following breakdown, it can be implemented within the software with relative ease:

$$Q^{New}(s_t, a_t) = (1 - lr) * Q(s_t) + lr * (r_t + Y * maxQ(s_{t+1}))$$

were

$$Q^{New}(s_t, a_t)\ is\ the\ new\ Q\ Value\ for\ some\ action$$

$$Q(s_t)\ is\ the\ old\ Q\ Value$$

$$maxQ(s_{t+1}))\ is\ the\ predicted\ future\ reward$$

$$r_t is\ the\ current\ reward$$

$$lr\ is\ the\ learning\ rate$$

$$Y\ is\ the\ discount\ rate$$

This implementation uses 2 hyper parameters, the learning rate and the discount. The learning rate controls how quickly the values stored in the Q-Table change when the algorithm is run. This is usually low value, approximately 0.1, to prevent a constantly changing Q-Table. The discount is a value representing how much the AI agent cares about the future reward. This is typically close to 1 in this project as a higher reward means a better performing simulation. Therefore, the rewards table, Table 5, applied negative rewards to make it reflective of the real interpretation to give the Q-Learning model value. The Q-Table learns the best action to apply based on the predicted reward.

To effectively develop the Q-Table the simulation was run many times, using an episode approach as detailed in section 2.2.6. While each episode is running, with each step applying an action is sent and a reward is received, and the Q-Table is updated based on the output of the Bellman Equation. The action applied will be selected by the Q-Table based on the current state of the system. However, based on the learning rate a random action will be applied intermittently with the aim of enhancing the model's learning.

The next chapter further reviews the results of applying different variables to the Bellman equation and the results of reusing the same Q-Learning algorithm on different scenarios.

4. Experimentation and Results

This chapter discusses the experimentation and final implementation of the simulation on AI Agent. It details the results achieved by applying Q-Learning to the custom made OpenAI Gym environment.

4.1 Test Results

The test results were obtained by running the full simulation cycle with the Q-Learning AI Agent on each scenario, varying the Q-Learning setup. The hyperparameters were initially varied to understand the range of effect they would have. Before analysing the Q-Learning performance on each scenario, an initial baseline was taken by running the simulation with only one action. In all cases action 0, which was sending message "all clear" as detailed section 3.4.2, was initially taken as the baseline. This is the safest action as all other actions added an element of risk. This is because there is a much larger negative reward for sending the same message constantly than not changing the route of the vehicle.

This chapter will first analyse the Hyperparameter and Q-Learning Configuration before discussing the scenario results.

4.1.1 Hyperparameter and Q-Learning Configuration

Initially the Learning Rate was tested to find the most optimal rate. Four runs were performed to trial learning rates of 0.05, 0.1. 0.25 and 0.9 over 150,000 episodes, whilst the Discount value was ~~a~~ consistent 0.95. The results of these tests are shown in Figure 9. Across all 4 learning rates the results stabilized after 75,000 episodes so it could be selected as the desired episode count moving forward. Additionally, 0.1 and 0.25 learning rates performed well demonstrating a mean reward score of 60, with little variance between them. A value of 0.1 was selected as a lower Learning rate, which allowed greater stabilization within the Q-Table. This aligns with the results observed in other use cases [66].

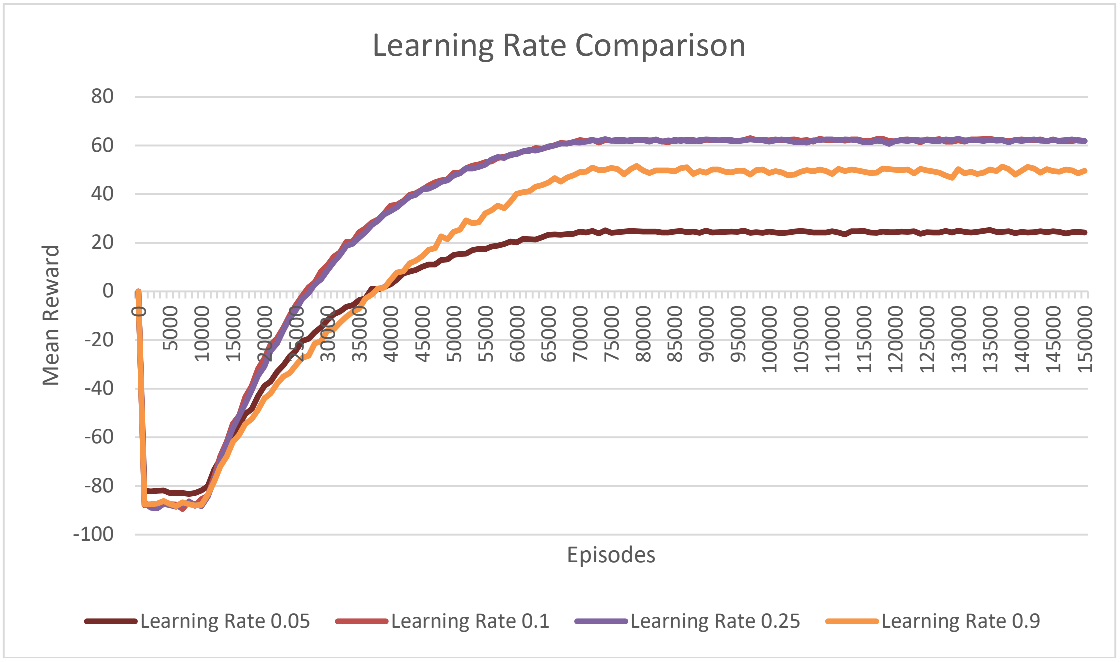


Figure 9: Comparison of Learning Rate for Q-Learning

The Discount value was configured next. By testing the Q-Learning algorithm with consistent learning rate of 0.1, the Discount value was tested with a value of 0.1, 0.5, 0.8, 0.9 and 0.95. As shown in Figure 10 the Discount Value 0.8 and 0.9 under performed, never achieving a high mean reward score. The other three values were much similar but 0.95 was the most stable and so it was selected for running the algorithms on the various scenarios.

