Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)

Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод тестування захисту від програм-вимагачів на основі навчання з підкріпленням
A method for anti-ransomware testing based on reinforcement learning
(тема)

Виконав:
студент 2 курсу, групи АМСЗІм-20-1
Бен Гуррам Мохамед Тайеб
(прізвище, ініціали)

Спеціальність: 125 Кібербезпека
(код і повна назва спеціальності)

Тип програми: освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма: Адміністративний менеджмент у сфері захисту інформації
(повна назва освітньої програми)

Керівник: доцент кафедри АПОТ
Адамов О.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Лемешко О.В.
(підпис) (прізвище, ініціали)

2022р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)

Кафедра Інфокомунікаційної інженерії імені В.В. Поповського

(повна назва)

Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Адміністративний менеджмент у сфері захисту інформації
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри_____
(підпис)
«____» _____2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Бен Гуррам Мохамед Тайеб
(прізвище, ім'я, по батькові)

1. Тема роботи: Метод тестування захисту від програм-вимагачів на основі навчання з підкріпленням.
затверджена наказом по університету від «26» березня 2022р. №175 Стз
2. Термін подання студентом роботи до екзаменаційної комісії 31.05.2022р.
3. Вихідні дані до роботи: початковий код дослідження для моделювання, документація python, документація tensorflow.

4. Перелік питань, що потрібно опрацювати в роботі:
1. Чи можемо ми використовувати штучний інтелект, щоб зробити програму-вимагач сильнішою? Чи може програма-вимагач обійти детектори програм-вимагачів, використовуючи комбінацію відомих методів?
2. Що ефективніше: Deep Q-Network чи простий Q-Learning Algorithm для навчання таких моделей?
3. Як використання штучного інтелекту вплине або повинно вплинути на поточні методи виявлення програм-вимагачів.

5. Перелік графічного матеріалу із зазначенням креслень, плакатів, комп'ютерних

ілюстрацій: Демонстраційний матеріал у вигляді ppt-презентації; набір тестів із використанням різних алгоритмів із різними гіперпараметрами для пошуку найкращого методу обходу антивимагань.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Основна частина	доц. Адамов Олександр Семенович		31.05.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	15.02.2022	Виконано
2	Збір матеріалів для дослідження	28.02.2022	Виконано
3	Розробка 1 розділу	19.03.2022	Виконано
4	Розробка 2 розділу	02.04.2022	Виконано
5	Розробка 3 розділу	12.04.2022	Виконано
6	Розробка 4 розділу	23.04.2022	Виконано
7	Розробка 5 розділу	01.05.2022	Виконано
8	Розробка 6 розділу		Виконано
9	Розробка 7 розділу		Виконано
10	Оформлення кваліфікаційної роботи	10.05.2022	Виконано

Дата видачі завдання 15 лютого 2022 року

Студента _____ Бен Гуррам Мохамед Тайеб
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Адамов О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка – 89 с., кількість таблиць – 1, кількість рисунків – 34, кількість джерел – 15.

ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, DQN, DQL, QL, НАВЧАННЯ З ПІДКЛЮВАННЯМ

Метою дослідження є виявлення можливості використання навчання з підкріпленням за допомогою поширених методів для обходу виявлення антивимагань.

Мета роботи полягає в тому, щоб знайти слабкі місця в поточних захистах від програм-вимагачів і виправити їх до того, як станеться реальна атака. Спочатку буде використовуватися Q-навчання, потім буде вивчений алгоритм Deep Q-Network для кращих результатів.

Метою дослідження є оцінка можливості обходу захисту від програм-вимагачів за допомогою комбінації відомих методів. Атака програм-вимагачів стала дуже популярною в останній рік, новини про компанії та окремих осіб, які постраждали від втрат через зараження програм-вимагачів на їхніх машинах, почастішали. Дослідники безпеки дуже добре працюють над вирішенням і запобіганням цих атак.

У цьому документі ми проведемо експерименти, щоб вивчити деякі майбутні прийоми, які хакери можуть використовувати в майбутньому, щоб обійти захист від програм-вимагачів. Дуже важливо передбачити та захиститися від таких методів до того, як станеться реальна атака. Оскільки атаки програм-вимагачів є дуже критичними, особливо тому, що компанії отримали зашифровані дуже важливі документи, і ще гірше, якщо вони не мають резервних копій, люди в більшості випадків втрачають важливі особисті файли, а в більшості випадків вони не мають резервних копій.

ABSTRACT

The report contains: 89 pages, 1 table, 34 figures and 15 sources.

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, DQN, DQL, QL, REINFORCEMENT LEARNING

The object of the research is to discover the possibility of using reinforcement learning with common techniques to bypass anti-ransomware detection.

The aim of the work is to find weaknesses in the current anti-ransomware defenses and fix them before a real-world attack happens. In the beginning Q-learning will be used, then Deep Q-Network algorithm will be explored for better results. The research aims to evaluate the possibility of bypassing anti ransomware protection using combination of known techniques.

Ransomware attack becomes very populaire in the recent year, the news about companies and individuals that suffered from losses because ransomware infection in their machines becomes more frequent. Security researchers are doing very good job addressing and preventing these attacks.

In this document, we will run experiments to explore some future techniques that black hat hackers could use in the future to bypass anti ransomware protection, it is very important to predict and defend against such techniques before a real-world attack happens. Since ransomware attacks are very criticial especially that companies got very important documents encrypted, and it is worse if they don’t have backups, individuals lose in most cases important personal files, and in most cases, they don’t have backups.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS 8
1 INTRODUCTION 9
1.1 Problem statement 9
1.2 Aim and objectives 9
1.3 Research questions 10
1.4 Scope and limitations 10
1.5 Ethical consideration 11
1.6 Document outline 11
2 BACKGROUND 13
2.1 Reinforcment learning 13
2.1.1 Q-Learning Algorithm 14
2.1.2 Deep Q-Learning 15
2.2 Ransomware 16
2.3 Reinforcement learning in cybersecurity 16
3 METHOD 18
3.1 Research method 18
3.2 Experiment 19
3.2.1 Environment setup 19
3.2.2. Tools and techniques 20
3.2.3 Test cases 23
4 RESULTS 26
4.1 Experiment results 26
4.1.1 Q-Learning: Test case 1 26
4.1.2 DQN: Test case 1 31
4.1.3 Q-Learning Test case 2 35
4.1.4 DQL Test case 2 37
5 ANALYSIS AND DISCUSSION 40
5.1 Research limitations 40

5.2. Experiment analysis and discussion 40
5.2.1 Q Learning Test case 1 40
5.2.2 DQL Test case 1 41
5.2.3 Comparing DQL and QL test case 1 results 41
5.2.4 Q Learning test case 2 42
5.2.5 Q Learning test case 2 42
5.2.6 DQL vs QL in test case 2. 42
6 DISCUSSIONS OF EXPERIMENTS AND FINDINGS 44
6.1 Research question 1 44
6.2 Research question 2 44
6.3 Research question 3 44
7 CONCLUSIUONS AND FUTURE WORK 46
7.1 Future work 46
8 BIBLIOGRAPHY 47
9 APPENDIX A 49
9.1 File: gambler.py 49
9.2 File: deepgambler.py 51
9.3 File: observer.py 55
9.4 File: train.py 65
9.5 File: cryptosim.py 70
9.6 File: ransomeware_simulator.py 76
10 APPENDIX B 82

LIST OF ABBREVIATIONS

AI – Artificial Intelligence

ML – Machine Learning

DQN – Deep Q-Network

DQL – Deep Q-Learning

QL - Q-Learning

RL – Reinforcement Learning

1 INTRODUCTION

The total cost of a ransomware breach was an average of $4.62 million in 2021, not including a ransom. (IBM) The average cost for education institutions to rectify the impacts of a ransomware attack, including the ransom itself, was $2.73 million in 2021–48% higher than the global average for all sectors.

As ransomware detection becomes good, these malwares evasion techniques become even better, so it is necessary to conduct experiments to find weaknesses in the defense/detecting system and solve them quickly.

Using Reinforcement Learning, Q-Learning algorithm in the beginning, then optimize it by using Deep Q-Network algorithm is the best way to start conducting this experiment. Since supervised learning would only provide statistical results rather than practical. The robustness of these methods is not certain, especially methods using supervised learning, which tends to extract static features and statistical characteristics, instead of doing dynamic or in-depth analysis. That may make supervised learning vulnerable to be attacked or evaded.

This work aims to expose the weaknesses in the ransomware detection systems, we will have a ‘Ransomware Defense Simulation’ and a ‘Ransomware Attack Simulation’. The result can be used to improve the efficiency of ransomware detectors.

1.1 Problem statement

The area of research has been investigated previously, first off there is similar published research study that try to bypass anti ransomware detection, there was also another study that studies possibility of using RL in penetration testing. The new thing about this research is using DQN instead of Q-Learning algorithm for training the model.

1.2 Aim and objectives

The project focuses on researching the possiblity of using RL using Q-Learning Algorithm or DQN and run many simulations in a game like style.

The player is the crypto simulator that tries to encrypt all the documents in each folder, but there's a guardian which is the ransomware detector.

Using combination of common bypass techniques in the crypto simulator vs using different ransomware detecting techniques. and let both run for different sessions of the game, and for each game running, the player gains and collect experience and becomes better by time.

Objectives:

a) Explore the possibility of encrypting files inside a folder withouyt being detected by the ransomeware detector;
b) Optimize the model by using more sophisticated training algorithm that would make it usable in real world;
c) Evalute the differences between DQN and Q-Learning;

1.3 Research questions

Ransomwares often uses different techniques to bypass detection, anti ransomwares themselves are using different smart technques to detect. For that reason, we are trying to answer the following question:

a) is it possible to train a model that uses combination of common bypass techniques to detect strict anti ransomware rules;
b) How far can we go in using AI, especially RL that does not require any data, in cyber security;
c) We know that DQL is potentially better than QL, but how much is the difference between the two algorithms in RL;

1.4 Scope and limitations

The research and experiments are limited to Windows OS, since the most famous ransomware attacks, targeted primarly Windows OS such as WastedLocker, Maze, Netwalker.

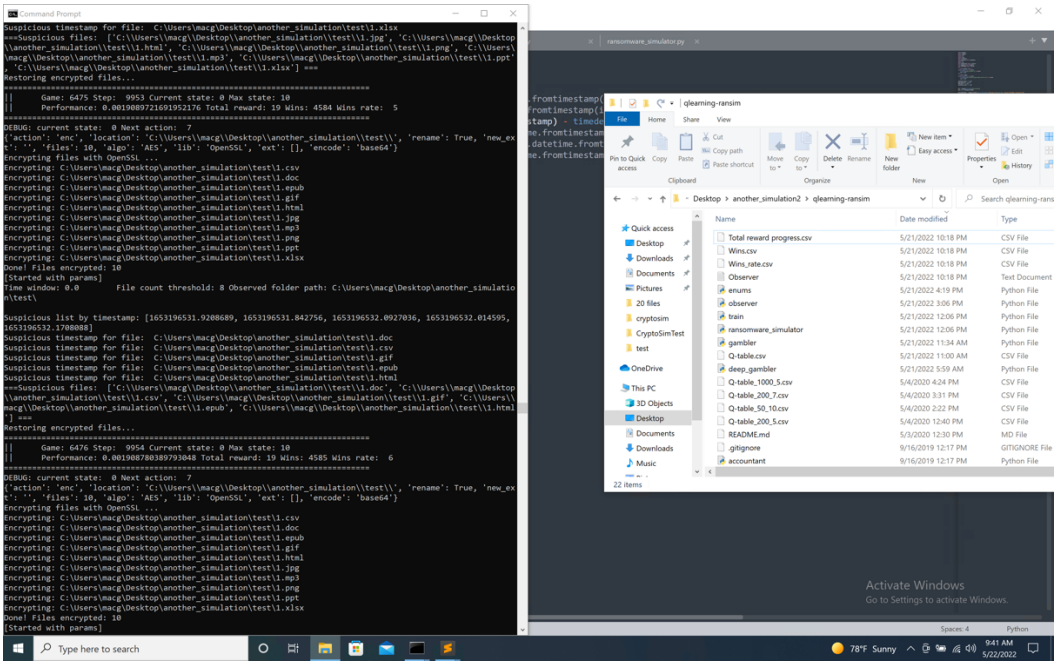

Figure 1.1 – A screenshot of the ransomware simulation environment.

