

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

**Deep Reinforcement Learning for
Artificial Intelligence-enabled Autonomous
Penetration Testing in Cyber Security**

by

Hoang Khuong Tran

THESIS SUBMITTED
IN FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

Supervisors: Prof. Chin-Teng Lin
Prof. Yu-Kai Wang

Sydney, Australia

December, 2022

Certificate of Authorship/Originality

I, Hoang Khuong Tran, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: December 30th, 2022 Place: Sydney, Australia

© Copyright 2022 [Hoang Khuong Tran]

ABSTRACT

Deep Reinforcement Learning for Artificial Intelligence-enabled Autonomous Penetration Testing in Cyber Security

by

Hoang Khuong Tran

Penetration Testing (PT) is the set of methods used to enhance the security of a networked system by exploiting potential vulnerabilities. It is the practice of simulating attacks on computer systems, networks, or web applications to test their security and identify vulnerabilities that an attacker could exploit. Penetration testers use various tools and techniques to probe the defenses of a system and uncover weaknesses. These methods require significant time and resources for training and execution. There is a shortage of skilled professionals to deal with the increasingly complex cyber security landscape. Conventional approaches in penetration testing require threat model of the target network to gain context to the exploits and vulnerabilities. These requirements present intractable challenges for penetration testing tools in dealing with rapidly changing network software and attack dimensions.

Artificial intelligence (AI) and reinforcement learning (RL) can potentially be used in penetration testing to automate certain tasks and improve the efficiency of the testing process, such as identifying targets, generating attack strategies, and adapting to changes in the target system. AI-enabled PT has been under research in recent years due to the insurgence of new deep learning advances in which RL is considered an appropriate learning framework to develop such applications thanks to its capability in learning a sequential decision-making process without any labelled dataset through interacting with the environment. For example, an AI-powered tool could analyse the network architecture and system configurations and suggest

potential attack vectors based on this information. An RL system could learn from previous testing experiences and use this knowledge to adapt to new situations and improve its performance over time.

However, there are multitudes of challenges in using RL to develop an automatic penetration testing application. The main challenges are the large and structured configuration of the state space and action space which are unconventional in typical deep RL works. Other challenges include the partial observability, the scarcity of rewards and the stochastic dynamics of the environment.

This research aims at understanding the technical challenges presented by an autonomous penetration testing application and developing novel Deep Reinforcement Learning (DRL) frameworks to deal with two problems of scalable autonomous PT, which involve the complexity of the *action space* and the *state space*. By leveraging the recent advances in Multi-Agent Reinforcement Learning (MARL) paradigm, we re-formulate the conventional approach of using a single-agent DRL into a multi-agent learning framework, enabling the decomposition of the complex and structured *action space* into manageable sub-modules each of which is controlled by a DRL agent. The agents are trained cooperatively to develop PT policies under two different representations of the *action space*: a large and discrete action space and a multinomial parameterised action space. We introduced two new frameworks called *Cascaded reinforcement learning agents for large discrete action spaces* and *Multi-agent reinforcement learning for parameterised action spaces* for each of the aforementioned *action space* representations.

The complexity of the *state space* representation in autonomous PT consisting of the non-visual and binary-valued description of the cyber networks, the highly stochastic state transition probability coupled with the sparsity of the reward signals makes it challenging for DRL algorithms to learn in such application. We adapted Hierarchical Reinforcement Learning (HRL), a multi-layer learning approach wherein the high-level layer is trained to assign different sub-goals to the lower level, which in turn learns a primitive policy to achieve the given subgoals. This integration of HRL into the MARL training is innovative and can be devel-

oped into a more general framework for handling complex problem space in different domains. The subgoal learning is facilitated by using the Successor Representation (SR) as it enables the learning of a state abstraction under environment with sparse or no reward. All the proposed approaches can be integrated end-to-end to develop an AI-enabled autonomous penetration testing application.

Dissertation directed by Distinguished Professor Chin-Teng Lin
Australian Artificial Intelligence Institute (AAIL)
School of Computer Science
University of Technology Sydney

To my loved ones

Acknowledgments

The 4-year PhD research is one of the toughest but worthwhile experiences in my life. It is a testimony to my perseverance, intellectual ability, and hard work. The Covid-19 pandemic has put a lot of constraints both mentally and physically to my research as well as my well-being. I experienced the passing of my grandpa and the birth of my first child during my PhD. I could not have done this without the unconditional support of my loved ones. There is no word that I can use to express my gratitude to my parents and my wife who have been standing with me throughout the journey.