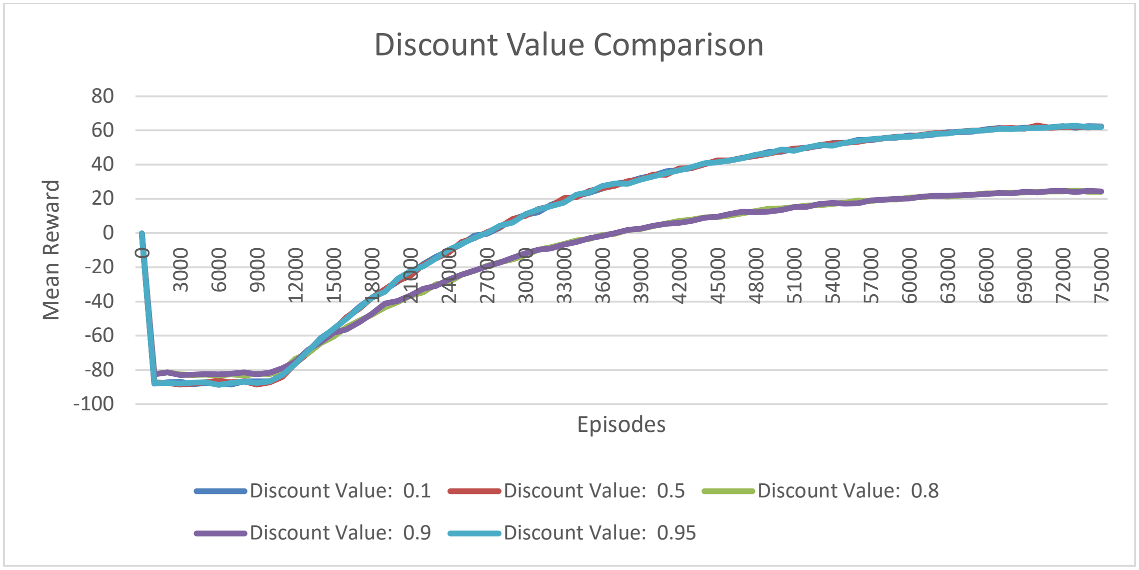


Figure 10: Comparison of Discount Values for Q-Learning

4.1.2 Scenario 1

Scenario 1 consists of ~~a~~ simple crossroads. The simulation map is 9x7 units big, with 3 routes. All routes start at the left road from a top-down perspective and route 1, route 2, route 3 proceed to go straight, turn right, or turn left respectively. Initially both vehicles are on route 1, going straight and travelling at the same speed. When running with Q-Learning with a baseline of reward 10 when only action 0 is used, an increase of three times the mean average reward is seen in Figure 11. This took 33,000 episodes before a better mean reward is achieved. It is after 70,000 episodes when the maximum average mean reward is achieved at reward 35.9. For the remaining 5,000 episodes the mean reward started to decline.

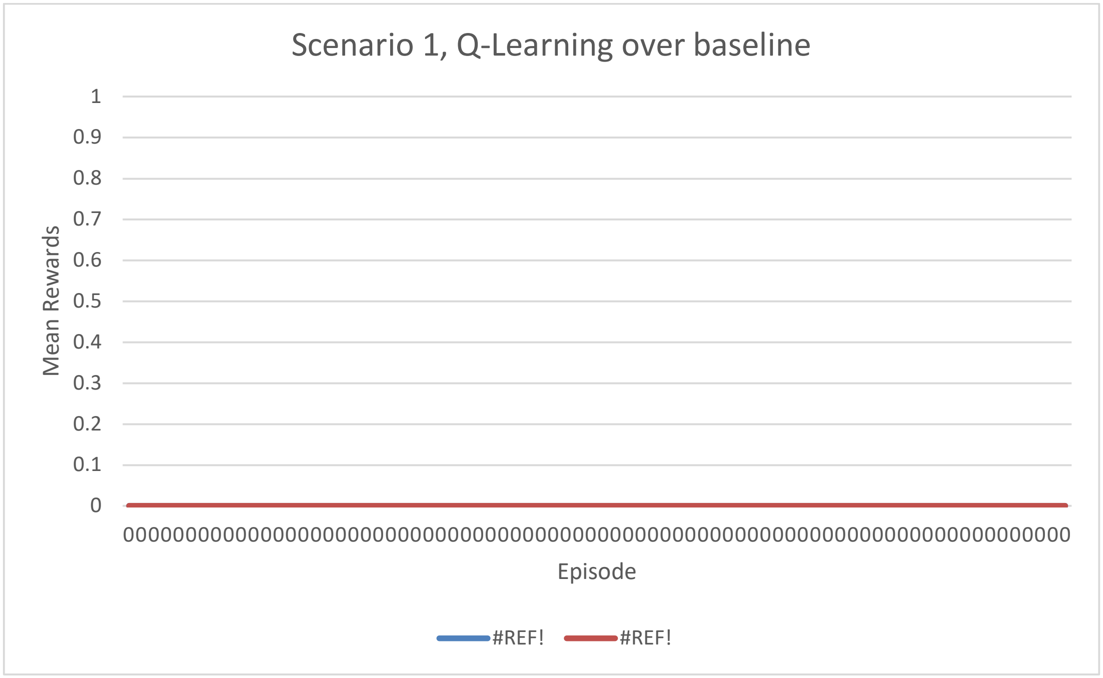


Figure 11: Scenario 1 Baseline and Q-Learning

4.1.3 Scenario 2

Scenario 2 is like Scenario 1 but the junction is a staggered crossroads with the right turn being closer than the left. The simulation map is 16x7 units big, with 3 routes. All routes start at the left road from a top-down perspective and route 1, route 2, route 3 proceed to go straight, turn right, or turn left respectively. Initially each vehicle is on route 1, going straight. This time the rear vehicle is twice as fast as the lead vehicle at 2 units per second, but this will only have an impact if the lead vehicle is redirected. The baseline reward of the simulation using only action 0 is 20.

Running with Q-Learning yielded three times the improvement as presented in Figure 12. Scenario 2 ran with 80,000 episodes instead of 75,000 as an initial run did not show stabilization after 75,000 episodes. With 80,000 episodes the highest mean reward achieved was 62.875.