1.5 Ethical consideration

The experiments are executed in a controlled envirement, a fresh isolated windows virtual machine, that is not connect to any other network, the experiment is done locally. And uses python languages for executing the simulation, the encryption is isolated into a single specified and controlled folder.

1.6 Document outline

Background describes information necessary to understand the content of the thesis. It explains the basics of Reinforcement learning, ransomware attacks, anti ransomware techniques, Q-Learning algorithm, Deep Q-Network. Related works informs of relevant papers that research closely connected with this thesis work. Method defines the research method, the environment, the programming languages and libraries such as tensorflow, as well as the table of actions, and ransomware detector methods. Results present the outcome of the different controlled experiments that has been done using 2 different RL algorithms, Q-Learning and DQN. Analysis and Discussion highlights the interesting and val-

uable findings of the experiment. It discusses the experiment findings and answers the research questions based on the information gathered. Conclusion and Future work summarise the thesis total outcome and suggests relevant research topics for future studies.

2 BACKGROUND

2.1 Reinforcment learning

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning, and unsupervised learning.

Reinforcement learning differs from supervised learning in not needing labeled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). Partially supervised RL algorithms can combine the advantages of supervised and RL algorithms.

The environment is typically stated in the form of a Markov decision process (MDP) because many reinforcement learning algorithms for this context use dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP, and they target large MDPs where exact methods become infeasible.

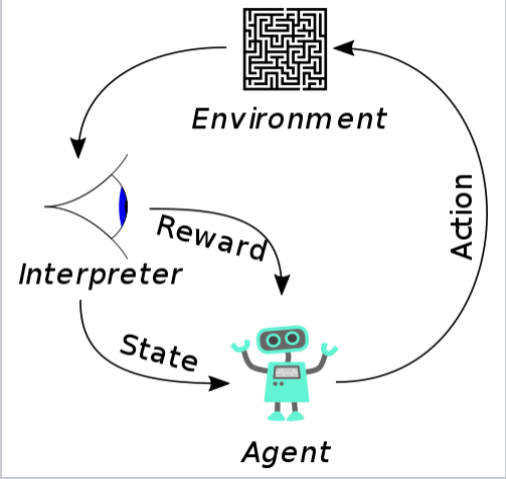

Figure 2.1 – Demonstration of Reinforcement Learning

2.1.1 Q-Learning Algorithm

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state.

It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations. For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over all successive steps, starting from the current state. Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly random policy. "Q" refers to the function that the algorithm computes – the expected rewards for an action taken in a given state.

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Figure 2.2 - Q-Learning table of states by actions that is initialized to zero, then each cell is updated through training.

When to use reinforcement learning? Reinforcement learning is useful when you have no training data or specific enough expertise about the problem.

2.1.2 Deep Q-Learning

In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output. The comparison between Q-learning & deep Q-learning is wonderfully illustrated in Figure 2.3.

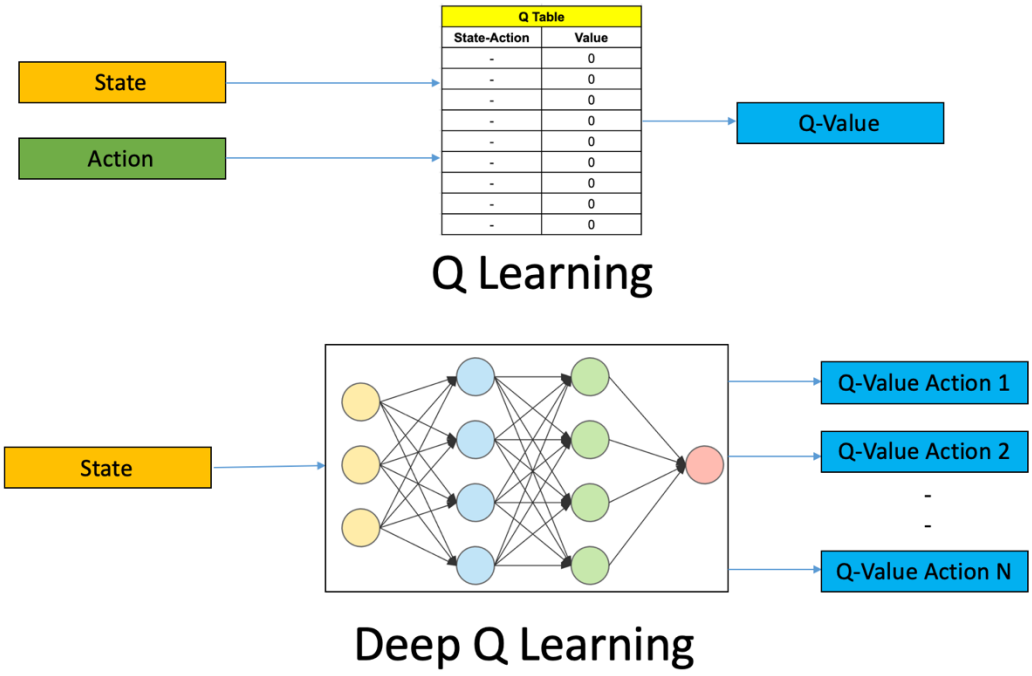

Figure 2.3 – Demonstrating the difference between Q-Learning and Deep Q-Learning

So, what are the steps involved in reinforcement learning using deep Q-learning networks (DQNs)? All the experience is stored by the user in memory the next action is determined by the maximum output of the Q-network, The loss function here is mean squared error of the predicted Q-value and the target Q-value – Q*. This is basically a regression problem. However, we do not know the target or actual value here as we are deal-

ing with a reinforcement learning problem. Going back to the Q-value update equation derived fromthe Bellman equation.

The section in green represents the target. We can argue that it is predicting its own value, but since R is the unbiased true reward, the network is going to update its gradient using backpropagation to finally converge.

2.2 Ransomware

Ransomware is a type of malware from cryptovirology that threatens to publish the victim's personal data or perpetually block access to it unless a ransom is paid. While some simple ransomware may lock the system without damaging any files, more advanced malware uses a technique called cryptoviral extortion.

It encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them. In a properly implemented cryptoviral extortion attack, recovering the files without the decryption key is an intractable problem – and difficult to trace digital currencies such as paysafecard or Bitcoin and other cryptocurrencies that are used for the ransoms, making tracing and prosecuting the perpetrators difficult. Ransomware attacks are typically carried out using a Trojan disguised as a legitimate file that the user is tricked into downloading or opening when it arrives as an email attachment. However, one high-profile example, the WannaCry worm, traveled automatically between computers without user interaction.

2.3 Reinforcement learning in cybersecurity

The scale of Internet-connected systems has increased considerably, and these systems are being exposed to cyber attacks more than ever. The complexity and dynamics of cyber attacks require protecting mechanisms to be responsive, adaptive, and scalable. Machine learning, or more specifically deep reinforcement learning (DRL), methods have been proposed widely to address these issues. By incorporating deep learning into traditional RL, DRL is highly capable of solving complex, dynamic, and especially high-

dimensional cyber defense problems. One example of using RL in cybersecurity other than this paper, is the using it in penetration testing.

3 METHOD

This chapter describes the different processes, methods and tools used to conduct these experiments. It will also show the tools that will be used for different training techniques, QL and DQL.

3.1 Research method

To conduct this research, we need first set the rules for game for the reinforcement learning. The player will be able to take 16 actions, which is a combaination of 4 variables and methods that are widely used to bypass antiransomware detection. The variables which creates a combination of actions will be: number of files to be encrypted for that state (limited to 1, 2, 5 or 10), to use or to not use base64 encryption, to add or not to add '.enc' extension.

The defender will use common ransomware detection techniques in real time during each game. The methods for detections are, timestamp, if the large number of files changed at the same time inside the folder. The entropy of the file, it measures the randomness of the data, and type of file through magic number and analysis of general structure of file, if common structure is found, then it is not a suspicious file, even if the entropy is high, which the case for most files such as images, videos, audios. The defender may also use double extension detection, since lot of ransomwares add '.enc' or '.encrypted' extension.

The playground of the game is a folder, it is called TEST_FOLDER, in which there will be 10 files initialy. The files are chosen to be common files such as: PDF, XLSX, DOC, DOCX, PNG, JPG, CSV, etc

The first experiments or RL algorithms starts by settings and Q-Table full of 0, and these table to be optimized while running games. When the. Game starts, the player runs a step according to the state, with a maximum number of 10 states. And it begins by chosing random action out of the 16 actions, and from there it optimizes the value of the Q-Table

using Bellman equation. The second experiment is using Deep RL, it trains a neural network, to better predict values for the best action to take in the current state.

3.2 Experiment

3.2.1 Environment setup

The Operating system is windows, it needs python 3 to be installed. And then to download multiple python libraries that are required such as:

a) Numpy
b) Python-magic
c) Tensorflow

Then we need the files to be encrypted in each session of the game, for this research, we take different samples of common filetypes such as CSV, EPUB, XLSX, DOCS.

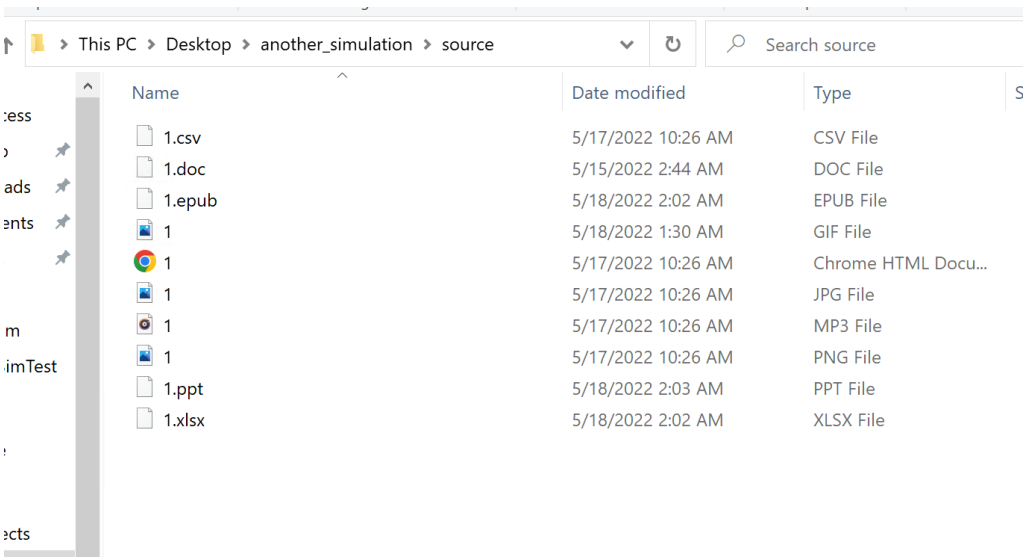

Figure 3.1 – Test folder with 10 sample files of common types

After setting the source folder, a test folder will be needed, in which the actual simulation of the game will be executed and played, and after each game it will reset, means reseting it is using the 10 files on the test folder.