I would like to sincerely thank Professor Chin-Teng Lin for giving me the opportunity to join the Brain Computer Interface lab at UTS to conduct my PhD research, and for continually supporting me in both work and personal life. I would also like to express my appreciation to Dr. Yu-Kai Wang for being my co-supervisor and his constant encouragement whenever we meet.

Last but not least, I extend my gratitude towards Dr. Junae Kim, Dr. Toby Richer and Max Standen from the Defence Science and Technology Group, Australia for supporting this work by having regular discussions with us and giving valuable feedbacks on the development of the involved researches.

This work was supported by the Australian Defence Science Technology Group (DSTG) under Agreement No: MyIP 10699.

Hoang Khuong Tran
Sydney, Australia, 2022.

List of Publications

Conference

- C-1. **Tran, Khuong** and Akella, Ashlesha and Standen, Maxwell and Kim, Junae and Bowman, David and Richer, Toby and Lin, Chin-Teng, “Deep hierarchical reinforcement agents for automated penetration testing” *Proceedings of the 1st International Workshop on Adaptive Cyber Defense, IJCAI 2021*

Journal

- J-1. **Tran, Khuong** and Standen, Maxwell and Kim, Junae and Bowman, David and Richer, Toby and Lin, Chin-Teng, “Cascaded reinforcement learning agents for large discrete action space” *Special Issue Machine Learning for Cybersecurity Threats, Challenges, and Opportunities II*
- J-2. **Tran, Khuong** and Standen, Maxwell and Kim, Junae and Bowman, David and Richer, Toby and Lin, Chin-Teng, “A Multi-Agent Reinforcement Learning Approach for Multinomial Parameterised Action Space” *IEEE Transactions on Information Forensics and Security* (Under Review)
- J-3. **Tran, Khuong**; Lin, Chin-Teng, “Subgoals Discovery Using Deep Successor Feature Representation in Autonomous Penetration Testing” *IEEE Transactions on Systems, Man, and Cybernetics* (In progress)

Contents

Certificate	ii
Abstract	iii
Dedication	vi
Acknowledgments	vii
List of Publications	ix
List of Figures	xv
List of Tables	xxi
Abbreviation	xxii
1 Introduction	1
1.1 Motivation	1
1.2 Scope	4
1.3 Thesis overview	7
2 Background and Literature Review	10
2.1 Penetration Testing	10
2.1.1 Introduction	10
2.1.2 Threat Models in Penetration Testing	12

2.1.3	Automated Penetration Testing Tools	13
2.1.4	CybORG Overview	14
2.2	Reinforcement Learning	16
2.2.1	Formulation	17
2.2.2	Reinforcement Learning Algorithms	22
2.2.3	Hierarchical Reinforcement Learning	27
2.2.4	Multi-Agent Reinforcement Learning	29
2.2.5	Hierarchical Multi-Agent Reinforcement Learning	32
2.3	Deep neural networks	33
2.3.1	Multi-layer perceptron	33
2.3.2	Autoencoders	33
2.4	Literature Review	34
2.4.1	Reinforcement learning in penetration testing	34
2.4.2	Deep reinforcement learning with large action space	35
2.4.3	Deep reinforcement learning with parameterised action space	39
2.4.4	State representation learning in deep reinforcement learning	42
3	Cascaded reinforcement learning agents for action space decomposition	49
3.1	Introduction	49
3.2	Background	51
3.3	Methodology	54
3.3.1	Algebraic Action Decomposition Scheme	54
3.3.2	Cooperative Multi-Agent Training	59
3.3.3	CRLA Implementation	61

3.3.4	Algorithms Pseudocode	62
3.4	Experiments	64
3.4.1	Toy Maze scenario	64
3.4.2	The CybORG Simulator	65
3.4.3	Neural Network Architecture	67
3.5	Results	67
3.5.1	Maze	67
3.5.2	CybORG	69
3.5.3	Cooperative Learning with QMIX	71
3.5.4	Discussion	72
3.6	Chapter summary	73
4	A multi-agent reinforcement learning approach to multi-nomial parameterised action space in autonomous penetration testing	75
4.1	Introduction	76
4.2	Background	80
4.3	Methodology	83
4.3.1	Multi-agent learning for parameterised action space	83
4.3.2	Cooperative learning	87
4.3.3	Auxiliary State Representation Training	88
4.3.4	Invalid Action Masking	89
4.4	Experiments	89
4.4.1	CybORG	89
4.4.2	Neural network architecture	93