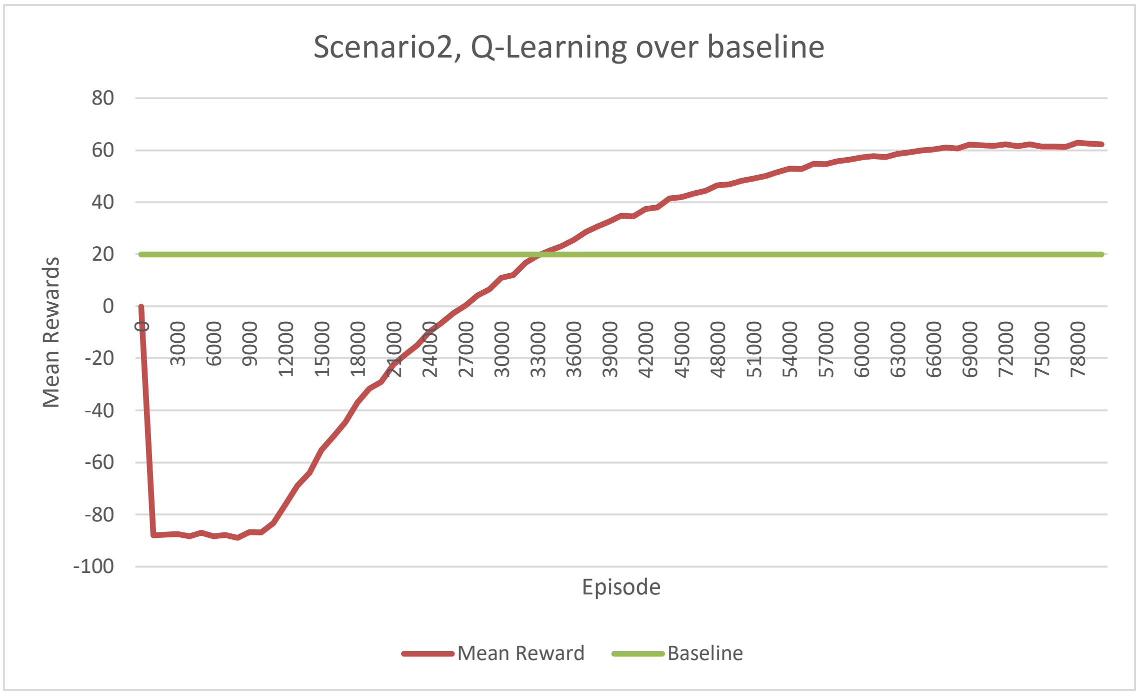
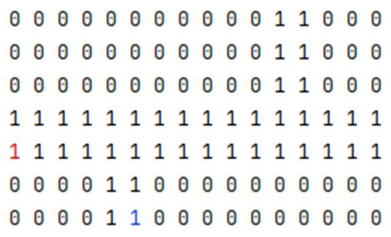


Figure 12: Scenario 2 Baseline and Q-Learning

4.1.4 Scenario 3

Scenario 3 leverages the same road map as scenario 2 however in this case route 2 starts at the lower right and turns right onto the staggered crossroads and route 3 starts in the lower edge, meets the staggered crossed roads and turns left. Vehicle 2 is following route 2. Figure 13 shows the initialised simulation where the red vehicle is the attacker vehicle, and the blue is vehicle 2.



```
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
```

Figure 13: Scenario 3 Initialized State

This is a much harder scenario for the simulation environment to handle as vehicle 2 and vehicle 1 start on different routes and vehicle 2 is not immediately in the way. It only gets in the way when it turns onto the staggered crossroads. The baseline reward is 90 when running action 0 (all default routes being followed) however it takes 15 steps to achieve this.

Q-Learning performs poorly even with 150,000 episodes in terms of mean reward, but it only took 8 steps to complete the simulation. This is a vast improvement in runtime and is prime result of this experimentation. The reward graph is shown here in Figure 14.

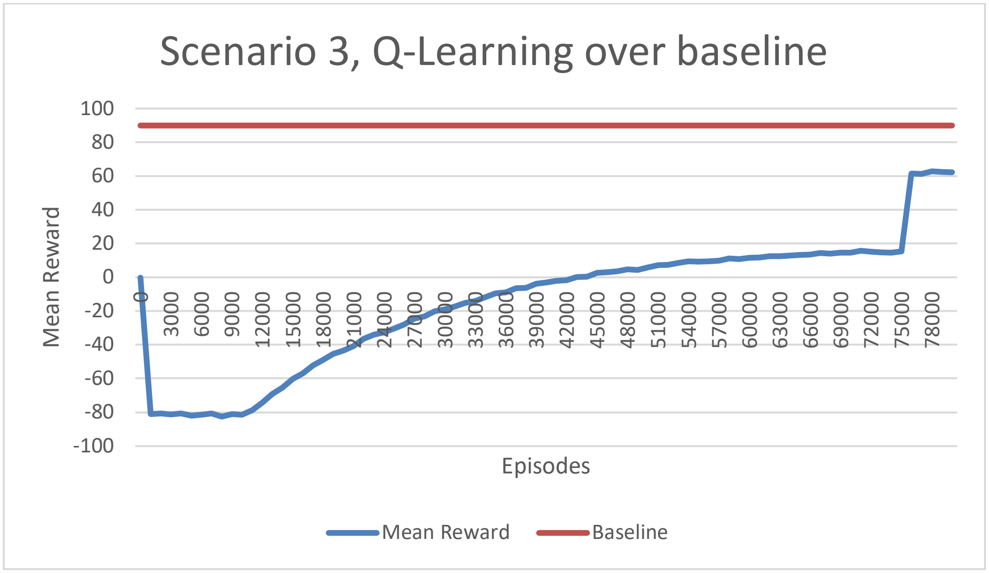


Figure 14: Scenario 3 Baseline and Q-Learning

4.1.5 Reuse the Q-Table between Scenarios

This section reviews the reusability of the Q-Table between scenarios. As the same simulated environment is always used but with different scenarios, extra experimentation was necessary to investigate if there is a benefit to reusing the Q-table between scenarios. Initially the Q-Table was trained with 75,000 episodes of Scenario 2 like section 4.1.3. This Q-Table was then reused as a starting point for running 75,000 episodes of Scenario 1. The result shows that there is no benefit in reusing the same Q-Table between scenarios despite the actions and simulation maps being similar. The plot for Scenario 1 in Figure 15 is very close to the plot for running Scenario 1 through the Q-Learning algorithm as shown in Figure 11.