3.2.2 Tools and techniques

The code was written to run different agents, what we will use in this experiment is one agent that outputs a Q-Table, and other one that trains a Deep Q-Network. The encryption method used for this experiment is OpenSSL and AES 256 encryption.

```
def encrypt_openssl(fullpath, enc_ext, *args):
    outputfile = fullpath + "temp"
    cmd_status = os.system("openssl aes-256-cbc -e -in %s -out %s -pass pass:%s -pbkdf2" % (fullpath, outputfile, PASSWORD))
    if cmd_status != 0:
        print("ERROR: Error executing OpenSSL")
    else:
        os.remove(fullpath)
        os.rename(outputfile, fullpath + enc_ext)
        if args[0].lower() == 'base64':
            encoding(fullpath + enc_ext)
    return cmd_status
```

Figure 3.2 The OpenSSl function used to encrypt files during ransomware simulation

After each successful simulation we get data that consist of:

a) Number of wins compared to number of games
b) Wins per 10 games

Also, for each simulation here is the table showing the 16 actions

Table 3.1 – Table of actions

Action	Extension	Base64	Number of files(code)	Num of files to be encryped
0	0	0	0	1
1	0	0	1	2
2	0	0	2	5
3	0	0	3	10
4	0	1	0	1
5	0	1	1	2
6	0	1	2	5
7	0	1	3	10
8	1	0	0	1
9	1	0	1	2
10	1	0	2	5
11	1	0	3	10
12	1	1	0	1
13	1	1	1	2
14	1	1	2	5
15	1	1	3	10

The ransomware detector uses the following techniques for detecting suspicious files:

a) Measuring the entropy level and filetype;
b) Calculating timestamp, if file was modified just in nanoseconds, in the experiment it's set to 0.08s;
c) Detecting double extension;
d) The threshold for this experiment is 8, meaning if the ransomware detector, counts 8 suspicious files, a ransomware attack is detected, and the game is lost;

If we consider these methods for detections, here's the methods that the player will use to bypass them:

a) Base64 encoding recuces the files entropy;
b) Avoiding using double extension will also bypass the double extension detection method;
c) As shown in the table 3.1 the number of files per iteration can bypass timestamp detection;

If we let the model trains, i twill learn the best policy to bypass this anti ransomware detection using just common techniques.

In the Q table simulation, we need to calculate the reward using bellman equation after each iteration outcome.

In other words, for one encrypted file the player earns 2 points of reward, and one action costs 1 point. The algorithm starts with the Random actions and then slowly reduce the probability of random choice for an action from 1 in the beginning to 0 at the end of the learning process.

That was for Q-Learning, for Deep Q-learning, things are different, we will use neural network, so there will no Q-Table. There will a neural network, that will be trained using Tensorflow.

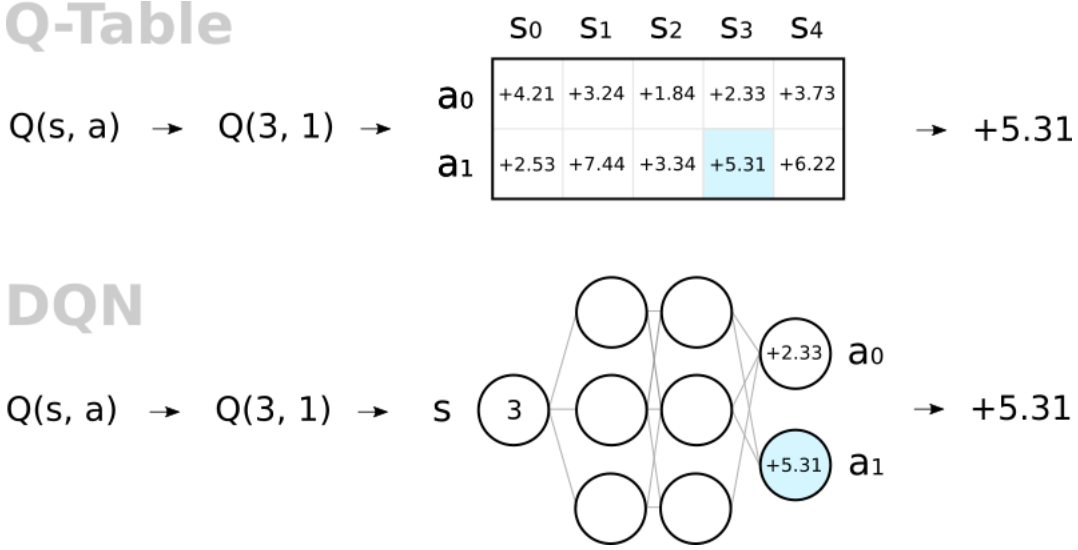

Figure 3.3 – Difference between Q-table and Deep Q-Network

As we see in figure 3.4, in the DQN, we just give the neural network an input of State and action and it gives us a predicted value.

In the Q-table, it does run and optimize the value, for that exact game, but the neural network is flexible, and it trains to predict the best value of Q-Table. This value may not be optimal, but the most important that it will work.

The Q-Table is not felxbile since it will give you the best policy for that exact game variables, if the game variables changed, the policy may not be the best to follow.

3.2.3 Test cases

There will be 4 experiments, in which 2 are Q-learning and 2 others are DQL, first two simulation, will be on an easy game game 10 states, 10 files to encrypt and 16 actions, then we will make it harder by increasing the number of states and number of files to be encrypted to 20, to compare the efficiency of each algorithm in RL.

a) Q-Table test case 1:
    1) In this test case we will use the following parameters;
    2) Q-Learning algorithm agent, called Gambler;
    3) 10000 iterations;
    4) 10 states;
    5) 10 files to encrypt;
    6) 16 possible actions;
    7) Timestamp delta of 0.08;
    8) OpenSSL for file encryption;
    9) Ransomeware detector threshold of 8;

b) DQN test case 1:
    1) In this test case we will use the following parameters;
    2) Deep Q-Learning algorithm agent, called DeepGambler;
    3) 10000 iterations;
    4) 10 states;
    5) 10 files to encrypt;
    6) 16 possible actions;
    7) Timestamp delta of 0.08;
    8) OpenSSL for file encryption;
    9) Ransomeware detector threshold of 8;

c) Q-Table test case 2:

1) In this test case we'll use the following parameters;
2) Q-Learning algorithm agent, called Gambler;
3) 10000 iterations;
4) 20 states;
5) 20 files to encrypt;
6) 16 possible actions;
7) Timestamp delta of 0.08;
8) OpenSSL for file encryption;
9) Ransomeware detector threshold of 8;

d) DQN test case 2:

1) In this test case we'll use the following parameters;
2) Q-Learning algorithm agent, called Gambler;
3) 10000 iterations;
4) 20 states;
5) 20 files to encrypt;
6) 16 possible actions;
7) Timestamp delta of 0.08;
8) OpenSSL for file encryption;
9) Ransomeware detector threshold of 8;

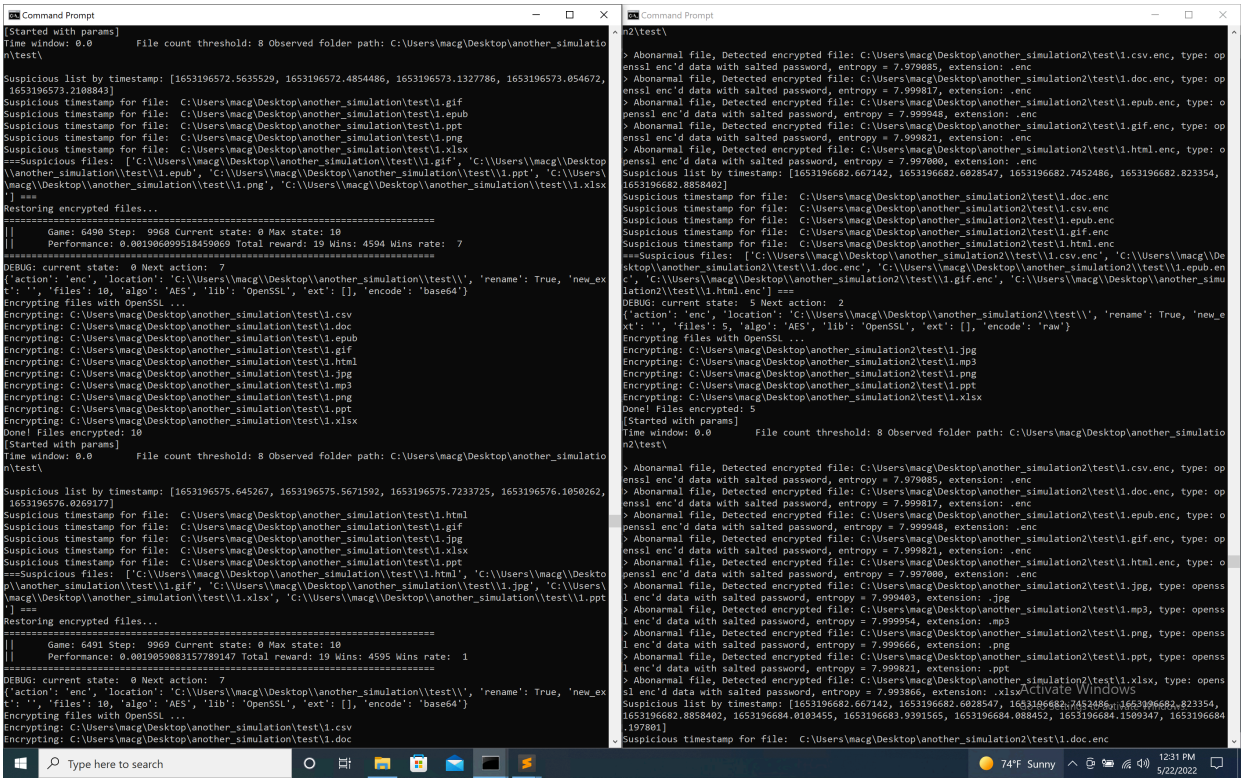

Figure 3.4 – Simulations of both test cases, on the left, the Q-Learning, the right the Deep Q-Learning

4 RESULTS

4.1 Experiment results

In this chapter we are going to explore the data gathered from each test case.

4.1.1 Q-Learning: Test case 1

After the model finished the training we got lot of data plus the Q-Table that represents the best policy for follow.

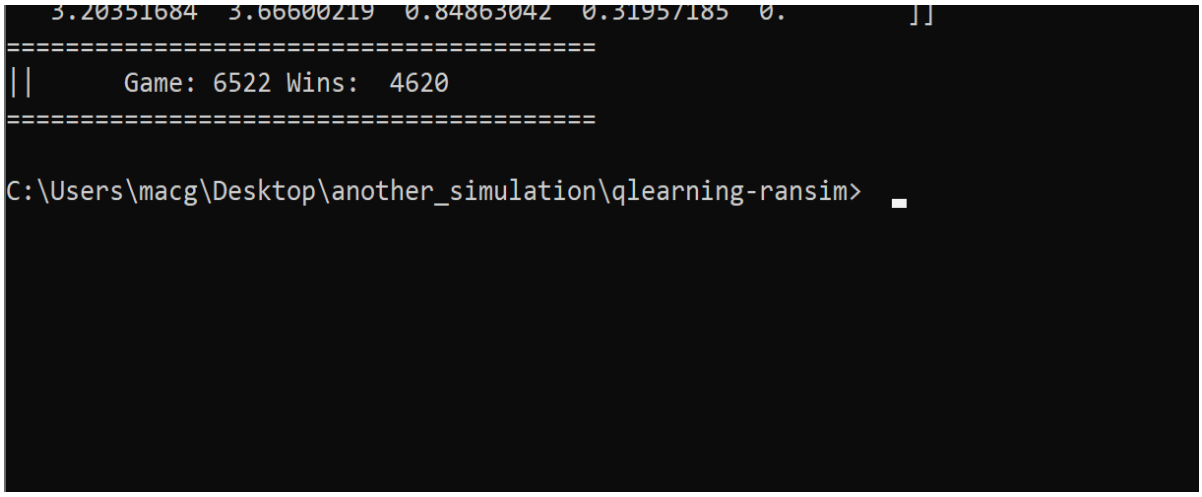

Figure 4.1 – The end of simulation of Q-Learning algorithm

As We see for 10000 iterations, 6522 games were played, in which the model won majority of them 4620.