4.5	Results	93
4.6	Chapter summary	95
5	State representation for effective learning in sparse re-ward environment	98
5.1	Introduction	98
5.2	Problem formulation	102
5.3	Methodology	104
5.3.1	State representation learning using the auto-encoder architecture	104
5.3.2	State representation learning with successor representation . .	107
5.3.3	Successor feature with hierarchical reinforcement learning . . .	112
5.3.4	Algorithm pseudocodes	112
5.4	Experiments	113
5.4.1	CybORG 2021	113
5.4.2	Neural network architecture	114
5.5	Results	116
5.6	Chapter summary	121
6	Conclusion	123
6.1	Overview	123
6.2	Summary of contributions	124
6.2.1	Cascaded Reinforcement Learning Agents for large discrete action space	124
6.2.2	Multi-Agent for Parameterised Action in CybORG	125

6.2.3	State representation for effective learning in complex environment with sparse reward	126
6.3	Limitations and Future works	126
6.3.1	Variable state space and action space	127
6.3.2	Adversarial cyber operations	127
6.3.3	Human-machine collaboration	128
A	Appendix	129
A.1	Appendix: CRLA Network Comparison	129

List of Figures

1.1	Research objectives	8
2.1	A general penetration testing process (Sarraute 2013).	11
2.2	CybORG diagram	15
2.3	Agent - Environment Interaction	17
2.4	A POMDP agent with a state estimator (SE) to assist itself in interpret the world state from the observation.	20
2.5	Reinforcement Learning methods (Silver 2015).	23
2.6	DQN algorithm	26
2.7	Feudal Architecture with spatial abstraction	27
2.8	A screen of the Atari game called Montezuma’s Revenge. The agent has to climb different ladders, avoid the ghosts and get to the key in order to transition to a new screen. The agent has to learn this entire sequence of actions without any reward signal.	28
2.9	HRL with temporal abstraction sub-goals	29
2.10	Feudal structure in MARL	32

2.11	The general architecture of an autoencoder. The input x is mapped into a latent representation h via the encoding function f . The decoding function g reconstructs h back to the original input r	34
2.12	An illustration of a typical cyber security network used in pen-testing simulators (Zhou et al. 2021).	36
2.13	The P-DQN algorithm by Xiong et al. (2018)	42
2.14	Parameterised multinomial action space	42
2.15	Zoom-in version of the hierarchical meta-controller/controller framework (Rafati and Noelle 2019a)	46
3.1	Action space composition. The final action space \mathcal{U} is constructed by chaining multiple action component's subspaces \mathcal{A}^i	54
3.2	An illustration of a tree-based structure for hierarchical action selection. The primitive action identifiers are located on the leaf nodes. Each internal node contains the action range of its children.	58
3.3	The operational diagram of the proposed CRLA architecture. DRL agents are grouped into a cascaded structure of L levels. The state-action values of all agents are fed into a MixingNet implemented as a hyper-network (at bottom of the figure) to optimise the agents altogether. The paths of the gradient are shown as red arrows.	59
3.4	Parallel training and execution. All agents share the replay buffer from which experiences can be sampled in batches and used for training in parallel.	62
3.5	Two simulated scenarios. (a) A toy-maze-based scenario. (b) A CybORG scenario of 24 hosts.	65

3.6	Results of CRLA and single-agent DDQN on a maze scenario with 4096 actions. Left panel: the cumulative scores throughout the training. Right panel: the required number of steps to capture the flag throughout the training.	69
3.7	Results of CRLA and single DDQN on different CybORG scenarios. Sub-figures from left to right, top to bottom: (a) 50-hosts scenario, (b) 60-hosts scenario, (c) 70-hosts scenario, (d) 100-hosts scenario. . .	70
3.8	Performances of CRLA with and without QMIX on the 60-hosts scenario.	72
3.9	State representation from the 3 agents of the 50-hosts scenario. The clusters are coloured based on the action components.	73
4.1	CybORG action components.	77
4.2	An example of a 15-hosts scenario in CybORG	78
4.3	Overview of the MAPA architecture. DQN1 is used to select the attack type. The state-action value of the chosen attack action is passed to DQN2 for parameters' selection.	85
4.4	A detailed view of the second DQN agent. Sub-networks are used for different parameters. Invalid masks are applied to the outputs of the sub-networks to mask out invalid parameters.	86
4.5	Dueling deep q-network architecture.	90
4.6	Invalid action mask.	90
4.7	A 1-dimensional tensor describing the observed state of the network. . .	91
4.8	An adaptation of the action branching architecture used as the baseline method.	92

