MODELING ADVANCED PERSISTENT THREATS USING STACKELBERG GAME THEORY AND REINFORCEMENT LEARNING UNDER PARTIAL OBSERVABILITY

BY

FATIMA MUHAMMAD ANIS

A Thesis Presented to the

COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

MAY 2025

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **FATIMA MUHAMMAD ANIS** under the direction of her thesis adviser and approved by her thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Mohammad Hammoudeh (Adviser)

Dr. Md Mahfuzur Rahman (Member)

Dr. Waleed Al-Gobi (Member)

Dr. Motaz Abdulrahman Alfarraj
Department Chairman

Dr. Sulıman Al-Homidan
Dean of Graduate Studies

Date

©Fatima Muhammad Anis
2025

To my parents, husband, and daughter, whose love and support made this journey possible.

ACKNOWLEDGMENTS

All praise and thanks are due to Allah for granting me the strength, guidance, and opportunity to complete this thesis.

I am sincerely grateful to my advisor, Dr. Mohammad Hammoudeh, for his valuable guidance, steady encouragement, and kind support. His patience and belief in me, especially during challenging times, made this work possible. I would also like to express my appreciation to my committee members, Dr. Waleed Al-Gobi and Dr. Md Mahfuzur Rahman, for their assistance and understanding during this process. I extend my gratitude to the university for providing this opportunity and to Dr. Sajjad Mahmood for his consistent help and willingness to assist whenever needed.

I extend heartfelt thanks to my family. My husband, Taha Abbasi, has been my greatest motivator and source of strength. I am especially thankful to my mother, whose help with my daughter's care allowed me to focus on my work. My daughter, Leena, though very young, showed remarkable patience for her age. I also thank my father, Muhammad Anis, for his prayers, and my sisters, Amna Anis and Hafsa Anis, who have been a constant presence throughout this journey. I am also grateful to my father-in-law, Muhammad Ijaz Abbasi, my mother-in-law, and my sister-in-law, Hira Abbasi, for their encouragement and consideration.

Lastly, I would like to thank the friends who stood by me in every possible way,

emotionally, academically, and through their ongoing support: Maryam Abdullatif, Nora Alowayed, Alanoud Alowayed, Alanood Alsubaie, Afrah Shahid, Munazzah Izzat, and Dorrieh Alomari. Each of you played an important role, and I am truly thankful for your presence and care during this time.

TABLE OF CONTENTS

ACKNOWLEDGMENTS **v**

LIST OF TABLES **xi**

LIST OF FIGURES **xii**

ABSTRACT (ENGLISH) **xvi**

ABSTRACT (ARABIC) **xviii**

CHAPTER 1 INTRODUCTION **1**

1.1 Motivation 4

1.2 Problem Statement 6

1.3 Objectives 7

1.4 Research Questions 8

1.5 Research Design 9

1.6 Thesis Structure 11

CHAPTER 2 BACKGROUND **13**

2.1 Advanced Persistent Threats (APTs) 13

2.2 Attack Graphs and CyberSpace Modeling 16

2.3 Stackelberg Security Games (SSGs) 18

2.3.1 Model Variations 20

2.4 Partially Observable Stochastic Games (POSGs) 22

2.5 Reinforcement Learning and Proximal Policy Optimization 24

2.5.1 PPO 25

CHAPTER 3 LITERATURE REVIEW 27

3.1 Game-Theoretic Models in Cybersecurity 27

3.1.1 Classical and Zero-Sum Game-Theoretic Models 28

3.1.2 Stackelberg and Leader–Follower Models for Strategic Defense . 31

3.1.3 Stochastic, POSG, and Learning-Based Formulations 33

3.2 RL for Strategic Cyber Operations 35

3.2.1 Single-Agent RL for Offensive Cyber Operations 36

3.2.2 Self-Play and Game-Theoretic RL for Strategic Interaction . . . 39

3.2.3 Multi-Agent and Hierarchical RL for Coordinated Cyber Defense 41

3.3 Gap Analysis 42

CHAPTER 4 GAME-THEORETIC FRAMEWORK 46

4.1 Stackelberg Game Formulation and Equilibrium 50

4.2 State Space Representation and Attack Graph Integration 52

4.3 Attacker and Defender Action Spaces 54

4.3.1 Attacker Action Space (A_A) 55

4.3.2 Defender Action Space (A_D) 57

4.4 State Transition Dynamics 61

4.5 Observation Model and Belief Updates under Partial Observability . . 64

4.5.1 Attacker's Observation O_A 65

4.5.2 Defender's Observation O_D 66

4.5.3 Belief Update via Bayesian Inference 68

4.6 Reward and Cost Structure 70

4.6.1 Attacker's Reward R_A 70

4.6.2 Defender's Reward R_D 73

4.7 RL Approach 76

4.7.1 Training Procedure with PPO 78

4.7.2 PPO Hyperparameter Configuration 80

4.7.3 Exploration Strategy 82

4.8 Training Procedure and Pseudocode 83
4.9 Discussion: Complexity, Assumptions, and Scalability 85
4.9.1 State Space Complexity 86
4.9.2 Action Space Complexity 87
4.9.3 Computation and Convergence 88
4.9.4 Modeling Assumptions 88
4.9.5 Scalability and Hierarchical Modeling 90
4.9.6 Equilibrium Analysis 91

CHAPTER 5 EXPERIMENTAL SETUP & RESULTS ANALYSIS 93

5.1 Environment Setup and Network Topology 93
5.2 Experiment Design and Methodology 96
5.3 Results and Analysis 101
5.3.1 Core Learning Dynamics 101
5.3.2 Equilibrium Behavior and Strategic Insights 105
5.3.3 Defender Strategies Comparison 108
5.3.4 Attacker Progression and Behavior Analysis 113
5.3.5 Detection and Observability Analysis 118
5.3.6 Ablation Study: Early Versus Late Training Behavior 122

CHAPTER 6 LIMITATIONS AND FUTURE DIRECTIONS 126

6.1 Limitations of the Study 126
6.1.1 Simulated Environment Accuracy 127
6.1.2 Network Size and Topology 127
6.1.3 Defender Strategy Constraints 128
6.1.4 Detection Model Simplifications 128
6.1.5 Reward Function and Objectives 129
6.1.6 Single-Stage Attack Focus 129
6.1.7 Generality and Scalability of the RL Approach 130
6.2 Future Directions 130
6.2.1 Adversarial Training and Game-Theoretic Approaches 131

6.2.2 Scaling to Larger and Richer Environments 131
6.2.3 Enhanced Attack Models 132
6.2.4 Improved Defense Mechanisms 132
6.2.5 Uncertainty and Partial Observability 133
6.2.6 Integration of Learning with Domain Knowledge 133
6.2.7 Autonomous Red Teaming Tools 133
6.2.8 Interactive Learning and Explainability 134

CHAPTER 7 CONCLUSION **135**

APPENDIX A GLOSSARY OF NOTATION **138**

A.1 Notation Table 138

REFERENCES **141**

VITAE **151**

LIST OF TABLES

Table 1.1 Framework (Chapter 4) vs. Prototype Implementation (Chapter 5) 10

Table 3.1 Summary of Game-Theoretic Models in Cybersecurity 29
Table 3.2 Summary of selected RL approaches in cybersecurity 37
Table 3.3 Comprehensive Gap Analysis with Explicit Thesis Contributions 43

Table 5.1 Reward structure used during PPO training. 97
Table 5.2 Key performance metrics recorded during training evaluations. . 98
Table 5.3 Summary of experimental environment and training setup. . . . 100
Table 5.4 Best recorded success rates and corresponding rewards during training. 103
Table 5.5 Correlation between major metrics during training. 105
Table 5.6 Defender patching metrics at final evaluation stage. 112
Table 5.7 Summary table averaged over the last 5 evaluation intervals for both defender strategies. 119
Table 5.8 Ablation study results comparing first and last 100k training steps for both defender strategies. 123

Table A.1 Notation used in the Stackelberg game-theoretic APT simulation framework. 138

LIST OF FIGURES

Figure 1.1 Timeline of Major APT Attacks (2020-2025) 5

Figure 4.1 High-level architecture of the AI-driven APT modeled framework. 50
Figure 4.2 Defender patching strategies. 60
Figure 4.3 PPO training loop. 80
Figure 4.4 Stackelberg-POSG attacker–defender interaction. 92

Figure 5.1 Cyber attack network topology with base probability of a successful compromise for each node. 94
Figure 5.2 Attacker performance during 500,000 training steps under both defender strategies. 101
Figure 5.3 Attacker failure rates during training under each defender strategy. 104
Figure 5.4 Average episode length during training under random and neighbor patching defenders. 105
Figure 5.5 Episode reward versus episode length during training 108
Figure 5.6 Attacker success rate versus detection rate under each defense strategy. 111
Figure 5.7 Inverse relationship between attacker success and timeout rate for both defender strategies. 111
Figure 5.8 Attacker performance over training: average new compromises and successful moves per episode under each defender strategy. 114
Figure 5.9 Time distribution of reaching the target and detection events across defender strategies. 116

Figure 5.10 Average number of patches applied per episode during the final 50,000 training steps. 118
Figure 5.11 Early versus late training performance under each defender strategy. 122
Figure 5.12 Combined performance changes from early to late training for both defender strategies. 124

THESIS ABSTRACT

NAME: Fatima Muhammad Anis

TITLE OF STUDY: Modeling Advanced Persistent Threats Using Stackelberg Game Theory and Reinforcement Learning under Partial Observability

MAJOR FIELD: Computer Science

DATE OF DEGREE: May 2025

Advanced Persistent Threats (APTs) are stealthy, long-term cyberattacks that target critical systems using a series of carefully planned steps. This thesis presents a framework that models APT behavior through a combination of Stackelberg game theory and Reinforcement Learning (RL) under partial observability. The attacker is treated as the leader who selects multi-stage attack paths, while the defender is the follower who responds with limited information about the attacker's actions. The model is designed as a partially observable stochastic game, where uncertainty is introduced through noisy intrusion detection alerts and belief updates based on Bayesian inference. An attack graph, built using the Attack Dynamics framework and mapped

to MITRE ATT&CK tactics, defines how the attacker can move through the network. Both attacker and defender agents are trained using the Proximal Policy Optimization (PPO) algorithm, with the attacker learning to evade detection and reach high-value targets, while the defender learns to patch vulnerabilities and respond to intrusions using incomplete observations.

To validate the framework, a lightweight prototype was implemented on a simulated 10-node network. The attacker was evaluated against two baseline defense strategies: random patching and neighbor-aware patching. The results show that the PPO-trained attacker was able to learn effective attack strategies, especially under the weaker random defense, achieving about an 80% success rate in compromising the network while remaining undetected. The neighbor-aware strategy, however, slowed down the attacker and increased detection rates. Performance metrics such as cumulative reward, detection frequency, episode length, and success-to-failure ratio highlight the strong influence of the defender's visibility and response strategy on the attacker's outcomes. This work contributes both a scalable simulation environment for modeling intelligent cyber threats and valuable insights into how APTs evolve against different defensive behaviors, supporting the development of smarter and more adaptive cybersecurity defenses.

ملخص الرسالة

الاسم: فاطمة محمد أنيس

عنوان الدراسة: نمذجة التهديدات المستمرة المتقدمة (Advanced Persistent Threats - APTs) باستخدام نظرية لعبة ستاكلبيرج والتعلم المعزز ضمن بيئة الملاحظة الجزئية

التخصص: علوم الحاسب

تاريخ الدرجة العلمية: مايو ٢٠٢٥

التهديدات المستمرة المتقدمة (Advanced Persistent Threats - APTs) هي هجمات إلكترونية متطورة تتميز بالتخفي وطول الأمد، وتستهدف الأنظمة الحساسة عبر سلسلة خطوات مخطط لها بعناية. تقدم هذه الأطروحة إطارًا جديدًا لنمذجة سلوك هذه التهديدات بدمج نظرية لعبة ستاكلبيرج والتعلم المعزز (Reinforcement Learning - RL) ضمن بيئة تتسم بالملاحظة الجزئية. يُعامَل المهاجم كقائد يختار مسارات متعددة المراحل للهجوم، في حين يكون دور المدافع تابعًا يستجيب بناءً على معلومات جزئية حول تحركات المهاجم. تم تصميم النموذج كلعبة عشوائية جزئية الملاحظة، حيث يُعبَّر عن عدم اليقين عبر تنبيهات كشف التسلل المشوشة وتحديث الاعتقادات بالاستدلال البايزي. تُحدد مسارات المهاجم عبر الشبكة من خلال المخطط البياني للهجمات، المستند إلى إطار ديناميكيات الهجوم (Attack Dynamics) والمتوافق مع تكتيكات MITRE ATT&CK. يُدرَّب كل من وكيلي المهاجم والمدافع باستخدام خوارزمية تحسين السياسة المُقارب (Proximal Policy Optimization - PPO)، حيث يتعلم المهاجم الوصول إلى أهداف عالية القيمة مع تجنب الكشف، ويتعلم المدافع معالجة الثغرات الأمنية والاستجابة للهجمات بناءً على ملاحظات جزئية.

للتحقق من صحة الإطار المقترح، طُوِّر نموذج أولي مبسط على شبكة حاسوبية محاكاة مكونة من عشر عقد. قُيِّم أداء المهاجم مقابل

استراتيجيتين دفاعيتين: السد العشوائي للثغرات وسد الثغرات المدرك للجيران. أظهرت النتائج قدرة المهاجم المُدرَّب على تعلّم استراتيجيات فعالة، وحقق نسبة نجاح تجاوزت 80% مع المحافظة على التخفي ضد استراتيجية الدفاع العشوائي. في المقابل، أدت استراتيجية سد الثغرات المدرك للجيران إلى إبطاء المهاجم وزيادة معدلات الكشف. يُسهم هذا العمل في توفير بيئة محاكاة قابلة للتوسع لتقييم التهديدات السيبرانية الذكية وتعزيز قدرات الدفاعات الأمنية التكيفية.

CHAPTER 1

INTRODUCTION

In the modern digital landscape, cyber adversaries have developed the ability to bypass traditional security mechanisms by adopting long-term, strategically driven attack methodologies. Among these, Advanced Persistent Threats (APTs) constitute a particularly dangerous class of intruders. Rather than launching short-term or opportunistic attacks, APT actors design sustained, multistage campaigns aimed at infiltrating networks, remaining hidden, and achieving high-value objectives. These campaigns typically involve systematic privilege escalation, lateral movement across networked systems, and the exfiltration of sensitive information, all while avoiding detection by conventional monitoring tools [1].

APT operations are not just technically complex. Their sophistication also comes from the decision-making strategies of the attackers, who adapt based on feedback from the environment. In most real-world scenarios, it is the attacker who controls the pace of engagement. Defenders often lack complete situational awareness and must react to signs of intrusion that may be delayed, subtle, or misleading. This dynamic calls for a modeling approach that places the attacker at the center of the strategic

interaction [1].

Classical security models, including traditional Stackelberg Security Games (SSGs), often assume that defenders commit to their strategies first, while attackers respond accordingly. Such assumptions do not reflect the reality of APT campaigns, where the attacker typically initiates the intrusion and continuously adapts while progressing through the network [2]. Therefore, this thesis reinterprets the Stackelberg framework by designating the attacker as the strategic leader and the defender as the reactive follower. The model reflects the actual sequence of events in persistent attacks, where defenders must respond to partial and delayed information.

To formalize this asymmetry, the attacker–defender interaction is modeled as a Stackelberg game embedded within a Partially Observable Stochastic Game (POSG). The POSG structure captures the inherent limitations of defensive visibility in real networks, including stealth techniques, noisy sensors, and delayed alerts. The defender's view of the system is incomplete and indirect; as a result, probabilistic reasoning, implemented through Bayesian belief updates, becomes necessary to estimate the attacker's behavior [3]. Meanwhile, the attacker operates with a clearer strategic view and executes actions that force the defender to make decisions under uncertainty.

To address the complexity of the environment and the limitations of exact analytical methods, this thesis takes a learning-based approach. Large, partially observable games are especially difficult to solve directly, so Reinforcement Learning (RL) offers a practical alternative. Specifically, Proximal Policy Optimization (PPO), a state-of-the-art RL algorithm, is used to train both attacker and defender agents through repeated adversarial self-play [4]. The attacker learns to pursue long-term objectives

such as reaching high-value targets, remaining undetected, or triggering specific alert patterns, while the defender attempts to infer and intercept the attacker's plans. This dual-agent setup supports the emergence of sophisticated and adaptive strategies, mirroring the dynamics observed in real-world cyber conflict.

To ensure operational relevance, the simulation environment is grounded in actual vulnerability data. Attack graphs, constructed using the Attack Dynamics framework, define the attacker's state and action space based on feasible exploitation paths. These paths are derived from Common Vulnerabilities and Exposures (CVEs) and are annotated using the MITRE ATT&CK framework to ensure alignment with recognized adversarial tactics [5], [6]. As a result, the attacker agent does not operate within a simplified or artificial environment but navigates a threat landscape informed by real configurations, vulnerabilities, and behavioral patterns.

In summary, this thesis introduces an attacker-led, game-theoretic framework for modeling and analyzing APT scenarios under conditions of partial observability and demonstrates its feasibility through a lightweight prototype simulation. The proposed framework integrates the strategic structure of Stackelberg games, the uncertainty modeling of POSGs, the exploit realism of attack graphs, and the adaptive capacity of PPO-based RL model. While the full framework supports training for both attacker and defender agents, the prototype implementation trains only the attacker in a simulation informed by real-world threat intelligence. This proof-of-concept establishes a foundation for more realistic and scalable simulations of APT behavior. This framework can be extended as a tool for red-team simulation, proactive defense testing, or academic benchmarking of security models under uncertainty.

1.1 Motivation

APTs are among the most challenging forms of cyber attacks. These threats involve skilled and well-resourced attackers who remain inside a network for a long time while carrying out a series of carefully planned steps. Their goals often include stealing sensitive information, spying on systems, or disrupting services. Unlike ordinary attacks, APTs can change their behavior during the campaign in response to the defender's actions. For this reason, defenders must be prepared to respond not just to individual attacks, but to a chain of possible events that unfold over time. A realistic way to model such threats is necessary, so that defenders can find weak points in the system and test how well their protection measures work before a real attack happens [1].

In recent years, many real-world incidents have shown how serious APTs can be. For example, APT29 was behind the SolarWinds breach in 2020, where attackers compromised supply chain software to reach thousands of organizations [7]. In early 2021, the Hafnium group used unknown vulnerabilities in Microsoft Exchange servers to break into networks [8]. The same year, a critical flaw in Log4j software (known as Log4Shell) allowed attackers to remotely run code on many systems, affecting organizations across the world [9]. More recently, other groups such as Lazarus have stolen large amounts of cryptocurrency [10], while Volt Typhoon has focused on gaining access to infrastructure systems, often without being detected [11]. APT29 has also been reported using cloud-based identity abuse to expand its access in targeted networks [12]. These examples, which are also shown in Figure 1.1, highlight the growing complexity and variety in how APTs operate.

Figure 1.1: Timeline of Major APT Attacks (2020-2025)

A key challenge in defending against APTs is the limited visibility defenders have into real-time attacker behavior. This lack of situational awareness results from both sensor noise and the attackers using stealthy techniques. Security tools like intrusion detection systems may only give partial or delayed alerts, and attackers often take steps to avoid being noticed. Because attackers often avoid detection, defenders operate with limited insight into where the attacker is or what they intend to do next. This uncertainty makes it harder to react effectively in real time. A good model should therefore reflect not just the actions taken, but the information gap defenders face when choosing how to respond. A Stackelberg game model is useful in this case because it reflects the step-by-step interaction between the attacker and the defender. In this model, the attacker acts first, choosing a path through the network, while the defender reacts based on the limited signals they receive. This setup matches how APTs work in the real world, where defenders respond to attacks with limited visibility [3].

To model attacker behavior in such complex situations, RL offers a practical approach. Traditional solution methods for these types of games are often too slow or cannot handle the large number of possible scenarios. RL techniques can help by letting the attacker and defender learn their strategies over time through repeated

practice in a simulated environment. In this thesis, both the attacker and the defender use a learning algorithm called PPO to improve their actions based on past experience. The possible attacker moves are based on an attack graph, which is created using the Attack Dynamics tool [5]. Each action in the graph is also related to a known tactic from the MITRE ATT&CK framework [6], which describes how real attackers operate. By focusing the simulation on real data and threat behaviors, the model supports more realistic training and evaluation of defensive strategies. These challenges highlight the need for a structured simulation framework that can capture both the strategic reasoning of the attacker and the uncertainty faced by the defender. This leads to the core problem this thesis addresses.

1.2 Problem Statement

The central research problem addressed in this thesis is to model and simulate the strategic interaction of an APT attacker and a defender in a realistic cyber environment with hidden information. The APT scenario is formalized as a dynamic Stackelberg game in which the attacker (leader) selects a sequence of multi-step exploits derived from a network attack graph, and the defender (follower) allocates detection and mitigation resources based on partial observations.

This formulation captures essential elements of APT campaigns: an intelligent adversary that plans multi-stage attack paths through known vulnerabilities, and a defender who must infer the attacker's intent from noisy alerts and incomplete network state. In this setup, the defender's challenge is not simply responding to attacks, but

reasoning about the attacker's likely path and goals using indirect signals. To capture this aspect, the framework introduces an observation model and uses Bayesian belief updates to maintain a probability distribution over possible attacker states.

Solving this game is challenging due to the large state and action spaces and the presence of hidden information. Computing an exact Stackelberg equilibrium is intractable for realistic networks. Therefore, attacker and defender policies are learned through RL in a simulated environment. Specifically, PPO is used to train both agents through iterative self-play. Overall, the goal is to integrate game-theoretic modeling, realistic attack-graph structures, and RL-based policy learning under partial observability to develop a coherent and adaptable simulation framework for APT scenarios.

1.3 Objectives

The objectives of this research are as follows:

- Analyze existing approaches for simulating APT campaigns and security games, identifying limitations in capturing strategic behavior using RL models.
- Formulate an attacker-led Stackelberg game under partial observability using a POSG to model sequential attacker–defender interaction.
- Integrate realistic attack graph data into the model using the Attack Dynamics framework and map vulnerabilities to MITRE ATT&CK tactics, thereby constructing a multi-stage action space for the attacker.

- Design an RL-based pipeline for the full framework in which both attacker and defender can be trained using PPO, with Bayesian belief updates enabling inference under uncertainty.
- Implement a lightweight prototype simulation using PPO-based attacker learning on a realistic 10-node network, demonstrating the feasibility of the proposed model.
- Evaluate attacker adaptation against two fixed defender strategies (random patching and neighbor-aware patching), and analyze how these strategies influence attacker progression, stealth, and performance outcomes.

1.4 Research Questions

The research is guided by the following questions:

- **RQ1 (Model Design):** How can a Stackelberg game under partial observability be formalized to model attacker–defender interactions in APT campaigns, and how can RL be integrated into this structure?
- **RQ2 (Model Design):** What role do Bayesian belief updates play in enabling inference and decision-making under uncertainty within the full game-theoretic model?
- **RQ3 (Prototype Analysis):** To what extent can a PPO-trained attacker, operating over a realistic attack graph, learn effective multi-stage strategies against fixed defender baselines?

- **RQ4 (Prototype Analysis):** How do different rule-based defender strategies (random vs. neighbor-aware patching) affect attacker success, detection timing, and behavioral adaptation during training?

1.5 Research Design

This thesis follows a two-stage methodological approach that balances theoretical depth with practical feasibility.

- **Stage 1:** A detailed game-theoretic simulation framework is developed to model APT scenarios under partial observability. This framework, presented in Chapter 4, formulates the attacker–defender interaction as a Stackelberg game embedded in a POSG. It formally defines the state and action spaces, integrates realistic attack graph data, and incorporates a full RL pipeline using PPO for both agents, along with Bayesian belief updates for attacker and defender under partial observability.

- **Stage 2:** To validate the proposed design, a lightweight prototype is implemented in Chapter 5. This proof-of-concept implements a reduced version of the full framework by training only the attacker agent using PPO within a 10-node simulated network. The defender's behavior is modeled using two fixed predefined strategies: random patching and neighbor-based patching. Belief updates and defender learning are not implemented in the prototype, which serves to evaluate attacker adaptation under realistic but constrained settings.

Table 1.1: Framework (Chapter 4) vs. Prototype Implementation (Chapter 5)

Aspect	**Full Framework**	**Lightweight Prototype**
Game Model	Stackelberg + POSG	Same structure assumed
Attacker Policy	PPO-based RL	PPO-based RL
Defender Policy	PPO-based RL + belief updates	Rule-based (random, neighbor); no learning
Belief Modeling	Bayesian updates for defender	Not implemented
Observability	Formal observation models (OA, OD)	Simplified defender view; attacker has full state
Attack Graph	Realistic CVE-based graph (Attack Dynamics + MITRE ATT&CK)	Reduced 10-node version from same tool
Reward Functions	Defined for both agents	Only attacker reward used
Network Scale	Generalizable to large networks	Fixed small network (10 nodes)
Agent Adaptation	Both agents learn adaptively	Only attacker learns
Evaluation Focus	Strategic completeness and design scalability	Attacker behavior under fixed defenses

This clear separation between the full theoretical framework and its simplified prototype implementation highlights the dual contributions of this thesis: the development of a scalable and generalizable framework for modeling APT scenarios, and a minimal empirical demonstration focused on attacker learning and initial strategy evaluation. Table 1.1 summarizes the key conceptual and functional differences be-

tween the two. While the prototype validates core attacker dynamics in a constrained setting, it does not implement the full capabilities of the framework, such as adaptive defender learning or belief-based reasoning.