Q(S, A)	States								Threshold		
Actions	0	1	2	3	4	5	6	7	8	9	10
0	17.064138	11.373914	9.434797	3.333476	4.437139	3.524384	2.747256	2.789401	0.594475	0.347245	0
1	15.65915	11.729954	9.363538	5.28466	3.349453	6.165858	3.647353	3.483226	0.852475	0.141511	0
2	17.39207	12.450127	12.214027	5.93241	0.166427	8.158896	2.39941	3.007965	0.579618	0.375648	0
3	17.995947	12.793953	9.668399	4.17875	1.770299	7.984168	1.598049	3.455996	0.3	0.235137	0
4	17.048778	11.361066	9.886199	4.734564	2.765516	4.323236	1.824859	2.74448	0.353823	0.957609	0
5	15.678352	10.160259	9.418679	11.368987	1.926289	6.840074	3.202167	2.946587	0.494003	0.271	0
6	17.369262	11.877071	12.361254	5.822014	9.523121	8.810861	2.536652	4.491754	0.559509	0.271	0
7	17.382124	16.876812	11.135758	6.817596	3.04458	8.107177	3.696525	4.0096	2.311116	0.468559	0
8	17.054053	11.68494	10.499314	2.442453	4.593637	4.585682	2.698563	3.414391	0.340277	0.342681	0
9	15.673266	10.794902	9.471515	5.84378	1.627716	6.939511	1.752751	3.052629	0.40781	0.232865	0
10	17.384413	11.539272	13.312174	3.499957	3.032259	8.286399	3.340707	2.985576	0.607114	0.355743	0
11	18.000647	13.925807	10.009137	3.37174	3.41559	7.802027	4.791765	3.314857	0.422058	0.233376	0
12	17.058852	7.45085	9.96248	3.453374	3.846584	4.219919	1.723634	3.157403	0.517162	0.143804	0
13	15.663129	11.669743	9.228592	5.045617	1.898836	6.799079	1.831609	2.755004	0.551517	0.229195	0
14	18.893459	11.154631	12.43444	7.433068	2.292787	7.788171	2.509881	3.112026	0.976161	0.032671	0
15	17.989217	11.34613	10.859368	1.063472	4.387827	7.737456	3.203517	3.666002	0.84863	0.319572	0

Figure 4.2 – Q-Table from running 10000 iterations on the first test case.

As we see this table value swere initially zeros, and as per games played, the Q values are calculated following, and per each iteration we can conclude the best action to take. For example, the first best action to take is in State 0 is 14 since it has the highest Q-values.

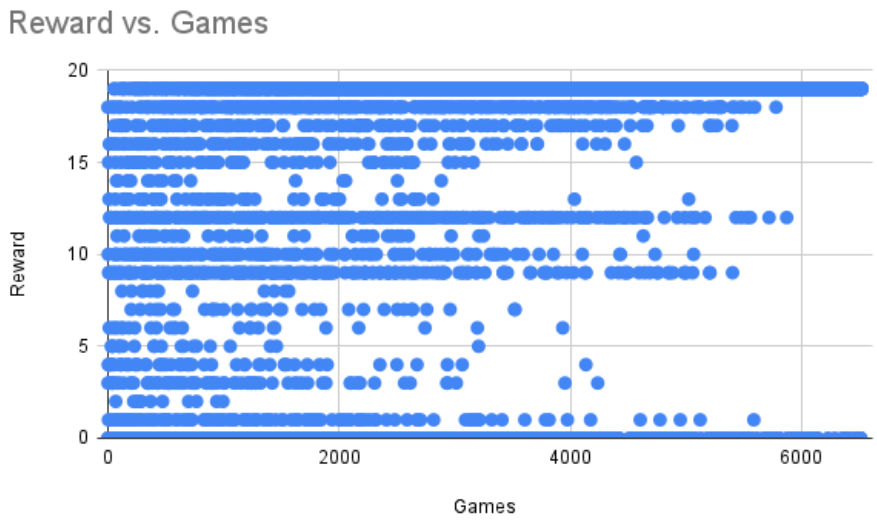

Figure 4.3 – Q-Learning Learning Progress Reward vs Games

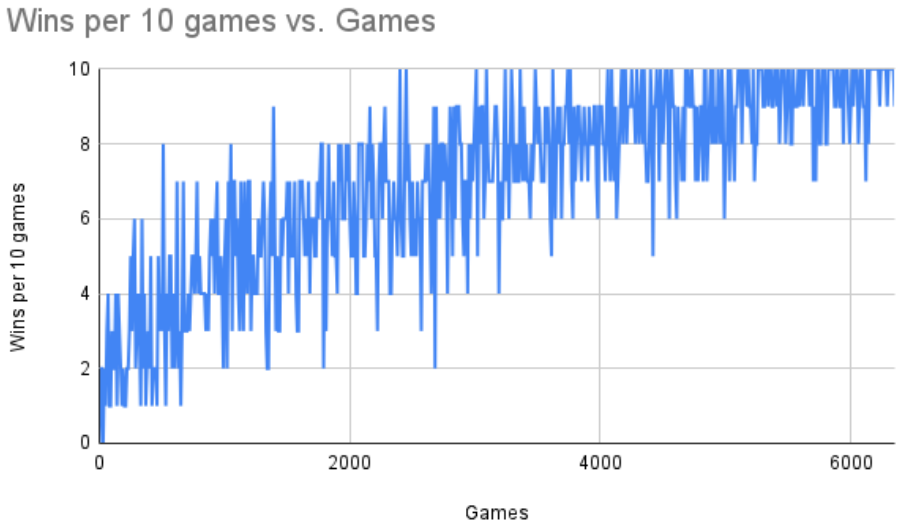

Figure 4.4 – Q-Learning Wins per ten games for Q-Learning

We can see that the wins per 10 games was increasing constantly.

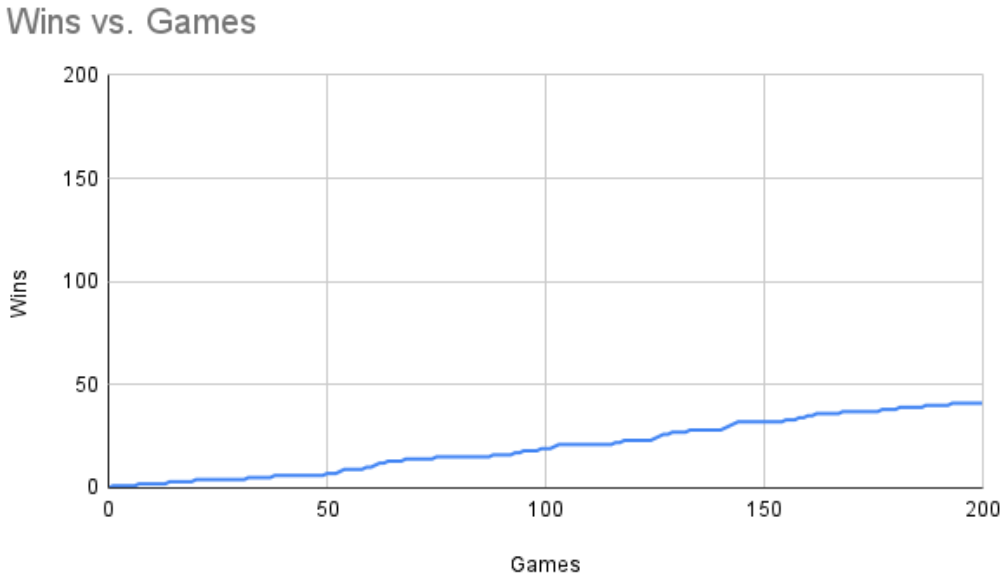

Figure 4.5 – Q-Learning Wins vs Games 200 games progress

We can see the the model is winning only few games in the first 200 games.

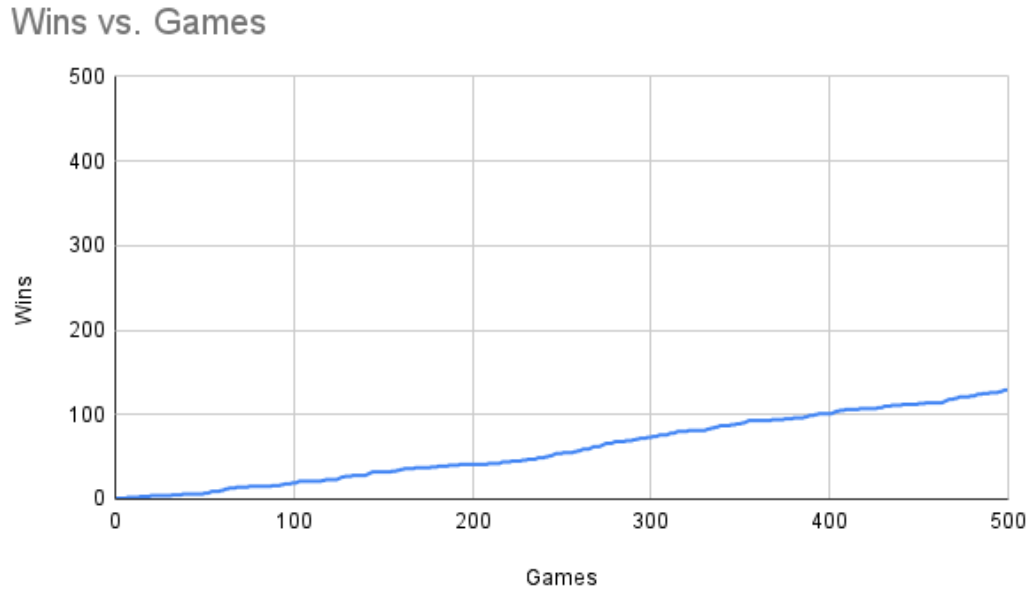

Figure 4.6 – Q-Learning Wins vs Games 500 games progress

The model slightly better after 500 games.

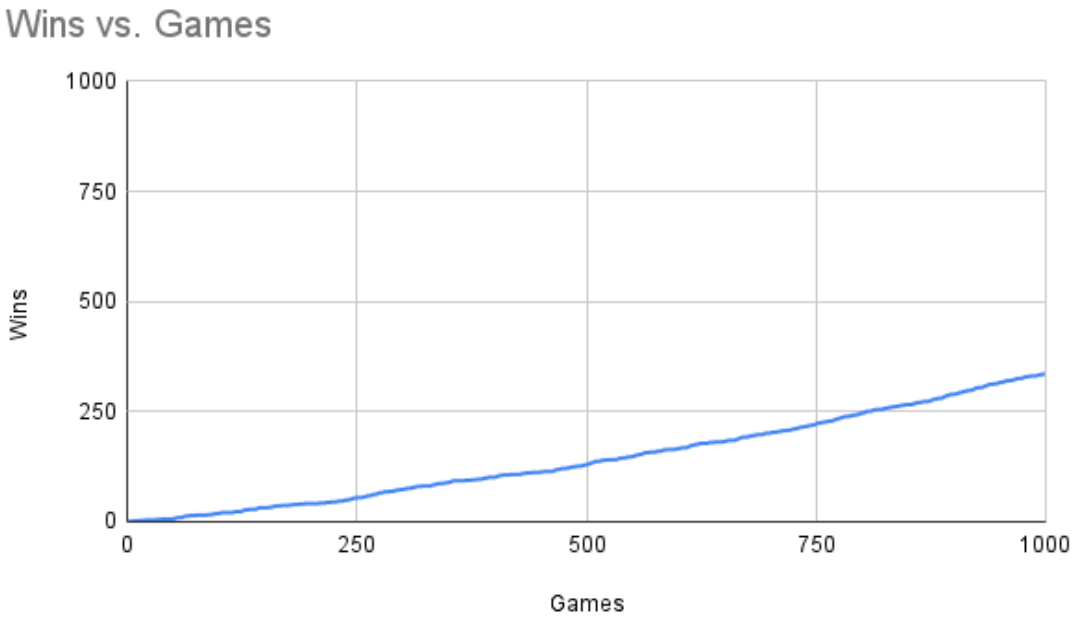

Figure 4.7 – Q-Learning Wins vs Games 1000 games progress

We see that the model starts to win more games as per 1000 games played.