4.9	Performance comparison between MAPA and the baseline with action branching. Left panel: the accumulative score during training. Right panel: the number of steps to capture the flag during training.	96
4.10	Performance with the 12, 15 and 18-hosts scenarios. Left panel: the accumulative score during training. Right panel: the number of steps to capture the flag during training.	96
4.11	Performance with the 50, 60 and 70-hosts scenarios. Left panel: the accumulative score during training. Right panel: the number of steps to capture the flag during training.	97
4.12	Performance with the 80, 90 and 100-hosts scenarios. Left panel: the accumulative score during training. Right panel: the number of steps to capture the flag during training.	97
5.1	The agent, represented by the humanoid figure, and the star are at spatially closed states but temporally distant.	100
5.2	Hierarchical reinforcement learning framework. The meta-controller generates a goal g_t every N^{th} time steps (e.g., $\mathbf{T} = N$ and $N > 1$), during which the controller executes a sequence of primitive actions $a_t \dots a_{t+N}$ to achieve the assigned subgoal. The learning of the controller is guided by the intrinsic reward while the meta-controller is trained by the extrinsic reward(Kulkarni et al. 2016).	101
5.3	A CybORG scenario configuration with 50 hosts organised into 10 subnets.	103
5.4	A <i>medium</i> CybORG scenario configuration with 18 hosts organised into 6 subnets.	103
5.5	A <i>hard</i> CybORG scenario configuration with 18 hosts organised into 6 subnets.	104

5.6	An Auto-encoder for reduced state representation.	105
5.7	A variational auto-encoder for learning state distribution.	105
5.8	Multi-Agent for Parameterised Action or MAPA from Chapter 4. . .	106
5.9	The auto-encoder training conducted on 12-hosts scenario. From left to right, top to bottom: cumulative episodic returns, epsilon rate for exploration, mean of the gradient norm, variance of the gradient norm, mean of the loss function, variance of the loss function, mean of the estimated state-action values, variance of the estimated state-action values, the number of steps to capture the flag.	107
5.10	Performance comparison between with and without auto-encoder. The green curves are the MAPA performance with auto-encoder while the gray curves show the MAPA performance without auto-encoder	108
5.11	Successor feature with MAPA architecture	111
5.12	Hierarchical reinforcement learning with successor feature. Samples of exploratory trajectories are used to train a good SF representation. They are then clustered into k groups. The centroids of these clustered are treated as sub-goals in HRL. The meta-controller learns to assign a sequence of subgoals to the controller for execution.	113
5.13	CybORG scenario (Standen et al. 2021)	114
5.14	An example of 15-hosts scenario	115
5.15	Deep auto-encoder training losses on 15-hosts scenario. Each column contains the loss metrics for each of the 4 agents. Top row shows the average or mean of the loss while the variances are shown in the second row.	116

5.16	Successor feature training losses on 15-hosts scenario. Each column contains the loss metrics for each of the representative 2 agents. Top row shows the average or mean of the loss while the loss variances are shown in second row.	117
5.17	The heat map of the SF matrix for a sampled random trajectory. Small values represent similar SF representations while larger values represent variant degree of dissimilarity between any SF pair of states.	118
5.18	CybORG observation transition graph	119
5.19	Cumulative rewards on 15 hosts scenario. The orange curve displays the performance of the HRL approach and while the gray curve is the performance of the original MAPA without HRL.	120
5.20	Steps to finish on 15 hosts scenario. The orange curve displays the performance of the HRL approach and while the gray curve is the performance of the original MAPA without HRL.	120
5.21	Cumulative returns (left) and number of steps to capture the flag (right) on the <i>hard</i> -level 18 hosts scenario using the MAPA with HRL approach.	121
5.22	The SF training losses of the 4 agents using the MAPA with HRL approach.	121
5.23	Cumulative rewards during training for scenarios with 50, 60 and 70 hosts.	122
A.1	Neural network architectures comparison between the single-agent duelling deep Q-learning and the proposed cascaded reinforcement learning agents approach.	130

List of Tables

3.1	Hyper-parameter settings	68
3.2	Configurations of tested scenarios.	71
4.1	The action components and their possible values in the 15-hosts scenario in CybORG	78
4.2	Notation table	82
4.3	Hyper-parameter settings	94
5.1	Hyper-parameter settings	115

Abbreviation

RL - Reinforcement Learning

MARL - Multi-agent Reinforcement Learning

DRL - Deep Reinforcement Learning

MLP - Multi-layer perceptron

CybORG - Cyber Operation Reseach Gym

RNN - Recurrent Neural Network

GRU - Gated Recurrent Unit

DQN - Deep Q-Network

HRL - Hierarchical Reinforcement Learning

MDP - Markov Decision Process

SMDP - Semi-Markov Decision Process

PAMDP - Parameterised Action Markov Decision Process

POMDP - Partially Observable Markov Decision Process

CTF - Capture The Flag

SR - Successor Representation

SF - Successor Feature

DSR - Deep Successor Representation

ICT - Information Communication Technology