1.6 Thesis Structure

This thesis is organized as follows:

- **Chapter 1 (Introduction):** Introduces the motivation, research problem, objectives, and research questions.
- **Chapter 2 (Background):** Reviews fundamental concepts and tools used in this work, including APT characteristics, attack graph generation (e.g., Attack Dynamics), the MITRE ATT&CK framework, and key ideas in game theory and RL under partial observability.
- **Chapter 3 (Literature Review):** Surveys related research on APT modeling, security games, and RL in cybersecurity, identifying gaps that the current work addresses.
- **Chapter 4 (Game-Theoretic Framework):** Details the proposed model, including the formal Stackelberg game formulation, integration of attack graphs, observation models, Bayesian belief updates for both agents, and an RL pipeline using PPO for attacker and defender.
- **Chapter 5 (Experimental Setup and Results Analysis):** Describes the simulation environment and training procedures for the lightweight prototype,

which applies PPO-based training to the attacker only, with fixed rule-based strategies for the defender. It also presents the analysis of attacker behavior, performance metrics, and the impact of different defender strategies.

- **Chapter 6 (Limitations and Future Directions):** Summarizes the limitations of the approach and outlines directions for future work and improvements.
- **Chapter 7 (Conclusion):** Concludes by summarizing the key findings reflecting on the theoretical and practical implications of combining Stackelberg game modeling with RL for adaptive threat simulation.

CHAPTER 2

BACKGROUND

This chapter provides the theoretical background necessary to understand the proposed research. It covers key cybersecurity concepts such as APTs, attack graph modeling, and Stackelberg game theory. The discussion extends to POSGs to account for hidden information in adversarial settings, and concludes with an overview of RL, focusing on PPO as the learning method applied in this work.

2.1 Advanced Persistent Threats (APTs)

APTs refer to well-resourced and highly skilled adversaries, often backed by nation-states or organized cybercriminal groups, that infiltrate target networks and remain undetected for prolonged periods [13]. In contrast to opportunistic attacks, APT campaigns are characterized by their use of zero-day exploits, custom malware, and carefully planned operations aimed at high-value targets such as government agencies, critical infrastructure, and large enterprises [13]. The primary objective of APTs is not to cause immediate disruption but to steal sensitive data or maintain control over

systems while avoiding detection. These attackers often exhibit patience, adaptability, and strategic thinking, conducting extensive reconnaissance and tailoring their techniques to the specific context of the targeted environment [14].

APT operations typically proceed through multiple stages, commonly described using the cyber kill chain or attack lifecycle model [15]. The major phases of a representative APT campaign are outlined below:

- **Reconnaissance:** The adversary gathers intelligence on the target through methods such as system scanning, employee profiling, or credential phishing. The goal is to identify vulnerable entry points and exploitable weaknesses [14].
- **Initial Compromise:** The attacker gains initial access, often through spearphishing, software vulnerabilities, or stolen credentials. After establishing entry, they install backdoors or malware to ensure persistence [13].
- **Privilege Escalation and Lateral Movement:** Once inside, the attacker escalates privileges and moves across the network. These actions are conducted stealthily while the adversary identifies critical assets and transitions between systems [14], [13].
- **Collection and Exfiltration:** When the objective system or data is reached, the attacker collects the target information and exfiltrates it from the network. This phase typically involves techniques designed to avoid detection, such as encrypted channels or staged transfers [16].
- **Post-Exfiltration Actions:** The attacker may erase logs, implant additional

persistence mechanisms, or remain dormant for future operations. In some cases, exfiltration continues incrementally, or delayed disruptive actions are launched [15].

This multi-phase lifecycle highlights why APTs are considered both advanced and persistent. These campaigns are substantially more sophisticated than isolated attacks, and once access is gained, attackers often remain for extended periods to exploit the environment strategically [13]. Common objectives include espionage involving intellectual property or classified data, and sabotage of critical systems, as demonstrated by notable cases such as the Stuxnet operation targeting nuclear facilities. Well-documented APT groups, often designated by labels such as APT28 or APT29, have been actively tracked and analyzed by cybersecurity organizations based on their distinctive tools and tactics [13].

Tactics and Techniques

The behavior of APT actors is commonly analyzed using structured taxonomies such as the MITRE ATT&CK framework. This knowledge base catalogues adversarial tactics and techniques identified in real-world cyber incidents [17]. Tactics represent high-level adversarial objectives, such as Credential Access or Lateral Movement, while techniques denote specific methods used to achieve those goals, such as DLL injection or Pass-the-Hash [18]. By correlating observed behaviors with ATT&CK techniques, defenders can infer the attacker's methods and deploy targeted countermeasures. For instance, identifying use of the "Remote Desktop Protocol Hijacking" technique (T1563) enables mitigation through measures such as RDP monitoring or

enforcing multi-factor authentication. Furthermore, statistical analysis of technique co-occurrence can support prediction of unobserved adversarial behavior [18]. Overall, the complexity and stealth of APT operations demand equally strategic and informed defense approaches.

2.2 Attack Graphs and CyberSpace Modeling

To accurately model the complex, multi-stage strategies of APT campaigns, researchers employ attack graphs. An attack graph is a directed graph that captures the potential exploitation paths through a network, based on known vulnerabilities and system configurations [19]. In this representation, nodes correspond to security-relevant system states or conditions, such as unauthorized access to a host, while edges represent attacker actions or exploits that transition the system between states.

Attack Dynamics Framework

The Attack Dynamics framework [5] is a data-driven method for generating attack graphs from real-world network configurations. It integrates network topology, host-level settings, and vulnerability data, producing an attack graph enumerating feasible multi-step paths. Each edge is labeled with specific exploits along with their associated preconditions and postconditions. The framework relies on standardized databases, including CVE, CWE (Common Weakness Enumeration), and CAPEC (Common Attack Pattern Enumeration and Classification), to map known exploits onto the network model.

Simulation Integration

By encoding concrete vulnerabilities and network parameters, attack graphs provide a realistic basis for simulating adversarial behavior. Rather than assuming a generalized or abstract action space, the attacker's choices are constrained to valid paths defined in the graph. Each edge represents a specific exploit, such as targeting a vulnerability on a particular server, thereby ensuring the simulation remains grounded in operationally realistic attack scenarios [19], [16].

Defensive Utility

Attack graphs are also instrumental for defensive strategy design. They enable security analysts to visualize the network's exposure and identify key nodes or vulnerabilities whose mitigation can disrupt multiple potential attack paths [20]. In large or resource-constrained environments, where addressing every vulnerability is impractical, attack graph analysis helps prioritize defensive investments by identifying minimal critical intervention sets [20]. This technique has seen successful applications in both enterprise networks and specialized systems such as SCADA and industrial control environments.

Hybrid Modeling

Recent studies propose hybrid approaches that combine attack graphs with lifecycle models such as the kill chain. A notable example is the "kill chain attack graph," which maps attack stages to specific graph transitions. This approach enhances situational awareness and allows defenders to tailor responses to each stage of an unfolding APT

campaign [21].

In the context of this study, attack graphs constitute the environment's structural foundation for game-theoretic simulation. They define the state space and permissible attacker actions used during training and evaluation (as detailed in Chapter 4). By embedding actual vulnerability data and aligning actions with MITRE ATT&CK tactics, the resulting model ensures that attacker behavior remains representative of real-world threat scenarios [5].

2.3 Stackelberg Security Games (SSGs)

SSGs model strategic interactions in which a defender, acting as the leader, commits to a strategy before the attacker, who observes this commitment and then responds as the follower. This sequential structure reflects asymmetric information flow in cybersecurity environments, where attackers often conduct reconnaissance and adapt their behavior based on the defender's visible actions [22].

Formally, let x denote the defender's strategy, typically represented as a mixed strategy over a finite set of defensive actions. Let y denote the attacker's response strategy. The attacker selects

$$y(x) \in \arg\max_{y} U_F(x, y), \tag{2.1}$$

where U_F is the attacker's utility function. Anticipating this response, the defender

selects their strategy by solving

$$x^* \in \arg\max_x U_L(x, y(x)), \tag{2.2}$$

where U_L is the defender's utility function, and $y(x)$ represents the best response of the attacker to x. The resulting solution constitutes a Stackelberg equilibrium. In practical security scenarios, the defender's strategy x may encode the probabilities of deploying protections across a set of targets, while the attacker's strategy y corresponds to the selection of a specific target.

For example, in a multi-target setting, the defender's expected utility can be expressed as

$$\mathbb{E}[U_L] = \sum_t q_t(x) \cdot U_D(t) \cdot (1 - x_t), \tag{2.3}$$

where x_t is the probability of defending target t, $q_t(x)$ is the probability of the attacker selecting target t given defense strategy x, and $U_D(t)$ is the utility the defender derives from protecting t [22]. This formulation captures the adversarial adaptation to defense levels and enables optimal allocation of limited defensive resources.

This bi-level optimization is typically solved via backward induction: for each defense strategy, the attacker's best response is computed first, followed by selecting the defense that maximizes the defender's expected utility [23].

2.3.1 Model Variations

To address practical considerations such as uncertainty and resource limitations, several extensions of the standard Stackelberg model have been proposed.

Bayesian Stackelberg Games

In this variant, the defender faces uncertainty over the attacker's type, which may reflect varying capabilities, objectives, or resources. The type is drawn from a known prior distribution, and the defender seeks a strategy that maximizes expected utility across this distribution. Zhang and Malacaria [24] formulated a multi-stage cyber defense system using a Bayesian Stackelberg game. Their bi-level model simultaneously selects an initial control configuration and an adaptive response strategy. By leveraging mixed-integer conic programming solvers, they efficiently addressed the complexity introduced by uncertainty in attacker types.

Robust Stackelberg Games

When attacker payoffs or capabilities are only partially known, robust formulations prepare defenses for worst-case attacker scenarios. Shukla et al. [23] introduced a cost-based Stackelberg equilibrium model to address ambiguity in attacker resources. Their formulation integrates action costs for both players into a zero-sum framework, allowing the defender to manage trade-offs between performance and cost. Their defense strategy remained effective under uncertainty, as demonstrated in power grid case studies where defender actions were robust to attacker budget variations.

Stackelberg Games with Bounded Rationality

Classical Stackelberg models assume perfect rationality, although real attackers often behave stochastically or inefficiently. Quantal response models assign probabilities to attacker choices based on utility, accounting for bounded rationality. Temghart et al. [2] applied this model to cloud security, allowing each attack option to have a probability that increases with its payoff and decreases with the defender's coverage. Their analysis showed that randomized defense strategies, where each defense action is selected with non-zero probability, improve resilience against attackers who do not behave perfectly rationally.

Dynamic and Multi-Stage Stackelberg Games

Traditional Stackelberg games are static, single-round interactions. However, real-world cyber conflicts typically unfold over time. Dynamic Stackelberg games extend the model to multi-stage scenarios in which both the attacker and defender act across evolving system states. Liu et al. [25] proposed a dynamic Stackelberg game for securing Internet-of-Things (IoT) systems. The defender, acting as the leader, allocates limited security resources over time, while attackers select intrusion strategies accordingly. The authors employed Bellman's principle of optimality to derive both open-loop and feedback equilibrium solutions that adapt to the system's evolving risk state.

2.4 Partially Observable Stochastic Games (POSGs)

While SSGs assume that the follower can observe the leader's strategy, real-world cyber conflicts frequently involve uncertainty and limited observability. In practice, defenders often have only partial visibility into attacker activities, and attackers may lack full knowledge of the network environment. This motivates the use of POSGs, a class of multi-agent decision-making models that extends Partially Observable Markov Decision Processes (POMDPs) to competitive settings involving two or more players [26].

A POSG is defined by a set of hidden system states, action spaces for each player, probabilistic state transitions, and observation functions. These observation functions determine the information each player receives about the true state at any point in time. For example, in cybersecurity applications, the system state may represent which hosts are compromised or which network segments are at risk, while the players' actions correspond to possible attack or defense operations. Crucially, the true state is not fully observable by either agent. Instead, each player receives noisy or partial observations, such as alerts, scan results, or sensor signals [26].

Each player maintains a *belief state*, which is a probability distribution over possible system states, and updates this belief using Bayesian inference based on the sequence of observations received. Players then select actions based on their current beliefs, aiming to maximize their respective expected cumulative rewards. State transitions depend stochastically on both players' actions, and outcomes reflect the success or failure of cyber operations such as exploit attempts or defensive countermeasures [26].

POSGs in Cyber–Attack Defense Scenarios

POSGs have been increasingly applied to model scenarios involving stealthy intrusions and incomplete information. A notable example is the modeling of APTs, where attackers gradually progress through a network while remaining undetected. Wang et al. [27] introduced the "Catch the Cyber Thief" game, a POSG-based framework that captures the strategic asymmetry between an advanced attacker and a defender operating under uncertainty.

The POSG structure naturally accommodates these asymmetries. The attacker observes local features during exploration, while the defender receives partial information through alerts and other indicators. By formulating the interaction as a POSG, the model allows security researchers to analyze how APTs exploit information gaps and how defenders can best deploy limited monitoring resources. In particular, it facilitates the derivation of optimal attacker policies and informed defender strategies, such as sensor placement or targeted scanning, that would be difficult to identify through fully observable or static models. This positions POSGs as a compelling framework for understanding and countering real-world, multi-stage cyber threats under uncertainty [27].

2.5 Reinforcement Learning and Proximal Policy Optimization

RL is a foundational paradigm for sequential decision-making under uncertainty. In this framework, an agent interacts with an environment to maximize cumulative reward over time. The interaction is typically formalized as a Markov Decision Process (MDP), defined by the tuple $\langle S, A, P, R, \gamma \rangle$, where S denotes the state space, A the action space, $P(s' \mid s, a)$ the transition probability, $R(s, a)$ the reward function, and $\gamma \in [0, 1)$ the discount factor [28]. The agent learns a policy $\pi(a \mid s)$, which maps each state to a probability distribution over actions, and aims to maximize the expected discounted return:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{2.4}$$

Classical approaches to RL include value-based methods such as Q-learning, and policy-based methods that directly optimize the policy parameters using gradient ascent. Among the latter, policy gradient methods are particularly well-suited for environments with continuous or high-dimensional action spaces. The gradient of the expected return with respect to the policy parameters θ is given by [29], [30]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) \, A^\pi(s, a) \right], \tag{2.5}$$

where $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ is the advantage function, representing the relative benefit of taking action a at state s.

To improve the stability and efficiency of learning, actor–critic methods combine two components: a policy (actor) $\pi_\theta(a \mid s)$, and a value function (critic) $V_w(s)$, where w denotes the critic's parameters. The critic estimates the expected return to reduce the variance of the policy gradient updates. Notable implementations include the Asynchronous Advantage Actor-Critic (A3C) algorithm, which uses parallel environments to stabilize training [31].

2.5.1 PPO

PPO is a widely used actor–critic algorithm that improves learning stability by constraining updates to the policy. PPO achieves this by maximizing a clipped surrogate objective, which limits how much the new policy can diverge from the previous one [32]. The clipped objective is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\, \hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\, \hat{A}_t \right) \right], \tag{2.6}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and $\hat{A}_t$ is an estimate of the advantage. The clipping operation ensures that $r_t(\theta)$ remains within a small trust region around 1, typically controlled by a hyperparameter $\epsilon \in [0.1, 0.2]$, thereby discouraging abrupt changes in the policy.

To further enhance exploration and value estimation, PPO often includes an entropy bonus and a value loss term. The full training objective is given by:

$$L^{\text{total}}(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{value}} + c_2 H[\pi_\theta], \tag{2.7}$$

where L^{value} is the squared error loss for the value function, $H[\pi_\theta]$ denotes the entropy of the policy distribution, and c_1, c_2 are scalar coefficients weighting each term.

Partially Observable Settings

In many practical applications, including cybersecurity, the agent cannot access the full environment state. This setting is formalized as a POMDP, in which the agent must act based on observations or belief states [33]. One common approach is to use a recurrent neural network within the policy or critic to retain memory of past observations [34]. Alternatively, belief-state representations can be constructed and passed as input to the policy network [35].

Extensions of PPO to partially observable environments have included recurrent PPO agents that learn effective policies under uncertainty [36]. These modifications are especially relevant to adversarial cybersecurity scenarios, where attacker and defender agents operate with incomplete information.

CHAPTER 3

LITERATURE REVIEW

This chapter surveys existing research in cybersecurity that applies game theory and RL to model strategic attacker–defender interactions. It reviews classical and hierarchical game-theoretic models, explores recent advancements involving partial observability and stochastic dynamics, and examines RL approaches for simulating cyber operations. The review concludes with a gap analysis that positions this thesis within the broader research landscape.

3.1 Game-Theoretic Models in Cybersecurity

Game theory has emerged as a foundational tool in cybersecurity research, enabling the structured modeling of strategic interactions between adversarial agents. In particular, attacker–defender scenarios are naturally suited to game-theoretic formulations due to the competitive and sequential nature of cyber conflicts. As modern cyber threats become increasingly sophisticated, including APTs and coordinated intrusions, the need for mathematically grounded decision models that account for uncertainty,

stealth, and adaptive behavior has intensified. This section surveys game-theoretic models applied in cybersecurity across various domains, from smart grids and IoT systems to cyber-physical and software-defined networks.

The reviewed works are categorized into three major groups based on their theoretical formulation and application focus. The first subsection introduces classical game-theoretic approaches, including static and simultaneous-move games, with emphasis on risk modeling and strategic investment. The second subsection explores hierarchical defense strategies modeled via Stackelberg games, particularly in deception-driven environments. The third subsection presents advanced models based on stochastic games, partial observability, and RL, which enable the simulation of complex attacker behavior and learning-based defense mechanisms. This structure provides a logical progression from foundational formulations to highly adaptive and scalable solutions. A comparative summary of the surveyed models is presented in Table 3.1.

3.1.1 Classical and Zero-Sum Game-Theoretic Models

Classical game-theoretic approaches in cybersecurity primarily employ static and simultaneous movement formulations to model attacker–defender interactions. These models emphasize strategic investment, risk quantification, and optimization under fixed assumptions.

Hyder et al. [38] develop a static zero-sum game model that integrates attack-defense trees to formalize attacker–defender interactions in smart grids, drawing strategy spaces from structured representations. Attack strategies represent paths to critical targets, while defense strategies involve investment in protective measures as-

Ref	Year	Game Type	Initiative / Role Order	Key Purpose
[37]	2024	Multi-domain zero-sum and stochastic game	Defender-first setup, simultaneous execution	Strategic deception across cyber-physical layers
[38]	2022	Zero-sum normal form (ADT + GTF)	Defender initiates; attacker reacts	Defense investment planning in smart grids
[39]	2025	Stackelberg (CoASG) in SDN	Defender-leads (SE)	Proactive thresholding in SDN and DDoS mitigation
[40]	2025	Markov signaling game	Defender signals first	Multi-stage deception with Bayesian update
[41]	2025	Stochastic game with BAGs	Attacker initiates per state	Dynamic defense in power systems
[42]	2025	Static mixed-strategy game	Attacker-first logic (risk-based)	ICS risk mitigation
[43]	2023	Zero-sum matrix APT game	Attacker leads; defender reacts periodically	Stealth APT defense in robotics
[44]	2022	Matrix game + knapsack optimization	Attacker initiates (path choice)	Budgeted defense planning
[45]	2025	Mixed-strategy game over CKC	Attacker-first per kill chain	Multistage response mapping
[3]	2024	POSG with attack graph	Attacker-first (simultaneous frame)	IoT mitigation via cost-based strategy
[46]	2025	Stackelberg stochastic game	Defender leads explicitly	Deception via honeypoints

Table 3.1: Summary of Game-Theoretic Models in Cybersecurity

sociated with access nodes. The model incorporates real-world elements including attack costs, defense costs, impact assessment, and attack probabilities. A synthetic cyber-physical smart grid test case validates the approach. The study highlights the effectiveness of this hybrid model in identifying optimal defense strategies; however, it does not incorporate partial observability or large-scale simulations.

Zhang et al. [42] develop a game-theoretic framework to quantify cybersecurity-induced risks in process industry systems and to guide optimal protective strategy selection. The model uses a static game with complete information, where attack and defense payoffs are derived from risk metrics based on process deviations caused by cyber-induced anomalies in control variables. Attack strategies are modeled using known threats like Stuxnet, while defense strategies are categorized into low- and high-cost options. Although the model does not use attack graphs, it integrates tolerable losses and evolving risk to calculate Nash equilibria. A simulated case study on a catalytic cracking unit validates the framework, although field-scale validation and defense cost modeling remain open challenges.

Kalderemidis et al. [44] propose GTM, a software-supported methodology that combines game theory, attack graphs, and budget optimization to design optimal cybersecurity defense strategies under budget constraints. The model is based on a static zero-sum game between a defender and multiple attacker types, incorporating attack graph data to simulate multistage attack paths. Defense strategies are evaluated using Nash equilibrium, and the integration of the 0-1 Knapsack algorithm ensures cost-effective security control selection. The methodology supports both technical and environmental vulnerabilities, and it is validated through three case studies including

real-world inspired attack graphs. Although GTM demonstrates high adaptability and efficiency, its reliance on manually defined threat probabilities may affect scalability in complex environments.

Kour et al. [45] present a game-theoretic model integrated with the Cyber Kill Chain (CKC) to analyze cybersecurity strategies across multiple stages of an attack lifecycle. The model adopts a simultaneous-move, non-cooperative game framework using mixed strategies, where attackers and defenders make probabilistic choices over predefined actions. The defender selects among monitoring, detection, and response actions, while the attacker chooses strategies aligned with CKC stages such as reconnaissance, delivery, and installation. The study includes a case simulation using random probability distributions to evaluate expected payoffs. While the model provides insights into equilibrium behavior and decision-making under uncertainty, it does not incorporate partial observability or dynamic adaptation mechanisms.

Collectively, these models provide analytical insights into optimal defense decisions under simplified or complete information settings.

3.1.2 Stackelberg and Leader–Follower Models for Strategic Defense

In contrast to static formulations, Stackelberg game models are well-suited for cybersecurity scenarios involving hierarchical decision-making, where defenders proactively commit to strategies that attackers observe and respond to.

Razvan et al. [39] adopt the SEC-SDN framework, which enhances the security

of Software-Defined Networking (SDN) environments by integrating game-theoretic modeling with a Snort-based intrusion detection system. The model adopts a dynamic Stackelberg game structure where the SDN controller acts as the defender (leader) and the attacker is the follower, enabling strategic deployment of honeypots and rate-limiting mechanisms. Attack and defense strategies are defined through bandwidth control and flow rule adjustments. The framework does not use explicit attack graphs but models adaptive thresholds and strategic responses based on attacker behavior. Experimental validation using Mininet and OpenStack testbeds confirms improved DDoS mitigation, though scalability and dynamic threat adaptation remain areas for improvement.

Wang et al. [46] introduce the BagSg model, a proactive deception defense approach that integrates Bayesian attack graphs and Stackelberg games to guide honeypot deployment against covert threats. It introduces a novel deception technique called Honeypoint, which operates covertly by mimicking legitimate services and dynamically adjusting its configuration. The model uses Bayesian attack graphs to assess attack paths and vulnerabilities, while the Stackelberg game captures strategic attacker-defender interactions under incomplete information. The framework supports partial observability, attack graph analysis, and differentiates between ordinary and advanced attackers. Experimental results demonstrate Honeypoint's superiority in capturing stealthy threats, although quantifying payoffs and scaling deployment under constrained resources remains a challenge.

Hemida et al. [37] present a game-theoretic framework for modeling cyber deception in cyber-physical systems, emphasizing the integration of deception strategies

across both cyber and physical domains. It introduces a novel multi-domain model where the attacker and defender operate within interdependent action spaces in each domain. The proposed approach builds on one-shot normal-form and stochastic game structures, incorporating partial observability to simulate realistic uncertainty. Attack strategies are derived from attack graphs, while defensive actions include deploying honeypots and crafting misleading topologies. The framework evaluates Nash equilibrium existence under domain consistency constraints. The authors apply their model to modern warfare scenarios, illustrating cross-domain deception effectiveness. However, scalability and computational complexity in multi-domain settings remain key challenges.

These works demonstrate how leader–follower dynamics can enhance proactive defense, particularly when integrated with attack graph structures and deception mechanisms.

3.1.3 Stochastic, POSG, and Learning-Based Formulations

Recent cybersecurity research increasingly uses stochastic games, partial observability, and learning techniques to simulate complex attacker behavior and derive adaptive defense policies.

Bai et al. [40] present a Moving Target Markov Signaling Game Defense Model (MTMSGDM) to address asymmetric information and strategic uncertainty in network security. The model integrates a Markov decision process with a signaling game, where defenders emit deceptive signals and attackers form posterior beliefs to decide optimal strategies. Attack and defense actions are optimized across multiple game

stages, with stage importance weighted using the PageRank algorithm. The refined Bayesian equilibrium is used to compute strategy profiles. Partial observability is modeled through belief updates, and simulation results show significant gains in detection rate and attacker deterrence compared to Stackelberg and equitable distribution models. A real-world inspired testbed with vulnerabilities is used; however, the model's complexity may limit scalability.

Yao et al. [41] model a dynamic game-theoretic defense model for Cyber-Physical Power Systems (CPPS) under coordinated cyber-attacks. The authors develop a stochastic game framework integrated with Bayesian Attack Graphs (BAGs) to capture the multi-stage attack penetration process from the cyber layer to physical components. The attacker and defender interact strategically over discrete time steps, with the defender optimizing security controls to minimize expected system degradation. RL is employed to compute Nash equilibrium strategies under incomplete information. Partial observability is handled via posterior probability updates based on detected attack evidence. Experimental validation using the IEEE 57-bus system demonstrates significant improvements in adaptive decision-making, although scalability and computational efficiency remain challenging.

Hamza et al. [3] propose AGA-POSG, a formal approach that transforms attack graph analysis in IoT networks into a two-player POSG. The model addresses the uncertainty and complexity of IoT environments by integrating state transitions, observation probabilities, and cost functions derived from real device vulnerabilities. Attack and defense strategies are derived from known exploits and countermeasures, and analyzed through iterative elimination of dominated strategies in normal-form

representation. A case study involving a multi-device IoT topology illustrates strategy computation and attack path cost evaluation. The approach supports partial observability and attack graphs, though scalability to large networks remains a challenge.