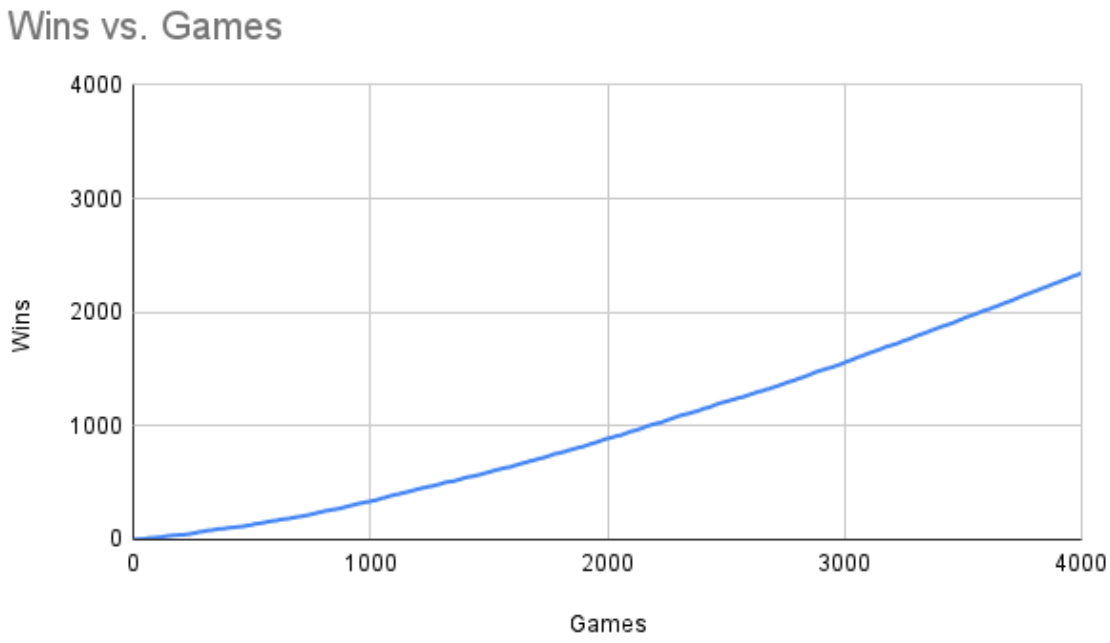

Figure 4.8 – Q-Learning Wins vs Games 4000 games progress

The model starts to perform extremely well from 2000 to 4000 games.

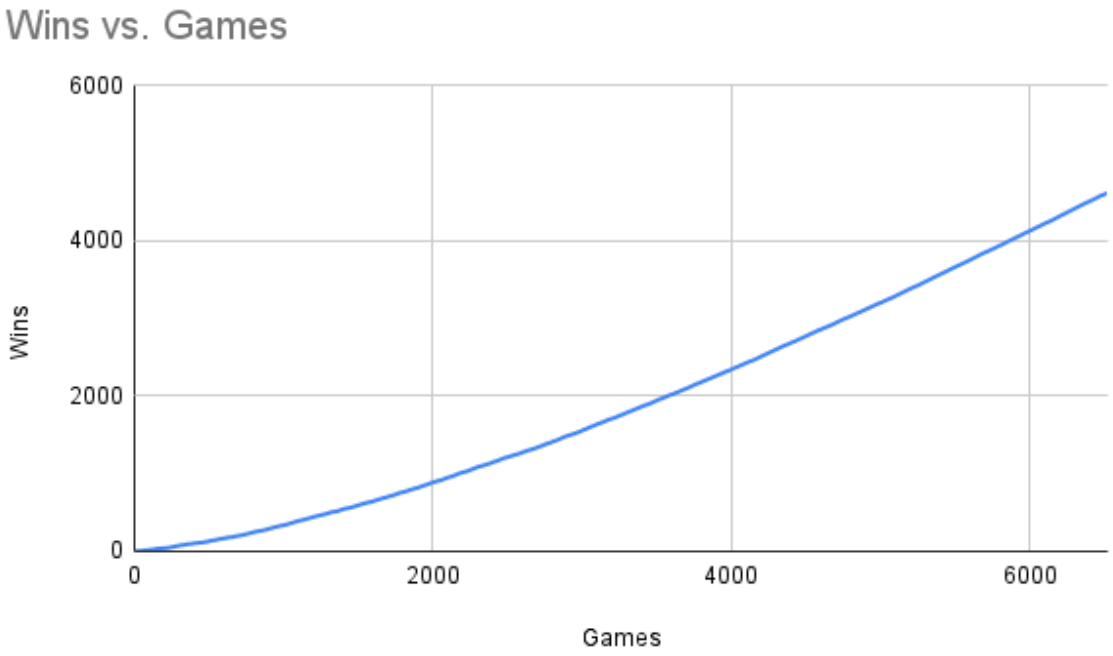

Figure 4.9 – Q-Learning Wins vs Games 6000 games progress

The model winning is finally growing exponentially.

4.1.2 DQN: Test case 1

This model uses tensorflow v1 for the neural network. To predict the Q-values of the states. The neural network will learn to predit the Q-value that related to the reward.

```
Suspicious timestamp for file:  C:\Users\macg\Desktop\another_simulati
Suspicious timestamp for file:  C:\Users\macg\Desktop\another_simulati
===Suspicious files:  ['C:\\Users\\macg\\Desktop\\another_simulation2\\
sktop\\another_simulation2\\test\\1.doc.enc', 'C:\\Users\\macg\\Deskto
c', 'C:\\Users\\macg\\Desktop\\another_simulation2\\test\\1.gif.enc',
lation2\\test\\1.html.enc'] ===
========================================
||      Game: 6495 Wins:  4621
========================================

C:\Users\macg\Desktop\another_simulation2\qlearning-ransim>
```

Figure 4.10 – DQN simulation results

As we see, the DQN played slightly less games, but won slightly more games. Now we will see more details about these wins.

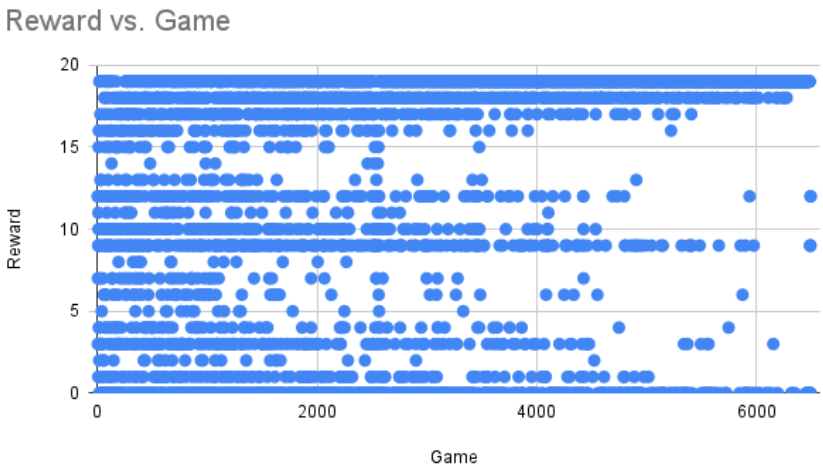

Figure 4.11 – DQN Reward vs Games

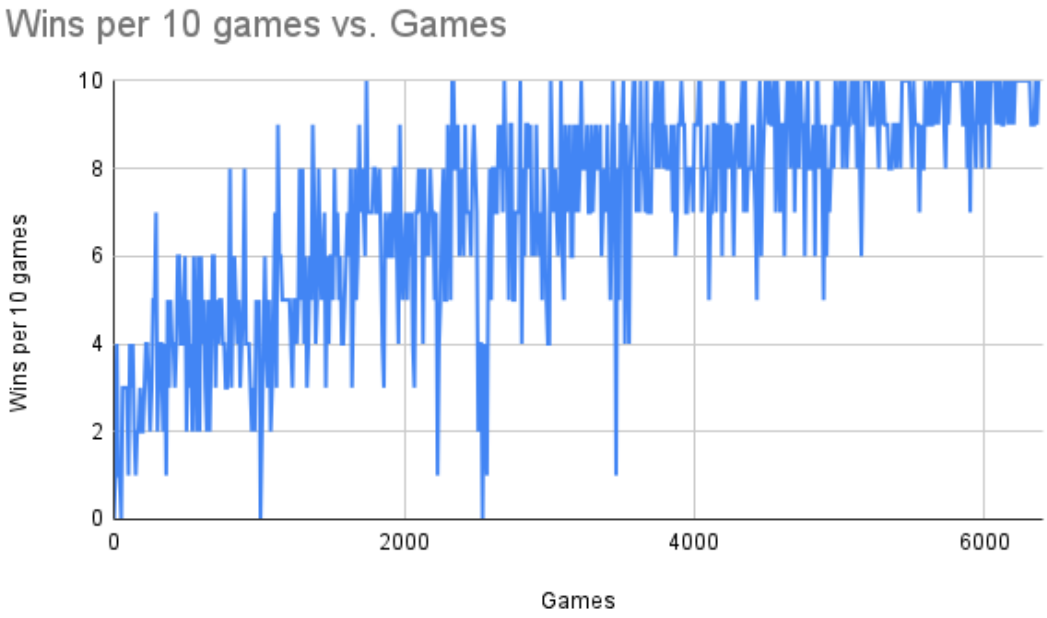

Figure 4.12 – DQN Wins per 10 games

DQN also learns to to win games 10/10 in the end and scope of errors is becoming less and less slight as the games progresses, which we can consider very good result.

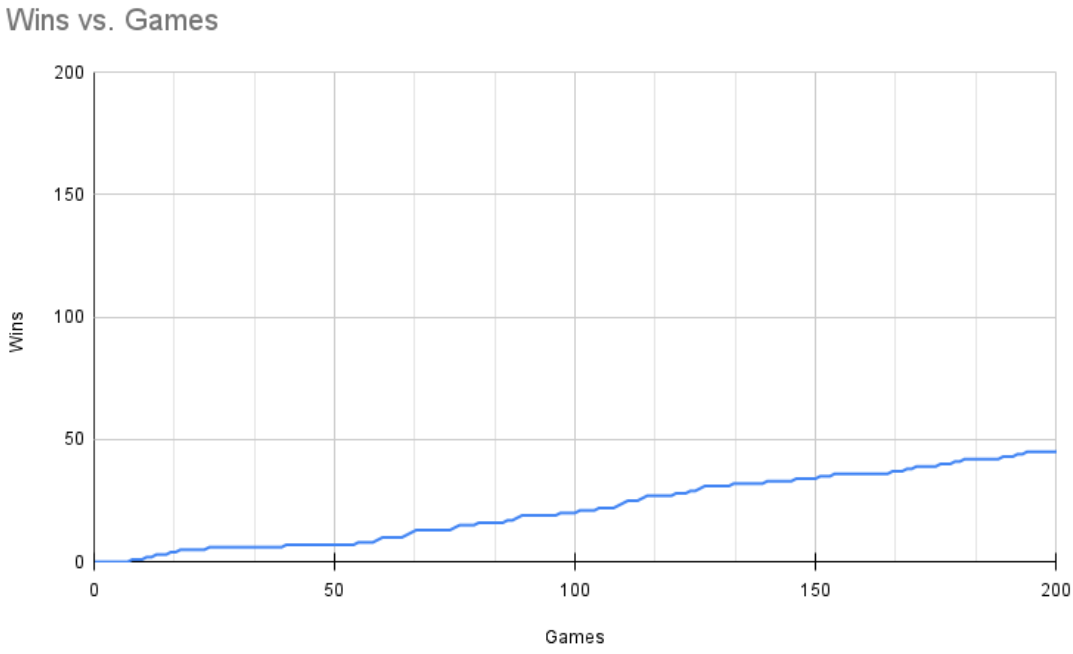

Figure 4.13 - DQN Wins 200 games progress

The wins are very low on the first 200 games.