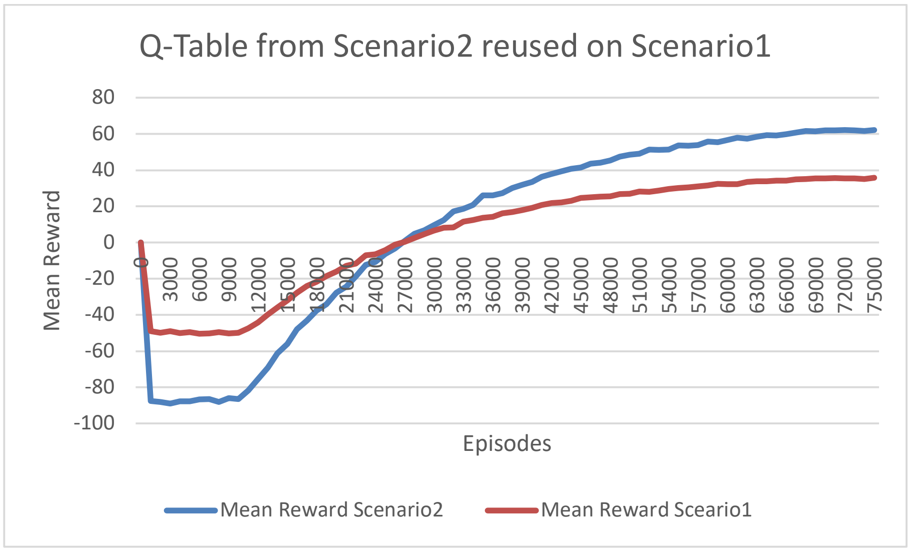


Figure 15: Scenario 2's Q-Table reused for Scenario 1

4.2 Test Results Summary

The results of Scenarios 1,2 and 3 are super imposed on each other in Figure 16. This presents a clear comparison between them. From Figure 16 it is clear where Scenario 3 struggles to perform, especially when compared with Scenarios 1 and 2.

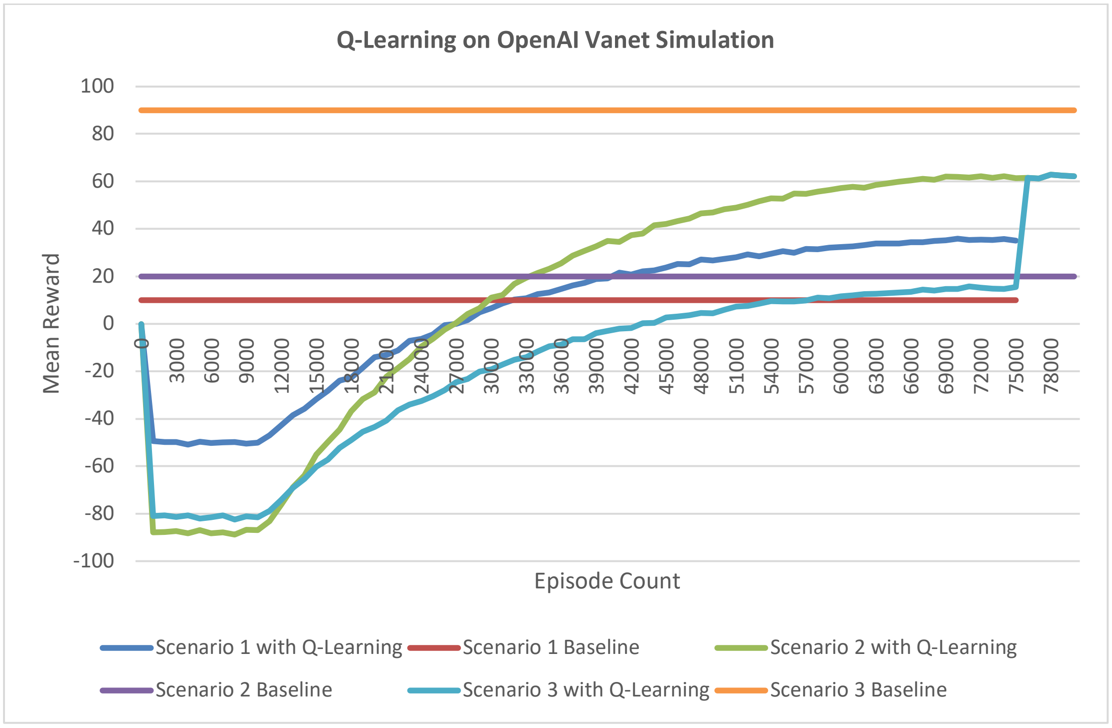


Figure 16: Scenarios 1, 2 and 3 Baseline and Q-Learning

5. Conclusion and Evaluation

5.1 Evaluation of Results

The previous chapter presented several findings from this research. The current Q-Learning algorithm shows a lot of value in evaluating Scenario 1 and 2. As a penetration test the overall implementation shows where the tester can leverage different actions to misdirect another vehicle. In the case where the application under test requires a built-in misbehaviour protection, further testing from different angles can be performed efficiently by using this Q-learning approach. This is one of the main benefits of using Q-Learning in penetration testing.

More complex cases such as Scenario 3 challenged both the simulator and Q-Learning and it flags a significant gap in the simulation design reward table where finishing early is not correctly rewarded. The reward table needs further optimisation as Scenario 3 with Q-Learning finishes 7 steps early, almost 50% less steps than the baseline, but is not reflected in the mean reward. Like Scenario 1 and 2, this shows the benefit of using a Q-Learning algorithm in the penetration test to find alternate options. In this case the optimised route had Vehicle 2 end up in a completely different direction which could be identified easily by a threat detection mechanism due to the degree of change. This is due to typical redundancy of sensors, Inertial Measurement Unit, Geo-Tracking etc. This a valid test approach as although it may be detected easily, the Q-Learning algorithm still has a chance to redirect to another route.

The experiment of reusing a Q-Table proved that further changes to the Q-Learning algorithm would be required rather than simply reusing the trained Q-Table as was done in section 4.1.5. Taking an approach like Q-Transfer Learning may be a more suitable approach to effectively reuse training from a previous run [67].

5.2 Conclusion

This research gives a complete end-to-end report on how a VANET simulation was designed, what was required to implement it in both VEINs and OpenAI Gym, and how a Q-Learning algorithm can be used to launch thousands of cycles of the simulation that produces a Q-Table.

The initial simulation implementation, VEINS, was unusable for this project instance due to the necessity for a strong C++ skillset for building the simulation which the author did not have. However, the alternative approach of using OpenAI Gym environment met the requirement for a functional VANET simulation. By researching typical VANET simulations and reviewing the design of VEINs, it was relatively straightforward to implement a custom OpenAI Gym environment. Testing a Q-Learning algorithm revealed there are improvements that can be made to the simulation. These are discussed further in the next section.

The custom OpenAI Gym environment provided much better integration with Reinforcement Learning algorithms. As shown in Scenarios 1, 2 and 3, integration of Q-Learning with a VANET simulation creates an effective testbed for performing penetration tests, proving the premise of this project. There are however more tweaks required to future proof this approach.

5.3 Future Work

There is a lot of scope for future development and research in this concept. This project implemented Q-Learning for a VANET simulation, but other machine learning techniques may also be leveraged. As attempted in section 4.1.5, Q-Transfer learning could be applied between simulations to save time learning. Runtime of the algorithm is not currently an issue and takes approximately 10 minutes to run 75,000 episodes on an Intel i7-9750 with 16Gb RAM and NVIDIA GeForce RTX 2060. However, as the simulation develops to add more functionally this may become more of an issue.

The simulation itself is an area that requires improvement. From testing there are a few areas in the simulation that can be improved as listed below:

- Further test and modify the reward table, currently the Q-Learning result from Scenario 3 does not represent the improvement in the number of steps taken.
- Increase the length of the road network, add larger routes and more junctions. A trial where multiple vehicles are run through a road network would be an interesting test case.

Future work may also focus more on the application under test. The current simulation does not safeguard the messages it receives, and the non-attackers merely follow what they are told. A real-world example is to send messages from an RSU and have the attacker vehicle manipulate the data and resend this message to other vehicles, claiming to be from a trusted source.

Another approach is to reuse a similar structure but with a CAV network rather than a VANET. This project scope was originally set out to investigate how a full penetration test could be done using reinforcement learning but this requires advanced skills and access to costly hardware. Further experimentation should be performed to investigate the use of AI in penetrating a CAV network from an external source.

6. Works Cited

[1] P. Seuwou, E. Banissi, G. Ubakanma, "The Future of Mobility with Connected and Autonomous Vehicles in Smart Cities," in *Digital Twin Technologies and Smart Cities*, Cham, Springer, 2020, pp. 37-52.

[2] C. Miller, "Downtown Toronto Simcoe Place," *IEEE Design & Test,* vol. 36, no. 6, pp. 7-9, 2019.

[3] G. Samara, W. A. H. Al-Salihy, R. Sures, "Security Analysis of Vehicular Ad Hoc Networks (VANET)," in *Second International Conference on Network Applications, Protocols and Services*, Alor Setar, Malaysia, 2010.

[4] F. Gonçalves, J. Macedo, A. Santos, "An Intelligent Hierarchical Security Framework for VANETs," *Mdpi Information,* vol. 12, no. 11, p. 455, 201.

[5] C. Schmittner et al., "Automotive Cybersecurity- Training the Future," in *Systems, Software and Services Process Improvement*, Springer, 2021, pp. 211-219.

[6] M. Humayun, M. Niazi, NZ Jhanjhi, M. Alshayeb, S. Mahmood, "Cyber Security Threats and Vulnerabilities: A Systematic Mapping," *Arabian Journal for Science and Engineering,* vol. 45, p. 20, 2020.

[7] CIST, "Security Tip (ST04-015) - Understanding Denial-of-Service Attacks," 20 11 2019. [Online]. Available: https://us-cert.cisa.gov/ncas/tips/ST04-015. [Accessed 04 12 2021].

[8] Cisco, "What Is Malware?," 11 04 2018. [Online]. Available: https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html. [Accessed 4 12 2021].

[9] J. Fruhlinger, "What is phishing? How this cyber attack works and how to prevent it," 4 09 2020. [Online]. Available: https://www.csoonline.com/article/2117843/what-is-phishing-how-this-cyber-attack-works-and-how-to-prevent-it.html. [Accessed 4 12 2021].

[10] D. Swinhoe, "What is a man-in-the-middle attack? How MitM attacks work and how to prevent them," 13 2 2019. [Online]. Available: https://www.csoonline.com/article/3340117/what-is-a-man-in-the-middle-attack-how-mitm-attacks-work-and-how-to-prevent-them.html. [Accessed 4 12 2021].

[11] M. Lezzi, M. Lazoi, A. Corallo, "Cybersecurity for Industry 4.0 in the current literature: A reference framework," *Computers in Industry,* vol. 103, pp. 97-110, 2018.

[12] H. Murray, D. Malone, "Costs and benefits of authentication advice.," *ArXiv,* vol. abs/2008.05836, p. 34, 2020.

[13] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, Y. Koucheryavy , "Multi-Factor Authentication: A Survey and Challenges in V2X Applications," *MDPI Cryptography,* vol. 2, no. 1, p. 31, 2018.

[14] M. Sergey, S. Nikolay. E. Sergey, "Cyber security concept for Internet of Everything (IoE)," in *2017 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO)*, Kazan, 2017.

[15] R. Sharma, S. Dangi, P. Mishra, "A Comprehensive Review on Encryption based Open Source Cyber Security Tools," in *2021 6th International Conference on Signal Processing*, Solan, 2021.

[16] H. Xu, Y. Zhou, J. Ming, M. Lyu , “Layered obfuscation: a taxonomy of software obfuscation techniques for layered security,” 03 04 2020. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00049-3. [Accessed 20 11 2021].

[17] A. Hassanzadeh, S. Modi, S.Mulchandani, “Towards effective security control assignment in the Industrial Internet of Things,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, 2015.

[18] C. Sandberg, B. Hunter, “Cyber security primer for legacy process plant operation,” in *2017 Petroleum and Chemical Industry Technical Conference (PCIC)*, Calgary, 2017.

[19] R. Chaturvedi, “UL testing standards to mitigate cybersecurity risk ~ UL's approach with complement to the other standards for SICE 2017,” in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, 2017.

[20] A. Beliba, A. Kukoba, Vladyslav V., “Automotive Security Testing 101: Requirements, Best Practices, Tips on Overcoming ChallengesIt was originally published on https://www.apriorit.com/,” 09 09 2021. [Online]. Available: https://www.apriorit.com/dev-blog/742-cybersecurity-automotive-security-testing. [Accessed 21 11 2021].

[21] R. Shimonski, “How to Perform a Penetration Test,” in *Penetration Testing For Dummies* , Wiley, 2020, p. 256.

[22] O. Security, “Kali Linux,” https://www.kali.org/, 2013.

[23] J. Dando, “A guide to ethical hacking — Understanding Nmap,” 01 02 2019. [Online]. Available: https://medium.com/@dandobusiness/a-guide-to-ethical-hacking-understanding-nmap-7aea71f65554. [Accessed 15 04 2022].

[24] R. Deraison, “Nessus,” Tenable Network Security, 1998.

[25] Wireshark, “Wireshark,” https://www.wireshark.org/, 2022.

[26] M. C. Hanem, T. M. Chen, “Reinforcement Learning for Efficient Network Penetration Testing,” *MDPI - Information 2020,* vol. 11, no. 6, p. 23, 2019.

[27] I. H. Sarker, M. H. Furhad, R. Nowrozy, “AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions,” *SN Computer Science ,* vol. 2, no. 173, p. 18, 2021.

[28] G. A. Francia, E. El-Sheikh, “Applied Machine Learning to Vehicle Security,” in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*, Springer, Cham, 2020, pp. 423-442.

[29] Dean Richard McKinnel, Tooska Dargahi, Ali Dehghantanha, Kim-Kwang Raymond Choo, “A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment,” *Computers and Electrical Engineering ,* vol. 75, pp. 175-188, 2019.

[30] Z. Hu, R. Bueran, Y. Tan, “Automated Penetration Testing Using Deep Reinforcement Learning,” in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Genoa, 2020.

[31] K. Kim, J. S. Kim, S. Jeong, J. Park, H. K. Kim, “Cybersecurity for autonomous vehicles: Review of attacks and defense,” *Computers & Security,* vol. 103, no. 102150, p. 27, 2021.

[32] J. Locke, “What Is Connected Vehicle Technology and What Are the Use Cases?,” 17 6 2020. [Online]. Available: https://www.digi.com/blog/post/what-is-connected-vehicle-technology-and-use-cases. [Accessed 1 12 2021].

[33] X. Luom Y. Liu, H. Chen, Q. Guo, “Physical Layer Security in Intelligently Connected Vehicle Networks,” *IEEE Network ,* vol. 34, no. 5, pp. 232-239, 2020.

[34] X. Ji, H. Cheng, Security Improvements in Connected Cars - Case Study: CEVT Connected Cars, 1st ed., Gothenburg: Chalmers tekniska högskola , 2019.

[35] C. Zhai, W. Wu, “Designing continuous delay feedback control for lattice hydrodynamic model under cyber-attacks and connected vehicle environment,” *Communications in Nonlinear Science and Numerical Simulation,* vol. 95, no. 105667, p. 17, 2020.

[36] G. Rajbahadur, A. Malton, A. Walenstein, A. Hassan, “A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety,” Changshu, 2018.

[37] G. Burzio, G. F. Cordella, M. Colajanni, M. Marchetti, D. Stabili, “Cybersecurity of Connected Autonomous Vehicles : A ranking based approach,” Milan, 2018.

[38] Technical Committee ISO/TC 22/SC 32 , “ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering,” 08 2021. [Online]. Available: https://www.iso.org/standard/70918.html. [Accessed 20 11 2021].

[39] Issuing Committee: Vehicle Cybersecurity Systems Engineering Committee, “Cybersecurity Guidebook for Cyber-Physical Vehicle Systems J3061_201601,” 14 01 2016. [Online]. Available: https://www.sae.org/standards/content/j3061_201601/. [Accessed 2021 11 20].

[40] H. Hoeberechts, “How ISO 21434 Will Transform the Automotive Industry,” Mirai, 21 10 2020. [Online]. Available: https://www.miraisecurity.com/blog/how-iso-21434-will-transform-the-automotive-industry. [Accessed 20 11 2021].

[41] T. Team, “Top 10 Biggest Car Manufacturers by Revenue (2021),” Thread in Motion, 27 07 2021. [Online]. Available: https://www.threadinmotion.com/blog/top-10-biggest-car-manufacturers-by-revenue. [Accessed 20 11 2021].

[42] N. I. o. S. a. Technology, “National Vulnerability Database,” U.S. Department of Commerce, Gaithersburg, 2021.

[43] D. S. Fowler, M. Cheah. S. A. Shaikh, J. Bryans, “Towards A Testbed for Automotive Cybersecurity,” TOkyo, 2017.

[44] A. Costley, C. Kunz, R. Gerdes, R. Sharma, “Low Cost, Open-Source Testbed to Enable Full-Sized Automated Vehicle Research,” *Systems and Control,* 2020.

[45] T. Toyama, T. Yoshida, H. Oguma, T. Matsumoto, “PASTA: Portable Automotive Security Testbed with Adaptability,” London, 2018.

[46] S. Baar, “Cheap Car Hacking for Everyone A Prototype for Learning about Car Security,” Regensburg, 2020.

[47] K. J. Higgins, “Toyota Prepping 'PASTA' for its GitHub Debut,” 2019. [Online]. Available: https://www.darkreading.com/vulnerabilities-threats/toyota-prepping-pasta-for-its-github-debut. [Accessed 2021 11 14].

[48] D. Jiaa, J. Suna, A. Sharma, Z. Zhenga, B. Liu, “Integrated simulation platform for conventional, connected and automated driving A design from cyber–physical systems perspective,” *Transportation ResearchPartC,* vol. 124, no. 102984, p. 19, 2021.

[49] F. AAlhaidari, A. Alrehan, “Asimulation work for generating a novel dataset to detect distributed denial of service attacks on Vehicular Adhoc NETwork systems,” *international Journal of Distributed Sensor Networks,* vol. 17, no. 3, p. 25, 2021.

[50] C. Sommer, R. German and F. Dressler, “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis,” *IEEE Transactions on Mobile Computing (TMC),* vol. 10, pp. 3-15, 2011.

[51] K. Pfosi, “Stress Tests for Automotive Cybersecurity,” 23 9 2021. [Online]. Available: https://www.aptiv.com/en/insights/article/stress-tests-for-automotive-cybersecurity. [Accessed 27 11 2021].

[52] UKAutodrive, “Connected and autonomous vehicles: a hacker’s delight?,” Gowling WLG, 2017.

[53] Tesla, “Product Security,” 2021. [Online]. Available: https://www.tesla.com/legal/security. [Accessed 04 10 2021].

[54] {S. Marksteiner, N. Marko, A. Smulders, S. Karagiannis, F. Stahl, H. Hamazaryan, R. Schlick, S. Kraxberger and A. Vasenev, “A Process to Facilitate Automated Automotive Cybersecurity Testing,” in *IEEE 93rd Vehicular Technology Conference*, Helsinki, 2021.

[55] J. Kamel, M. R. Ansari, J. Petit, A. Kaiser, I. Jemaa and P. Urien, “Simulation Framework for Misbehavior Detection,” *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY,* vol. 69, no. 6, pp. 6631-3344, 2020.

[56] C. Kyrkou, A. Papachristodoulou, T. Theocharides, A. Kloukiniotis, A. Papandreou, A. Lalos, K. Moustakas, “Towards artificial-intelligence-based cybersecurity,” in *IEEE Computer Society Annual Symposium on VLSI*, Limassol, 2020.

[57] F. M. Zennaro, L. Erdodi, “Modeling Penetration Testing with reinforcement learning using capture-the-flag challenges and tabular Q-learning,” 26 May 2020. [Online]. Available: https://arxiv.org/abs/2005.12632. [Accessed 30 September 2021].

[58] D. R. Mickinnel, T. Dargahi, A. Dehghantanha, K. R. Choo, “A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment,” *Computers and Electrical Engineering,* vol. 75, pp. 175-188, 2019.

[59] R. S. Sutton; A. G. Barto, Reinforcement Learning: An Introduction, Cambridge: MIT Press, 2018.

[60] Y. Feng, “Create a customized gym environment for Star Craft 2,” 25 11 2019. [Online]. Available: https://towardsdatascience.com/create-a-customized-gym-environment-for-star-craft-2-8558d301131f. [Accessed 17 04 2022].

[61] V. Lyashenko, P. Januszewki, “The Best Tools for Reinforcement Learning in Python You Actually Want to Try,” Neptune.ai, 17 01 2022. [Online]. Available: https://neptune.ai/blog/the-best-tools-for-reinforcement-learning-in-python#:~:text=KerasRL%20is%20a%20Deep%20Reinforcement,with%20different%20algorithms%20quite%20easily.. [Accessed 19 01 2022].

[62] F. Valle, S. Cespedes, A.S. Hafid, “Automated Decision System to Exploit Network Diversity for Connected Vehicles,” *IEEE Transactions on Vehicular Technology,,* vol. 70, no. 1, pp. pp 858-871, 2021.

[63] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schelman, J. Tang et al., “Openai Gym,” arXiv:1606.01540, 2016.

[64] M. Behrisch, R. Hilbrich, “SUMO User Documentation,” German Aerospace Center (DLR), 04 01 2022. [Online]. Available: https://sumo.dlr.de/docs/Tutorials/index.html. [Accessed 02 04 2022].

[65] A. King, “Create custom gym environments from scratch — A stock market example,” TowardsDataScience, 10 04 2019. [Online]. Available: https://towardsdatascience.com/creating-a-custom-openai-gym-environment-for-stock-trading-be532be3910e. [Accessed 01 03 2022].

[66] R. MacWha, “Q Learning - From the Basics,” 12 07 2021. [Online]. Available: https://medium.com/nerd-for-tech/q-learning-from-the-basics-b68e74f97254. [Accessed 09 03 2022].

[67] T. V. Phan, S. Sultana, T. F. Nguyen, T. Bauschert, “Q - TRANSFER: A Novel Framework for Efficient Deep Transfer Learning in Networking,” in *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Fukuoka, Japan, 2020.

[68] J. Axelsson, “Safety in Vehicle Platooning: A Systematic Literature Review,” *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS,* vol. 18, no. 5, p. May, 2017.

[69] Alam et al, “Heavy-Duty Vehicle Platooning for Sustainable Freight Transportation,” *IEEE Control Systems,* pp. 34-56, 29 10 2015.

[70] T. Kim et al, “Camera and Radar-based Perception System for Truck Platooning,” Busan, 2020.

[71] Mani et al., “Security Vulnerabilities of Connected Vehicle Streams and Their Impact on Cooperative Driving,” *AUTOMOTIVE NETWORKING AND APPLICATIONS,* pp. 126-132, June 2015.

[72] Hao et al., “REPLACE: A Reliable Trust-Based Platoon Service Recommendation Scheme in VANET,” *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY,* vol. 66, no. 2, pp. 1786-1797, 2017.

[73] Markham et Al., “A Balanced Approach for Securing the OBD-II Port,” *SAE International J Passenger Cars - Electronic Electrical Systems,* vol. 10, no. 2, p. 11, 2017.

[74] Turri et al., “Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning,” *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY,* vol. 25, no. 1, p. 17, 2017.

[75] Khan et al., “Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions,” *Accident Analysis & Prevention,* p. 16, 26 October 2020.

[76] A. Petrillo et al., “A collaborative approach for improving the security of vehicular scenarios: The case of platooning,” *Computer Communications,* vol. 122, pp. 59-75, 2018.

[77] M Hossein Basiri et al., “Security of Vehicle Platooning: A Game-Theoretic,” *IEEE Access,* p. 185565–185579, 19 December 2019.

[78] F. Utesch , A. Brandies, P. Fouopi, C. Schießl, “Towards behaviour based testing to understand the black box of autonomous cars,” *European Transport Research Review,* p. 11, 2020.

[79] F. Lambert, “The Big Tesla Hack: A hacker gained control over the entire fleet, but fortunately he’s a good guy,” Electrek, 27 08 2020. [Online]. Available:

https://electrek.co/2020/08/27/tesla-hack-control-over-entire-fleet/. [Accessed 10 06 2021].

[80] A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway - With Me in It," Wired, 21 July 2015. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. [Accessed 17 06 2021].

[81] D. Tobok, "How Does Penetration Testing Work?," 20 03 2019. [Online]. Available: https://cytelligence.com/how-does-penetration-testing-work/. [Accessed 24 10 2021].

[82] J. Jadaan, S. Zeater, Y. Abukhalil, "Connected Vehicles: an Innovative Transport Technology," in *10th International Scientific Conference Transbaltica 2017: Transportation Science and Technology*, Vilnius, 2017.

[83] C. V. Hoof, "Learn by example Reinforcement Learning with Gym," 2019. [Online]. Available: https://www.kaggle.com/code/charel/learn-by-example-reinforcement-learning-with-gym/notebook#Markov-decision-process(MDP). [Accessed 13 02 2022].

[84] P. Garrad, "GitHub - Project Files," 10 04 2022. [Online]. Available: https://github.com/Dazack/v2v_sim/. [Accessed 10 04 2022].

[85] A. Violante, "Simple Reinforcement Learning: Q-learning," 18 03 2019. [Online]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56. [Accessed 08 05 2022].

7. Appendices

Appendix I – Raw Results

Episode	Scenario 1 with Q-Learning	Scenario 1 Baseline	Scenario 2 with Q-Learning	Scenario 2 Baseline	Scenario 3 with Q-Learning	Scenario 3 Baseline
0	-0.1	10	-0.09	20	-0.115	90
1000	-49.43	10	-87.905	20	-81.085	90
2000	-49.79	10	-87.71	20	-80.8	90
3000	-49.82	10	-87.395	20	-81.385	90
4000	-50.87	10	-88.25	20	-80.785	90
5000	-49.64	10	-86.885	20	-82.03	90
6000	-50.165	10	-88.265	20	-81.565	90
7000	-50	10	-87.86	20	-80.83	90
8000	-49.88	10	-88.85	20	-82.525	90
9000	-50.405	10	-86.69	20	-81.13	90
10000	-50.045	10	-86.81	20	-81.565	90
11000	-47.015	10	-83.24	20	-78.865	90
12000	-42.785	10	-76.235	20	-74.2	90
13000	-38.51	10	-68.795	20	-69.145	90
14000	-35.705	10	-63.89	20	-65.335	90
15000	-31.685	10	-55.175	20	-60.28	90
16000	-28.28	10	-49.85	20	-57.115	90
17000	-23.99	10	-44.51	20	-52.285	90
18000	-22.7	10	-36.83	20	-49.09	90
19000	-18.455	10	-31.625	20	-45.505	90
20000	-14.075	10	-28.94	20	-43.42	90
21000	-13.16	10	-22.325	20	-40.765	90
22000	-11.21	10	-18.56	20	-36.49	90
23000	-7.175	10	-14.915	20	-34.09	90
24000	-6.305	10	-9.605	20	-32.56	90
25000	-4.595	10	-6.29	20	-30.52	90
26000	-0.62	10	-2.525	20	-28	90
27000	0.01	10	0.43	20	-24.595	90
28000	1.585	10	4.195	20	-23.17	90
29000	4.9	10	6.58	20	-20.08	90
30000	6.61	10	11.065	20	-19.105	90
31000	8.605	10	12.1	20	-17.245	90
32000	10.15	10	16.81	20	-15.22	90
33000	10.81	10	19.435	20	-14.005	90
34000	12.595	10	21.49	20	-11.68	90
35000	13.225	10	23.2	20	-9.505	90
36000	14.74	10	25.525	20	-8.8	90
37000	16.09	10	28.645	20	-6.535	90
38000	17.32	10	30.7	20	-6.385	90

39000	18.805	10	32.665	20	-3.94	90
40000	19.18	10	34.855	20	-2.98	90
41000	21.61	10	34.57	20	-1.99	90
42000	20.665	10	37.36	20	-1.765	90
43000	22.135	10	38.05	20	0.245	90
44000	22.42	10	41.455	20	0.455	90
45000	23.695	10	42.04	20	2.75	90
46000	25.105	10	43.24	20	3.035	90
47000	25.045	10	44.425	20	3.56	90
48000	26.98	10	46.54	20	4.64	90
49000	26.71	10	46.93	20	4.43	90
50000	27.475	10	48.265	20	5.87	90
51000	28.015	10	49.045	20	7.25	90
52000	29.17	10	50.14	20	7.46	90
53000	28.375	10	51.625	20	8.51	90
54000	29.515	10	52.885	20	9.59	90
55000	30.535	10	52.735	20	9.425	90
56000	29.98	10	54.76	20	9.44	90
57000	31.42	10	54.64	20	9.845	90
58000	31.39	10	55.765	20	11.105	90
59000	32.11	10	56.395	20	10.835	90
60000	32.35	10	57.235	20	11.57	90
61000	32.65	10	57.715	20	11.9	90
62000	33.085	10	57.355	20	12.56	90
63000	33.835	10	58.54	20	12.65	90
64000	33.895	10	59.125	20	12.905	90
65000	33.91	10	59.89	20	13.25	90
66000	34.345	10	60.34	20	13.505	90
67000	34.375	10	60.97	20	14.33	90
68000	34.96	10	60.64	20	13.985	90
69000	35.095	10	62.095	20	14.735	90
70000	35.905	10	61.915	20	14.765	90
71000	35.305	10	61.66	20	15.725	90
72000	35.455	10	62.23	20	15.275	90
73000	35.275	10	61.585	20	14.825	90
74000	35.65	10	62.215	20	14.705	90
75000	35.005	10	61.45	20	15.455	90
76000			61.495	20	61.495	90
77000			61.345	20	61.345	90
78000			62.875	20	62.875	90
79000			62.47	20	62.47	90
80000			62.23	20	62.23	90

Appendix II – Code

All results and code are available on GitHub - https://github.com/Dazack/v2v_sim