Rass et al. [43] develop and experimentally validate a game-theoretic defense model named Cut-The-Rope for securing cyber-physical systems against APTs. The model formulates a zero-sum, infinite-horizon attacker-defender game on attack graphs, where the defender optimizes spot-checking policies without knowing the attacker's exact position. The framework models stealthy intrusions, excludes feedback or signaling mechanisms, and supports movement patterns such as Poisson, geometric, and complexity-weighted exploit traversal. The study uses real attack graphs from industrial robots (MARA and MiR100) to demonstrate the defense policy's effectiveness compared to heuristic baselines. Despite promising results, the model assumes complete attack graph knowledge and lacks support for reactive strategies.

These contributions reflect a transition toward dynamic modeling approaches capable of addressing uncertainty and strategic adaptation across multi-stage attack scenarios.

3.2 RL for Strategic Cyber Operations

RL has emerged as a powerful paradigm for automating strategic decision-making in cyber operations, enabling agents to learn complex attack and defense behaviors through environmental interaction. This section categorizes existing research into

three distinct thematic areas to clarify methodological and conceptual advances. The first subsection reviews single-agent RL approaches for modeling offensive actions such as penetration, exfiltration, and malware generation. The second subsection examines self-play and game-theoretic frameworks that simulate adversarial co-evolution between attackers and defenders. The final subsection focuses on multi-agent and hierarchical RL models designed to coordinate defensive strategies under partial observability and dynamic threats. Table 3.2 summarizes these recent RL approaches applied to cybersecurity, highlighting their game types, agent roles, and execution order.

3.2.1 Single-Agent RL for Offensive Cyber Operations

This subsection groups studies that use RL to model attacker behavior in isolation. These papers focus on payload selection, penetration testing, exfiltration, ransomware simulation, and synthetic attack generation, typically without defender involvement or multi-agent dynamics.

AlMajali et al. [57] propose a RL-based agent that automates vulnerability exploitation using Q-learning within the Metasploit framework. The agent selects the most effective payload for a given operating system and vulnerability to establish a reverse shell. The approach models attacker behavior and uses an epsilon-greedy strategy under partial observability. Experiments exploiting CouchDB 3.1.0 show consistent success across multiple systems, significantly reducing exploitation time compared to manual methods.

Li et al. [50] introduce EPPTA, a penetration testing agent based on PPO, de-

Ref	Year	RL Technique	Game Type	Player Roles	Execution Order
[47]	2024	Hierarchical MARL	Stackelberg Hypergame	Attacker as leader, Defender acts first	Defender to Attacker
[48]	2022	PPO, A2C	Multi-agent turn-based	Red and Blue agents (jointly trained)	Attacker to Defender
[4]	2024	Self-Play PPO	OODA-loop with POMDP	Red vs Blue (symmetric)	Alternating / self-play
[49]	2024	DQN, Actor-Critic, PPO	MDP attacker simulation	Attacker only	Attacker first
[50]	2023	PPO, LSTM belief module	Single-agent POMDP	Attacker only	Attacker first
[51]	2023	PPO	MDP exfiltration pathing	Attacker only	Attacker first
[52]	2024	PPO, Fictitious Play	Zero-sum Markov game	Both agents learn	Blue to Red
[53]	2024	Hierarchical MARL (IPPO)	Zero-sum Dec-POMDP	FSM Red, RL Blue	Red initiates attack
[54]	2024	PPO with risk aversion tuning	Single-agent MDP	Attacker only	Attacker first
[55]	2023	Q-learning	Mutation-based RL tasks (Malware, DoS)	Attacker only	Attacker first
[56]	2024	TD3, PPO, Q-Learning	Markov game with self-play	Both agents train	Simultaneous; attacker starts
[57]	2023	Q-learning	Single-agent MDP	Attacker only (Metasploit)	Attacker first

Table 3.2: Summary of selected RL approaches in cybersecurity

signed for partially observable environments. The agent incorporates a novel belief update module aligned with POMDP principles, enabling accurate state estimation under uncertainty. EPPTA models attacker behavior only and operates within the NASim simulation platform. Integrated with Sample Factory, it significantly improves convergence speed and scalability. Evaluations show EPPTA outperforms baselines across six network configurations, especially under node isolation and large action spaces.

Rishu et al. [51] present a RL framework for discovering exfiltration paths while considering protocol usage and payload volume. The attacker agent, trained using PPO, operates under partial observability and navigates dynamic network defenses such as firewalls. The environment models protocol-aware path selection and variable upload rates. Experiments on two networks (56 and 1444 hosts) show that the agent learns optimal, stealthy exfiltration paths while evading detection, validating scalability and realism in adversarial simulations.

Wang et al. [54] implement a RL-based ransomware simulation framework for red teaming, using PPO to train attacker agents in a time-constrained enterprise network. The agent models ransomware behavior, balancing stealthy lateral movement with rapid encryption, while avoiding decoy honeyfiles under partial observability. Experiments on a 152-host network reveal risk-sensitive strategies, showing that retraining after defensive changes leads to adaptive evasive behavior. The study demonstrates iterative attacker-defender learning in a dynamic environment.

Hernandez-Suarez et al. [55] propose ReinforSec, a RL framework for generating synthetic malware samples and denial-of-service (DoS) attacks. The agent uses

Q-learning within a POMDP-based OpenAI Gym environment to mutate Portable Executable binaries and craft DoS attack traffic from PCAP traces. The study models attacker behavior under partial observability without multi-agent interaction. Experiments demonstrate that the synthetic outputs are effective, with over 97% of malware and 64% of DoS samples detected by real-world sensors, validating their realism and threat relevance.

Oh et al. [49] investigate the application of deep RL to cyber-attack simulation using a scenario based on MITRE ATT&CK. The study compares PPO, actor–critic, DQN, and tabular Q-learning to model attacker behavior in a simulated network of PCs. The agents operate under partial observability and interact with the environment through node discovery and exploitation stages. The actor–critic algorithm achieved the highest success rate and learning efficiency, highlighting its advantage in dynamic cyberattack environments.

3.2.2 Self-Play and Game-Theoretic RL for Strategic Interaction

This subsection focuses on research incorporating self-play and adversarial learning where attacker and defender agents co-evolve or compete strategically, often within formal game-theoretic or POMDP/zero-sum setups.

Zhang and Luo [4] employ a RL framework with self-play to solve dynamic cyber attack-defense games modeled via POMDPs and guided by the OODA loop. The method trains attacker and defender agents using PPO within an Actor–Critic ar-

chitecture, under conditions of partial observability. Self-play enables strategic co-evolution through alternating training. Simulations show improved attacker performance and stable equilibria when compared to static or naive training setups. A novel two-stage opponent sampling strategy enhances learning robustness and convergence stability.

Jaber [56] utilize a self-play RL framework for intrusion detection and prevention systems, integrating the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm with enhanced autoregressive policy representation. The framework models attacker-defender dynamics using zero-sum Markov games under partial observability. Evaluations in a simulated environment demonstrate improved policy convergence, reduced false positives, and adaptive defense behavior. The approach highlights the utility of TD3 in continuous action spaces for real-time cybersecurity adaptation without expert-crafted rules.

McDonald et al. [52] present a competitive RL framework for autonomous cyber operations using fictitious play to approximate game-theoretic optimal strategies. The study models both attacker and defender agents within a partially observable zero-sum Markov game and trains policies using PPO. Conducted in a modified CybORG environment, the agents iteratively learn minmax strategies, achieving convergence near Nash equilibrium. The work demonstrates that opponent sampling stabilizes training and reduces exploitability in adversarial policy learning.

3.2.3 Multi-Agent and Hierarchical RL for Coordinated Cyber Defense

This subsection centers on defensive modeling and multi-agent collaboration using hierarchical or decentralized learning, often grounded in CybORG or Dec-POMDP environments.

Kunz et al. ([48] extend CyberBattleSim to support autonomous blue (defender) agents alongside red (attacker) agents in a multi-agent RL framework. They employ PPO and A2C algorithms and demonstrate joint training of attackers and defenders using the MARLon system. The environment models realistic enterprise networks under partial observability. Simulation results show that jointly trained defenders significantly reduce red agent success, although red agents struggle with overfitting and constrained exploration. Invalid action handling and network availability constraints were key training challenges.

Tang et al. [47] introduce a Hierarchical Multi-Agent RL (HMARL) framework for autonomous cyber defense using a Stackelberg hypergame model. The approach models both attacker and defender roles, incorporating partial observability via information loss parameters. PPO is used for low-level agent training, while Q-learning guides high-level coordination. Simulated in the CybORG platform, the method improves defense strategy efficiency, coordination, and adaptability compared to non-hierarchical baselines.

Singh et al. [53] develop a HMARL framework for scalable cyber network defense in partially observable environments. The approach models a team of blue agents

using PPO with decentralized critics and sub-task-specific policies coordinated by a master policy. The environment is formulated as a Dec-POMDP and instantiated using CybORG CAGE 4, involving stealthy red agents and shared team rewards. Experimental results show faster convergence, improved defense precision, and successful sub-policy transferability, outperforming centralized and non-hierarchical baselines.

3.3 Gap Analysis

Existing literature on cybersecurity modeling using RL and game-theoretic frameworks contains several limitations that constrain the design of adaptive and strategic attack simulations. A systematic evaluation of 23 recent contributions reveals significant gaps in areas critical to the advancement of dynamic cybersecurity operations. These gaps are particularly evident in aspects involving strategic learning integration, adversarial realism, and the modeling of asymmetric attacker–defender scenarios. Table 3.3 provides a structured summary of these gaps along with the corresponding contributions proposed in this thesis.

One of the most evident shortcomings in the reviewed literature is the limited integration of RL into security-focused modeling. Although RL has the capacity to enable adaptive, sequential decision-making in dynamic environments, only slightly more than half of the surveyed studies incorporate any form of RL. Among those that do, multi-agent RL remains insufficiently addressed, appearing in fewer than one-fourth of the RL-based papers. This significantly limits the potential to simulate interactive, revolving behaviors between attackers and defenders, which are central to

Identified Gap	**Paper Count**	**%**	**Ref**	**Thesis Contribution**
Use of RL	13	56.5	[41], [52], [48]	Incorporates PPO-based RL for attacker-defender strategy learning.
Use of Multi-Agent RL	5	21.7	[52], [48], [47]	Employs multi-agent RL for realistic adversarial interaction modeling.
Integration of RL with Stackelberg Games	1	4.3	[47]	Combines RL explicitly with Stackelberg strategic leadership modeling.
Modeling APT	3	13	[43], [46], [3]	Simulates multistage APT behavior using RL under uncertainty.
Non-RL Studies Using Stackelberg Models	2	8.7	[39], [46]	Integrates Stackelberg leadership explicitly with adaptive RL approaches.
Attack Graph Utilization	7	30.4	[44], [43], [3], [41]	Uses realistic attack graphs based on MITRE ATT&CK and CVE frameworks for practical modeling.
Partial Observability Modeling	7	30.4	[3], [41], [47], [53]	Employs Bayesian-based partial observability.

Table 3.3: Comprehensive Gap Analysis with Explicit Thesis Contributions

real-world cyber operations.

Another important gap relates to the almost complete absence of frameworks that integrate RL within a Stackelberg game structure. Stackelberg models are widely acknowledged as effective representations of hierarchical strategic interactions, yet their application within RL systems is rarely observed. The disconnection between these two domains creates a barrier to modeling asymmetric initiative dynamics, where one agent strategically leads and the other responds.

Furthermore, the majority of existing literature implicitly assumes an attacker-first model. While this assumption aligns with certain real-world conditions, it overlooks a wide range of proactive defense strategies. This thesis addresses the issue by explicitly modeling the attacker as the strategic leader in a Stackelberg environment, thereby enabling an empirical investigation of adversarial initiative and leadership.

Only two studies employ Stackelberg models outside of a RL framework, and none explore the strategic leadership dimension within a learning environment. This absence limits the applicability of results to dynamic, data-driven cyber defense scenarios. Additionally, a minority of the works analyzed use attack graphs that are grounded in realistic frameworks such as MITRE ATT&CK and CVE databases. The simplification or abstraction of these graphs limits the operational relevance and effectiveness of the proposed defense strategies.

Partial observability is another key area where literature remains underdeveloped. Despite the fact that cyber defenders operate with incomplete and often uncertain information, only a limited portion of the studies incorporate observation uncertainty into their models. The remaining works rely on idealized full-information setups that

rarely occur in practical scenarios. Finally, only a small fraction of the reviewed studies explicitly simulate APTs. These threats are characterized by stealth, long-term planning, and multi-stage exploitation, yet their complexity is often reduced to short-term or reactive attacker behaviors in most models.

This thesis addresses all of the identified gaps. It introduces an RL-based framework built upon PPO that captures strategic decision-making under partial observability. The model integrates Stackelberg leadership, designating the attacker as the strategic leader, and supports multi-agent interaction to better reflect adversarial dynamics. Realistic attack graphs based on MITRE ATT&CK and CVE vulnerabilities are utilized to simulate plausible intrusion pathways. Furthermore, the model reflects APT behavior by enabling the attacker to operate under uncertainty across multi-stage scenarios, allowing the evaluation of long-term threats and defender responses.

CHAPTER 4

GAME-THEORETIC FRAMEWORK

This chapter formalizes an integrated framework for modeling APT scenarios using a game-theoretic approach with RL-based decision-making. The interaction between an attacker and a defender is modeled as a two-player POSG with a Stackelberg leader-follower structure, meaning one player (the leader) commits to a strategy that the other (the follower) then responds to optimally. In this framework, the attacker assumes the role of the Stackelberg leader, and the defender acts as the follower, adapting its strategy in response to the observed attack. Both agents employ policy gradient methods to learn effective decision policies within this game-theoretic framework.

In summary, the proposed framework effectively gamifies the cybersecurity engagement by representing it as a strategic game between the attacker and defender. This chapter introduces the formal model components and assumptions that underlie this framework, setting the stage for the simulation and analysis in subsequent chapters.

Formally, the game can be defined by the tuple:

$$\mathcal{G} = \langle S,\ \{A_A,\ A_D\},\ T,\ \{O_A,\ O_D\},\ R_A,\ R_D \rangle, \tag{4.1}$$

where S is the set of environment states capturing the security status of the system. A_A and A_D are the action sets of the attacker and defender, respectively. The state transition function $T : S \times A_A \times A_D \to \Delta(S)$, where $\Delta(S)$ denotes the set of all probability distributions over S, defines the probability distribution over next states given the current state and both agents' actions. O_A and O_D are the observation models for the attacker and defender, respectively. Each defines a conditional distribution $O_i(s, a, s')$, which corresponds to the probability distribution $P(o_i \mid s, a, s')$ over possible observations o_i. Finally, R_A and R_D are the reward (or utility) functions for each agent. A complete summary of all mathematical symbols used in this framework is provided in Appendix A.1.

This specific formulation is a reduction from the more general structure of a POSG, formally given by:

$$\mathcal{G}_{\text{POSG}} = \langle N,\ S,\ \{A_i\}_{i \in N},\ T,\ \{O_i\}_{i \in N},\ \{R_i\}_{i \in N} \rangle, \tag{4.2}$$

where $N = \{A, D\}$ is the set of agents (attacker and defender), S is the set of environment states, A_i is the action set for agent i, T is the state transition function, O_i is the observation model, and R_i is the reward function. The instantiation used in this framework aligns this general form with a Stackelberg security game structure, where

the attacker commits to a strategy as a leader and the defender responds optimally as a follower under partial observability.

The attacker's objective is to progress through the Cyber Kill Chain (CKC), such as escalating privileges and moving laterally, to ultimately compromise critical assets, while the defender's objective is to detect, delay, or block the attack to protect those assets. This formulation captures a general-sum stochastic game where the goals of the two players are adversarial but not strictly zero-sum, as both may incur independent costs irrespective of the other's reward outcomes.

A defining characteristic of APT engagements is partial observability, where both the attacker and defender have limited knowledge of the true underlying state of the system. The attacker might not know which vulnerabilities are present or which defenses are active in the network, and the defender might not know the attacker's presence or location until some alert is triggered. Therefore, each agent must maintain a belief state which is a probability distribution over possible states and update this belief as observations are obtained. This reformulates the problem as a POSG, as introduced earlier, which is significantly more complex than a fully observable game due to the uncertainty each player has about the true state and the information available to the other.

To ground the modeled environment in real-world cyber scenarios, the framework integrates a data-driven attack graph generation tool called *Attack Dynamics* by Sönmez et al. [5]. Attack Dynamics automatically produces an attack graph for a given scenario by taking as input the network topology, host configurations, and known vulnerabilities identified by CVE. It maps these vulnerabilities to known weakness types

under the CWE and known attack techniques under the CAPEC, then enumerates all feasible multi-step attack paths in the network. In essence, Attack Dynamics encodes the cyber terrain of the scenario by providing the set of possible exploits represented as edges in the attack graph, together with their prerequisites (preconditions) and consequences (postconditions).

The framework also utilizes the MITRE ATT&CK knowledge base as an authoritative reference to ensure tactical and behavioral realism, for example, linking each identified vulnerability or technique to known attacker tactics and corresponding mitigations. By using Attack Dynamics to generate the attack graph and referencing MITRE ATT&CK for tactics and mitigation measures, the state space S, the action sets A_A, A_D, and the transition dynamics T in the model are firmly grounded in real-world data and security knowledge.

Finally, solving such a game model directly, such as computing an exact Stackelberg equilibrium, is analytically intractable in realistic settings due to the huge state space and the partial information available to each player. Instead, the framework uses RL to approximate optimal or near-optimal policies for the attacker and defender. The attacker and defender agents are trained through repeated simulated engagements using self-play, gradually improving their strategies over time. Figure 4.1 illustrates the full interaction structure between the attacker, environment, and defender within the proposed Stackelberg game-theoretic APT simulation framework.

A formal definition of the Stackelberg equilibrium concept for this game is presented in the next section, followed by detailed descriptions of each component of the framework: the state representation and attack graph integration, the action spaces

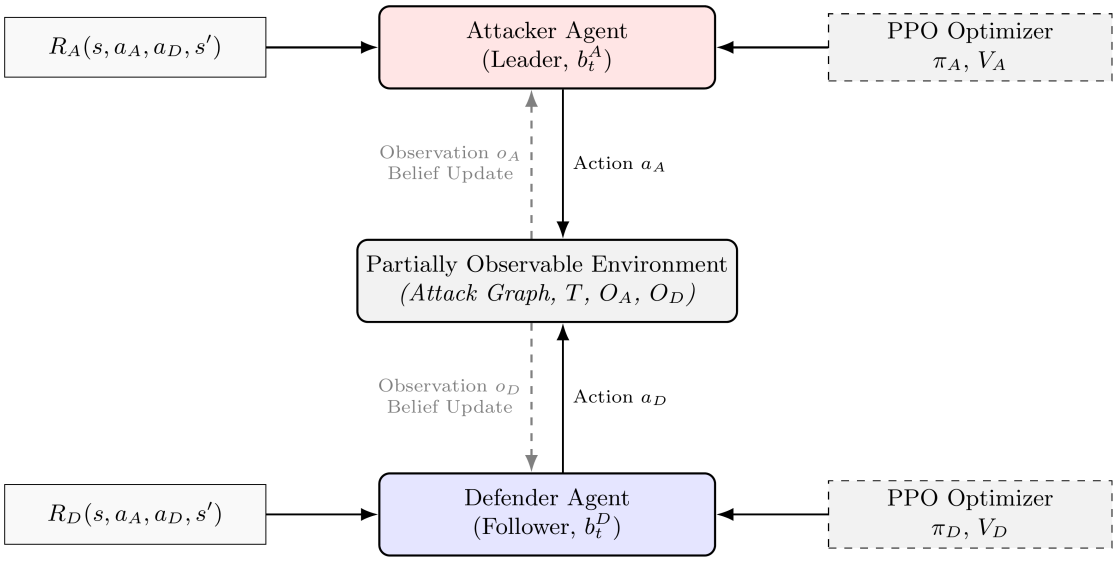


Figure 4.1: High-level architecture of the AI-driven APT modeled framework.

for both players, the state transition dynamics, the observation model with Bayesian belief updates, the reward structures for both players, and the learning algorithm PPO along with exploration strategies. Pseudocode is provided to illustrate the overall training procedure. The following sections are structured to provide a rigorous, formal treatment of each aspect of this AI-driven APT simulation framework.

4.1 Stackelberg Game Formulation and Equilibrium

In this model, the attacker is designated as the Stackelberg leader and the defender as the follower. This means the attacker is assumed to commit to a policy or plan of attack that the defender can observe or anticipate, and the defender then optimizes its strategy in response. Formally, let π_A denote a possibly randomized policy for the attacker and π_D a policy for the defender. The outcome of their interaction, in terms of expected cumulative rewards, can be denoted $U_A(\pi_A, \pi_D)$ for the attacker's utility

and $U_D(\pi_A, \pi_D)$ for the defender's utility. In a Stackelberg setting with the attacker as leader, the attacker seeks a policy π_A^* that maximizes its utility, assuming the defender will respond optimally. The defender's best response to any given attacker policy π_A is defined by:

$$\pi_D^*(\pi_A) = \arg\max_{\pi_D} U_D(\pi_A, \pi_D), \tag{4.3}$$

This means the defender chooses the policy that maximizes its own expected reward given the attacker's fixed strategy π_A. A Stackelberg equilibrium is achieved at the policy pair (π_A^*, π_D^*) when the attacker's policy π_A^* satisfies:

$$\pi_A^* = \arg\max_{\pi_A} U_A\left(\pi_A, \pi_D^*(\pi_A)\right), \tag{4.4}$$

and the defender's policy is $\pi_D^* = \pi_D^*(\pi_A^*)$. In words, the attacker's equilibrium strategy π_A^* is the best possible plan for the attacker, taking into account that the defender will respond optimally, and π_D^* is the defender's optimal reaction to that attack strategy. This equilibrium concept formalizes the leader-follower dynamic. The attacker, as the leader, optimizes its choice while anticipating the defender's rational reaction. Notably, making the attacker the leader tends to favor the attacker's outcome. The leader can secure a payoff at least as high as in a simultaneous move scenario. This represents a challenging, worst-case condition for the defender who is relegated to a reactive role.

For our cybersecurity game, finding an exact Stackelberg equilibrium (with the attacker as leader) is computationally infeasible. The game is a large-scale stochas-

tic game with partial observability and sequential interactions, which defies closed-form solution via traditional game-theoretic methods. Traditional Stackelberg security game algorithms, which are designed for one-shot games with full observability, are not applicable due to the sequential and partially observable nature of this domain.

Therefore, this framework does not attempt to solve the equilibrium analytically. Instead, it approximates the solution via self-play RL. The attacker and defender policies are iteratively improved through simulation. If training converges to stable policies where neither agent finds a significant advantage in unilaterally deviating, this aligns with the notion of an approximate equilibrium under learning dynamics in multi-agent systems. In essence, the learning process searches for a policy pair (π_A, π_D) that is stable in the Stackelberg sense. The attacker has no incentive to deviate from its strategy, given that the defender is playing a best response. Assigning leadership to the attacker induces a first-mover advantage, providing insight into how a well-prepared attacker might perform against an optimally responding defender.

4.2 State Space Representation and Attack Graph Integration

State Definition: The state $s \in S$ represents the security status of the system at a given time step. It is modeled as a set of variables that capture all relevant conditions required for decision-making within the modeled environment. Key components of the state include the attacker's presence and privilege levels on various hosts, the vulnerability status of hosts, the defender's deployed configurations, and any environ-

mental factors that influence the feasibility of attacks. For example, the state may indicate which hosts the attacker has compromised, whether the access is user-level or administrative, which vulnerabilities are present and unpatched, which systems are patched or isolated, and whether intrusion detection sensors are active. Additional conditions may include whether specific prerequisites for exploits are satisfied, such as obtained credentials or active services.

An abstract representation of the state may be written as $s = \{X_1, X_2, \ldots, X_n\}$, where each X_i is a proposition such as "host H is compromised" or "vulnerability V on host H is unpatched." The initial state s_0 reflects the scenario's starting point, typically where the attacker has no access or only a limited foothold, and many vulnerabilities across the network remain unpatched. A goal state is any state in which the attacker has achieved a critical objective, such as obtaining Domain Administrator privileges or accessing sensitive data.

Attack Graph Integration: The state space is constructed based on the attack graph produced by the Attack Dynamics tool. This graph represents a structured set of feasible attacker actions and corresponding state transitions. Each node in the graph corresponds to a partial state condition, such as the attacker having user access on a specific host or administrative access on another. Directed edges represent exploits or actions that change the attacker's state, for instance, by escalating privileges or moving laterally to a connected host. These edges are generated by Attack Dynamics based on the input topology, host configurations, and known vulnerabilities. Each action is tied to specific preconditions and postconditions and is often linked to mitigation techniques referenced in the MITRE ATT&CK knowledge base.

The integration of the attack graph constrains the state space S to include only those states that are logically reachable through valid exploit sequences, thereby excluding states that are not attainable in the modeled scenario. In practice, states are encoded using Boolean or multi-valued variables. For example, a Boolean variable `compromised(H)` may indicate whether a host H has been compromised, and a variable `vuln(H, v)` may indicate whether a vulnerability v on host H is currently unpatched. The preconditions and postconditions associated with each action define how these variables evolve as the attacker progresses. These state variables are typically Boolean (true or false), although some may be multi-valued to capture privilege levels or other categorical information.

This structure enforces realism by grounding state transitions in actual exploit logic, which reduces the combinatorial complexity of the state space and eliminates infeasible paths. It also introduces a natural form of abstraction by excluding states and transitions unsupported by valid attack sequences, which helps focus learning and planning on realistic adversarial behavior.

4.3 Attacker and Defender Action Spaces

Both players have action spaces defined in correspondence with the attack graph and conditioned on the current state. Let $A_A(s)$ be the set of actions available to the attacker in state s, and $A_D(s)$ the set of actions available to the defender in state s. These action sets can vary with the state s since the available actions evolve as the system state changes and certain options become unavailable. Each action set is

formally defined below with respect to the current system state.

4.3.1 Attacker Action Space (A_A)

An attacker's action corresponds to performing an exploit or other tactic at a given step. Let E be the set of all exploit actions enumerated by the attack graph. Each edge $e \in E$ in the graph represents a specific attack step, such as exploiting a particular vulnerability on a particular host. Each exploit e has an associated set of preconditions, denoted by preconds(e), that must hold in the current state in order for the exploit to be applicable. For example, an exploit e might require that Host H is reachable from a host that the attacker already controls, and that vulnerability v is present and unpatched on Host H. If those conditions are not satisfied in the current state s, then the attacker cannot perform e at that time. The attacker's allowed actions in state s are all exploits whose preconditions are satisfied in state s:

$$A_A(s) = \{e \in E \mid \text{preconds}(e) \subseteq s\}, \tag{4.5}$$

where s is interpreted as the set of all propositions that are true in the current state. In this formulation, $A_A(s)$ includes any exploit action e that is logically applicable given the attacker's current foothold and knowledge in state s, thereby excluding actions that are impossible or irrelevant.

For example, if the attacker has not yet compromised any host inside the corporate network, then it cannot directly exploit a server in a protected area, since the required precondition of access to an internal pivot point is not met. Consequently, any exploit

targeting that enclave would not appear in $A_A(s)$ at that time. Similarly, the attacker cannot exploit a vulnerability that has already been patched, nor repeat a previously executed attack step that provides no further gain. Once the state has evolved beyond a certain point, repeating the same exploit would either be invalid or yield no additional privileges, and is therefore excluded from $A_A(s)$. Defining the attacker's actions in this manner, based on the attack graph, serves as a filtering mechanism that restricts choices to only those that are feasible and meaningful. This improves both the realism of the modeled interaction and the efficiency of the learning process.

In addition to exploit actions, the attacker's action space may be extended to include information-gathering or stealth-oriented actions, such as scanning a host for vulnerabilities, conducting network reconnaissance, or attempting to remain undetected by clearing logs or installing persistence mechanisms. These would be modeled as additional actions, potentially represented as special edges in the attack graph whose effect is to reveal information or maintain access without necessarily escalating privileges. Although the current framework focuses on active exploit actions that directly alter the state, it remains straightforward to incorporate such reconnaissance or persistence actions. These would have preconditions and postconditions that modify the attacker's knowledge or ensure continued access to a compromised host, without immediately changing privilege levels.

For instance, an action to scan Host H for vulnerabilities may require Host H to be reachable. If successful, this action does not change the ground truth state, but updates the attacker's belief state in accordance with the partial observability assumption. Similarly, an action to establish persistence on Host H might not extend

access but could preserve the attacker's control of Host H even if the defender attempts to revoke it. This would alter future transition probabilities and increase resilience. Including such actions aligns the simulation with MITRE ATT&CK tactics such as Reconnaissance and Persistence. Although these have not been modeled in the current implementation in favor of learning efficiency, their integration would follow the same formal approach and can be incorporated to support more stealth-oriented or long-term campaigns.

Each attacker action $e \in A_A(s)$, if executed successfully, affects the system state. Let $\text{apply}(e, s)$ denote the resulting state after applying action e in state s. For example, if e corresponds to exploiting CVE-2021-41773 on WebServer1 to gain administrative privileges, and if s indicates that the attacker currently holds user-level access on WebServer1, then $\text{apply}(e, s)$ results in a new state s' in which the attacker has administrative access on that host. This apply operator will be formally used in the next section to define the state transition function.

4.3.2 Defender Action Space (A_D)

The defender's actions correspond to countermeasures taken to slow, disrupt, or mitigate the attacker's progress. These may include patching vulnerabilities, modifying network segmentation policies, deploying intrusion detection sensors, revoking compromised credentials, or isolating machines. The defender's action set is also derived from the attack graph, thereby ensuring that each defensive action is grounded in an identifiable attack path or condition.

Let D denote the set of all defined defensive actions, including patching, seg-

mentation, monitoring, and isolation operations derived from the attack graph. The defender's available actions are defined in state s as all mitigation and monitoring actions whose preconditions are satisfied:

$$A_D(s) = \{d \in D \mid \text{preconds}(d) \subseteq s\} \tag{4.6}$$

Each action $d \in D$ corresponds to a specific defensive measure tied to a condition identified in the attack graph. Each defender action $d \in A_D(s)$, if executed, modifies the system state. Let $\text{apply}(d, s)$ denote the resulting state after executing action d in state s. This notation parallels the attacker transition operator introduced earlier.

For each exploit or vulnerability captured in the attack graph, there typically exists an associated mitigation. The output of Attack Dynamics, along with security knowledge bases such as MITRE ATT&CK and CWE, specifies recommended mitigations including patch deployment, feature deactivation, or enhanced monitoring. Based on these recommendations, $A_D(s)$ is constructed as follows:

- **Patching actions:** For each vulnerability v on each host H represented in the attack graph, the defender may take the action to patch vulnerability v on host H. Executing this action removes or disables the vulnerability on that host, preventing the attacker from using any exploit that depends on v. This sets the state variable $vuln(H, v)$ to false, and renders inapplicable any exploit edge in the attack graph that relies on v on H.

- **Network configuration actions:** For each connection or pivot route in the attack graph, the defender may act to restrict that connectivity. For example,

if an edge in the graph permits lateral movement from Host A to Host B, the defender may harden firewall rules or isolate segments to block this path. The effect is to disable the corresponding edge in the attack graph. This is reflected in the state by a variable indicating that the network path is closed.

- **Detection and monitoring actions:** The defender may deploy monitoring mechanisms to improve detection capabilities. For instance, actions such as deploying an intrusion detection system on Host H or enabling enhanced logging on Server S increase visibility. These actions update the state by modifying variables such as $sensor(H)$, indicating whether a detection mechanism is active on Host H. These measures do not prevent exploits but increase the probability of detection, thereby influencing the defender's observation model O_D.

- **Do-nothing or wait action:** Although the defender may have the option to take no action in a given turn, the nature of APT scenarios often demands timely intervention. Hence, this framework assumes the defender will typically select a meaningful action at each step. Nonetheless, a no-op action may be included in the action space to represent a passive decision in specific scenarios, such as when no urgent countermeasure is required or when the cost of the action outweighs its immediate benefit.

Formally, the defender's available actions $A_D(s)$ at state s include all mitigation and monitoring actions that are applicable in that state. If $d \in A_D$ is a defender action, its effect is to update either the state variables or the structure of the attack graph. For example, executing d as "Patch vulnerability v on Host H" sets $vuln(H, v)$

to false in the next state s', which invalidates any future exploit dependent on that vulnerability. Similarly, a network isolation action may eliminate multiple attacker paths, while a detection action alters the defender's observational probabilities by setting $sensor(H)$ to active.

In summary, the defender's action space $A_D(s)$ comprises all patching and mitigation actions for unpatched vulnerabilities and open connections, as well as sensor deployment actions where visibility can be improved. If a vulnerability is already patched, the corresponding patching action is no longer available. If a sensor is already deployed or a path has been closed, those actions do not appear in $A_D(s)$. By constructing $A_D(s)$ from the attack graph and its associated vulnerabilities, the defender's actions are ensured to be grounded in the scenario rather than abstract or generic. Figure 4.2 illustrates an example of the defender's patching strategy in action. After the attacker compromises Host H1, the defender may choose between patching a neighboring host (e.g., H3) or randomly selecting a non-neighboring host (e.g., H5), depending on the selected mitigation strategy.

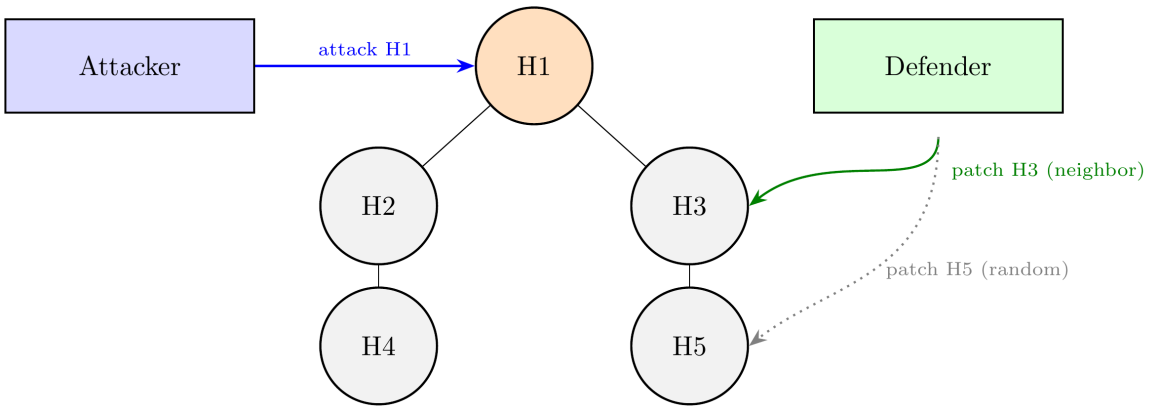


Figure 4.2: Defender patching strategies.

Both $A_A(s)$ and $A_D(s)$ are dynamic functions of the current state. As the modeled

interaction and actions are executed, available options diminish. Once a vulnerability is patched or a host is compromised, the corresponding actions are removed from the respective sets. This causes the game to become more constrained over time. Early in the simulation, the attacker has many potential actions and the defender has a broad set of possible mitigations. As the interaction unfolds, each player's strategic space narrows. This behavior reflects the operational reality of APTs and reduces the effective decision space, which improves the tractability of learning and planning processes.

4.4 State Transition Dynamics

Having defined the states and actions, the state transition function T is now formalized to describe how the environment evolves when the attacker and defender take actions. The game proceeds in discrete turns. At each time step t, the attacker selects an action $a_A^t \in A_A(s_t)$ and the defender selects an action $a_D^t \in A_D(s_t)$, based on the current state s_t, or based on their respective observations and belief states due to partial observability. These actions are taken concurrently, and the environment transitions to a new state s_{t+1} according to the function T:

$$T : S \times A_A \times A_D \to \Delta(S) \tag{4.7}$$

where $\Delta(S)$ denotes the set of all probability distributions over the state space S. This function defines a stochastic transition model: for a given state and joint action,

it outputs a probability distribution over possible next states. The transition function forms the basis for belief updates under partial observability. While the Stackelberg formulation suggests a leader-follower sequence, the modeled interaction adopts concurrent decision-making to reflect realistic interactions. Action resolution is handled as follows:

- Both agents select their actions simultaneously at the start of each time step, without knowledge of the opponent's immediate move.
- The outcome of the step depends on both actions. If the defender's action directly interferes with the attacker's chosen action, the effect of the attacker's action is nullified for that turn.

Stochasticity is introduced in the outcome of attacker actions. Each exploit action has an inherent probability of success, even if unopposed. Let e be the attacker's exploit and d the defender's action in a given turn. A success probability $p_e \in [0, 1]$ is assigned to each exploit e, which may be derived from empirical sources such as the CVSS Exploitability Score or the Exploit Prediction Scoring System. This value reflects the inherent difficulty or reliability of the exploit. For instance, a well-understood exploit may have $p_e = 0.9$, while a less stable one may have $p_e = 0.4$.

Given state s, attacker action e, and defender action d, the transition outcome depends on whether d nullifies e and on the value of p_e. Let $s_{\text{new}} = \text{apply}(e, s)$ denote the state resulting from a successful exploit, and let $\text{apply}(d, s)$ denote the state resulting from applying the defender's action. Two distinct cases arise:

- If the defender's action d blocks the exploit e, for example by patching the relevant vulnerability or severing network connectivity, then the exploit fails with probability 1. The resulting state s' will reflect only the changes caused by the defender's action.

- If the defender's action does not interfere with the preconditions or outcome of e, the exploit proceeds with its intrinsic probability p_e. In this case:

$$P(s' = \text{apply}(e, \text{apply}(d, s))) = p_e \tag{4.8}$$

$$P(s' = \text{apply}(d, s)) = 1 - p_e \tag{4.9}$$

  That is, with probability p_e, the exploit succeeds and its effects are applied in addition to the defender's action. With probability $1 - p_e$, the exploit fails and only the defender's changes are reflected.

As an example, consider an attacker exploiting a host H with a base success probability of 0.5. If the defender simultaneously patches a different host $H2$, then the exploit e will either succeed with probability 0.5 and yield a state where H is compromised and $H2$ is patched, or it will fail with probability 0.5 and yield a state where only $H2$ is patched. However, if the defender instead patches host H itself, the exploit is neutralized, and the next state will reflect only that H is now patched.

This model captures the concurrent and probabilistic interaction between attacker and defender. It reflects the realistic dynamics of cyber operations, where a timely defensive intervention may block an attack just before execution. Over multiple time

steps, this stochastic transition model gives rise to distributions over state trajectories. These trajectories are explored by RL agents that aim to maximize cumulative reward: the attacker seeks to gain privileges and persistence, while the defender attempts to prevent compromises and minimize impact.

4.5 Observation Model and Belief Updates under Partial Observability

The state transition function T, introduced in the previous section, defines how the environment evolves as a result of both attacker and defender actions. It also forms the foundation for probabilistic belief updates under partial observability, as the agents must infer hidden aspects of the system state from indirect signals. However, in realistic adversarial scenarios, agents do not have direct access to the true underlying state. Instead, they receive partial information in the form of observations.

Partial observability in this framework implies that neither agent directly observes the system's full true state, s_t. Instead, after each turn, each agent receives an observation that provides some information about what happened or about the current state, and they use this information to update their belief about s_t. Observation models are defined for the attacker and defender, which are inherently asymmetric due to their different perspectives and information sources. Let O_A and O_D denote the observation models or probability distributions over observations for the attacker and defender, respectively. Formally, the attacker's observation model is defined as a conditional distribution $O_A(s, a_A, s')$, which represents the probability of receiving a

particular observation o_A given that the system transitioned from state s to state s' due to attacker action a_A. Similarly, the defender's observation model is denoted as $O_D(s, a_D, s')$. These models capture the fact that observations are generated based on transitions rather than static states, and depend on the dynamics of both agents' actions.

4.5.1 Attacker's Observation O_A

The attacker is generally aware of the outcome of their own action. It is assumed that after executing an exploit, the attacker knows whether it succeeded or failed. If the attacker attempts to compromise a system, they typically can determine if access was gained, for instance by obtaining a shell or receiving confirmation, or if the attempt failed due to lack of response or an error. However, the attacker may not know the cause of failure. The vulnerability may not have been present, the exploit may have failed probabilistically, or the defender may have patched the system in time. From the attacker's viewpoint, only the result is observed.

The attacker's observation o_A at time t is defined as a discrete signal such as success or failure of the attempted action. More detailed outcomes can be incorporated if necessary, such as inference based on loss of access or unexpected behavior, but the default assumption is that the attacker only observes success or failure. Detection by the defender is assumed to be hidden unless it has a visible effect on the attacker's operations.

Formally, if the attacker's action e is executed at state s and results in new state s', the attacker's observation o_A is defined as:

$$o_A = \begin{cases} \text{success}, & \text{if the attacker gained new access or made progress in } s' \\ \text{failure}, & \text{if the attacker gained nothing new in } s' \end{cases} \tag{4.10}$$

The attacker cannot observe defender actions unless they directly alter the attacker's outcome or state. Defender actions and detection mechanisms remain unobserved unless they lead to observable disruption.

4.5.2 Defender's Observation O_D

The defender receives observations from intrusion detection systems, logs, alerts, and other passive monitoring mechanisms. The defender's observation is probabilistic and is tied to the attacker's activity. Each attacker action e has a detectability probability $\delta(e)$, which represents the likelihood that the action produces an observable signal. This probability depends on the state's monitoring coverage. For instance, if host-level sensors are deployed, the probability of detecting an exploit increases.

When the attacker executes action e, the defender may receive:

- An alert describing the action's nature or location with probability $\delta(e)$
- No alert with probability $1 - \delta(e)$

The absence of an alert is also informative. It may lead the defender to lower the probability of compromise or infer stealthy attacker behavior. If no actions are taken or a stealthy method is used, the defender receives no signal beyond routine noise.

The observation model is described by the conditional distribution $P(o_D \mid s, a_A = e, a_D = d, s')$. For simplicity, observations are primarily generated by attacker activity, while outcomes of defender actions such as active scanning are modeled separately.

This conditional probability distribution can be denoted compactly as $O_D(s, a_D, s')$, where $o_D \sim O_D(s, a_D, s')$. This formulation models attacker detectability as a probabilistic observation emission conditioned on both the attacker's action and the defender's sensor configuration. Formally, the defender's observation model is defined as a conditional probability distribution over possible alerts given the following system dynamics

$$P(o_D \mid s, a_A, a_D, s') = \begin{cases} \delta(e), & \text{if } o_D \text{ corresponds to an alert caused by } a_A = e \\ 1 - \delta(e), & \text{if } o_D = \varnothing \end{cases} \tag{4.11}$$

where $\delta(e) \in [0, 1]$ denotes the detectability of attacker action e, which depends on the defender's monitoring configuration in state s. This distribution specifies the likelihood of receiving an observation o_D, such as an intrusion detection alert, conditioned on the transition from state s to s' under actions a_A and a_D.

For example, if attacker action e was executed:

$$P\left(o_D = \text{alert related to } e\right) = \delta(e), \quad P\left(o_D = \varnothing\right) = 1 - \delta(e) \tag{4.12}$$

4.5.3 Belief Update via Bayesian Inference

Due to partial observability, agents maintain beliefs over possible states. Let b_t^A and b_t^D denote the attacker's and defender's belief distributions at time t, respectively. Each agent uses observations to update its belief using Bayes' rule. Formally, the belief state for agent $i \in \{A, D\}$ at time t is a probability distribution over the true state space S, denoted $b_t^i(s) = P(s_t = s \mid h_t^i)$, where h_t^i represents the agent's action-observation history up to that point. This distribution encodes the agent's subjective uncertainty about the system's true state and is updated as new information becomes available.

The attacker's initial uncertainties concern unknown vulnerabilities and defensive configurations. For instance, the attacker may maintain a prior belief $P(X)$ that a host H has an unpatched vulnerability v. If an exploit targeting (H, v) fails, the attacker updates this belief using Bayes' rule as

$$P(X \mid o_A = \text{failure}) = \frac{P(o_A = \text{fail} \mid X) \cdot P(X)}{P(o_A = \text{fail} \mid X) \cdot P(X) + P(o_A = \text{fail} \mid \neg X) \cdot P(\neg X)} \quad (4.13)$$

where $P(o_A = \text{fail} \mid X) = 1 - p_e$ and $P(o_A = \text{fail} \mid \neg X) \approx 1$, assuming that the exploit has negligible success probability if the vulnerability is absent. Conversely, if the exploit succeeds, the attacker updates its belief as

$$P(X \mid o_A = \text{success}) = \frac{P(o_A = \text{success} \mid X) \cdot P(X)}{P(o_A = \text{success} \mid X) \cdot P(X) + P(o_A = \text{success} \mid \neg X) \cdot P(\neg X)}, \quad (4.14)$$

where $P(o_A = \text{success} \mid X) = p_e$ and $P(o_A = \text{success} \mid \neg X) \approx 0$. These expressions explicitly consider both the presence and absence of vulnerabilities and update the

attacker's belief accordingly. Repeated failures may lead to the inference that a vulnerability is either patched or nonexistent, whereas successful exploits reinforce the belief in its presence.

The defender's belief b_t^D reflects uncertainty about the attacker's presence or movement across the network. Upon receiving an alert o_D, the defender updates beliefs for relevant propositions, such as Y: "attacker is present on host H," using

$$P\left(Y \mid o_D = \text{alert on } H\right) = \frac{P\left(o_D = \text{alert on } H \mid Y\right) \cdot P(Y)}{P\left(o_D = \text{alert on } H\right)} \tag{4.15}$$

In the absence of alerts, the defender may reduce the likelihood of compromise, though it can never rule it out entirely. The defender also reasons along the structure of the attack graph to anticipate follow-up actions and to reposition defenses accordingly. This dynamic process enables adaptive defense under uncertainty.

For both agents, the general belief update rule, given a prior belief b_t^i, an action a_t^i, and an observation o_{t+1}^i, can be written as

$$b_{t+1}^i(s') = \frac{P(o_{t+1}^i \mid s') \sum_{s \in S} T(s, a_t^A, a_t^D, s') \cdot b_t^i(s)}{P(o_{t+1}^i)}, \tag{4.16}$$

where $T(s, a^A, a^D, s')$ is the state transition function and $P(o_{t+1}^i \mid s')$ is the observation likelihood. This provides a general formalism applicable when state spaces are large or continuous. In practice, exact belief updates may be intractable due to combinatorial complexity; therefore, approximations using particle filtering, neural state encoding, or RL may be used during training.

In summary, both agents use observations to incrementally refine their understanding of the environment. The attacker learns about vulnerabilities and defender configurations through interaction, while the defender uses passive monitoring to localize the attacker and respond effectively. These belief updates, whether exact or approximate, are essential for making strategic decisions under uncertainty in adversarial settings.

4.6 Reward and Cost Structure

A well-defined reward structure is crucial for guiding the attacker and defender agents toward their objectives during RL. In this game, each agent's reward function $R_i(s, a_A, a_D, s')$ quantifies the immediate payoff or cost to that agent when the system transitions from state s to state s' after the chosen actions a_A and a_D. R_A and R_D are constructed to reflect the goals of each side in the context of an APT, while also incorporating domain knowledge to aid effective learning. The rewards are designed in a general-sum fashion, allowing us to penalize or reward certain outcomes for each side independently. For example, both players might incur a cost for each time step to reflect the passage of time or resources used.

4.6.1 Attacker's Reward R_A

The attacker's primary goal is to reach high value targets in the network. These could be specific critical assets such as a database server with sensitive data, or a global condition such as obtaining domain administrator privileges. To capture this,

the reward function provides a strong positive payoff for achieving these end goals. Formally, let $G \subseteq S$ be the set of goal states for the attacker. If a transition leads the system into a state $s' \in G$, the attacker should receive a large reward. Denote this reward as R_A^{goal}, a large positive constant such as $+100$ to signify a successful mission.

However, if only the final success is rewarded, the attacker's learning process will suffer from extremely sparse rewards. To address this, intermediate rewards are included for progress. A function $I(s, s')$ is designed to measure the incremental progress the attacker made from state s to s'. For example, $I(s, s')$ could be defined as the number of new machines compromised or the number of privilege levels escalated that were not previously attained. Each time the attacker gains access to a new host or achieves a higher privilege on a host not previously owned, that is considered valuable progress. A smaller positive reward Δ is assigned for each such achievement. Additionally, a small stepwise penalty is imposed to encourage the attacker to be efficient. This can represent the cost in time or resources for each action, or the risk of being detected, which increases as the campaign continues. This can be modeled as a constant negative reward per action taken.

We can formulate the attacker's immediate reward as a combination of these terms. One possible formalization is

$$R_A(s, a_A, a_D, s') = R_A^{goal} \cdot \mathbb{I}(s' \in G)$$
$$+ \Delta \cdot \big(\#\text{ new hosts or privileges gained going from } s \text{ to } s'\big)$$
$$- C_A(a_A) \tag{4.17}$$

where $\mathbb{I}(s' \in G)$ is an indicator function that equals one if s' is a goal state, and #new hosts or privileges gained counts how many unique compromise milestones were obtained by the attacker in this transition. Δ is the reward for each such milestone. $C_A(a_A)$ represents a cost for the attacker's action a_A. Commonly, $C_A(a)$ is set to a small constant such as 0.1 to model a time or resource penalty per action and encourage efficiency. $C_A(a)$ may also be varied across different types of actions to discourage particularly noisy or risky behavior. For instance, if a_A is a known noisy malware that is likely to trigger alarms, an extra penalty can be applied to reflect the attacker's caution. In general, a uniform small penalty per action works as a proxy for the passage of time or risk accumulation.

We also account for negative consequences for the attacker when the defender successfully counters them. If the defender detects or blocks the attacker's action, that is considered a setback. An immediate penalty is imposed when the attacker's action is detected or thwarted. For example, if the defender's observation o_D indicates the action was detected, or the state transition shows that the action was stopped due to the defender's simultaneous move, a fixed penalty such as -5 is subtracted. This models the idea that being discovered or failing due to interference harms the attacker's

campaign, possibly forcing them to waste time, change tactics, or risk exposure. This reward design balances short-term tactical incentives, such as stealth and efficiency, with long-term strategic goals, such as the successful compromise of critical assets.

An illustrative attacker reward scheme might be:

- $+100$ if the attacker reaches a high value target state.
- $+1$ for each new host compromised or significant privilege gained.
- -1 if the attacker's action was detected by the defender or blocked outright.
- -0.1 per action as a time or resource penalty.

For example, if the attacker successfully compromises a new server, earning $+1$, but triggers an alert at the same time, incurring -1, the net reward might be zero. Adding the small penalty for taking an action, the net reward becomes slightly negative. The large $+100$ is only awarded upon achieving the final goal, ensuring that the agent does not endlessly optimize for small gains. The intermediate reward Δ alleviates reward sparsity and encourages exploration, while remaining small enough to not distract from the long term objective. The detection penalty is kept lower than the progress reward to favor stealth. The step penalty is minimal to promote action efficiency.

4.6.2 Defender's Reward R_D

The defender's goal is to prevent the attacker from reaching the defined goals and to detect or block them during the campaign. The reward function R_D is constructed to reward successful defense and penalize breaches, while also incorporating the cost

of defensive actions. Although a strictly zero-sum setting would use $R_D = -R_A$, a general-sum formulation that includes defender-specific tradeoffs that may not directly correspond to the attacker's outcomes is preferred.

Key components of R_D include:

- A large positive terminal reward if the attacker fails to reach a goal by the end of the episode. For example, +100 is awarded upon successful defense.
- Intermediate rewards for effective actions. A reward of +1 is given for thwarting an attacker's action, and +0.5 for detecting the attacker.
- A large penalty of −100 if the attacker reaches a goal state.
- Action cost penalty $C_D(a_D)$ for each defensive action, such as −0.2 for patching or −0.5 for deploying an intrusion detection sensor.

The combined defender reward can be formalized as:

$$\begin{aligned} R_D(s, a_A, a_D, s') = &+ 100 \cdot \mathbb{I}\big(s' \notin G \text{ and attacker never reaches } G \text{ by episode end}\big) \\ &+ 1 \cdot \mathbb{I}(\text{thwarted } a_A) \\ &+ 0.5 \cdot \mathbb{I}(\text{detected attacker}) \\ &- C_D(a_D) \\ &- 100 \cdot \mathbb{I}(s' \in G) \end{aligned} \tag{4.18}$$

In this formula, the reward is assigned at each step based on what happened in

the transition. If the attacker eventually fails, a terminal reward of +100 is applied. If the attacker's action was blocked without progress, the defender receives +1. If the attacker was detected, a smaller reward of +0.5 is granted. Costs are subtracted based on the specific action taken. If the attacker reaches a goal, −100 is applied. For example, if the defender patches a host and blocks an exploit, earning +1, and the patch costs −0.2, the net reward for that step is +0.8.

This structure encourages intelligent and selective action. Excessive patching or overly aggressive defense incurs cost, while doing nothing can result in heavy penalties. The defender is thus incentivized to act wisely and efficiently. The goal outcome captures the zero-sum part of the interaction, but the rest allows for nuanced modeling of security operations and operational costs.

Furthermore, this reward shaping supports convergence in RL. Intermediate feedback improves exploration and reduces the chance of unproductive behavior. Domain expertise can also be incorporated, such as penalizing unnecessary patching or rewarding realistic progress signals.

If desired, the defender or attacker reward may also be shaped using structural features from the attack graph. For example, the attacker may be rewarded for reducing the graph distance to a goal state, and the defender for increasing it by invalidating paths. While not implemented explicitly, this is compatible with the existing model and could improve planning quality.

4.7 RL Approach

Given the game model described above, which involves a Stackelberg-POSG with large state and action spaces, this framework employs RL techniques to compute effective policies for the attacker and defender agents. RL is used to approximate equilibrium policies via self-play under partial observability and strategic interaction. Classical game-theoretic solution methods are not feasible due to the computational complexity of solving such high-dimensional, partially observable games. As a result, a simulation-based learning approach is adopted to approximate optimal behaviors.

This work exclusively utilizes PPO, a state-of-the-art policy gradient algorithm known for its stability, sample efficiency, and ability to operate effectively in environments characterized by long-horizon planning, adversarial dynamics, and partial observability. The PPO algorithm is adapted to the multi-agent setting of the game, with tailored reward structures and policy updates for both attacker and defender agents. Furthermore, strategies are incorporated to balance exploration and exploitation during training, given the large and sparse space of possible attack-defense interactions.

PPO is particularly suitable for addressing the Stackelberg leader-follower dynamics inherent in this game model due to its gradual policy updates and stable convergence properties, effectively handling strategic interactions and nonstationarity common in adversarial cybersecurity environments. Additionally, the numerical values selected for rewards, such as large positive rewards for critical successes and smaller incremental penalties for detection or resource usage, were empirically tuned

based on typical cybersecurity scenarios to balance learning efficiency, exploration, and realistic attacker and defender behaviors.

Although other RL algorithms could also be applicable in this context, such as Deep Q-Networks (DQN), Asynchronous Advantage Actor Critic, or Soft Actor Critic, they were not implemented in this study. DQNs, for instance, are effective in fully observable discrete domains but are less suited for learning robust policies under partial observability and long decision horizons. Asynchronous Advantage Actor Critic and Soft Actor Critic offer promising mechanisms for parallelized training and entropy-driven exploration, respectively, yet their inclusion would significantly increase complexity without necessarily improving performance for the problem structure addressed here.

PPO was selected based on its demonstrated ability to learn stable and generalizable policies in high dimensional and adversarial environments. While the proposed framework is extensible and can accommodate alternative algorithms in future work, this framework focuses on PPO to maintain clarity, reduce computational overhead, and emphasize interpretability and training stability in a complex cyber threat modeling scenario. Supporting techniques such as opponent modeling, alternating policy updates, or the integration of centralized critics may be incorporated independently of the chosen learning algorithm to enhance convergence in adversarial multi agent training environments.

4.7.1 Training Procedure with PPO

This framework uses PPO to train stochastic policies $\pi_A(a \mid s)$ and $\pi_D(d \mid s)$ for the attacker and defender, respectively. In PPO and other actor critic methods, two sets of networks are typically maintained: actor networks and critic networks. The actor network for the attacker outputs a probability distribution over attacker actions given the state or the attacker's observation or belief, and similarly for the defender. The critic network estimates the value of a state or state action for the agent, which is used to compute advantage estimates. In a multi agent context, one can use a centralized critic that takes into account the global state and both actions or separate critics for each agent. For decentralization, separate critics $V_A(s)$ and $V_D(s)$ are used for the attacker and defender that estimate the value of the state from each agent's perspective.

Both the attacker and defender policy networks are initialized with initial weights, for example small random weights corresponding to near uniform random policies. Similarly, their value networks are initialized. During training, many episodes of the simulation are executed. At each time step within an episode, the agents sample their actions from their policy distributions: $a_A^t \sim \pi_A(\cdot \mid s_t)$ and $a_D^t \sim \pi_D(\cdot \mid s_t)$. The environment is simulated for one step with these actions, yielding next state s_{t+1} and rewards r_t^A, r_t^D. The tuple $(s_t, a_A^t, a_D^t, r_t^A, r_t^D)$ and policy probabilities or logits for a_A^t and a_D^t are recorded for training.

After collecting a batch of experience, a PPO update is performed, involving several steps:

1. **Compute advantages:** For each time step t in the batch and for each agent, compute the advantage A_t^A and A_t^D. The advantage represents how much better or worse the action performed compared to the critic's baseline expectation. For example, a one step advantage is estimated as $A_t^A = r_t^A + \gamma V_A(s_{t+1}) - V_A(s_t)$. In practice, Generalized Advantage Estimation is used for smoother estimates over multiple steps.

2. **Update critic:** The collected state, reward, and next state pairs are used to compute target values, and then the value network is updated by minimizing the mean squared error between $V(s_t)$ and these targets for each agent.

3. **Update actor:** The PPO clipped objective updates the policy. For the attacker, the probability ratio is computed as

$$\rho_t = \frac{\pi_A^{\text{new}}(a_A^t \mid s_t)}{\pi_A^{\text{old}}(a_A^t \mid s_t)}. \tag{4.19}$$

   The surrogate objective for the attacker is defined as

$$L_{\text{PPO}}^A = \mathbb{E}_t\Big[\min\big(\rho_t A_t^A,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t^A\big)\Big] \tag{4.20}$$

   where ϵ limits policy change magnitudes. Gradients of this objective with respect to the attacker's policy parameters are computed and used for updates. A similar objective is constructed for the defender's policy.

These updates are repeated for several epochs over the sampled batch. The episode or batch data is discarded or partially reused for Generalized Advantage Estimation,

and the process continues with new experience. This iterative PPO training process is summarized in Figure 4.3.

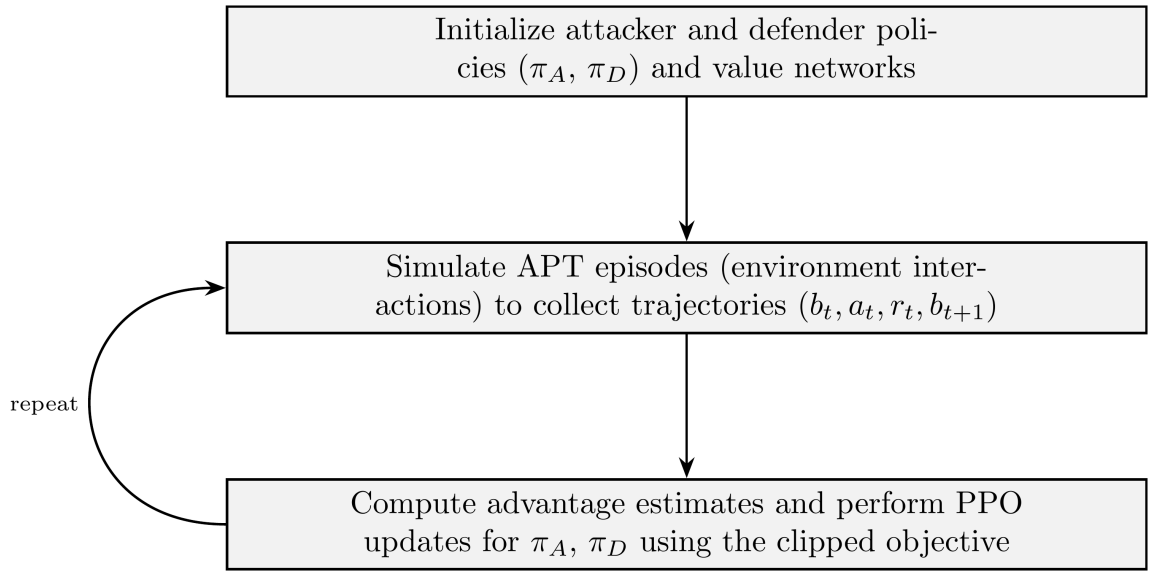


Figure 4.3: PPO training loop.

4.7.2 PPO Hyperparameter Configuration

The PPO algorithm used in this framework relies on a set of carefully selected hyperparameters, tuned to balance exploration, stability, and sample efficiency in a high-dimensional, partially observable, and adversarial environment. The configuration is based on empirical best practices established in prior PPO literature and adapted to the dynamics of the attacker-defender interaction modeled in this game. The hyperparameters are detailed below:

- **Discount factor** ($\gamma = 0.99$): Promotes long-term planning by assigning high weight to future rewards. Particularly relevant for sequential, multi-step APT scenarios where delayed payoff is common.

- **Clipping parameter** ($\epsilon = 0.2$): Prevents large, destabilizing policy updates by bounding the probability ratio during optimization. This stabilizes learning and prevents catastrophic forgetting.

- **Entropy regularization coefficient** ($\beta = 0.01$): Encourages exploratory behavior in the early training phases by discouraging premature convergence to deterministic policies. Gradually annealed over time.

- **Learning rate** (3×10^{-4}): Applied to both actor and critic networks using the Adam optimizer. Chosen to provide a good trade-off between convergence speed and gradient stability.

- **Generalized Advantage Estimation (GAE) parameter** ($\lambda = 0.95$): Smooths advantage estimates over time and reduces variance, leading to more stable updates.

- **Batch size** (2048 transitions): Defines the number of samples collected per PPO update. Larger batches help stabilize advantage estimation and reduce gradient noise in adversarial settings.

- **Epochs per PPO update** (10): Number of passes over the batch during optimization. Multiple epochs improve sample efficiency but risk overfitting if excessive.

- **Optimizer**: Adam is used for both actor and critic networks, due to its effectiveness in high-dimensional nonlinear optimization.

These hyperparameter choices were guided by empirical stability across adversarial RL environments and were not exhaustively tuned for each scenario. In future extensions, these values can be fine-tuned per scenario or dynamically adapted during training using meta-learning techniques or Bayesian optimization. This fixed configuration, however, ensures reproducibility and provides a robust baseline for the game-theoretic training procedure described in this work.

4.7.3 Exploration Strategy

Exploration is critical due to the complex environment. Several mechanisms ensure adequate exploration:

- **Entropy Regularization:** An entropy bonus maintains a broad action distribution, encouraging randomness throughout training. Initially, a relatively large entropy coefficient promotes diverse exploration, gradually reduced as training progresses.

- **Randomized Initial Conditions:** Elements of the initial state, such as unpatched vulnerabilities or attacker footholds, are randomized at the beginning of each training episode, improving strategy generalization.

- **Reward Shaping for Exploration:** Intermediate rewards guide learning, providing incremental feedback rather than delayed rewards. The attacker receives small rewards for compromising individual hosts, guiding toward effective paths. The defender is rewarded for detecting or blocking intermediate steps.

These exploration strategies promote the emergence of novel attack sequences and diverse defensive responses, enabling both agents to discover effective behaviors beyond deterministic patterns. High initial entropy coefficients facilitate broader exploration of the attack graph, improving the generalization capability of the learned policies. Collectively, these learning mechanisms support stable convergence in high-dimensional, adversarial environments. The resulting attacker and defender policies constitute an empirical approximation of a Stackelberg equilibrium under RL dynamics, wherein the attacker exploits its leadership role and the defender adapts optimally under uncertainty throughout the simulated interaction.

4.8 Training Procedure and Pseudocode

The training procedure for the APT game framework integrates the environment model and the RL algorithm previously described. The learning process proceeds in episodes, each modeling a complete APT attack scenario. An episode terminates if the attacker reaches a goal state, the defender successfully contains the campaign, or the predefined time limit is reached. **Algorithm 4.1** outlines the procedure using PPO, incorporating environment transitions, Bayesian belief updates, and policy refinement through trajectory based advantage estimation.

Throughout the training process, performance is monitored using several indicators, including the moving average of rewards, success and failure rates for both agents, and loss statistics reflecting policy stability. A policy is considered to have converged when metrics such as reward trends and advantage variance stabilize, indicating di-

minishing learning returns and behavioral saturation.

Algorithm 1: PPO-Based Training under Partial Observability for APT Simulation

Initialize:

Attacker and defender policy networks $\pi_A(\cdot \mid b_t^A; \theta_A)$, $\pi_D(\cdot \mid b_t^D; \theta_D)$

Value networks $V_A(b_t^A; \psi_A)$, $V_D(b_t^D; \psi_D)$

Belief states b_0^A, b_0^D, environment state s_0

Hyperparameters: learning rate α, clipping range ϵ, entropy coefficient β, epochs K

for *each training episode* **do**

// --- Phase 1: Interaction with Environment ---

for $t = 0$ *to* T_{max} **do**

Sample attacker action: $a_t^A \sim \pi_A(\cdot \mid b_t^A; \theta_A)$

Sample defender action: $a_t^D \sim \pi_D(\cdot \mid b_t^D; \theta_D)$

Transition to new state: $s_{t+1} \sim T(s_t, a_t^A, a_t^D)$

Generate observations:

$o_t^A \sim \mathcal{O}_A(o_t^A \mid s_{t+1}, a_t^A)$

$o_t^D \sim \mathcal{O}_D(o_t^D \mid s_t, a_t^A, a_t^D, s_{t+1})$

Update beliefs using Bayesian inference:

$b_{t+1}^A(s') = \eta_A \cdot O_A(o_t^A \mid s', a_t^A) \cdot \sum_{s \in \mathcal{S}} T(s' \mid s, a_t^A, a_t^D) \cdot b_t^A(s)$

$b_{t+1}^D(s') = \eta_D \cdot O_D(o_t^D \mid s', a_t^D) \cdot \sum_{s \in \mathcal{S}} T(s' \mid s, a_t^A, a_t^D) \cdot b_t^D(s)$

Compute rewards: $r_t^A = R_A(s_t, a_t^A, a_t^D)$, $r_t^D = R_D(s_t, a_t^A, a_t^D)$

Store trajectory data: $(s_t, b_t^A, a_t^A, r_t^A, \log \pi_A(a_t^A \mid b_t^A))$ and similarly for defender

if *attacker is detected* ***or*** $t \geq T_{max}$ **then**

break

// --- Phase 2: Policy Optimization ---

Compute estimated returns $\hat{R}_t^A$, $\hat{R}_t^D$ using discounted rewards or GAE

Compute advantages: $A_t^A = \hat{R}_t^A - V_A(b_t^A; \psi_A)$, $A_t^D = \hat{R}_t^D - V_D(b_t^D; \psi_D)$

for *each optimization epoch* $k = 1, 2, ..., K$ **do**

Compute PPO surrogate loss with entropy:

$$\mathcal{L}_A^{\text{CLIP+ENT}}(\theta_A) = \mathbb{E}_t \left[\min \left(r_t^A A_t^A, \text{clip}(r_t^A, 1-\epsilon, 1+\epsilon) A_t^A \right) + \beta \mathcal{H}[\pi_A(\cdot \mid b_t^A)] \right]$$

$$r_t^A = \frac{\pi_A(a_t^A \mid b_t^A)}{\pi_A^{\text{old}}(a_t^A \mid b_t^A)}$$

Update attacker parameters:

$\theta_A \leftarrow \theta_A + \alpha \nabla_{\theta_A} \mathcal{L}_A^{\text{CLIP+ENT}}$

Repeat surrogate loss computation and parameter update for defender θ_D

Update value networks via MSE loss:

$$\psi_A \leftarrow \arg\min_{\psi_A} \left(V_A(b_t^A; \psi_A) - \hat{R}_t^A \right)^2$$

$$\psi_D \leftarrow \arg\min_{\psi_D} \left(V_D(b_t^D; \psi_D) - \hat{R}_t^D \right)^2$$

This RL methodology, grounded in formal game-theoretic modeling and informed by reward shaping and exploratory regularization, enables the automated learning of strategic behavior in realistic APT simulations. The environment is instantiated using Attack Dynamics to produce feasible multi-step attack pathways based on known vulnerabilities, host configurations, and logical prerequisites. The attacker and defender interact through these paths in accordance with real-world tactics, while partial observability is addressed through Bayesian belief tracking and observation modeling.

Moreover, the use of MITRE ATT&CK mappings links observed behavior to standard adversarial tactics and mitigations, enabling a rigorous mapping between learned policies and enterprise relevant threat behaviors. Upon convergence, these learned attacker and defender policies are evaluated against scripted baselines and varied scenarios to assess both strategic robustness and alignment with equilibrium behavior, as presented in the subsequent evaluation chapter.

4.9 Discussion: Complexity, Assumptions, and Scalability

The automated APT decision-making framework described in this chapter is powerful but also intricate in structure. This section discusses the computational complexity, core modeling assumptions, and scalability considerations associated with the proposed methodology.

4.9.1 State Space Complexity

The combination of enterprise network configurations, attacker presence, and defender policies yields a state space of considerable magnitude. Even after pruning unreachable states from the attack graph, the number of feasible states increases combinatorially with the number of hosts, vulnerabilities, and credentials in the environment. This combinatorial explosion, often referred to as the curse of dimensionality, necessitates the adoption of function approximation using deep neural networks and model-based RL techniques in place of exact dynamic programming. Partial observability further enlarges the effective state space, as belief distributions over states are maintained in place of direct state information.

In practice, the framework relies on the generalization capabilities of neural networks to process high-dimensional state representations. The networks capture patterns, such as similarities among host configurations, and learn corresponding defensive responses, without requiring enumeration of every possible state. Furthermore, the attack graph provides a factored representation of state features. For instance, states may contain propositions such as "host H compromised", enabling the network to infer and generalize on a per-feature basis. Despite this abstraction, training duration increases with scenario size. Scenarios comprising several dozen hosts and vulnerabilities remain tractable in the current implementation, although scaling to hundreds may necessitate more advanced state abstraction or distributed training strategies.

4.9.2 Action Space Complexity

The action spaces for both the attacker and the defender can be extensive. The attacker may simultaneously have numerous exploits applicable across reachable hosts with multiple vulnerabilities, and the defender may be able to choose from various patching and mitigation actions. The framework utilizes the attack graph to eliminate infeasible actions. For example, exploits that require unreachable hosts or patched vulnerabilities are excluded, thereby reducing the effective branching factor at each decision point.

Hierarchical abstraction offers another means to manage action space complexity. For instance, a sequence of exploits could be abstracted into a single higher-level action such as "perform lateral movement into subnet X." Although macro-actions are not implemented in this framework, they offer a potential approach to achieve scalability through hierarchical modeling. Similarly, action abstraction can consolidate functionally equivalent low-level actions into generalized action types. For example, in an enterprise setting, the action of patching any workstation may be represented as a generic action parameterized by host identity. This framework achieves a similar effect implicitly by encoding the host as an input feature in the neural policy and value networks. Explicit parameterization would further support generalization and reduce the effective dimensionality of the action space.

4.9.3 Computation and Convergence

The process of training attacker and defender agents through self-play is computationally demanding. Each episode requires multiple simulation steps and corresponding policy updates via neural networks. Although convergence guarantees are absent in general-sum games, empirical results often show that policies stabilize following initial oscillations. The framework assumes that both agents are afforded sufficient training time and model capacity to approximate their optimal responses.

In the Stackelberg formulation, the attacker is considered the leader and commits to a strategy first, with the defender subsequently optimizing its response. This ordering can be operationalized during training by giving the attacker an initial advantage in learning or assigning a higher learning rate. In practice, alternating policy updates or introducing a minor bias in favor of the attacker facilitates convergence. Even in simultaneous training regimes, the agents tend to converge to equilibrium-like strategies, where neither policy can be easily exploited by the other. When one agent learns significantly faster, leading to destabilization, mitigation strategies such as temporarily freezing one policy or adjusting learning rates may be adopted to restore balance.

4.9.4 Modeling Assumptions

Several assumptions were introduced to ensure the tractability and analytical feasibility of the modeled environment:

- **Discrete time and single action per step:** Real-world cyber operations may occur in continuous time with agents taking multiple actions concurrently.

This framework simplifies interaction dynamics into discrete steps, where each agent performs one action per turn. This design assumes a balanced interaction pace, where significant attacker activities are matched by corresponding defender responses. While simplified, this model captures essential interactions in sequential adversarial engagements.

- **Complete knowledge of the attack graph:** It is assumed that both the attacker and the defender are aware of the structure of the attack graph. The defender is aware of all potential vulnerabilities, though not their active status, and the attacker is aware of the general network topology and possible vulnerabilities. This represents a semi-realistic scenario where the attacker performs reconnaissance to infer network layout and software vulnerabilities, and the defender knows its infrastructure but not the attacker's exact location or entry point. The framework does not currently incorporate unknown exploits or zero-day vulnerabilities, although this extension is conceptually feasible through probabilistic modeling of unanticipated actions.

- **Rational and reward-maximizing agents:** Both attacker and defender are modeled as rational agents aiming to maximize their respective reward functions. Although real-world attackers may behave sub-optimally or be risk averse, and defenders may prioritize other operational constraints, these behaviors are approximated in the framework via tailored reward structures. For instance, defender penalties for excessive actions simulate resource constraints and usability concerns.

- **Sensing and alert reliability:** The defender is modeled as receiving alerts with probabilistic reliability. False positives are not explicitly modeled; instead, a lack of alert is treated as background noise. False negatives are indirectly captured by the detectability probability of each attacker action. A more elaborate observation model incorporating explicit false alerts can be incorporated to further enhance realism, though this would increase the complexity of the defender's inference process. While false positives are not modeled explicitly, detection uncertainty is indirectly captured via probabilistic observability and belief noise.

- **Reward values and temporal discounting:** Numeric reward values (such as plus one hundred or minus one hundred) are assigned to ensure meaningful feedback during training. The discount factor is set close to unity to preserve long-term dependencies, ensuring that success or failure at later stages influences early decision-making. Although the specific value of the discount factor is not stated, values such as zero point nine nine are typically used in long-horizon adversarial simulations.

4.9.5 Scalability and Hierarchical Modeling

In larger scenarios, scalability can be achieved through hierarchical RL. The attacker's campaign may be decomposed into sub-goals such as initial compromise, lateral movement, and privilege escalation. Each sub-task can be addressed using specialized sub-policies, while a higher-level policy coordinates these objectives. Similarly, the

defender's tasks can be divided into detection and response modules. This hierarchical decomposition reduces the complexity of decisions at each level and enables policy reuse across scenarios. For example, a generalized lateral movement policy may be applicable across different network topologies. Although hierarchical policies are not implemented in this framework, the underlying structure, which reflects phases of the CKC, supports future integration of this approach.

Another method to improve scalability involves parallelized training. This may be accomplished using asynchronous environments that simulate multiple episodes in parallel, thereby accelerating experience collection and reducing training time. Such approaches are consistent with asynchronous actor-critic methods and distributed RL architectures.

4.9.6 Equilibrium Analysis

The framework aims to learn policies that approximate a Stackelberg equilibrium with the attacker in the leader role. Post-training, empirical evaluation may be conducted to assess the proximity of the resulting policies to equilibrium. One approach involves fixing the defender's policy and testing whether alternative attacker strategies yield significantly improved outcomes. Conversely, the attacker's policy is held constant while the defender explores alternative responses. If no significant gains can be obtained through unilateral deviation, the policy pair can be considered an approximate equilibrium. Due to the Stackelberg formulation, a slight advantage for the attacker is expected.

Policy stability is used as a heuristic indicator of equilibrium. When training

performance metrics stabilize and policy oscillations diminish, the learned strategies are considered converged. More formal evaluation may involve iterative improvement of one agent's policy while holding the other fixed. In this framework, such evaluations revealed no significant improvements at convergence, indicating that the final policies approximate best responses under the modeled dynamics.

In conclusion, the proposed framework addresses a realistic and challenging cyber defense scenario. Complexity is managed through function approximation, state and action abstraction, and domain knowledge integration. RL with appropriately shaped rewards and observational models enables tractable learning in high-dimensional adversarial environments. The assumptions made are consistent with enterprise cyber threat models and can be relaxed or extended for future studies. The next chapter presents experimental results that demonstrate the emergence of realistic strategies. For example, the attacker learns to maintain stealth until reaching the high-value target, while the defender prioritizes patching critical vulnerabilities and deploying detection capabilities. The modeled results also illustrate the implications of attacker leadership under the Stackelberg game structure and evaluate the resilience of learned strategies under scenario variations. Figure 4.4 provides a visual summary of the attacker–defender interaction dynamics formalized in this framework.

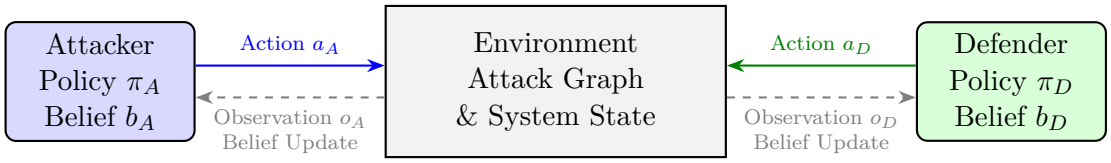


Figure 4.4: Stackelberg-POSG attacker–defender interaction.

CHAPTER 5

EXPERIMENTAL SETUP & RESULTS ANALYSIS

This chapter describes the experimental design, environment setup, and results analysis for the lightweight prototype of the proposed cybersecurity framework. A custom Gym-based environment was developed to model attacker-defender dynamics over a 10-node network. The simulation incorporates elements of partial observability, defender adaptation, and RL to reflect realistic APT progression scenarios. Key aspects such as the network topology, environment parameters, attacker and defender observability, and metric definitions are presented first, followed by a detailed presentation and discussion of the experimental results.

5.1 Environment Setup and Network Topology

The experiments were conducted in a custom Gym-based environment modeling a 10-node computer network. This size was selected to balance structural diversity with

computational feasibility, in line with prior RL-based cybersecurity studies [58], [59]. A 10-node topology enables realistic modeling of attack paths, lateral movement, and progressive privilege escalation without overwhelming the training process or inflating the state space.

The network topology is defined by a directed attack graph with nodes labeled 0 through 9. Node 0 serves as the attacker's initial foothold, while node 9 designates the target domain controller. Connectivity is defined by an adjacency list. The structure is illustrated in Figure 5.1, where the green node indicates the entry point, the red node highlights the critical target, and edges represent possible attack paths.

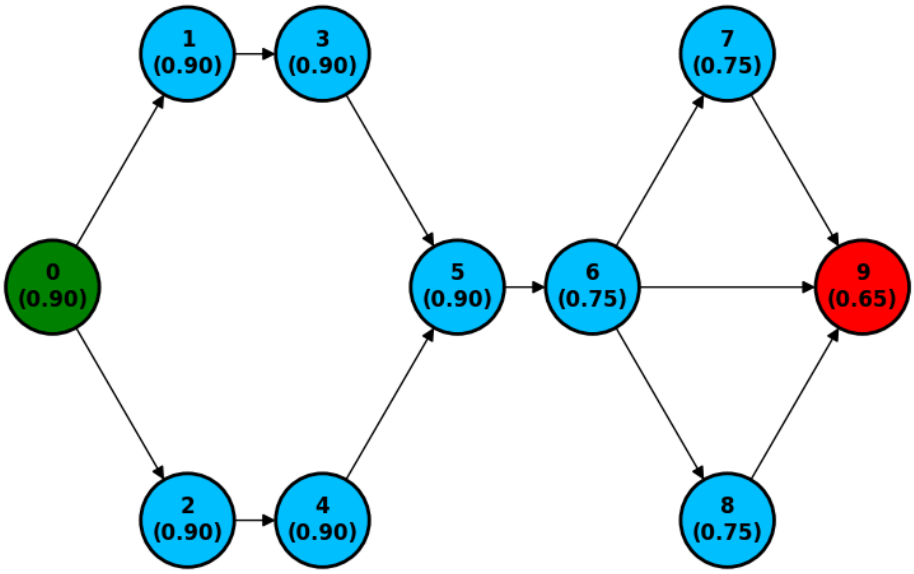


Figure 5.1: Cyber attack network topology with base probability of a successful compromise for each node.

Each node has an associated base success probability of being compromised. As depicted in Figure 5.1, perimeter nodes (0 through 5) have a higher base compromise probability (0.90), whereas deeper nodes (6, 7, and 8) are moderately harder to compromise (0.75). The domain controller (node 9) has the lowest base success

probability (0.65). These probabilities reflect an increasing level of difficulty as the attacker progresses toward critical assets.

The defender's detection mechanism is asymmetrical. Any failed exploit attempt triggers an 8% chance of detection (i.e., `detect_prob_on_fail` = 0.08), while successful compromises do not immediately trigger detection (`detect_prob_on_success` = 0.0). This models the real-world assumption that failed, overt actions are more likely to expose an attacker than stealthy, successful breaches.

The defender periodically patches nodes in response to attacker actions. After each attacker action, the defender selects a node to patch: if a previously compromised node is available, it is prioritized for patching; otherwise, a random unpatched node is selected. Patching removes any existing attacker foothold on the node and prevents further compromises, thereby dynamically altering the attack surface during an episode.

The environment provides the attacker with full visibility of the current network state after an initial exploration phase. Specifically, the attacker observes a binary vector indicating which nodes are compromised and which are patched. Although the underlying topology remains static, the network is partially dynamic due to defender-driven patching, which alters node states over time.

The attacker's action space consists of all ten nodes, where the agent selects a target node to attempt a compromise at each step. Episodes have a maximum length of 50 steps. Termination occurs earlier if the attacker successfully compromises node 9 (successful breach), if the defender detects the attacker (attacker caught), or if 50 steps elapse without achieving success (timeout). Early failures can occur if the defender

patches key paths, resulting in the attacker being unable to progress before the episode limit.

5.2 Experiment Design and Methodology

PPO, an RL algorithm, was employed to train the attacker's decision-making policy. The attacker aims to maximize cumulative rewards, where the reward function incentivizes reaching the target quickly while avoiding detection. The PPO implementation from Stable-Baselines3 was used with a multi-layer perceptron policy network and standard hyperparameters, including a discount factor of $\gamma = 0.99$ and a PPO clipping range. A small entropy coefficient of 0.03 was applied to encourage persistent exploratory behavior, which was observed to contribute to performance oscillations in later training stages.

The prototype adopts a numerically scaled reward structure that enables smooth training and stable convergence in the cyber-attack simulation environment. The attacker agent receives immediate scalar rewards based on the outcome of each action. The table below summarizes the actual reward values implemented during training.

Table 5.1: Reward structure used during PPO training.

Action or Outcome	**Reward**
Reaching the domain controller (goal success)	+50.0
Compromising a new host (non-goal)	+10.0
Redundant attack on already compromised node	−0.2
Attempting an unreachable (non-adjacent) node	−0.5
Failed attack on reachable target (not patched)	−0.5
Detection after failed attempt	−20.0
Timeout (episode ends after 50 steps)	−3.0
Step penalty (resource/time cost per action)	−0.05

This structure encourages the attacker to prioritize stealthy, efficient compromises while penalizing noisy, redundant, or invalid moves. Each training run was conducted for 500,000 timesteps against a fixed defender strategy. Two distinct defender patching strategies were evaluated:

- **Random Patching**: the defender randomly selects a node to patch after each attacker action, independent of the attacker's current state. This strategy serves as a baseline, providing unpredictable but untargeted defense behavior.
- **Neighbor-Aware Patching**: the defender prioritizes patching a neighbor of the last compromised node when possible. This approach models a targeted containment strategy, aiming to disrupt the attacker's lateral movement by securing adjacent nodes.

During training, the agent was periodically evaluated without exploration noise. Every 10,000 training steps, defined as one evaluation interval, 20 evaluation episodes were conducted, and performance metrics were recorded. The core metrics collected at each interval are included in Table 5.2.

Table 5.2: Key performance metrics recorded during training evaluations.

Metric	**Definition**
Success Rate (%)	Episodes where the attacker reached node 9 (target)
Detection Rate (%)	Episodes where the defender detected the attacker
Timeout Rate (%)	Episodes that reached the maximum step limit (50) without success
Average Episode Reward	Mean cumulative reward obtained per episode
Average Episode Length	Mean number of steps taken per episode, including timeouts
Average New Compromises	Average number of distinct nodes compromised per episode
Average Time to Domain	Average number of steps to reach node 9 (successful episodes)
Average Time to Detection	Average number of steps before detection (detected episodes)
Detected-after-Compromise Rate (%)	Among detected episodes, percentage where attacker had compromised at least one node
Detected-before-Compromise Rate (%)	Among detected episodes, percentage where attacker was caught prior to any successful compromise

The network topology used in all experiments is depicted in Figure 5.1. The attacker always begins at node 0, with that node initially considered compromised as a foothold. The defender applies one patch action per time step, immediately after the attacker's move, according to the selected strategy. A patched node is permanently immunized against compromise for the remainder of the episode; any attack attempt

on a patched node results in an automatic failure.

This dynamic patching behavior, particularly under the neighbor-aware strategy, creates a partially dynamic environment where the attacker must continually adapt its path based on the defender's interventions. In several episodes, particularly under aggressive neighbor-aware patching, early failures were observed when key paths were completely patched before the attacker could advance, resulting in timeouts well before reaching 50 steps. A concise overview of the environment setup and experimental methodology is summarized in Table 5.3.

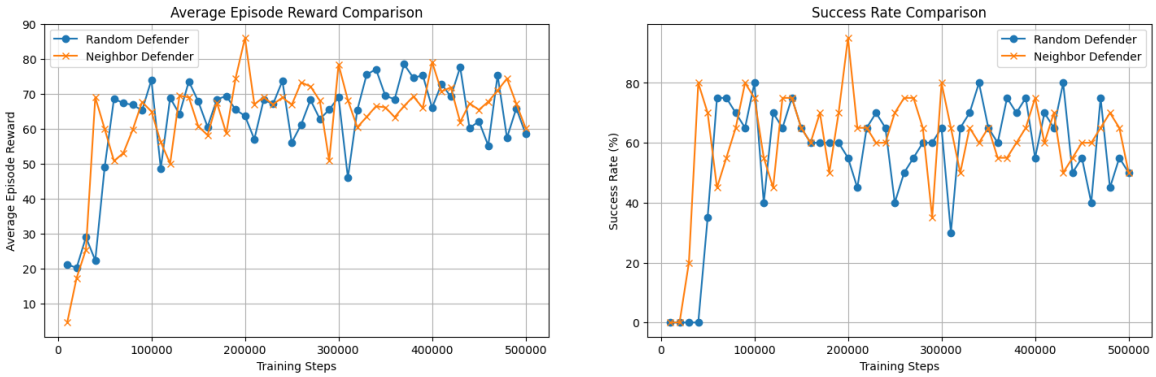
Table 5.3: Summary of experimental environment and training setup.

Network Environment	
Network Topology	- 10 nodes, directed attack graph - Node 0: Entry point - Node 9: Domain Controller
Base Compromise Probabilities	- Perimeter nodes: 0.90 - Intermediate nodes: 0.75 - Domain Controller: 0.65
Defender Detection Probabilities	- Failed exploit: 8% detection chance - Successful compromise: 0% detection chance
Attacker Observability	Binary vector: compromised vs patched nodes
Attacker Actions	Select one of 10 nodes to attack per step
Episode Termination	Success, Detection, & Maximum of 50 steps
Training and Evaluation Setup	
Training Algorithm	PPO
Defender Strategies	*Random Patching* and *Neighbor-Aware Patching*
Evaluation Frequency	Every 10,000 training steps (20 evaluation episodes)
Key Metrics Collected	Success rate , Detection rate, Timeout rate Episode reward, Episode length, New compromises Stealthiness

5.3 Results and Analysis

5.3.1 Core Learning Dynamics

The attacker was trained for 500,000 steps against each defender strategy, after which the learning outcomes and performance metrics were analyzed. Figure 5.2 presents the core learning dynamics, including the attacker's cumulative performance in terms of average reward and success rate throughout the training period.



(a) Average episode reward (b) Attack success rate

Figure 5.2: Attacker performance during 500,000 training steps under both defender strategies.

Reward Trends

The average episode reward of the attacker increased rapidly during the initial stages of training, indicating the swift development of effective attack policies. Against the neighbor-aware patching defender, the reward peaked at approximately 86.0 around 200,000 steps. Against the random patching defender, the peak occurred earlier, around 100,000 steps, with a maximum reward of approximately 74.0. These peak values correspond to the agent reaching the target in most evaluation episodes while

incurring minimal penalties. Following the peak, the average reward declined slightly under both defender strategies and stabilized at lower values by the end of training. For instance, against the random patching defender, the reward decreased to approximately 58.5 by 500,000 steps. This behavior indicates that the attack policy oscillated around a quasi-equilibrium rather than achieving full convergence. The continued policy exploration, driven by the entropy coefficient of 0.03, contributed to these late-stage fluctuations. Overall, the attacker attained a high-reward policy relatively quickly, although maintaining peak performance was not sustained consistently over time. The trends in average episode rewards are depicted in Figure 5.2(a).

Success and Failure Rates

The success rate of attacks followed a trajectory similar to that of the reward. Beginning at 0%, the success rate increased sharply as the attacker identified effective strategies. Figure 5.2(b) shows the success rate over the training period for both defense strategies. Against the random patching defender, the success rate reached approximately 80% by 100,000 steps, representing the highest level recorded for this strategy. Subsequently, it fluctuated between approximately 50% and 75%, stabilizing near 50% by the end of training. Against the neighbor-aware patching defender, the success rate peaked at approximately 95% around 200,000 steps, with the attacker succeeding in nearly every evaluation episode at that point. This high level of performance was not sustained, as the success rate dropped to approximately 35% around 290,000 steps before rising again. In the later stages, success against neighbor patching ranged between 60% and 70%, with specific values of approximately 65%

at 310,000 steps, 60% at 320,000 steps, and a subsequent increase to approximately 80%. Table 5.4 summarizes the training steps at which the highest success rates and corresponding average rewards were observed for both strategies.

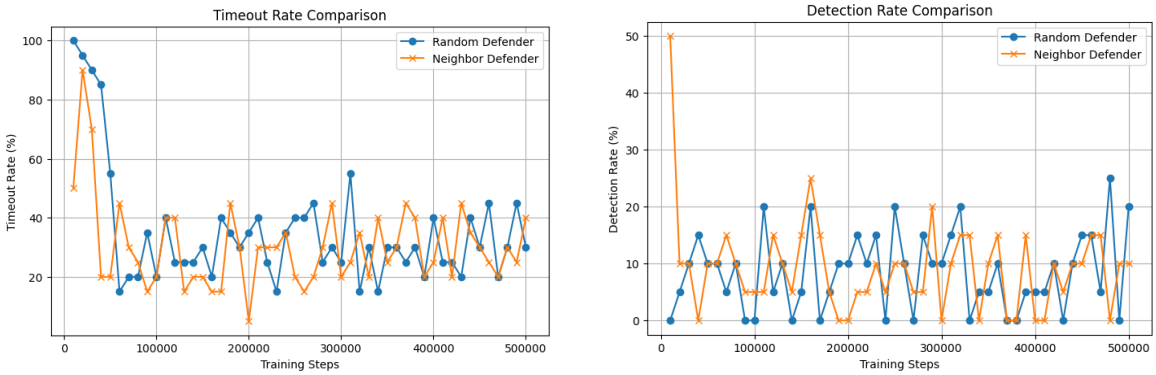
Table 5.4: Best recorded success rates and corresponding rewards during training.

Metric	**Random Defender**	**Neighbor Defender**
Best Training Step	100,000	200,000
Best Success Rate (%)	80.0	95.0
Detection Rate at Best Step (%)	0.0	0.0
Average Reward at Best Step	74.01	86.05

Timeout failures, defined as episodes where the attacker failed to complete its objective within the allowed number of steps, exhibited an inverse correlation with the success rate. As illustrated in Figure 5.3 (a), when success against the neighbor-aware patching defender peaked at 95%, timeout failures decreased to approximately 5%. Conversely, at a success rate of 35% around 290,000 steps, timeouts increased to approximately 45%. By 500,000 steps, timeout rates stabilized in the range of 25% to 40% for both defender strategies. Specifically, timeout rates were approximately 30% against random patching and between 25% and 30% against neighbor patching at the end of training.

Detection rates remained low for the majority of the training period. Figure 5.3(b) illustrates the progression of detection rates for both strategies. Detection occurred predominantly during early training phases when the attack policy was still exploratory and suboptimal. Against random patching, detection peaked at approximately 20% between 110,000 and 120,000 steps, then declined to a long-term average of between 0% and 5%. Against neighbor patching, detection remained generally

within the 0% to 10% range during the later stages of training. In some evaluations, detection fell as low as 0%, for example, at 480,000 steps. At the final evaluation point, the detection rate was approximately 20% against random patching and between 0% and 10% against neighbor patching. By the end of training, most failures by the attacker were attributed to timeout events rather than detection.



(a) Timeout rate (b) Detection rate

Figure 5.3: Attacker failure rates during training under each defender strategy.

Episode Length Trends

As the attacker refined its policy, a significant reduction in average episode length was observed, indicating faster and more efficient attack execution. Figure 5.4 presents episode lengths over time under both defense strategies. At the beginning of training, episodes frequently lasted close to the maximum of 50 steps, reflecting an untrained and ineffective policy. Over time, as the attacker became more proficient, episodes were completed more quickly. In the later stages, the attacker typically completed an episode, successfully or unsuccessfully, in substantially fewer steps than at the start, reflecting improved efficiency.

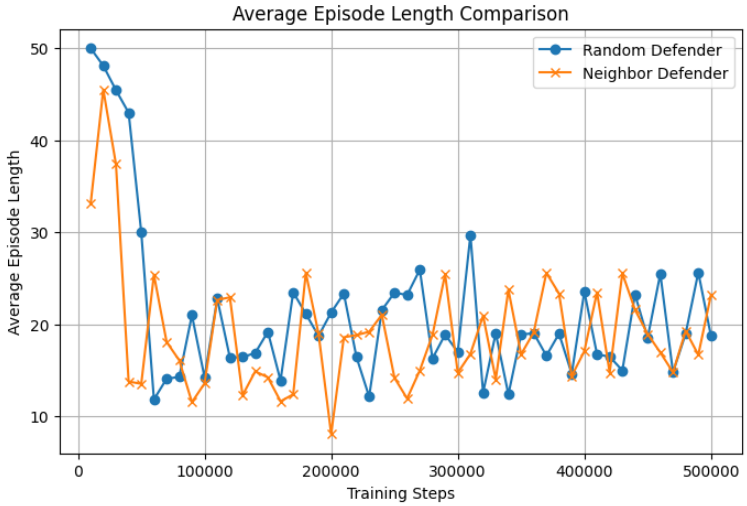


Figure 5.4: Average episode length during training under random and neighbor patching defenders.

5.3.2 Equilibrium Behavior and Strategic Insights

During training, the interaction between the attacker and each fixed defender strategy progressed toward a form of quasi-equilibrium, marked by oscillations in performance. Instead of converging to a single static policy, the attacker continued to explore a range of high-performing strategies and demonstrated ongoing adaptation. These performance fluctuations were more pronounced against the neighbor-patching defender. Table 5.5 presents observed correlations among key metrics, including success rate, detection rate, episode length, and compromises, for each defense strategy.

Table 5.5: Correlation between major metrics during training.

Relation	**Random Defender**	**Neighbor Defender**
Success fate vs detection rate	-0.293	-0.556
Average episode reward vs episode length	-0.894	-0.766
Any compromise rate vs detection rate	-0.300	-0.009

By approximately 200,000 training steps, the attacker had identified a highly effective tactic against the neighbor-patching defense, achieving a peak success rate of

approximately 95 %. During this period, the attacker was able to avoid most of the neighbor's patches. However, sustaining this level of performance was difficult. Small adjustments to the policy, caused by ongoing learning updates, led to declines in success rate following the peak. The attacker alternated between very high and moderate success rates, indicating a tradeoff between two competing goals: reaching the target rapidly to stay ahead of the defender's patching, and proceeding cautiously in the presence of patched obstacles.

The attacker's behavior stabilized around a mixed strategy. It alternated between aggressive, fast attack attempts that succeeded when patch timing was favorable, and more conservative approaches that occasionally resulted in timeout failures when critical paths were blocked. This pattern of dynamic stabilization reflects training within an environment that, although fixed in policy, introduced variability through randomized patch placement and timing. Such variability required the attacker to continue adapting. Figure 5.5 shows the correlation between episode duration and episode reward. Shorter episodes were generally associated with higher rewards and therefore with higher success rates. Rapid completion of an episode typically indicated that the attacker reached the target efficiently, avoiding both detection and failure.

Minor fluctuations in the attacker's policy frequently influenced the balance between its competing objectives, leading to temporary regressions in performance. Ultimately, the attacker and defender interaction reached a dynamic steady state. In this state, the attacker was successful in the majority of episodes but could not achieve success consistently, primarily due to occasional but significant defensive actions, particularly under the neighbor-patching strategy.

These outcomes provide several strategic insights. Targeted defense strategies, such as neighbor-aware patching, created a more difficult environment for the attacker during early training. However, when the defender's behavior became predictable, the attacker was able to adapt and exploit this predictability. The neighbor-patching strategy created a form of moving obstacle that prompted the attacker to increase its speed. When successful, this approach led to short periods of very high reward and success, such as those seen around 200,000 steps. However, the possibility of unfavorable patch alignments also led to sudden drops in performance.

In contrast, the random patching strategy did not directly block specific paths but introduced unpredictability. This randomness made it more difficult for the attacker to achieve high levels of success consistently. As a result, the attacker's policy against random patching stabilized at a moderate success level relatively early in training. For instance, after reaching an early peak of approximately 70 %, the success rate decreased to approximately 60 % by the final evaluation. Continued exploratory behavior and random disruptions prevented further improvement. The possibility of disruption at any point, regardless of policy quality, ensured that the attacker's success rate remained below the theoretical maximum.

In conclusion, each defense strategy led to a distinct performance profile for the attacker. The neighbor-patching strategy produced high variance in outcomes. The attacker occasionally achieved near-perfect success, while at other times it was halted early, due to the deterministic and aggressive nature of the defense. The random-patching strategy resulted in more stable outcomes within a moderate success range, illustrating that unpredictability can serve as an effective barrier to consistent attacker

dominance. Rather than achieving complete superiority, the attacker settled into a dynamic equilibrium. Unpredictability in the defense, as in random patching, limited the attacker's maximum achievable performance. Deterministic strategies, such as neighbor patching, could be partially anticipated and adapted to, but still succeeded in disrupting the attacker periodically. The final equilibrium reflected a scenario in which the attacker succeeded frequently but not always. These findings emphasize that the presence or absence of adaptivity in the defense strategy has a significant impact on the attacker's ultimate performance.

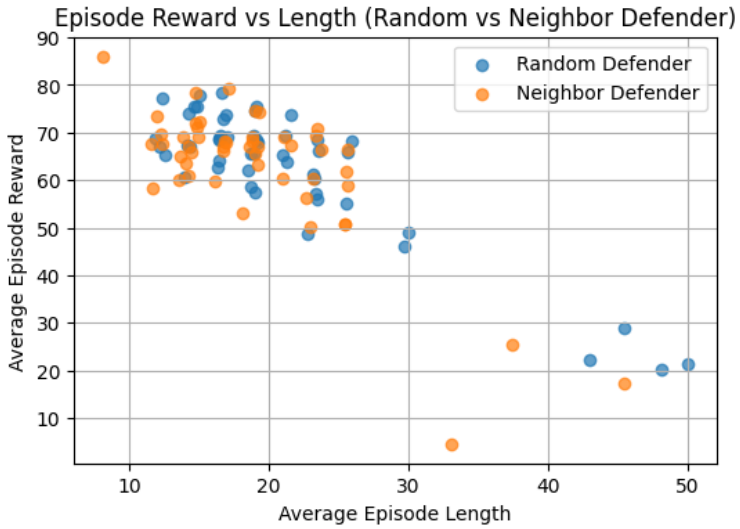


Figure 5.5: Episode reward versus episode length during training

5.3.3 Defender Strategies Comparison

This section examines how the two defensive patching strategies influenced attacker performance and behavior, in order, to assess their relative effectiveness. The analysis highlights the differing strengths and limitations of each approach.

Attacker Success and Rewards

The neighbor-patching strategy initially proved more effective in delaying attacker success. At 50,000 training steps, the attacker's success rate remained at 0% against the neighbor-patching defender, with many episodes ending in timeouts or detections. In contrast, under the random patching strategy, the attacker reached a success rate of approximately 35% by the same stage. This outcome suggests that targeted patching imposed a greater early burden on the attacker. However, as training progressed, the attacker adapted to the neighbor-patching behavior. Ultimately, the attacker achieved a higher peak success rate against the neighbor defender (95%) compared to the random defender (80%), along with higher peak rewards. This indicates that although targeted defenses were initially more disruptive, an adaptive attacker eventually exploited their predictability, achieving higher performance than under a random defense.

Attacker Behavior Adaptation

The neighbor-patching strategy influenced both the attacker's path planning and execution pace. The attacker learned to anticipate that, after compromising a node, adjacent nodes would likely be patched. In response, the attacker frequently adopted a rapid attack strategy, aiming to complete multi-stage sequences before patching restricted further movement. This led to short, high-intensity paths, typically resulting in either swift success or early failure. In contrast, under random patching, the attacker employed more cautious behavior, exploring alternatives when patches obstructed the current path. Since patching in this case did not correlate directly with

attacker actions, there was less urgency to complete attacks quickly. As a result, successful attacks under random patching involved longer paths, averaging 15–20 steps. Under neighbor patching, successful paths typically consisted of 8–14 steps. This difference reflects how the predictability of neighbor patching encouraged the attacker to optimize for speed, whereas the unpredictability of random patching necessitated exploratory planning.

Detection versus Prevention

A successful defense may either detect the attacker or prevent attack success. The neighbor-patching strategy emphasized prevention, often stopping the attacker by inducing a timeout rather than by detection. In later training stages under this strategy, detection rates declined to near zero. For example, at 200,000 steps, the detection rate was 0%, while the timeout rate reached approximately 5%. In contrast, the random patching strategy occasionally resulted in detection events, particularly when the attacker attempted to exploit a recently patched node. At 150,000 steps, detection occurred in approximately 5% of episodes under the random defender, while timeouts accounted for around 30%. By the end of training, both strategies exhibited low detection rates (generally between 0–10%), indicating that prevention, rather than detection, became the primary defense mechanism.

The trade-off between prevention and detection is illustrated by the relationship between attacker success and the frequency of timeouts or detections. Figure 5.6 shows that higher attacker success rates correlate with lower detection rates, especially under the neighbor-patching strategy. Figure 5.7 illustrates a corresponding inverse trend

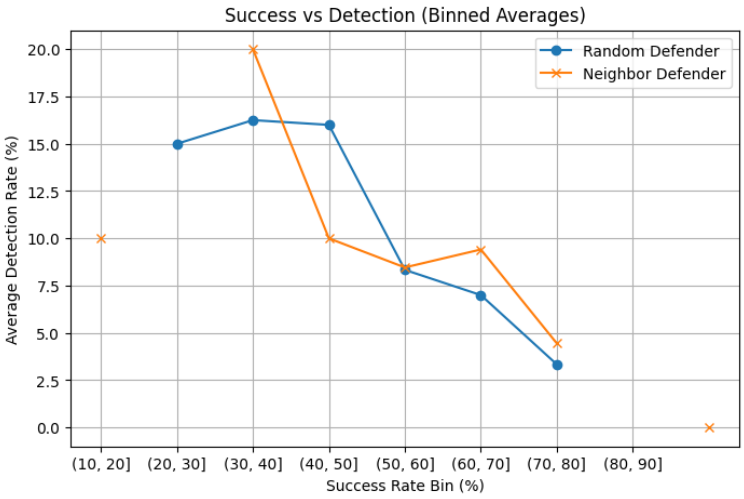
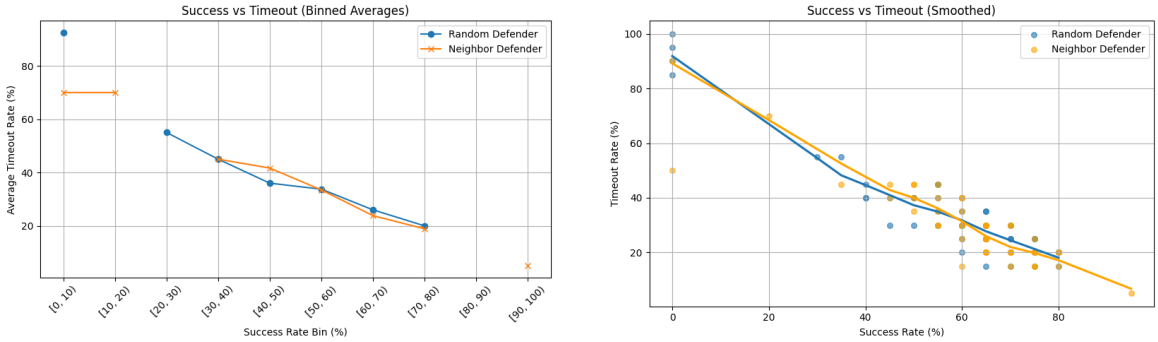


Figure 5.6: Attacker success rate versus detection rate under each defense strategy.

between success and timeout rates. These trends confirm that the neighbor-patching strategy primarily prevented success without detection, whereas the random strategy occasionally allowed failures to be detected.



(a) Binned correlation (b) Smoothed trend

Figure 5.7: Inverse relationship between attacker success and timeout rate for both defender strategies.

Patching Effectiveness

Patching efficiency was assessed by examining how often a patch directly interfered with the attacker's next action. *Patch efficiency* is defined as the percentage of patches

that blocked the next attempted action. By the end of training, this metric had declined to 0% for both strategies, indicating that the attacker consistently avoided patches. The *patch prevent rate*, representing the fraction of attacker actions that encountered patched nodes, was similarly reduced to 0%. These outcomes suggest that patches functioned primarily to redirect the attacker or slow progression, rather than to immediately terminate actions.

Despite this convergence, the neighbor-patching strategy demonstrated greater impact earlier in training. On average, it prevented 0.118 attacker actions per episode, compared to 0.020 under random patching. This indicates that targeted defenses were more likely to block attacker movement, particularly during earlier phases, although this advantage diminished as the attacker refined its strategy. By the final evaluation, both strategies exhibited negligible patch effectiveness, highlighting the adaptability of the attacker. Table 5.6 reports patching efficiency and prevention statistics achieved by each defender strategy in the final evaluations.

Table 5.6: Defender patching metrics at final evaluation stage.

Metric	**Random Defender**	**Neighbor Defender**
Final Patch Efficiency (%)	0.0	0.0
Final Patch Prevent Rate (%)	0.0	0.0
Average Patches per Episode	1.756	1.816
Average Patch Hits (per episode)	0.006	0.020
Average Prevented Actions (per episode)	0.020	0.118

In conclusion, neither defensive strategy fully prevented a well-trained attacker from achieving its objective. Each method demonstrated specific advantages. Random patching introduced variability, complicating the attacker's optimization and resulting in moderate late-stage success rates (approximately 50–70%). Neighbor patching

initially imposed greater constraints, but its predictable behavior was eventually exploited by the attacker, at times enabling near-perfect success. Thus, neighbor patching is effective against naive or inflexible attackers but susceptible to more adaptive agents. Conversely, random patching imposed consistent uncertainty, which limited the attacker's ability to fully optimize its strategy. These observations suggest that a hybrid strategy, combining the strengths of both approaches, may offer improved robustness, presenting an opportunity for future investigation.

Overall, the comparison indicates that RL attackers are capable of adapting to both strategies. The neighbor-patching defense had a strong early effect but became less effective as the attacker adapted. The random defense, while less impactful initially, maintained uncertainty that constrained attacker success throughout. These findings highlight the importance of incorporating unpredictability into defense strategies, as static or deterministic approaches may ultimately be learned and circumvented by adaptive adversaries.

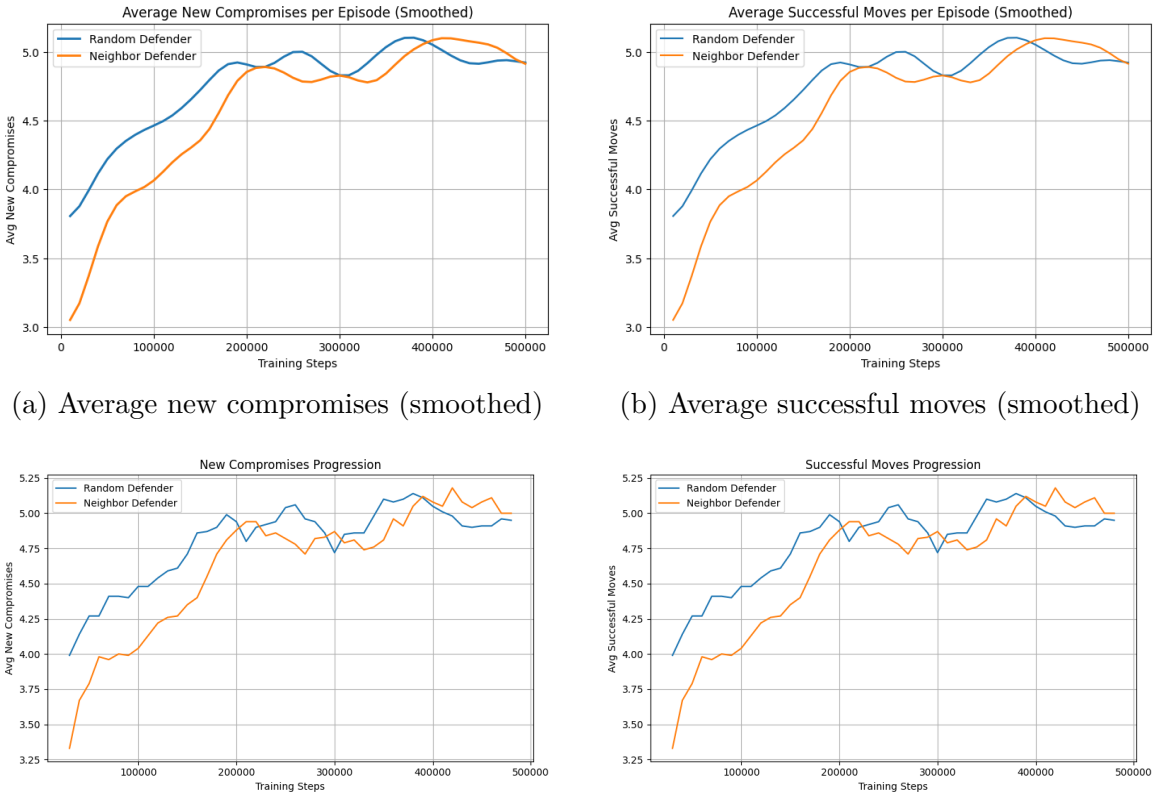
5.3.4 Attacker Progression and Behavior Analysis

This section examines the evolution of attacker strategies throughout training and analyzes behavioral characteristics in both successful and unsuccessful episodes.

Progression of Compromise Steps

In the early training phase, the attacker often compromised only a limited number of nodes before being detected or reaching the episode time limit. As training progressed, the average number of new compromises per episode increased, reflecting

deeper network penetration. By the final training stages, successful episodes typically involved compromising a chain of approximately four to five nodes, which corresponds to the minimum number of steps required to reach the target node (node 9) from the entry node (node 0) within the given network topology. This indicates that the agent learned to execute multi-step attack sequences efficiently. Figure 5.8 presents trends in attacker performance, showing both smoothed averages and raw values for new compromises and successful moves over the course of training for each defender strategy.



(a) Average new compromises (smoothed)

(b) Average successful moves (smoothed)

(c) New compromises (raw)

(d) Successful moves (raw)

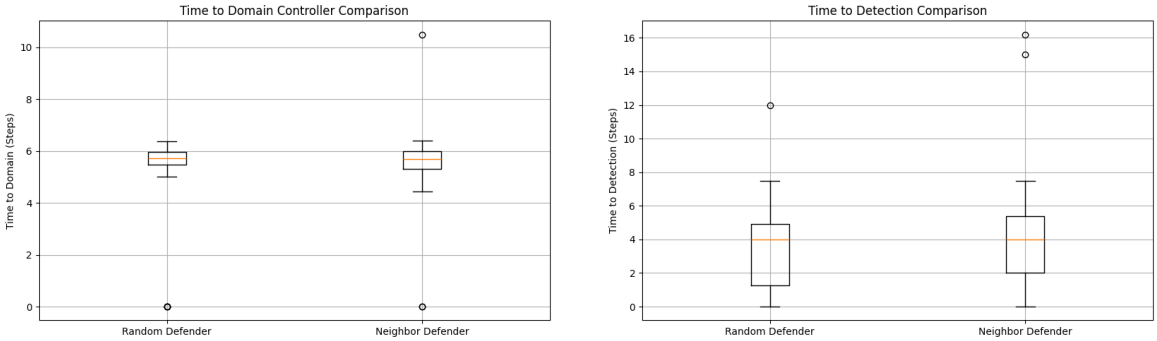
Figure 5.8: Attacker performance over training: average new compromises and successful moves per episode under each defender strategy.

Speed of Attack: First Compromise and Time to Target

The first compromise step metric indicates the step number at which the attacker initially compromised a node in each episode. Early in training, this value was high due to frequent failures, but it steadily declined to approximately 1.0, indicating near-immediate success in later training stages. This reflects the agent's improved ability to identify a vulnerable neighbor of the entry node at the beginning of an episode.

Similarly, the number of steps required to reach the domain controller in successful episodes declined significantly. In later stages, this value averaged approximately five to six steps. Given that the shortest path from the entry to the target node consists of four to five hops, the attacker was consistently executing near-optimal paths. Under neighbor patching, deviations from the shortest path occasionally occurred due to critical nodes being patched. For example, at 270,000 steps, successful episodes under neighbor defense required an average of 6.4 steps, compared to approximately 4.9 steps at 300,000 steps when patch placements were less obstructive. These results indicate that the attacker learned to reroute efficiently while maintaining near-optimal progression.

Figure 5.9 provides a summary of timing-related metrics. Figure 5.9 (a) shows the distribution of steps to reach the target, while Figure 5.9(b) shows when detection events occurred. Most successful attacks concluded in under 10 steps, with median values near five to six steps. Detections typically occurred early in the episode, often within the first few moves.



(a) Time to reach target

(b) Time to detection

Figure 5.9: Time distribution of reaching the target and detection events across defender strategies.

Behavior in Success and Failure Cases

Two primary failure modes were observed during training:

1. *Timeout failures*: The attacker failed to reach the target within the 50-step limit. These episodes frequently involved redundant or invalid moves, such as revisiting compromised nodes or attempting actions blocked by patches.

2. *Detection failures*: The attacker was detected before completing the attack, often due to attempting an exploit on a patched node, which carried an 8% probability of detection.

Both failure types became less frequent over time. Under random patching, approximately one-third of detections occurred before any compromise, while two-thirds occurred after partial progress. Under neighbor patching, detections, when they occurred, typically happened immediately after the first exploit attempt, aligning with the defense strategy's design for early containment. In successful episodes, the attacker followed a linear progression, compromising nodes along a direct route with minimal

backtracking. The stealthiness metric, defined as the proportion of the episode during which the attacker remained undetected, approached 100% in successful cases.

Adaptation to Defense Strategies

The attacker adapted its policy to each defense strategy. Against neighbor patching, the agent learned to avoid immediately attacking adjacent nodes that had just been patched. For instance, after compromising node 5 and observing node 6 being patched, the attacker would detour through other nodes and revisit node 6 later. This is reflected in the low patch hit rates during later stages of training, indicating that the attacker rarely targeted patched nodes. Although neighbor patching caused slightly more patch-prevented actions than random patching, these events remained infrequent, averaging fewer than 0.2 per episode by the end of training. After initial encounters with patched nodes, the attacker adjusted its behavior to avoid repeating failed attempts.

Figure 5.10 shows the average number of patches applied per episode during the final 50,000 training steps. As the attacker's efficiency improved, the number of patches applied per episode decreased. Under both defense strategies, this metric stabilized at a low level. The neighbor strategy applied slightly more patches per episode, consistent with its targeted behavior. Against random patching, adaptation was based on resilience. Since patches appeared in locations unrelated to attacker actions, the attacker learned to reroute efficiently. When a primary path was blocked, the agent quickly identified alternatives. This flexibility supported stable success rates of approximately 60%, even when random patching interrupted planned routes.

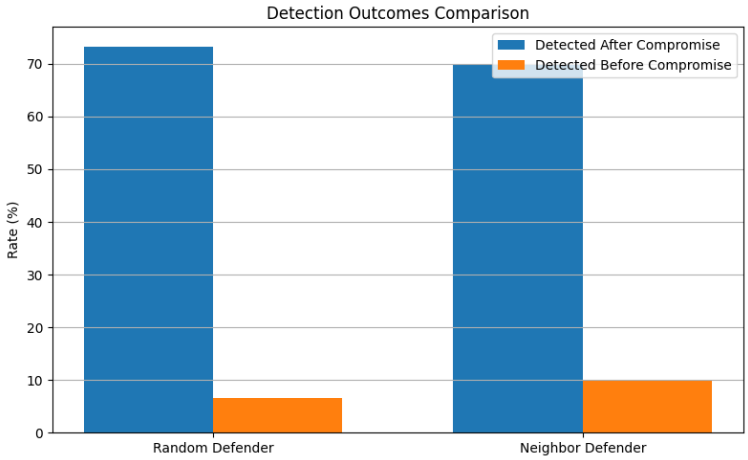


Figure 5.10: Average number of patches applied per episode during the final 50,000 training steps.

Overall, the attacker demonstrated adaptive behavior, including rapid initial compromises, efficient lateral movement, and responsive detours. The RL agent developed a streamlined attack policy that minimized redundant actions and responded dynamically to defender interventions. Even in failure cases, the attacker often achieved partial progress before being stopped. This behavior analysis demonstrates that RL can produce attackers with human-like adaptability. The agent identified efficient paths, adjusted to environmental constraints, and exhibited decision-making patterns comparable to those of experienced human adversaries.

5.3.5 Detection and Observability Analysis

An important aspect of attacker-defender interaction is the timing and nature of attacker detection. This factor has significant implications for practical cyber defense operations. In the simulation environment, detection events followed a probabilistic model: each failed exploit attempt had an 8% chance of triggering detection. Al-

though this model is simplified, it offers meaningful insights into the relationship between attacker behavior and observability. Table 5.7 reports the final evaluation metrics, averaged across the last evaluation episodes, for both defender strategies. These include new compromises, successful and failed moves, and detection timing.

Table 5.7: Summary table averaged over the last 5 evaluation intervals for both defender strategies.

Metric	**Random Defender**	**Neighbor Defender**
Avg New Compromises	4.95	5.00
Avg Successful Moves	4.95	5.00
Avg Failed Moves	1.30	0.97
Avg First Compromise Step	1.09	1.12
Avg Time to Domain Controller	5.75	5.94
Avg Time to Detection	3.94	3.70
Detected After Compromise (%)	73.33	70.00
Detected Before Compromise (%)	6.67	10.00

By the conclusion of training, detection rates were consistently low, often approaching 0% across evaluation episodes. This was partially due to the environment design—successful exploits had no chance of triggering detection—and partially due to the attacker's high efficiency in avoiding failed actions. The trained attacker rarely engaged in low-probability exploits, thus minimizing detection opportunities. This behavior illustrates how efficiency in action selection can result in operational stealth. An attacker can avoid triggering alerts simply by selecting targets with high success likelihood.

When detection did occur, the timing relative to compromise events differed between defender strategies. Under the random patching strategy, detections occurred both early and late in the attack sequence. Some detections followed initial compromises, while others occurred immediately after the first failed exploit. For example,

in one evaluation interval, approximately 66.7% of detections under random patching occurred after at least one successful compromise, while 33.3% occurred beforehand. Under the neighbor-patching strategy, detections predominantly occurred at the beginning of the attack. In several late-stage intervals, all detections under this strategy took place before the first compromise. This outcome reflects the neighbor strategy's design objective of early containment. Patching nodes adjacent to the entry point increased the likelihood of an initial failure and immediate detection. Once the attacker bypassed this first layer, subsequent detection was rare, and containment relied primarily on patching rather than alerts. This represents a qualitative difference: neighbor patching aims to detect early or not at all, while random patching permits detections throughout the attack sequence.

The analysis also included a measure of overall observability via an episode-level "stealthiness" score. This score was defined as the proportion of the episode during which the attacker remained undetected. Episodes with no detections achieved 100% stealthiness, whereas episodes with early detections had significantly lower scores. As training progressed and detection events declined, average stealthiness increased, approaching 100%. After approximately 100,000 training steps, the attacker consistently achieved average stealthiness between 95% and 100% under both defender strategies. Even in cases where the attacker failed due to timeout, these episodes typically remained undetected, meaning the attacker was halted without alert generation. This reveals a potential blind spot: many failed attacks left no trace unless subjected to post-incident analysis. For instance, under neighbor patching during the final evaluation phase, the detection rate was near zero while the attacker success rate was

approximately 65%, implying that about 35% of episodes ended in failure without any detection.

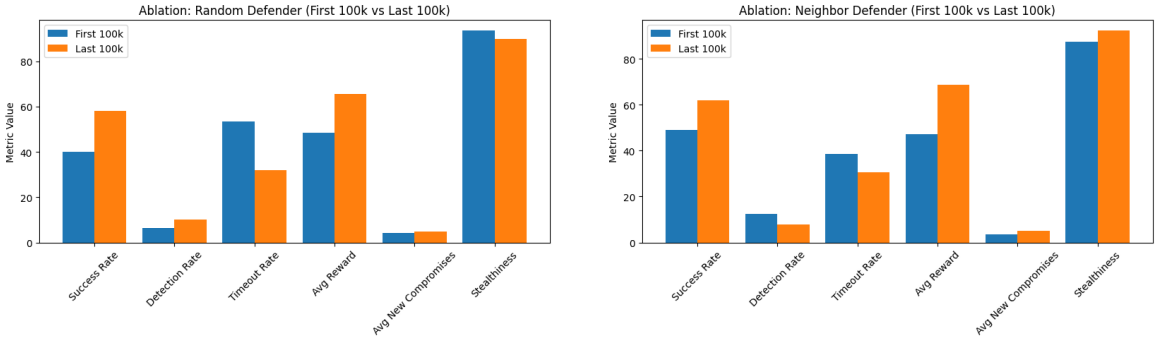
These findings illustrate a trade-off between prevention and detection. As the defender's patching became more effective—particularly in the neighbor strategy—detection rates declined to negligible levels. The attacker either succeeded quickly or was stalled without generating any alert. By contrast, the random strategy occasionally induced detectable failures by blocking a path and triggering a failed exploit. As a result, this strategy provided slightly greater visibility into attacker actions. Detection rates under random patching fluctuated up to 20% in certain intervals, while those under neighbor patching remained below 5%. From a defense planning perspective, this highlights the need to define priorities. A purely preventive defense may successfully halt attacks but could leave defenders unaware of attempted intrusions. Conversely, a defense that enables more detections may allow earlier response at the cost of higher breach likelihood.

In summary, the analysis of detection and observability shows that a RL-based attacker can operate with high stealth when the environment emphasizes patching over detection. Most detection events occurred at the initial stage of the attack, particularly under neighbor patching. Importantly, halting an attacker through patching is not equivalent to detecting its presence. Many failed attack episodes produced no alerts, suggesting that real-world defenders might remain unaware of near-compromises unless forensic analysis is conducted. An effective defense strategy should balance prevention and detection, aiming to both stop intrusions and generate signals that inform security response. In the experiments, the random strategy provided more such sig-

nals via occasional detections, while the neighbor strategy offered stronger breach prevention but with limited visibility. The attacker, in turn, learned that maximizing exploit efficiency was the most effective path to remaining undetected, exhibiting stealth through competence rather than through evasive behavior.

5.3.6 Ablation Study: Early Versus Late Training Behavior

To evaluate the training dynamics of the agent, an ablation study was conducted by comparing its performance during the early and late stages of training. Specifically, metrics were collected over the first 100,000 training steps and the last 100,000 training steps for both the random and neighbor defender strategies. Figure 5.11 illustrates changes in key performance metrics from early to late training under both defender scenarios. These metrics include success rate, detection rate, timeout rate, average episode reward, average new compromises per episode, and stealthiness. Table 5.8 presents the numerical values of these metrics for direct comparison. Figure 5.12 provides a combined visualization of improvements in attacker performance across both defender strategies.



(a) Random defender (b) Neighbor defender

Figure 5.11: Early versus late training performance under each defender strategy.

The results of the ablation study show that the attacker improved across all major metrics between the early and late stages of training. However, the nature and magnitude of improvement varied between the two defender strategies. For example, the success rate under the random defender increased from 40.00% to 58.18%, while under the neighbor defender it increased from 49.00% to 61.82%. This demonstrates consistent learning progress under both defenses, with success rates exceeding 60% by the final stage. Timeout rates decreased in both cases, from 53.50% to 31.82% for the random strategy and from 38.50% to 30.45% for the neighbor strategy. This indicates that the attacker became more capable of completing its objectives within the allowed episode length.

Table 5.8: Ablation study results comparing first and last 100k training steps for both defender strategies.

Metric	**Random First 100k**	**Random Last 100k**	**Neighbor First 100k**	**Neighbor Last 100k**
Success Rate (%)	40.00	58.18	49.00	61.82
Detection Rate (%)	6.50	10.00	12.50	7.73
Timeout Rate (%)	53.50	31.82	38.50	30.45
Average Episode Reward	48.38	65.52	47.25	68.83
Average New Compromises	4.20	4.95	3.67	5.07
Stealthiness (%)	93.50	90.00	87.50	92.27

Detection rates exhibited different trends depending on the defense. Under the random strategy, the detection rate increased slightly from 6.50% to 10.00%, whereas under the neighbor strategy, it decreased from 12.50% to 7.73%. This is consistent with previous observations indicating that under the neighbor strategy, the attacker learned to avoid detection almost entirely. In contrast, under the random strategy, a residual risk of detection remained.

The average episode reward increased from 48.38 to 65.52 under the random strategy and from 47.25 to 68.83 under the neighbor strategy. This reflects improved efficiency in achieving objectives. The average number of new compromises per episode also increased, from 4.20 to 4.95 for the random strategy and from 3.67 to 5.07 for the neighbor strategy, approaching the maximum of five compromises. Stealthiness remained high throughout training but followed the detection trend: it decreased from 93.50% to 90.00% under the random strategy and increased from 87.50% to 92.27% under the neighbor strategy.

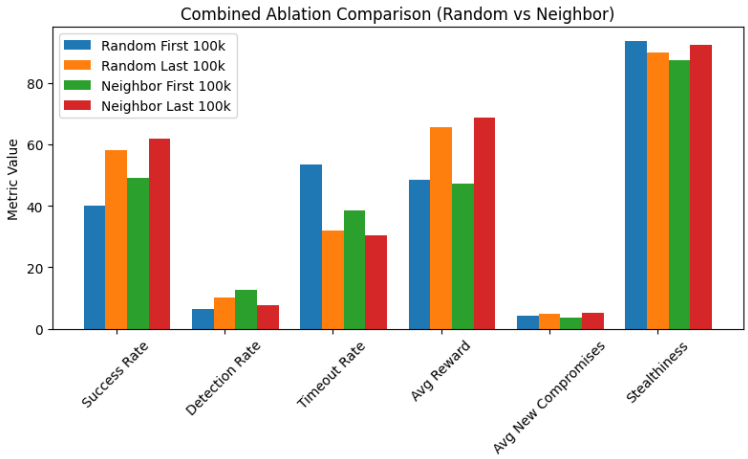


Figure 5.12: Combined performance changes from early to late training for both defender strategies.

These results indicate that the attacker improved its performance in a manner sensitive to the type of defense. Against the neighbor defender, the attacker improved primarily in avoiding detection and maintaining stealth, while also increasing its success rate. Against the random defender, the attacker increased success and reward but retained a small detection risk due to the inherent unpredictability of the defense. Figure 5.12 further highlights these trends, showing greater improvement in stealth under the neighbor strategy and comparable improvements in success rate across both

scenarios.

In summary, the ablation study highlights the attacker's ability to adapt its strategy over time. The attacker not only became more effective in terms of success rate and reward, but also tailored its behavior to the specific defensive environment. Against neighbor patching, the attacker focused on stealth and reliability after an initial learning phase. Against random patching, the attacker optimized for performance while tolerating a minor detection risk. These findings demonstrate the context-sensitive nature of the learning process and confirm the attacker's capacity to overcome both defense types through RL.

CHAPTER 6

LIMITATIONS AND FUTURE DIRECTIONS

This chapter introduces the limitations of the current study and outlines possible directions for future research. The focus of this work was on modeling intelligent attacker behavior using RL in a simplified simulation environment. While the framework demonstrates the potential of learning-based offensive strategies, several assumptions and design choices limit its realism and scalability. The following sections discuss these limitations in detail and suggest ways to improve the model in future work, particularly by expanding the attack scope, increasing environment complexity, and enhancing learning methods.

6.1 Limitations of the Study

While the results provide valuable insights, it is important to acknowledge the limitations of this research. These limitations arise from simplifying assumptions, the

experimental setup, and scope constraints.

6.1.1 Simulated Environment Accuracy

The study was conducted within a highly abstracted simulation. Although the Gym-based environment captured the essence of lateral movement through a network, it did not account for many real-world complexities, such as varied exploit techniques, heterogeneous vulnerabilities, user behaviors, and network traffic patterns. Nodes in the simulation had static compromise probabilities and binary states (compromised or patched), which oversimplifies the dynamic and layered security states of actual computer systems.

Consequently, the absolute performance levels observed for the attacker and defender in this simulation may not directly translate to real-world environments. The model lacks considerations such as exploiting noise, partial compromises, persistence mechanisms, and active defensive monitoring tools that would influence attack and defense dynamics in practice.

6.1.2 Network Size and Topology

The network was limited to ten nodes in a fixed, known topology. This small and static design was necessary for experimental tractability but is not representative of large, complex enterprise networks. In larger networks, the attacker would face many more possible paths, and the defender would have more complex topological features to protect, including multiple entry points and subnet segmentation.

Furthermore, the attacker in the simulation had full observability after an ini-

tial learning phase. Although the attacker had to learn the graph structure during training, it did not have to perform active network discovery actions that would be observable and potentially detectable in real-world scenarios. Thus, the simulation afforded the attacker an unrealistic level of information access, representing a simplifying assumption that would not hold in realistic attack settings.

6.1.3 Defender Strategy Constraints

Only two simple defender strategies were implemented: random patching and neighbor patching. These strategies were rule-based and reactive, showing no adaptation or strategic improvement as the simulation progressed.

In practice, defenders might employ a diverse set of tactics, including intrusion detection systems, dynamic network reconfigurations, deception techniques, and adaptive responses based on observed attacker behavior. The defender in this study also operated under a fixed action budget, patching only one node per time step without any explicit cost modeling or strategic optimization. As a result, the conclusions drawn are specific to the two simplistic strategies evaluated. A more sophisticated or adaptive defender could substantially alter the observed dynamics.

6.1.4 Detection Model Simplifications

The detection mechanism in the simulation was highly simplified, relying on a flat 8% chance of detection upon failed exploit attempts. Real-world detection systems are considerably more complex, incorporating anomaly detection, signature-based rules, behavior correlation, and varying sensitivity based on attacker behavior.

Moreover, real detection probabilities often increase with rapid or noisy attacker actions, a phenomenon not captured in the current model. The simulated attacker faced no additional risk for rapid lateral movement, whereas in practice, fast compromises might increase the likelihood of detection by human analysts or automated monitoring systems. Therefore, the attacker's optimal strategy in reality might differ significantly due to a more complex and risk-sensitive detection landscape.

6.1.5 Reward Function and Objectives

The attacker's reward function was designed to prioritize reaching the target quickly and avoiding detection. However, different reward formulations could influence learned behaviors substantially. For example, heavily rewarding speed could incentivize riskier behavior, whereas heavily penalizing detection could produce more cautious strategies.

Additionally, the defender was modeled as a static environment without its learning objective. The engagement was not formulated as a true adversarial RL problem, where both attacker and defender evolve their strategies in response to each other. Introducing a dynamic defender that adapts based on attacker behavior could significantly alter the learning dynamics and final outcomes.

6.1.6 Single-Stage Attack Focus

The simulation modeled a single-stage attack focusing on lateral movement toward a domain controller. Other stages of the cyber kill chain, such as initial exploitation, privilege escalation, or data exfiltration, were not represented.

Episodes concluded upon either successful compromise of the target or failure

through detection or timeout, without modeling partial achievements or varied attacker objectives. This binary success/failure framing omits consideration of attacks that achieve partial impacts, stealthy persistence, or multi-stage objectives.

6.1.7 Generality and Scalability of the RL Approach

Although PPO performed well within the constrained environment, scaling this approach to significantly larger state and action spaces could present challenges. Even in the relatively simple network used, performance oscillations and convergence instability were observed.

In more complex networks, the agent might require more sophisticated exploration strategies, hierarchical learning methods, or reward shaping techniques to achieve comparable success. The 500,000 training steps required for each scenario were non-trivial, and training demands would likely increase significantly in larger, more dynamic environments. Therefore, the results achieved here do not guarantee straightforward scalability to real-world enterprise-scale networks.

6.2 Future Directions

This research opens several avenues for further investigation, which could build on and extend the findings presented.

6.2.1 Adversarial Training and Game-Theoretic Approaches

A natural next step is to model the problem as a two-player game where the defender also learns and adapts. Instead of employing a fixed defender strategy, future work could introduce a defender agent that updates its policy in response to the attacker, potentially using RL or other artificial intelligence techniques. This could involve multi-agent RL frameworks or iterative training methods akin to generative adversarial networks or self-play.

Investigating how an intelligent defender adapts, and whether an equilibrium or cycles of dominance emerge, would offer deeper insights into long-term engagements. Game-theoretic analysis could further help in identifying optimal strategies for both attacker and defender, and in discovering potential Nash equilibria in the policy space.

6.2.2 Scaling to Larger and Richer Environments

Future work should expand the network size and complexity. Enhancements could include realistic network topologies with subnetworks, chokepoints, and varying connectivity, as well as heterogeneous node types such as workstations, servers, and IoT devices with diverse vulnerabilities.

Larger state spaces may necessitate more efficient algorithms or representations, such as neural network architectures with attention mechanisms or hierarchical RL approaches that decompose tasks into subgoals. Additionally, scaling the environment would allow the introduction of partial knowledge scenarios, requiring the attacker to perform reconnaissance activities, thereby increasing the realism of detection oppor-

tunities and modeling risks associated with discovery phases.

6.2.3 Enhanced Attack Models

Incorporating additional stages of the cyber kill chain would significantly enhance realism. Future simulations could involve multi-stage attacks requiring privilege escalation, maintenance of stealth through actions like log clearing, or establishing multiple footholds.

Agents could be tasked with balancing objectives such as persistence, data exfiltration, and lateral movement rather than simply reaching a target node. This would allow exploration of more sophisticated attacker strategies, including diversionary attacks and fallback planning, and analysis of how agents prioritize conflicting goals.

6.2.4 Improved Defense Mechanisms

Future research should implement more advanced and varied defensive mechanisms. This could include simulating intrusion detection systems that use anomaly detection, deploying honeypots and decoy nodes to mislead attackers, and dynamically reconfiguring network topologies or access control policies in response to detected threats.

Additionally, developing an intelligent defender agent capable of selecting from a playbook of defensive responses using RL would enable the simulation of more complex, adaptive defensive behaviors against RL-driven attackers.

6.2.5 Uncertainty and Partial Observability

Introducing partial observability for both attackers and defenders is a critical future direction. Realistic scenarios involve attackers operating without full network visibility and defenders possessing incomplete knowledge of attacker locations.

Formulations such as Partially Observable Markov Decision Processes (POMDPs) could be employed, alongside techniques like Deep Recurrent Q-Networks (DRQN) or Bayesian belief updates, to manage uncertainty. Agents would need to infer network conditions or attacker movements based on limited observations, enhancing realism and strategic complexity.

6.2.6 Integration of Learning with Domain Knowledge

Combining RL with domain-specific rules could improve performance and realism. For instance, hybrid attacker models could use RL for high-level decision-making while employing rule-based techniques for low-level exploits. Defenders could similarly incorporate threat intelligence into learning processes, shaping strategies based on known attack patterns.

Research could explore how best to integrate prior knowledge into RL frameworks through reward shaping, policy constraints, or hierarchical model architectures.

6.2.7 Autonomous Red Teaming Tools

An applied extension of this research could involve the development of autonomous red-teaming tools. Deploying RL-driven attackers within digital twins of enterprise

networks could help identify vulnerabilities and evaluate defense postures.

Future systems might allow operators to specify critical assets and then observe how an AI agent autonomously discovers and exploits pathways to those assets, uncovering non-intuitive attack vectors. Ensuring operational safety, environmental isolation, and interpretability of agent behavior would be essential for practical deployment.

6.2.8 Interactive Learning and Explainability

Incorporating human expertise into the learning loop is another promising direction. Techniques such as imitation learning or inverse RL could enable attacker agents to learn from human red team behaviors before advancing beyond them through autonomous refinement.

Similarly, defensive agents could be trained with human guidance or reward shaping. Explainable artificial intelligence (XAI) techniques should also be applied to derive understandable models of agent behaviors, providing actionable insights for human operators. Decision-tree approximations or feature analysis could aid in interpreting learned strategies.

CHAPTER 7

CONCLUSION

This thesis presented a detailed framework for modeling advanced cyberattacks as an intelligent, game-theoretic interaction between an APT adversary and a defender. At the core of the approach is a dynamic Stackelberg game under partial observability, the attacker, as leader, selects multi-step attack paths through a network, while the defender, as follower, responds with limited information about the attacker's actions. The state and action spaces were grounded in realistic data, notably, an attack graph derived from real system vulnerabilities and mapped to MITRE ATT&CK tactics, and the defender's observations were simulated as noisy, incomplete intrusion alerts requiring Bayesian belief updates. The complex attacker-defender interaction was solved approximately using RL. Both agents were modeled as independent PPO learners in a self-play training loop, allowing the attacker to refine its offensive strategy and the defender to react within its fixed strategy constraints to intrusions. This novel integration of game theory and RL enabled the study of APT engagements without relying on analytically intractable solutions. Using this framework, a series of simulation experiments were conducted to address the research questions and evaluate

the system's behavior.

- First, the study confirmed that PPO is effective in training the attacker agent within the proposed model. The learning curves demonstrated stable convergence of the attacker's policy, indicating that RL can navigate the large state space and partial observability of the scenario.

- Second, the experiments highlighted the value of Bayesian inference for the defender. A defender that maintained and updated beliefs about the attacker's location and intent consistently outperformed a naive defender, achieving higher detection rates and better containment of the attacker. This result underlines the importance of explicitly modeling uncertainty, even a simple probabilistic inference mechanism, markedly improved defensive performance.

- Third, analysis of the emergent behaviors revealed that the attacker learned tactics resembling those of real APT campaigns. For example, the trained attacker prioritized stealthy lateral movement and conditional privilege escalation, and it adjusted its path in response to the defender's actions. The defender, although following a fixed predefined policy in the experiments, was observed to focus its patching around recently compromised nodes, mimicking an intuitive human response to contain breaches. These realistic behavior patterns suggest that the simulation environment and reward structure successfully captured essential dynamics of APTs.

- Lastly, the framework allowed a comparative assessment of defense strategies. The neighbor-aware patching strategy, which proactively secures systems adjacent to

detected compromises, significantly reduced the attacker's lateral movement success relative to a purely random patching strategy. This demonstrates the framework's capability to evaluate how specific defensive measures can constrain or deter an adaptive attacker.

Together, these findings validate the framework and provide actionable insights as they show that RL agents can be used to model complex cyber adversaries, and they illustrate how certain defense strategies, especially those that are deterministic or myopic, might be exploited by a learning attacker over time. In summary, the research introduced and validated a new approach to modeling APT scenarios. The Stackelberg game formulation with partial observability captured the sequential and asymmetric information structure of attacker-defender interactions. By integrating real vulnerabilities, threat intelligence into a high-fidelity simulation and enabling both agents to adapt their strategies through learning, the framework was able to reproduce realistic attack patterns and outcomes. The core contributions of this work, bridging game-theoretic cyber modeling with RL, demonstrate considerable potential for advancing our understanding of cyber conflicts. The results not only contribute to academic knowledge in security games and machine learning but also inform the development of smarter defense mechanisms against intelligent, adaptive threats.

APPENDIX A

GLOSSARY OF NOTATION

A.1 Notation Table

Table A.1: Notation used in the Stackelberg game-theoretic APT simulation framework.

Symbol	Description
S	Set of environment states
$A_A,\ A_D$	Action sets of the attacker and defender
$A_A(s),\ A_D(s)$	Available actions in state s for attacker and defender
$T(s, a_A, a_D)$	State transition function; outputs distribution over next states
$\Delta(S)$	Set of all probability distributions over S

Continued on next page

Table A.1 – continued from previous page

Symbol	Description
$O_A,\ O_D$	Observation models for attacker and defender
$O_i(s, a, s')$	Conditional distribution over observations for agent i given state transition
$o_A,\ o_D$	Observations received by attacker and defender
$\pi_A,\ \pi_D$	Policies of the attacker and defender
$\pi_D^*(\pi_A)$	Defender's best response policy to attacker policy π_A
$U_A,\ U_D$	Expected utilities (cumulative rewards) for attacker and defender
$R_A,\ R_D$	Immediate reward functions for attacker and defender
$b_t^A,\ b_t^D$	Attacker and defender belief states at time t
G	Set of goal states for the attacker
$\mathbb{I}(\cdot)$	Indicator function; equals 1 if the condition is true, 0 otherwise

Continued on next page

Table A.1 – continued from previous page

Symbol	**Description**
p_e	Probability of success for attacker exploit e
$\delta(e)$	Detectability probability for attacker action e
$\text{apply}(a, s)$	Resulting state after applying action a in state s
$I(s, s')$	Progress metric from state s to s'
$C_A(a_A)$	Cost associated with attacker action a_A
$C_D(a_D)$	Cost associated with defender action a_D
$b^i_{t+1}(s')$	Belief update for agent i about state s' at time $t+1$
ρ_t	Probability ratio in PPO update
$L^A_{\text{PPO}},\ L^D_{\text{PPO}}$	Clipped surrogate PPO loss for attacker and defender
$V_A(s),\ V_D(s)$	Critic value estimates for attacker and defender

REFERENCES

[1] P. N. Bahrami, A. Dehghantanha, T. Dargahi, R. M. Parizi, K.-K. R. Choo, and H. H. Javadi, "Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures," *Journal of information processing systems*, vol. 15, no. 4, pp. 865–889, 2019.

[2] A. Ait Temghart, M. Marwan, and M. Baslam, "Stackelberg security game for optimizing cybersecurity decisions in cloud computing," *Security and Communication Networks*, vol. 2023, no. 1, p. 2811038, 2023.

[3] L. Hamza, M. Yousfi, and L. Bounehar, "Partially observable stochastic game for analysing complex attacks in iot networks," *Journal of Cyber Security and Mobility*, pp. 1039–1060, 2024.

[4] J. Zhang and Y. Luo, "Cyber attack-defense game strategy solving based on reinforcement learning and self-play," in *Proceedings of the 2024 3rd International Conference on Cryptography, Network Security and Communication Technology*, 2024, pp. 135–141.

[5] C. Hankin, P. Malacaria *et al.*, "Attack dynamics: An automatic attack graph generation framework based on system topology, capec, cwe, and cve databases,"

Computers & Security, vol. 123, p. 102938, 2022.

[6] B. Al-Sada, A. Sadighian, and G. Oligeri, "Mitre att&ck: State of the art and way forward," *ACM Comput. Surv.*, vol. 57, no. 1, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3687300

[7] R. Alkhadra, J. Abuzaid, M. AlShammari, and N. Mohammad, "Solar winds hack: In-depth analysis and countermeasures," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021, pp. 1–7.

[8] A. M. Pitney, S. Penrod, M. Foraker, and S. Bhunia, "A systematic review of 2021 microsoft exchange data breach exploiting multiple vulnerabilities," in *2022 7th international conference on smart and sustainable technologies (SpliTech)*. IEEE, 2022, pp. 1–6.

[9] R. Hiesgen, M. Nawrocki, T. C. Schmidt, and M. Wählisch, "The log4j incident: A comprehensive measurement study of a critical vulnerability," *IEEE Transactions on Network and Service Management*, 2024.

[10] A. Perdana, M. E. Aminanto, and B. Anggorojati, "Hack, heist, and havoc: The lazarus group's triple threat to global cybersecurity," *Journal of Information Technology Teaching Cases*, p. 20438869241303941, 2024.

[11] P. CISA, "State-sponsored actors compromise and maintain persistent access to us critical infrastructure," 2024.

[12] "CISA Details Cloud-Focused Attack Techniques Used by APT29 — duo.com," https://duo.com/decipher/cisa-details-cloud-focused-attack-techniques-used-by-apt29#:~:text=Historically%2C%20APT29%20has%20targeted%20exposed,adapt%20their%20techniques%20and%20targeting., 2024.

[13] A. Sharma, B. B. Gupta, A. K. Singh, and V. Saraswat, "Advanced persistent threats (apt): evolution, anatomy, attribution and countermeasures," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 7, pp. 9355–9381, 2023.

[14] M. Mahmoud, M. Mannan, and A. Youssef, "Apthunter: Detecting advanced persistent threats in early stages," *Digital Threats: Research and Practice*, vol. 4, no. 1, pp. 1–31, 2023.

[15] S. Quintero-Bonilla and A. Martín del Rey, "A new proposal on the advanced persistent threat: A survey," *Applied Sciences*, vol. 10, no. 11, p. 3874, 2020.

[16] A. Kumar and K. Guleria, "Comprehensive analysis and detection of multi-step cyber attacks: Insights from the multi-step cyber-attack dataset," in *2024 8th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2024, pp. 561–564.

[17] B. Al-Sada, A. Sadighian, and G. Oligeri, "Mitre att&ck: State of the art and way forward," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–37, 2024.

[18] R. Al-Shaer, J. M. Spring, and E. Christou, "Learning the associations of mitre att & ck adversarial techniques," in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[19] O. S. M. B. H. Almazrouei, P. Magalingam, M. K. Hasan, and M. Shanmugam, "A review on attack graph analysis for iot vulnerability assessment: challenges, open issues, and future directions," *IEEE Access*, vol. 11, pp. 44 350–44 376, 2023.

[20] A. T. Al Ghazo and R. Kumar, "Critical attacks set identification in attack graphs for computer and scada/ics networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 9, pp. 5604–5613, 2023.

[21] L. Sadlek, P. Čeleda, and D. Tovarňák, "Identification of attack paths using kill chain and attack graphs," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–6.

[22] K. Wang, A. Perrault, A. Mate, and M. Tambe, "Scalable game-focused learning of adversary models: Data-to-decisions in network security games." in *AAMAS*, 2020, pp. 1449–1457.

[23] P. Shukla, L. An, A. Chakrabortty, and A. Duel-Hallen, "A robust stackelberg game for cyber-security investment in networked control systems," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 856–871, 2022.

[24] Y. Zhang and P. Malacaria, "Bayesian stackelberg games for cyber-security decision support," *Decision Support Systems*, vol. 148, p. 113599, 2021.

[25] B. Liu, H. Xu, and X. Zhou, "Stackelberg dynamic game-based resource allocation in threat defense for internet of things," *Sensors*, vol. 18, no. 11, p. 4074, 2018.

[26] O. Tsemogne, Y. Hayel, C. Kamhoua, and G. Deugoue, "Partially observable stochastic games for cyber deception against network epidemic," in *Decision and Game Theory for Security: 11th International Conference, GameSec 2020, College Park, MD, USA, October 28–30, 2020, Proceedings 11*. Springer, 2020, pp. 312–325.

[27] W. Wang, X. Chen, Y. Li, and C. Zhu, "Catch the cyber thief: A multi-dimensional asymmetric network attack–defense game," *Applied Sciences*, vol. 14, no. 20, p. 9234, 2024.

[28] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[30] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PmLR, 2016, pp. 1928–1937.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[34] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps." in *AAAI fall symposia*, vol. 45, 2015, p. 141.

[35] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[36] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[37] A. Hemida, A. B. Asghar, C. Kamhoua, and J. Kleinberg, "A game theoretic framework for multi domain cyber deception," in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2024, pp. 443–447.

[38] B. Hyder and M. Govindarasu, "A novel methodology for cybersecurity investment optimization in smart grids using attack-defense trees and game theory," in *2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2022, pp. 1–5.

[39] F. Razvan and C. Mitica, "Enhancing network security through integration of game theory in software-defined networking framework," *International Journal of Information Security*, vol. 24, no. 3, p. 100, 2025.

[40] X. Bai and Y. Bai, "Equilibrium strategy of attack and defense in computer networks based on markov signal game theory," *Journal of Cyber Security and Mobility*, vol. 14, no. 1, pp. 127–153, 2025.

[41] P. Yao, B. Yan, and Q. Yang, "Game theoretical decision-making of dynamic defense in cyber-physical power systems under cyber-attacks," *ACM Transactions on Cyber-Physical Systems*, vol. 9, no. 2, pp. 1–21, 2025.

[42] F. Zhang, L. Chen, J. Yang, P. Wang, J. Zhang, Q. Wang, X. Diao, and Z. Dou, "Game-theoretic approach to cybersecurity risk assessment and protective strategy optimization in process industry production systems," *Computers & Chemical Engineering*, p. 109034, 2025.

[43] S. Rass, S. König, J. Wachter, V. Mayoral-Vilches, and E. Panaousis, "Game-theoretic apt defense: An experimental study on robotics," *Computers & Security*, vol. 132, p. 103328, 2023.

[44] I. Kalderemidis, A. Farao, P. Bountakas, S. Panda, and C. Xenakis, "Gtm: Game theoretic methodology for optimal cybersecurity defending strategies and investments," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–9.

[45] R. Kour, R. Karim, and P. Dersin, "Modelling cybersecurity strategies with game theory and cyber kill chain," *International Journal of System Assurance Engineering and Management*, pp. 1–12, 2025.

[46] R. Wang, C. Yang, X. Deng, Y. Zhou, Y. Liu, and Z. Tian, "Turn the tables: Proactive deception defense decision-making based on bayesian attack graphs and stackelberg games," *Neurocomputing*, p. 130139, 2025.

[47] Y. Tang, J. Sun, H. Wang, J. Deng, L. Tong, and W. Xu, "A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning," *Computers & Security*, vol. 142, p. 103871, 2024.

[48] T. Kunz, C. Fisher, J. La Novara-Gsell, C. Nguyen, and L. Li, "A multiagent cyberbattlesim for rl cyber operation agents," in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2022, pp. 897–903.

[49] S. H. Oh, J. Kim, J. H. Nah, and J. Park, "Employing deep reinforcement learning to cyber-attack simulation for enhancing cybersecurity," *Electronics*, vol. 13, no. 3, p. 555, 2024.

[50] Z. Li, Q. Zhang, and G. Yang, "Eppta: Efficient partially observable reinforcement learning agent for penetration testing applications," *Engineering Reports*, vol. 7, no. 1, p. e12818, 2025.

[51] R. Rishu, A. Kakkar, C. Wang, A. Rahman, C. Redino, D. Nandakumar, T. Cody, R. Clark, D. Radke, and E. Bowen, "Enhancing exfiltration path analysis using reinforcement learning," *arXiv preprint arXiv:2310.03667*, 2023.

[52] G. Mcdonald, L. Li, and R. Al Mallah, "Finding the optimal security policies for autonomous cyber operations with competitive reinforcement learning," *IEEE Access*, 2024.

[53] A. V. Singh, E. Rathbun, E. Graham, L. Oakley, S. Boboila, A. Oprea, and P. Chin, "Hierarchical multi-agent reinforcement learning for cyber network defense," *arXiv preprint arXiv:2410.17351*, 2024.

[54] C. Wang, C. Redino, R. Clark, A. Rahman, S. Aguinaga, S. Murli, D. Nandakumar, R. Rao, L. Huang, D. Radke *et al.*, "Leveraging reinforcement learning in red teaming for advanced ransomware attack simulations," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2024, pp. 262–269.

[55] A. Hernandez-Suarez, G. Sanchez-Perez, L. K. Toscano-Medina, H. Perez-Meana, J. Olivares-Mercado, J. Portillo-Portillo, G. Benitez-Garcia, A. L. Sandoval Orozco, and L. J. García Villalba, "Reinforsec: an automatic generator of synthetic malware samples and denial-of-service attacks through reinforcement learning," *Sensors*, vol. 23, no. 3, p. 1231, 2023.

[56] A. Jaber, "Transforming cybersecurity dynamics: Enhanced self-play reinforcement learning in intrusion detection and prevention system," in *2024 IEEE In-*

ternational Systems Conference (SysCon). IEEE, 2024, pp. 1–8.

[57] A. AlMajali, L. Al-Abed, R. Mutleq, Z. Samamah, A. A. Shhadeh, B. J. Mohd, and K. M. A. Yousef, "Vulnerability exploitation using reinforcement learning," in *2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE, 2023, pp. 281–286.

[58] J. Esteban, "Simulating network lateral movements through the cyberbattlesim web platform," Ph.D. dissertation, Massachusetts Institute of Technology, 2022.

[59] I. S. Thompson, A. Caron, C. Hicks, and V. Mavroudis, "Entity-based reinforcement learning for autonomous cyber defence," in *Proceedings of the Workshop on Autonomous Cybersecurity*, 2024, pp. 56–67.

VITAE

- Name: Fatima Muhammad Anis
- Nationality: Pakistan
- Date of Birth: 07-09-1999
- Email: *fatimaanis11@gmail.com*
- Permenant Address: H.No. 224, St. 100, Sector I-8/4, Islamabad, Pakistan
- Academic Background: Fatima completed her BS in Computer Science at Imam Abdulrahman Bin Faisal University (IAU) in 2022. She is currently pursuing an MS in Computer Science at King Fahd University of Petroleum and Minerals (KFUPM), with a focus on AI-driven cybersecurity, adversarial modeling, and offensive security.

Research Output and Recognition

- Weaponizing AI in Cyberattacks, A Comparative Study of AI-powered Tools for Offensive Security: The 8th International Conference on Future Networks & Distributed Systems (ICFNDS 2024), Marrakech, Morocco – (Accepted)

- Nominated for the National Cybersecurity Authority (NCA) Cybersecurity Research and Innovation Pioneer Grant (Grant Type: Master's Postgraduate Research Thesis, May 2025–May 2026).