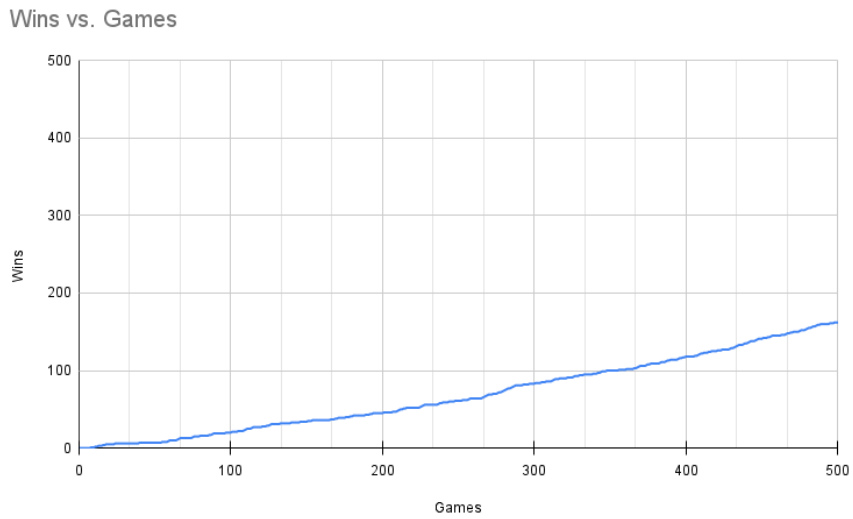

Figure 4.14 - DQN Wins 500 games progress

The model learns better and starts already to win a lot from 500 games progress

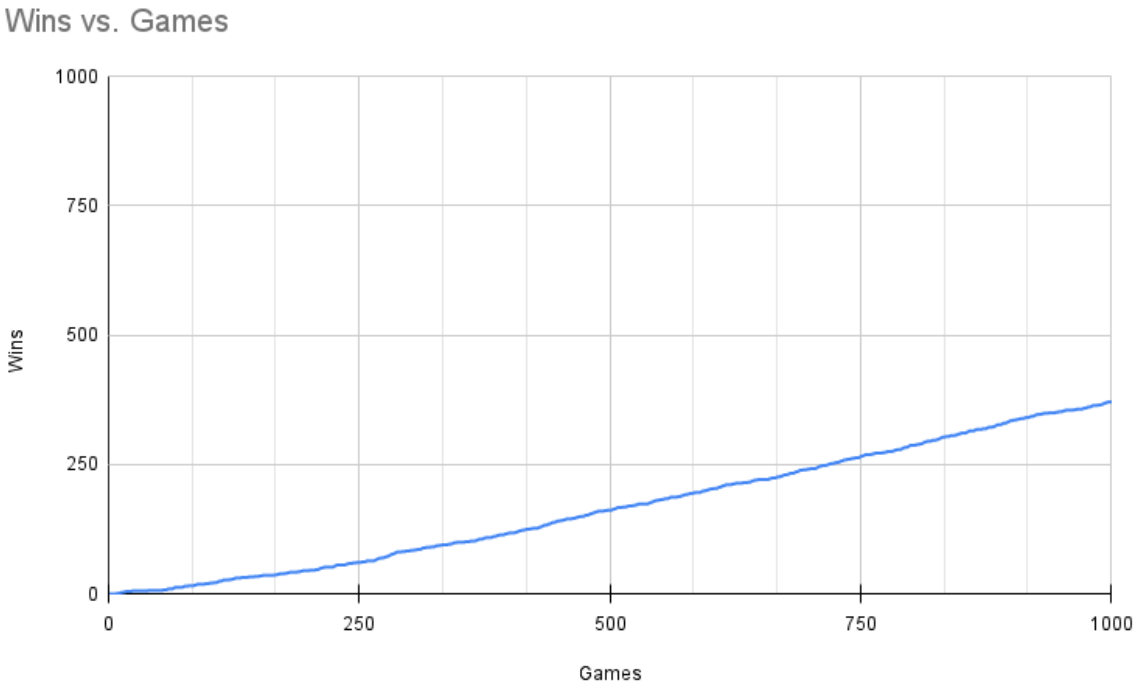

Figure 4.15 - DQN Wins 1000 games progress

The model is growing exponentially already on the 1000 games progress

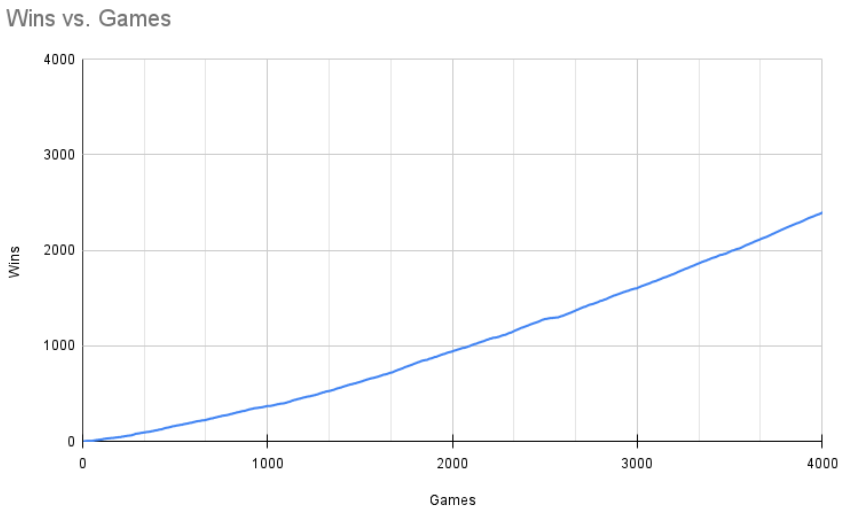

Figure 4.16 - DQN Wins 4000 games progress

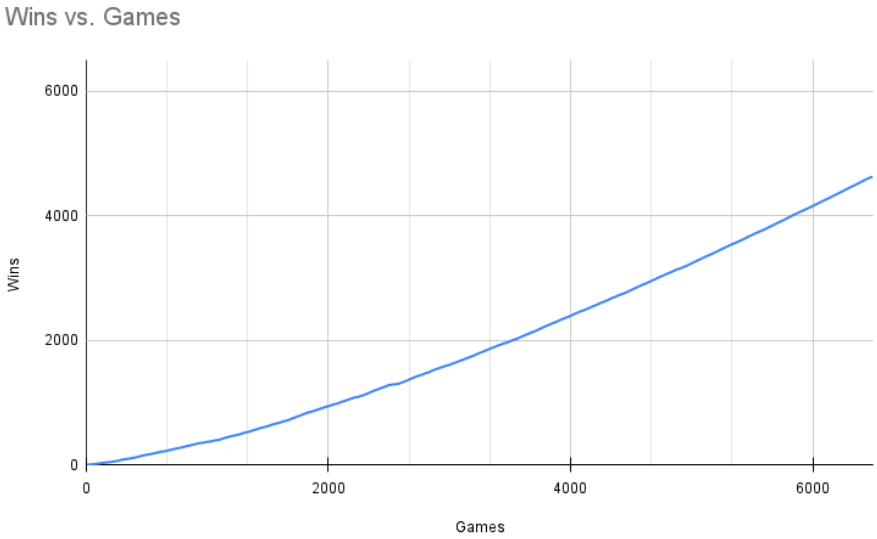

Figure 4.17 - DQN Wins 6500 games progress

Seems the model discovered the best policy for winning the game and encrypting all files without few little failures.

4.1.3 Q-Learning Test case 2

In the following figure, we will see the results from the second simulation in which we'll runn 4500 plus games, and as we'll see 1758 are wins, but what is interesting is the wins rate or wins per 10 games throught the 4587 games.

```
  0.3          0.1          0.         ]
 [34.81006517 15.00174427 15.13822744  2.5666834   0.07373425  9.95802266
   9.5495701   7.0643961   0.49354655  0.         17.24905424  3.343533
   1.95045445  1.6620707   3.15394745  2.29518533  1.79825107  0.58207043
   0.11671035  0.18905     0.         ]]
=======================================
||      Game: 4587 Wins:  1758
=======================================

C:\Users\macg\Desktop\another_simulation\qlearning-ransim>
```

Figure 4.18 – Output of games vs Wins at the end of q leaning test case 2

Q(S,A)	States																				
Action	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	30.917378	14.860792	6.809581	1.715921	0.44295	3.961406	6.554918	3.534599	0.421601	0.67508	15.359779	1.277411	5.41268	0.155865	1.393692	2.322459	1.985215	0.206383	0.250758	0	0
1	28.657275	7.141911	12.878813	7.903121	1.25902	6.797622	9.176728	4.207503	1.902531	0	15.299075	4.446743	4.701715	0.3	3.376047	1.985445	1.580567	0.407815	0.11671	0	0
2	34.852224	18.274566	11.979351	1.997468	2.064471	3.036457	8.354547	5.375127	1.576471	1.709237	16.93833	2.580444	3.884466	1.388416	2.759307	3.861257	2.308575	0.134423	0	0	0
3	34.856179	16.411216	16.214505	4.236124	0.724499	4.336103	8.764623	3.406667	0.344821	1.707507	17.411907	3.701006	3.697443	0.333012	2.47708	0.599955	0.915097	0.172393	0.162317	0.468559	0
4	30.873529	15.533645	5.505385	2.51544	3.28794	3.348339	3.747385	3.966309	1.20993	4.340769	14.857078	5.773794	4.196271	0.899832	3.121508	2.566443	0.362393	0.198012	0.23122	0.1	0
5	28.116139	7.928066	10.07677	3.133795	17.576883	6.000038	4.251744	3.56218	1.926057	0	14.802481	4.232103	6.54247	1.065138	2.649764	2.491498	1.607192	0.882696	2.313696	0.1	0
6	32.219172	32.498458	27.910613	2.108142	0.140095	25.665819	12.736608	21.243303	1.233012	11.075937	16.730909	15.602342	5.67589	2.541885	4.277742	8.862055	6.997813	3.544541	0.57	0	0
7	37.008172	19.69245	12.928879	14.316873	2.607591	8.614127	24.719863	6.081452	12.007565	0	18.997246	7.292466	9.241398	11.274147	10.813556	2.780473	2.194076	0.95	0.2985	0	0
8	30.453818	9.295287	6.217416	1.602922	0.626606	5.347945	5.061628	1.562457	1.072103	0.540938	14.733866	3.851745	3.709449	1.112584	0.891379	1.937276	0.877122	0	0	0	0
9	27.733729	6.543414	13.54461	1.614605	4.333387	4.31051	4.542066	5.0445	0.754623	0	15.535065	2.466689	13.250727	2.125805	2.777388	1.513817	2.542915	0	0.3	0.1	0
10	34.878789	17.611519	12.043693	2.303753	0.111606	2.006084	7.43636	4.026839	0.344821	1.324189	17.16787	3.712845	6.33988	1.196	3.75669	0.741605	1.961666	0	0	0	0
11	34.884062	20.558634	12.286449	0.498279	1.167896	4.259353	8.691858	5.596294	0.344622	1.343109	17.685016	0.409682	4.661793	0.333012	3.058674	2.97727	1.948662	0.089828	0.3	0	0
12	30.656353	9.271884	8.848778	3.051729	0.601763	6.905128	8.087271	0.282291	0.942842	0	15.343195	4.208232	4.051519	1.082423	3.748768	1.313729	0.369064	0.404094	0.285257	0.1	0
13	26.798283	9.972253	11.774326	6.726721	0.759302	2.0501	3.020267	4.448022	0	0.612496	15.914575	3.582003	7.405156	1.148328	3.727964	1.099228	2.243428	0.323248	0.3	0.1	0
14	34.847721	14.440732	13.191086	3.231649	0	3.17861	9.881766	3.556415	1.417783	3.165057	17.411308	3.127441	3.460034	1.852025	3.07209	2.620763	2.423433	0.10172	0.3	0.1	0
15	34.810065	15.001744	15.138227	2.566683	0.073734	9.958023	9.54957	7.064396	0.493547	0	17.249054	3.343533	1.950454	1.662071	3.153947	2.295185	1.798251	0.58207	0.11671	0.18905	0

Figure 4.19 – Q Learning test case 2, final Q-Table

As we see in Figure 4.18, 1758 out 4587 games were won.

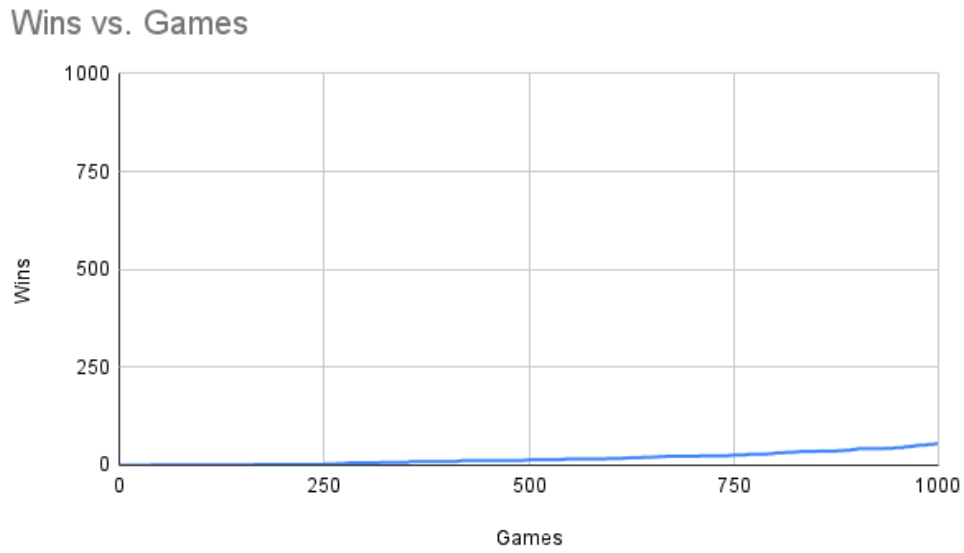

Figure 4.20 – Q Learning test case 2 Wins vs Games, 1000 games progress

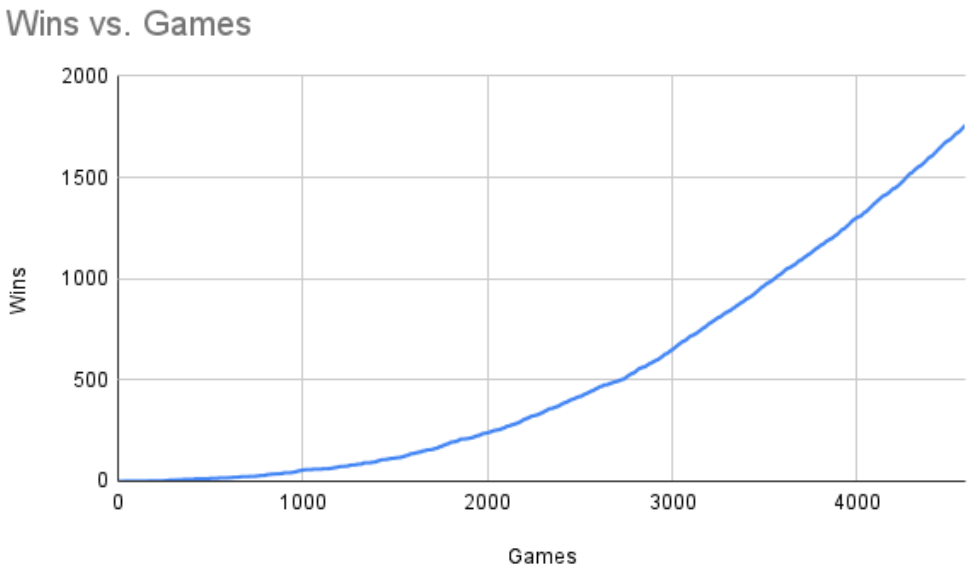

Figure 4.21 – Q-learning test case 2 wins vs games, all games progress

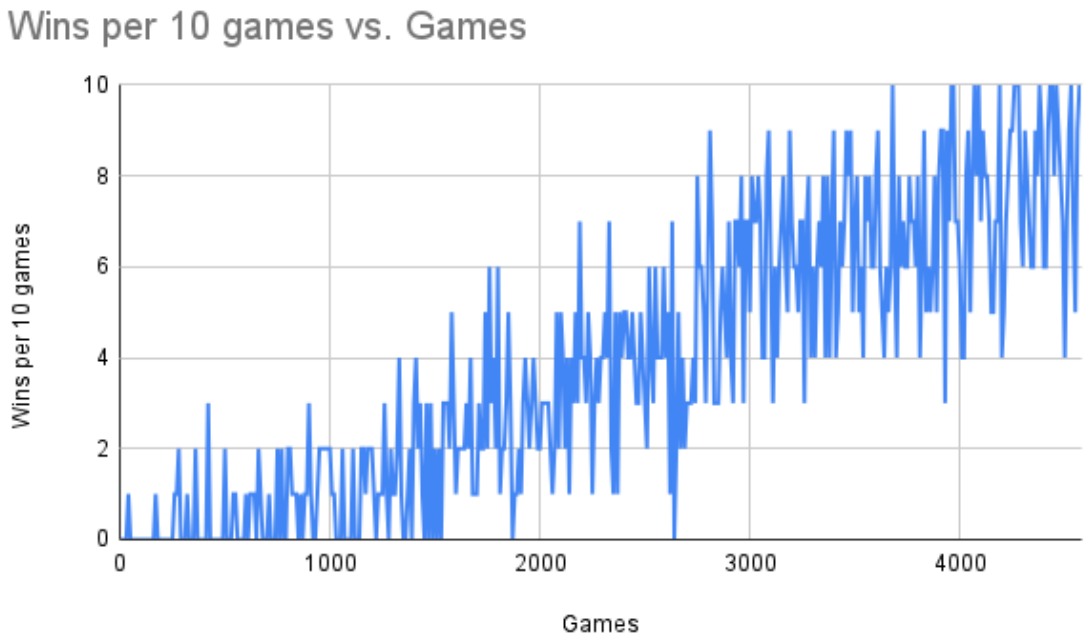

Figure 4.22 – Q learning test 2, wins per 10 games

4.1.4 DQL Test case 2

```
Suspicious timestamp for file:  C:\Users\macg\Desktop\another_simulation2\test
Suspicious timestamp for file:  C:\Users\macg\Desktop\another_simulation2\test
===Suspicious files:  ['C:\\Users\\macg\\Desktop\\another_simulation2\\test\\s
\Desktop\\another_simulation2\\test\\sample3.pdf'] ===
========================================
||      Game: 4188 Wins:  2565
========================================

C:\Users\macg\Desktop\another_simulation2\qlearning-ransim>
```

Figure 4.23 – DQL test case 2 command line results

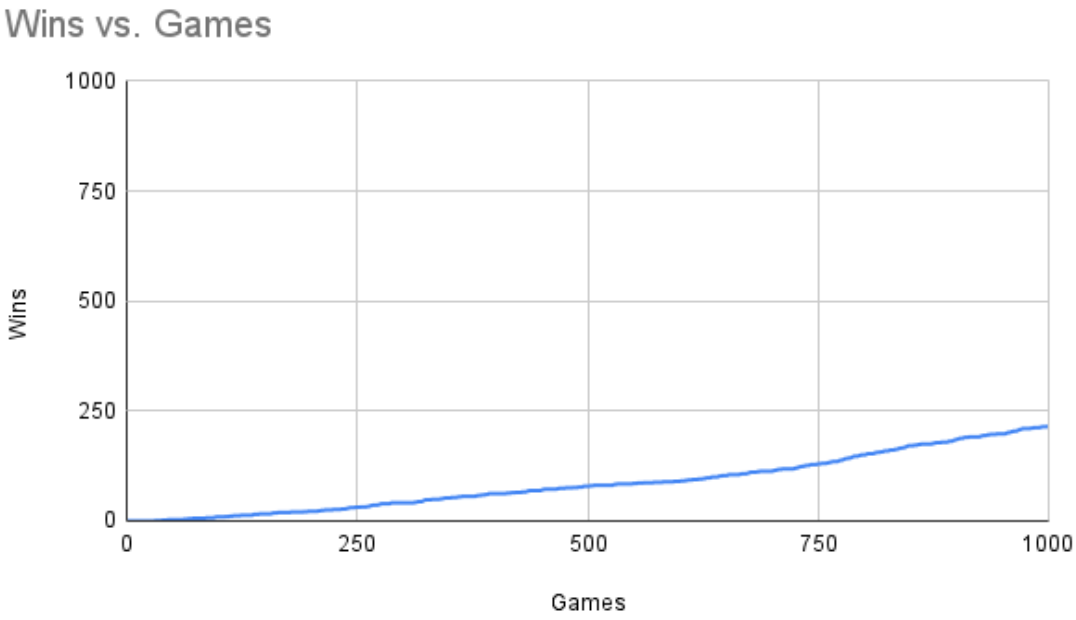

Figure 4.24 – DQL test case 2 wins vs games, 1000 games progress

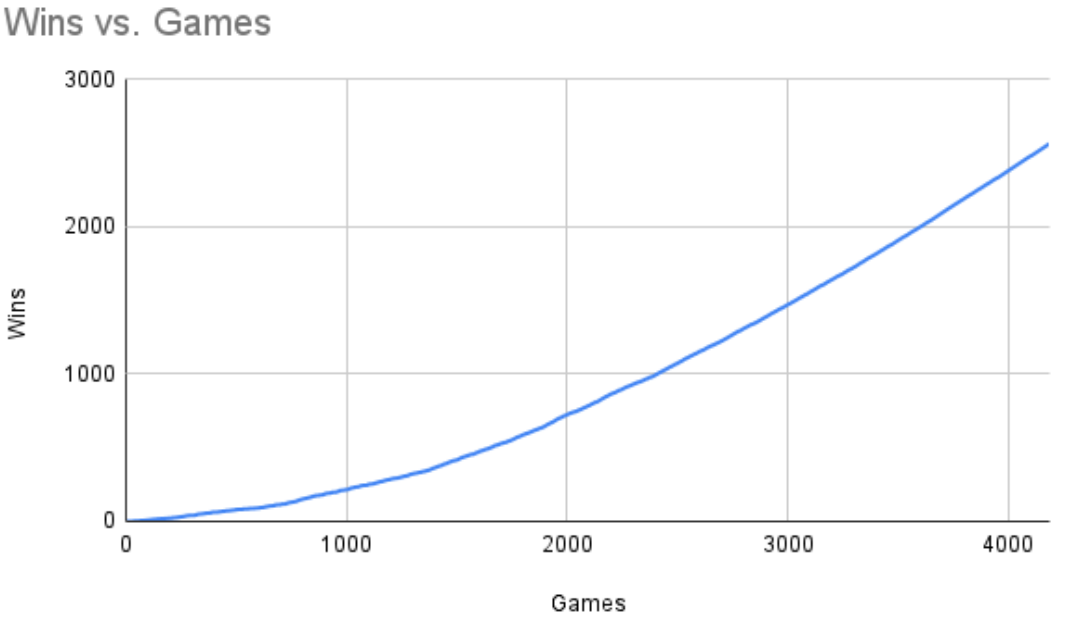

Figure 4.25 – DQL test case 2 wins vs games, all games progress

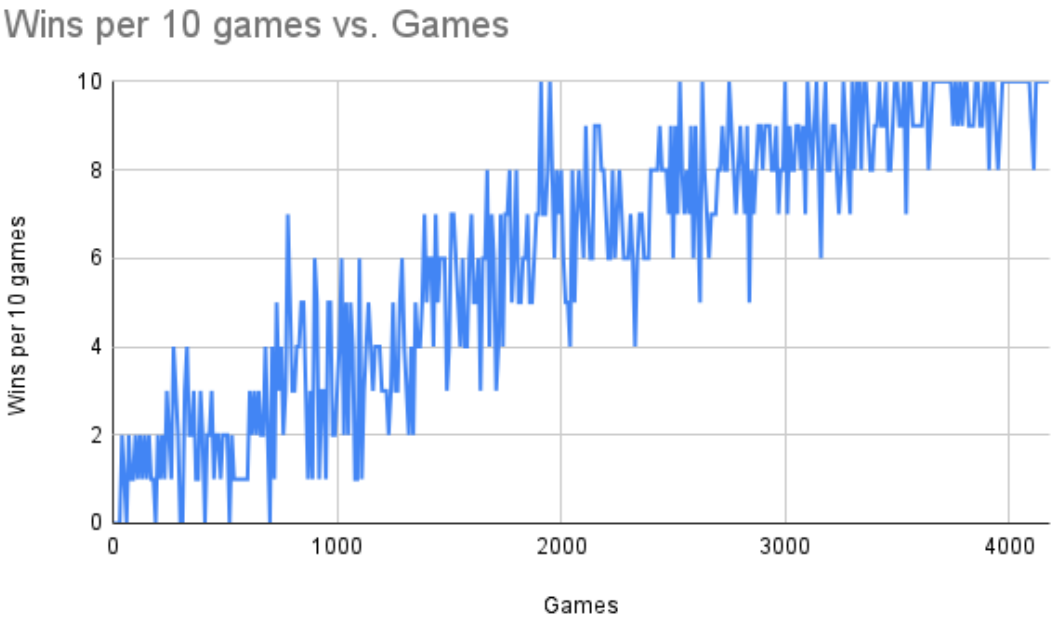

Figure 4.26 – DQL test case 2 wins per 10 games

5 ANALYSIS AND DISCUSSION

5.1 Research limitations

First, this reseach is very limited to the OS, to the few bypass tricks and few detecting methods. But this can be the beginning of more deep research into this. There are plently of actions that can be added, and plenty of defence tactics that can implemented as well. The goal is to study the possibility of training such model and bring good results and test different training algorithms, this makes it very easy for testing and experimenting. Later on, we can increase the action space, the number of states and the number of files to be encrypted, we can also make the ransomware detector stricter to the level of real world detectors.

5.2. Experiment analysis and discussion

5.2.1 Q Learning Test case 1

Looking at the Q-Table, we see that the model discovered the best policy in 2 steps, which is:

a) State 0: Action 14 with adding extension and base64 and encrypts 5 files;
b) State 5: Action 6 without adding extension and with encoding base64 and encrypts 5 files, now the game is won;

But to reach that policy 6522 games were played, in which there was 4560 wins as shown in figure 4.1. But as we look closer in the number of wins initially, we see that in the beginning the model was struggling to even win few games, in Figure 4.5, the progress on 200 games is low, we see that the model did not even reach 50 wins per 200.

But as the games progress, we see more and more wins, and see an exponential growth in the number of wins. This only means that the model is learning the best policy and adapting to it as we see in the Q-Table. We also see in Figure 4.4, the wins per 10 games is initially low, and as the model plays more, it learns to win more and therefore a highest win’s rate. In the end it reached 10/10 wins rate.

5.2.2 DQL Test case 1

In DQN, we see that the model also learns to beat the game, but it's different how it got there, first we see in Figure 4.10 that it played 6495 games and won 4621 games.

In DQN we only have a neural network, that takes State as input and predict the optimal Q-Value. This were DQN is superior to Q-Table. Since Q-Table is trying to find the optimal solution for that game, while DQN can be trained to make Q-Value prediction which makes it usable in real world.

While training we notice an unstability in wins per 10 games rate, in Figure 4.12, we see that as the model is getting better, it falls and won 0/10 games after 2400 iterations, but recovers quickly, but by time, we see that it wins 10/10 frequently and brings similar results as Q-Table.

5.2.3 Comparing DQL and QL test case 1 results

While looking at wins per 10 games for Q-Learning and DQN in Figures 4.4 and 4.12, we notice that Q-Learning is progressing systematically, while DQN is more chaotic, but both brings the best results possible. This is normal due to the nature of neural networks, it is powerful because it can actually do better job in much complicated simulations while Q-Learning would fail, since Q-Learning is trying to find the best options, while the neural network is trying to find something that just works. And Both learns through time.

With 10000 iterations, both played almost similar number of games, while DQN played slightly less games and won slightly more games in total. In 200, 1000 progress, we see that DQN performed slightly better than Q-Learning. DQN being slightly better is important, since this game is easy to learn, only 10 files, it's predicted that this DQN being slightly superior would make it much better if we conduct more complicated experiment.

5.2.4 Q Learning test case 2

As we see in the figure 4.19, the Q table is big, this is because of 20 states and 20 files to encrypt. We also see in figure 4.20 and figure 4.21, that the number of wins is growing slowly, at one time, the winning becomes more frequent, but we can say that it did not find the best policy yet for winning, we can also say that as the game became harder, Q Learning algorithm is less performant compared to the test case 2.

In figure 4.22, Q Learning is winning per 10 games we see that the model through the 10000 iterations is becoming better, but good enough, since it can't figure a policy that almost always winning.

5.2.5 Q Learning test case 2

In this experiment, the power of DQN is showing, as the game becomes harder, the neural network is doing such a good job to predict the write action to do in each state.

In Figure 4.24 and figure 4.25, see that the winning is very low compared to number of games played, but if we look at all games, we see that the wins are growing exponentially.

In figure 4.26 the winning per 10 games is growing slowly but, in the end, it figured out the best policy for winning. With stable improvments. We can see in the end that rate is between 9 and 10 which is very good results.

5.2.6 DQL vs QL in test case 2

In the first test case both QL and DQL have almost similar results, but as we made the game harder in test case 2, the difference is obvious, DQL performed much much better than QL.

The DQL figured how to win consistenly, but QL failed to do that. It also obvious in the q table in figure 4.19, its best strategy is to encrypt 10 files without extension and with base64 2 times, one after the other. This way will win games randomly, according to the

speed of encrypting, because the ransomware detector can detect files by modified time and with a threshold of 8, it will get caught most of time.

The Q-Table generally did a good job to find out that it's better to not use extension and to encode in base64, both bypasses detection by entropy and detection by double extension. But unfortunately, it fails to bypass the modified time defense. In the other hand, in DQL, we do not have a Q-Table, we have a trained neural network, that takes state as input and give 16 Q values for each action that is based on bellman equation and reward. DQL gives very good results even when the game becomes harder.

6 DISCUSSIONS OF EXPERIMENTS AND FINDINGS

6.1 Research question 1

Is it possible to tr ain a model that uses combination of common bypass techniques to detect strict anti ransomware rules?

Yes, it is possible to train the model to bypass the ransomware detector in the research scope. The fact that the models progress slowly and get better by time, only when the game is reasonable and can be won. If we create a game under conditions that are impossible to win, of course the model will always lose.

The actual challenge is when imitate the actual detection software with strict rules and bypass it, because if we can bypass the best tool, all other tools will be automatically bypassed.

6.2 Research question 2

How far can we go in using AI, especially RL that does not require any data, in cyber security?

It is obviously that very few research has been done, which means only that this is few steps toward using RL in security attacks, there is wide area in which RL can be applied, including penetration testing.

This research shows that it’s possible to train models in controlled envirement and reach the goal, without the need of data.

In this document we used RL to make the ransomware attack better, but we can also make the defence better using the same technique, and in this case the defender will be main player and if it defends successfully and stops the attack a reward will be given. But we need more research on that matter. In conclusion RL have huge potential in optimizing the security attacks and defense.

6.3 Research question 3

We know that DQL is potentially better than QL, but how much is the difference between the two algorithms in RL?

In the first test case of the simplified game, we saw that there is not much difference between DQL and QL, but as we made the game harder, The DQL performed lot better. So, to say in RL, it is better to use Q-Learning first, once everything works well, it highly recommended to switch to DQL.

7 CONCLUSIUONS AND FUTURE WORK

This thesis shows how RL can help to discover new attack strategies that can overcome behavior-based ransomware detector protection. This thesis also shows the potential of using DQN algorithm other than Q-Learning algorithm. It is worth noting that the experiments were conducted on a limited number of detection methods, and a limited number of bypass techniques. The presented results are very promising, especially when comparing Q-Learning with DQL, we see that DQL have very high potential that can be further applied to the anti-malware, anti-virus's products on behavior analysis.

RL can also be applied on network penetration testing, so the agent can be presented a large set of actions, and try to find the optimal attack path, or in case of DQN, to predict an attack path that works.

7.1 Future work

In the future work, it is suggested to increase the defense and increase action space to imitate what we have in real world. Other suggestions also, is to train the model in cloud where the model can train much faster, and to DQN instead of Q-Learning. Training the model in powerful cloud machines would be beneficial. DQN would work well if the states and actions are very high. The future works an also be conducted in many areas such as penetration test, bypass anti-malware defense.

8 BIBLIOGRAPHY

1. Reinforcement Learning for Anti-Ransomware Testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.nioguard.com/2020/10/reinforcement-learning-for-anti.html
2. Attention is All They Need: Combatting Social Media Information Operations With Neural Language Models [Електронний ресурс] – Режим доступу до ресурсу: https://www.fireeye.com/blog/threat-research/2019/11/combatting-social-media-information-operations-neural-language-models.html
3. Autonomous Penetration Testing using Reinforcement Learning by Jonathon Schwartz [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1905/1905.05965.pdf
4. Reinforcement Learning Tutorial Part 1: Q-Learning [Електронний ресурс] – Режим доступу до ресурсу: https://blog.valohai.com/reinforcement-learning-tutorial-part-1-q-learning
5. Reinforcement Learning Tutorial Part 2: Cloud Q-learning [Електронний ресурс] – Режим доступу до ресурсу: https://blog.valohai.com/reinforcement-learning-tutorial-cloud-q-learning
6. Reinforcement Learning Tutorial Part 3: Basic Deep Q-Learning [Електронний ресурс] – Режим доступу до ресурсу: https://blog.valohai.com/reinforcement-learning-tutorial-basic-deep-q-learning
7. Q Learning simple simulation code by valohai in GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/valohai/qlearning-simple
8. Named: Endpoint Threat Detection & Response [Електронний ресурс] – Режим доступу до ресурсу: https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/
9. WastedLocker’s techniques point to a familiar heritage [Електронний ресурс] – Режим доступу до ресурсу: https://news.sophos.com/en-us/2020/08/04/wastedlocker-techniques-point-to-a-familiar-heritage/

10. Ransomware Protection Test - April 2017 [Електронний ресурс] – Режим доступу до ресурсу: https://www.nioguard.com/2017/05/ransomware-protection-test-april-2017.html
11. Windows Virtual machine download [Електронний ресурс] – Режим доступу до ресурсу: https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
12. Reinforcement learning by Joseph M. Carew [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning
13. Reinforcement learning [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Reinforcement_learning
14. DEEP DIVE INTO CYBER REALITY Security Effectiveness Report 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.mandiant.com/sites/default/files/2021-09/rt-security-effectiveness-report-000287.pdf
15. Deep Q-Learning Tutorial: minDQN [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc