AUTONOMOUS CYBER DEFENSE: FORMAL MODELS AND APPLICATIONS

by

Ashutosh Dutta

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Software and Information Systems

Charlotte

2021

Approved by:

Dr. Bei-Tseng Chu

Dr. Jinpeng Wei

Dr. Samrat Chatterjee

Dr. Mark Pizzato

ABSTRACT

ASHUTOSH DUTTA. Autonomous Cyber Defense: Formal Models and Applications. (Under the direction of DR. BEI-TSENG CHU)

With technological advancements, cyber attacks are highly automated and organized with asymmetric advantages over defenders regarding cost and efforts. Attackers employ sophisticated and diversified approaches to achieve attack objectives while being stealthy [1]. Therefore, enterprises strive for *autonomous* frameworks to optimize cybersecurity planning through addressing uncertainties related to attackers and the environment. However, most existing cybersecurity defense solutions are static and highly rely on human expertise, which inevitably decreases odds for defenders. This dissertation aims to advance state of the art by developing new models and frameworks to enact automated and dynamic defense planning optimization for cyber risk and attack mitigation. This dissertation has three objectives: (1) developing a framework to compute optimal cyber risk mitigation planning, (2) developing a framework to achieve real-time defense optimization against strategical cyber attacks in a stochastic environment, and (3) developing defense models to cope with dynamic attack and environment behavior.

In the second chapter, this dissertation presents formal models of an automated cyber risk mitigation framework, named CyberARM, to compose an optimal set of cybersecurity defense controls as *cybersecurity portfolio* for an enterprise. Computing a cost-effective portfolio to optimize Return on Investment (ROI) is still a highly complex and error-prone task due to the large number of security controls, and correlated risk factors (e.g., vulnerabilities and attack techniques) of an escalated and diversified attack surface. CyberARM formulates the decision-making problem as Constraint Satisfaction Problem (CSP) to compute a correct-by-construction cybersecurity portfolio. The computed portfolio wants to maximize ROI considering the

increasingly evolving threat actions after satisfying all user requirements (e.g., budget and mission-oriented constraints). Moreover, the computed portfolio answers three fundamental questions: (1) "what" security controls are needed for "which" security function (i.e., Identify, Protect, Detect, Respond, and Recover), (2) "where" to enforce (Network, Device, People, Application, and Data), and (3) "why" it is effective in the cyber attack kill chain phases. The evaluation results show that CyberARM can approximate a cost-effective cybersecurity portfolio for large enterprises applying its model reduction and decomposition approaches.

In the third chapter, this dissertations presents a multi-agent distributed cyber defense framework, named *Horde*, to defend sophisticated Infrastructural Distributed Denial of Service (I-DDoS) attacks *autonomously*. In I-DDoS attacks, attackers target core backhaul links to impede the availability of critical networks or servers while avoiding end-system defenses [2, 3]. Despite the extensive efforts in developing DDoS mitigation solutions, the sophistication and potential impact of I-DDoS attacks continue to grow significantly. To protect critical network links, *Horde* assigns autonomous agents that compute cost-effective composition of defense tactics (*i.e.,* limiting, filtering, diversion, rerouting) dynamically at real-time, considering the expected behavior of I-DDoS attackers and the network. It establishes automated collaboration among agents to share spare bandwidth for rerouting prioritized traffic of congested links through alternative routes. *Horde* formulates the problem of an agent's decision-making using Reinforcement Learning (RL) and applies Partially Observable Markov Decision Process (POMDP) to solve it, after reasoning over uncertainties of decision parameters.

In the fourth chapter, this dissertation presents models aiming to infer *expected behavior* of attacker and environment to integrate into decision-making, in order to confront dynamic I-DDoS attackers in an uncertain environment. Most existing game-theoretic DDoS frameworks struggle due to static assumptions on attackers and

critical environmental parameters. This chapter presents *incremental and online* approaches to learn currently adopted attack strategies and critical decision parameters of the network without requiring deep domain knowledge. The autonomous defense model enables *Horde* agents to evolve dynamically through observing, hypothesizing, and investigating and deciding via the *interaction experience* with the environment and attackers. This enables to not only observe and respond to I-DDoS attacks timely and effectively but also exhibit a robust behavior against evasion and deception attacks. The evaluation results on diversified attack strategies show that *Horde* agents can serve more than 97% benign traffic despite dynamic attack and network behavior and attack detection inaccuracies.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude wholeheartedly to Dr. Ehab Al-Shaer who was my advisor for the last six years. He joined Carnegie Mellon University as Distinguished Career Professor in august 2020. Nevertheless, he maintained close contact with me and guided me throughout the process. Without his help, I would never be able to reach at this destination. When I look back at my previous self (before starting my Ph.D.) who had almost zero research and writing experience, I can realize the intensity of experience accrued during my Ph.D. journey. He always gives his best efforts to overcome my weakness and harness my skills. I want to thank him for those prolonged brainstorming sessions, discussions on research tasks, and constructive feedback, for which, I cannot be anything but grateful. I consider myself lucky enough to get an advisor and professor on whom I can rely without any doubt. I would also like to express my gratitude to Dr. Bei-Tseng (Bill) Chu who took my responsibilities at very crucial time. I will always be grateful to him for advising me throughout my dissertation writing process and giving his best efforts for me.

Ph.D. is a comparatively long academic journey that is hard to finish without constant support from family and friends. I want to thank my parents, wife, sisters, brothers, nephews, and nieces for their tremendous efforts of cheering my life. I consider myself lucky enough to get blessed with a family that can relieve my depression with love and desperation with happiness. I am grateful to my father and mother who care only about my smile – not academic or professional success. It is such a relief from stress throughout my journey of life, that eventually gives me more courage to confront challenges. While living 8,000 miles away from my close people, my wife, Moumita, is here with her utmost efforts – passionately compromising for my whimsical characteristics. She reminds me a quote of Paulo Coelho, "And, when you want something, all the universe conspires in helping you to achieve it." I thank the universe for conspiring me to choose her as my significant one. I want to express love

to my sisters, bordi (Shoma) and chordi (Shanta), for supporting me undoubtedly, pampering me, and loving me for whatever I am.

I would love to shout out to all my friends who are always with me with their support and inspiring words. From my childhood to Ph.D., I am always bestowed with a lot of friends, without whom, this journey would have lost many colors. I am grateful to my roommate, Sakib and Madiha, for tolerating me, which was definitely not so easy. I want to thank my colleagues and friends, Rakeb, Mohiuddin, and Ehsan, for their companion. I will never forget the difficult time that we all tried to cross in the same boat – quite an experience. Finally, I would like to acknowledge myself for being patient to finish the journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CyberARM Cyber Automated Risk Mitigation.

DNN Deep Neural Network.

MDP Markov Decision Process.

POMDP Partially Observable Markov Decision Process.

RL Reinforcement Learning.

RNN Recurrent Neural Network.

SMT Satisfiability Modulo Theories.

CHAPTER 1: Introduction

The advancement of technologies has engendered the emergence of diversified devices and applications. These devices and applications enable the processing and sharing of information with better flexibility and control by connecting with cyber. Indeed, applications of cyber have stretched from personal domains such as home automation to critical national infrastructures such as smart grid, gas, and oil grid. Due to such reliance on cyber, any interruption induces a significant negative impact on our lives and business. The incident of hacking an electric utility's SCADA (Supervisory Control and Data Acquisition) system in Ukraine is such an example, that caused approximately 6 hours of power blackout for a quarter of a million people [4].

The exponential rise of cyber connectivity with heterogeneous assets (e.g., devices, applications, services, and others) has made operating and managing cyber extremely complex. New vulnerabilities never cease to appear due to erroneous configurations or weakness in software/firmware. Moreover, many vulnerabilities remain undiscovered until being attacked (i.e., zero-day vulnerabilities), and many remain unpatched due to compatibility issues or tendencies to ignore software/firmware update. Hence, cyber expansion is also responsible for the rapid escalation of the cyber attack surface (i.e., available attack approaches to exploit system vulnerabilities). Therefore, any cyber system such as Cyber Physical Systems (CPSs) are more susceptible to cyberattacks than ever before.

Despite investing a lot of money for cyber risk mitigation, cyber incidents still tend to escalate. In fact, no money is adequate to deploy a bullet-proof risk mitigation planning against such an extensive and diversified attack surface. Therefore, enterprises quest for a cost-effective cyber risk mitigation planning, termed as cy-

bersecurity planning, to minimize the risk within the limited budget considering the potential attack surface and its sophistication. However, manual planning is infeasible due to complex cyber architecture, attack sophistication, and numerous correlated cyber factors (e.g., threat, attack techniques, vulnerabilities, and others). The situation exacerbates further due to diversified enterprise-oriented policies regarding business, operations, and others. Therefore, manual defense configurations are often not only non-optimal but also erroneous.

Cyber adversaries nowadays adopt automated tools to launch the attack at high speed with low traces. According to NETSCOUT, it only takes five minutes to attack an IoT (Internet of Things) device after being connected with the internet, and Official Annual Cybercrime Report (ACR) has predicted that there will be a ransomware attack in every 14 seconds [5]. Sophisticated adversaries discover deployed defense plans and evade these with innovative approaches. They follow a specific strategy that maximizes attack objectives by observing the network condition and consequences of past attack actions. In fact, seven out of ten predominant MITRE ATT&CK techniques [6] of last year belong to *Discovery* attack tactic [7]. They evolve to encroach into critical systems or applications by exploiting new vulnerabilities or old vulnerabilities with innovative attack techniques. The attacker infers entry points, current defense configuration, or effective attack vectors through reconnaissance and discovery. For example, the attacker can use CrackMapExec and Kwampirs to discover the password policy [8]. He can use BRONZE BUTLER to determine follow-on behaviors or check the infection status on victims [9]. Moreover, state-sponsored and coordinated attacks such as Advanced Persistent Threat (APT) attacks are well-resourced to exert advanced automated techniques to attack critical infrastructures persistently while being stealthy. As a result, they provide little time to react while disabling countermeasures, counter-attacks, or human interference in the first place.

On the contrary, enterprises relying on static defense planning and human interfer-

ence to detect or respond to cyber attacks are slow to defend such automated cyberattacks. Therefore, there exist profound demands for automated cyber decision-making frameworks to effectuate optimal defense plannings. These frameworks endeavor to minimize cyber risk while satisfying all requirements regarding budget, operation, safety, and security [10] of an enterprise or infrastructure. However, to compute an effective risk mitigation plan, defense optimization frameworks must address the following challenges: (1) asymmetric cyber warfare, (2) lack of domain-specific data, (3) establishing automated collaboration and coordination, and (4) scalability.

The first challenge is *asymmetric cyber warfare* that arises due to the attacker's asymmetric advantages over the defender regarding cost and efforts [10]. While the defender has to concern all possible attack approaches, a single loophole or vulnerability may be enough for the attacker. Additionally, cybersecurity countermeasures are generally more expensive, approximately four times of attack cost, due to installation and utility cost (e.g., loss of usability due to restricted access) [11]. According to InfraGard reports, a low-end cyberattack causing $34 to launch could return $25,000 in a month, while the expensive-end attack causing a few thousand dollars could return more than $1 million [12]. Importantly, attackers can easily afford to try various attack approaches due to the low cost of attack tools that can be as low as $1 in dark web [5]. Therefore, the defense optimization framework must predict the imminent potential cyberattacks to prioritize them for mitigating. Besides, to cope with failures of proactive approaches, defense configurations need to be reactive to respond to cyberattacks or recover from attack impact. Indeed, layered defense-in-depth to avoid single-point-of-failure is the fundamental requirement to implement a resilient defense configuration. Moreover, to react to the attacker's strategical adaptions effectively, the defense framework must infer the attacker's strategical adaptations, at least probabilistically, to minimize the cyber risk or attack impact.

The second challenge arises from the *lack of domain-specific data*, which makes for-

mulating the cyber environment with numerous correlated factors very hard. Besides, many of these factors exhibiting uncertain properties can only be observed partially with incomplete and imperfect information. For example, all sensitive network links may not be observed altogether to detect suspicious activities due to limited energy, which induces *incompleteness* into the gathered information. Without domain-specific data and deep domain knowledge, deterministic or static approaches of formulating stochastic environment behavior become overly conservative. Moreover, cyber infrastructures generally have diversified requirements that must not be violated to maintain their expected behavior, safety, and security. Many of these requirements depending on dynamic environment properties, can only be known after going to operations [13]. This necessitates the refinement of defense policies *dynamically* by *actively* learning the environment.

The third challenge, *establishing automated collaboration and coordination*, needs to be addressed to enable optimal defense configurations across the network. Distributed attacks such as link-flooding DDoS (Distributed Denial of Service) attacks [3, 2] demand collaborations among ISPs or ASPs for effective defense, which are generally indefensible at the target enterprise premises. Besides, flooding the downstream ISPs' links also incur unaffordable network utilization at upstream ISPs. Hence, upstream network points or cyber entities should step up and collaborate among themselves to defend such distributed attacks. Here, cyber entities represent ISPs, domains, ASPs, sub-domains, enterprises, or departments. However, there still lack of consensus collaborative intentions among these entities, who are mostly apathetic to it due to the lack of automation.

The fourth challenge, *scalability*, needs to be addressed to cope with the exponential growth of problem space due to the size and exponential expansion of cyber. The computational complexity of decision optimization algorithm must not grow significantly with the increase of cyber, especially for real-time optimization.

In both industry and academia, researchers have put tremendous efforts into automating cybersecurity planning applying different approaches such as formal modeling, game theory [14], machine learning, sequential decision process [15, 16], reinforcement learning [17, 18], and others for past several years. Though these works have provided insights about the basic properties of automated defense frameworks, these frameworks have several limitations regarding dynamic defense optimization. *Firstly*, most of these frameworks consider static or strict assumptions about attackers, which restrict the scope of attack models unrealistically. As a result, these frameworks fall short in defeating dynamic and adaptive attackers. *Secondly*, these frameworks ignore uncertainties of dynamic environmental factors (e.g., background traffic, hardware failures, sensor errors, and others), that deviate the reality far from the modeled or simulated environment. *Thirdly*, these frameworks' objective functions only focus on local cyber risk mitigation of an enterprise without exploring the scope of collaborations. *Fourthly*, these frameworks fail to tackle the exponential growth of computational complexity with the increase of critical resources. Hence, these are incompetent for real-time defense optimization considering dynamic attackers and correlated environmental factors.

The objective of the dissertation is to develop methods, models, and frameworks for automating optimization of cybersecurity decision-making to employ dynamic, context-aware, and cost-effective cybersecurity planning against sophisticated and adaptive attacks. Notably, context depends on the current environment/network condition and currently adopted attack strategy. This dissertation presents feasible and scalable models to solve real-world cybersecurity decision problems and automate collaborations among heterogeneous entities. However, this dissertation does not discuss any financial model to establish incentives for collaborations.

This dissertation mainly focuses on two problems: (1) optimal defense resource allocations of an enterprise through recognizing potential attack surface and satisfy-

ing diversified business or mission-oriented policies, and (2) distributed multi-agent *dynamic* decision-making against strategical and adaptive I-DDoS (Infrastructural Distributed Denial of Service) attacks. The first research problem investigates data-driven planning of coarse-grained security countermeasures, which is discussed in chapter 2. The other research problem investigates experience-oriented planning of fine-grained and dynamic defense strategies against I-DDoS attacks, which is divided into two problems and discussed in chapter 3 and chapter 4.

In summary, this dissertation solves the following research problems:

- The first research problem aims to compute a cost-effective Cybersecurity Portfolio for an enterprise after satisfying all of its requirements regarding business, mission, security, and others.

- The second research problem aims to develop a multi-agent distributed framework to optimize I-DDoS defense composition autonomously at real-time to defend adaptive I-DDoS cost-effectively. This framework to be deployed by upstream network points must be flexible to collaborate with other upstream network points.

- The third research problem aims to develop models to integrate the *Evolve* capability into I-DDoS defense framework to address the dynamic behavior of the network and attackers.

1.1 Motivation

This section discusses motivations of research problems of this dissertation.

- ***Automated Cyber Risk Mitigation Optimization:*** The exponential expansion of cyber has not only expanded the attack surface but also increased security countermeasures or controls to defend those. Security guidelines such as CIS Critical Security Control [19], MITRE [6] gather a comprehensive set

of defense approaches, but it is infeasible to deploy all these mitigation approaches with a limited budget. Moreover, enterprises generally have many requirements that, if violated, induce colossal loss. For example, a web application hosting advertisements should not adopt white-list based URL filtration, as it will not conform with business objectives. Therefore, enterprises wish to conduct cybersecurity investments in a way that it will minimize the risk within the affordable budget while conforming with all given requirements. According to MarketsandMarkets (a market research firm), the cybersecurity investment market will reach at $248 billion that is almost twice the current market by 2023 [20]. However, enterprises still fail to invest appropriately, and a survey of over 1,500 organizations reveals that approximately two-thirds of organizations' defense plannings are ineffective [21]. As a result, cyberattacks continue to haunt enterprises, and are predicted to appear with more intensity and sophistication in the near future. Only in 2020, induced global loss due to successful attacks has exceeded $1 trillion [21].

Deploying an optimal cybersecurity investment plan, known as Cybersecurity Portfolio, has become a pivotal responsibility of a CISO (Chief Information Security Officer) or CSO (Chief Security Officer) of an enterprise. Importantly, besides minimizing the cyber risk, a cybersecurity portfolio must satisfy all requirements regarding budget, operation, usability, safety, and others of an enterprise. However, it is very challenging because there exist (1) complex correlations among components of attack vectors and vulnerabilities, (2) heterogeneous effectiveness (i.e., success probability against an attack) of security countermeasures, (3) enterprise oriented resiliency requirements, (4) diversified business or mission oriented policies/requirements and regulations, and (5) non-linear growth of problem space with the increases of asset list. This is why previously adopted ad hoc and manual approaches to optimize cybersecurity

investment plan is not only error-prone but also infeasible for a modern-day enterprise with a large number of assets. Hence, computing the cost-effective cybersecurity portfolio is still the holy grail of cybersecurity.

- ***Autonomous Cyber Defense Against Adaptive I-DDoS:*** Despite extensive efforts to defend against Distributed Denial of Service (DDoS) attacks, the sophistication of DDoS continues to evolve to defeat advanced defense techniques. According to Neustar (a telecommunication company), there was a 151% increase of DDoS incidents only in the first half of 2020 compared to 2019, with a significant spike in attack innovations and sophistication [22]. Infrastructural Distributed Denial of Service (I-DDoS) is one variant of sophisticated attacks, that aims to bring devastating impact on critical infrastructures of victims [23]. In I-DDoS attacks, attackers target core/critical backhaul links carrying a significant portion of traffic of targeted critical networks or servers, in order to impede the availability of those critical servers or networks [3, 2]. There are two variants of I-DDoS attacks: (1) *Direct*, and (2) *Indirect*, based on intended destinations of attack traffic. In *Indirect* I-DDoS, instead of sending traffic directly to target networks, the attackers send traffic to selected decoy bots or decoy servers residing in the victims' neighborhood to congest critical links. Notably, in both *Direct* and *Indirect* I-DDoS, downstream end-system defenses cannot resist these attacks effectively as congestion at upstream links already blocks traffic before reaching to downstream network points [24, 25, 26].

  Defending sophisticated I-DDoS attackers is challenging because they adopt mixed strategies with varying (1) traffic rate (aggressive or low), (2) bots' location distribution (sparse or dense), and (3) number and location of decoys servers to maximize the stealthiness while maintaining menacing attack intensity. Instead of always relying on elephant flows (i.e., large continuous traffic flow) only, they may emphasize on shrewd attacks (i.e., comparatively low-rate

attacks) for imitating legitimate traffic behavior to deceive Intrusion Detection System (IDS) or attack flow classifiers. The situation aggravates for *Indirect* attacks as victims remain ignorant about attack traffic. Nowadays, attackers can afford to launch such distributed and adaptive attacks due to the easy availability of compromised end-points, commercialized DDoS servers, and automated tools for attack coordination [27]. In fact, I-DDoS attackers are highly advanced who learn about network conditions and adapt their strategies accordingly to sustain the attack impact while evading detection and mitigation. According to Neustar, almost all mitigated DDoS threats had multiple attack vectors while 52% attacks exerted three or more vectors [22]. Therefore, enterprises spanning from financial to manufacturing, healthcare to streaming, or small to large are at continuum risk of I-DDoS that can completely cripple network infrastructure to deny the availability of services.

1.2 Research Objectives

The main objective of this dissertation is to design automated cybersecurity decision-making frameworks to dynamically optimize defense strategies against adaptive and diversified attackers, through solving challenges related to limited resource, incomplete or imperfect knowledge about environment, mission or business oriented policies, state space explosion, and stochastic environment behavior. This section describes research objectives of chapter 2, 3, and 4 respectively:

- ***Automated Cyber Risk Mitigation Optimization:*** The goal of this research is to develop an automated decision-making framework for CISOs, called *CyberARM*, to compute the cost-effective Cybersecurity portfolio within a limited budget, considering all correlations among attack vectors, vulnerabilities, assets, and security countermeasures. Besides, the portfolio must be able to satisfy enterprise specific resiliency requirements and all mission, business, security, or usability oriented requirements. One of the fundamental contribution

of the work is prioritizing the attack surface of an enterprise that, if mitigated, improves Return On Investment (ROI) significantly. Moreover, the computed portfolio must reason over following questions: (1) *What* security functions (i.e., identify, protect, detect, respond, and recover) it will offer, (2) *Where* (i.e., at which layers of environment) it will apply security functions, and (3) *When* (kill chain phase) and against which threat actions it will defend. To optimize defense plan through answering mentioned questions, this dissertation aims to develop a systematic way to categorize security countermeasures based on their defensive traits. This framework ensures the scalability for enterprises with large number of assets and diversified requirements.

- ***Autonomous Cyber Defense Against Adaptive I-DDoS:*** The goal of this research is to develop a multi-agent distributed framework, called *Horde*, to optimize defense composition *autonomously* against adaptive and strategical I-DDoS attacks at a stochastic environment. The framework assigns a dedicated and autonomous agent for each of the critical links, that independently optimizes the defense composition of traffic limiting/filtering and traffic diversion (e.g., traffic rerouting) at its assigned link. *Horde* must enable automated collaborations among agents of same and other *Horde* for sharing link bandwidth, in order to deploy traffic rerouting to avoid congestions. An agent's decision model must be scalable to ensure defense optimization *dynamically* at real-time through addressing stochastic network behavior and integrating the probable attack behavior into decision-making. Besides, the decision model must address uncertainties related to environmental decision parameters such as understanding current link condition, IDS uncertainties in distinguishing attack traffic, and others. At any specific time-sequence, an agent tries to answer following questions: (1) *which flows need to be dropped?*, (2) *which flows need to be rerouted?*, and (3) *which flows need to be allowed through regular routes?* The aggregation

of all agent's independent decision-making collectively maximizes the benign traffic serving while also maximizing the benign traffic drop.

- ***Integrating Evolve Capability into Defense Model:*** The goal of this research is to integrate *Evolve* capability into *Horde* agent's decision process, so that, it can cope with dynamic behavior of the environment. The objective of this chapter is two-fold: (1) predicting the attack behavior, and (2) optimal tuning of critical decision parameters. Understandably, without understanding the attack behavior, defense optimization is not feasible. Due to lack of data on diversified and adaptive I-DDoS attack strategies, *Horde* must deploy an incremental approach that can learn the currently adopted attack strategy based on previous interactions with the attacker. Such decision model must not only be able to cope with attack adaptations but must also be robust against attack deceptions. This chapter focuses on tuning critical decision parameters such as system dynamics (i.e., regulating changes of link condition due to attack and defense interplays) and discount factor (i.e., regulating future impact). Moreover, *Horde* agents must cope with IDS inaccuracies that mainly change with the stealthiness or aggressiveness of attack behavior. Another objective of this chapter is to evaluate *Horde* against diversified I-DDoS attack strategies in a dynamic setting.

1.3 Background

This section describes some modeling approaches that are applied to solve problems of this dissertation.

1.3.1 Satisfiablity Modulo Theories (SMT)

SMT is a form of Constraint Satisfaction Problems [28], that generalizes boolean SAT [28] theories to deal with integers, real numbers, arrays, and other data structures efficiently. Moreover, SMT enables extensive formal modeling approaches com-

pared to SAT, and SMT-solvers are powerful tools to solve constraint satisfaction problems in many domains such as discovering security vulnerabilities, hardware and software verification, counterexample generation, scheduling, and planning, solving graph problems, and others [29]. SMT formulas are expressed in classical first-order logic, where predicates from many background theories replace binary variables of SAT. Modern SMT-solvers can satisfy formulas with thousands of variables and millions of clauses [30]. SMT-solver is applied to compute risk mitigation planning that optimizes decision-making with bounded rationality.

1.3.2 Sequential Decision Process (SDP)

A sequential decision process (SDP) is the process of optimizing an agent's decision making. The agent has to interact synchronously with the external environment by acting and observing [31] the consequences of actions. To apply SDP, the subjected environment needs to (1) exhibit stochastic behavior that means the consequence of an action is not always the same, and (2) satisfy the markovian property that specifies the transition to the next state depends on the current state and action. In a stochastic environment, the decision-making approach needs to be sequential rather than action planning to address the dynamic properties of the environment.

The sequential decision process model consists of the following parameters:

- Set of states, S, where a state defines a distinct condition of the environment,
- Set of actions, A,
- State transition function, $T : S \times A \longrightarrow S'$, where $T(s, a, s') = P(s'|s, a)$ specifies the probability of transition to next state $s' \in S$ from the current state $s \in S$ and action $a \in A$.
- Set of observations, Ω,
- Observability matrix function, $O : S \times A \longrightarrow \Omega$, where $O(s, a, o) = P(o|s, a)$

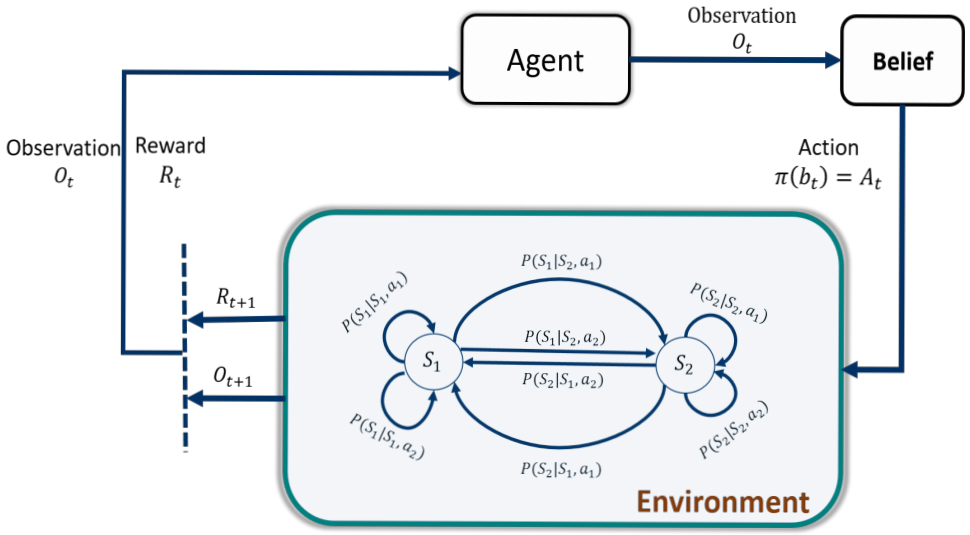


Figure 1.1: An example of Sequential Decision Process. The environment consists of two states S_1 and S_2, and two actions a_1 and a_2. The computed policy π determines the optimal action A_t for the given belief (probabilistic estimation of current state) b_t at current time-sequence t.

specifies the probability of observing $o \in \Omega$ at the current state $s \in S$ when the recent executed action is $a \in A$,

- Reward function, $R : S \times A \times O \times S' \longrightarrow \mathcal{R}$, where $R(s, a, s', o)$ specifies the reward, $\mathcal{R}$, achieved by executing action $a \in A$ that transits the environment from current state s to next state s' when the observation is $o \in \Omega$,

- Discount factor, $\gamma \in [0, 1)$, that regulates how far the agent will look to understand consequences of considered actions.

Several algorithms such as value iteration and policy iteration can solve the decision-process model to compute a policy [31, 32]. The computed optimal policy aims to maximize the accumulated rewards by executing the optimal sequence of actions throughout time-sequences based on conditions of the environment. The computed optimal policy, π, determines the optimal action for a given belief. Belief is a probabilistic distribution over possible states, that contains a value for each state $s \in S$ specifying the probability s as current state s_t. The belief function depends on state

transition probabilities, conditional observation probabilities, and current observation.

Fig. 1.1 illustrates an example of sequential decision process. At the start of current time-sequence t, the agent determines belief b_t based on the recent observations (e.g., network symptoms, senor values) o_t. The computed policy π determines the action A_t for the current belief b_t. After executing A_t, at the next time-sequence $t+1$, the agent observes O_{t+1} and determines reward R_{t+1}.

Based on observability and knowledge about the environment, there are three approaches of the sequential decision process:

- **Markov Decision Process (MDP):** MDP assumes the full observability of the environment [31]; hence, the agent can determine the underlying state certainly. Therefore, only one value in belief vector is one, and all other values are zero. Hence, MDP is only a tuple of 5 parameters: (S, A, T, R, γ).

- **Partially Observable Markov Decision Process (POMDP):** POMDP is used when the environment is only partially observable [33]. Due to imperfect and incomplete information, the agent cannot determine the underlying state certainly. Hence, the agent determines the belief that characterizes the current state probabilistically based on recent observation. POMDP model consists of all 7 parameters of SDP: $(S, A, T, \Omega, O, R, \gamma)$.

- **Reinforcement learning (RL):** RL is another type of sequential decision process, where the environment is unknown [34]. Therefore, the agent does not know state transitions or observation matrix. It differs from supervised learning as there is no dataset to dictate or model its optimal action. Instead, the agent learns or approximates the environment based on its observations about the consequences of executed actions.

  One of the challenges in RL is to balance the trade-off between exploration and

exploitation. Here, exploitation means leveraging previous knowledge gained by exploring consequences of past actions, and exploration specifies gaining new knowledge by executing unexplored actions. Most of the current model-free reinforcement learning approaches only require four parameters: S, O, A, and R, and mapping from current observations to the current state is handled implicitly in the decision optimization model. A relatively new approach, named Deep Reinforcement learning (DRL), solves RL using deep learning and has shown good results for complicated games like AlphaGo, Atari, and others [35].

1.4 Research Challenges

In order to achieve all research objectives, this dissertation addresses following challenges:

- Computing a cost-effective cybersecurity portfolio for an enterprise is a challenging task, because there exist various security countermeasures that diverge in their defense approaches. These security countermeasures despite having same objective (e.g., protecting web-browser) differ not only in defense effectiveness but also in cost regarding installment or deployment. Moreover, the problem intensifies due to numerous and diversified enterprise-oriented requirements.

  To address the challenge, this dissertation formulates the problem of optimal defense planning as Constraint Satisfaction Problem (CSP) that aims to maximize ROI. This model considers all requirements as constraints of CSP, that makes it flexible in considering any enterprise-specific requirement and expert knowledge such as "Do not deploy white-list based url filtration for the advertising web-application". Using the Markov chain property, this approach correlates security countermeasures having heterogeneous effectiveness and cost with attack vectors consisting of threat, threat actions, and vulnerabilities. The formulated model computes an optimal security planning that induces cost less

than the affordable budget (in dollars). It guarantees to keep the residual cyber risk (i.e., unmitigated cyber risk despite deploying security countermeasures) within a tolerance level after satisfying all enterprise-specific requirements.

- Recognizing the relevant attack surface for an enterprise requires formulating an appropriate metric for attack prioritization, which is still a well-sought research problem. Modeling such attack prioritization metric is challenging due to correlations among attack components (i.e., threat, threat actions, vulnerabilities), varying exertion likelihood of threat action depending on data and assets' types, non-homogeneous inter-dependencies among threat and threat actions, and missing attack incident reports. Additionally, multiple vulnerabilities existing at the same asset may have heterogeneous exploitability (i.e., risk of a vulnerability being exploited).

  To address the challenge, this dissertation presents a metric that formulates the risk due to the threat of a particular attack (e.g., hacking, phishing) by predicting the exertion likelihood of its associated threat actions (e.g., sql-injection, send email-attachment). CyberARM estimates the exertion likelihood of a threat action against an asset based on not only attack incident and vulnerability scanning reports but also based on its type (e.g., desktop, application) and domain (e.g., financial, public news). Instead of considering directly, CyberARM considers the impact of a vulnerability through its associated threat actions that have non-negligible exertion likelihood. Hence, even though multiple vulnerabilities with varying severity may exist in the same asset, risk due to them will be neither overestimated nor underestimated. Moreover, by leveraging associations among threat actions and existing vulnerabilities, CyberARM considers the impact of a *potential* threat action that is missing in available attack incident reports.

- Satisfying diversified requirements exacerbates the problem of defense optimization not only from the computational perspective but also from the implementation perspective. To address the high granularity of requirements, the planning must consider more in-depth reasonings for recommending specific subset of security countermeasures. Moreover, the framework must be flexible to incorporate any new constraint/requirement or changes of existing requirements, and inefficient modeling of any of such constraints deviates the defense plans from desired outcomes.

  To address the challenge, this dissertation presents a data model that defines the approach to select relevant security countermeasures (security products) through selecting security controls (i.e., particular mechanism of defense). This model provides details reasonings about why a security control is selected, and how it maximizes ROI. Moreover, CyberARM contains a categorization of security controls based on their defense traits. Thus, CyberARM is capable to integrate fine-grained requirements relevant to threat action and defense phases into its decision model.

- The computational complexity grows exponentially with the increases of assets of an enterprise, that makes computing optimal portfolio for a large enterprise *automatically* very hard. The increase of assets expands the attack surface, as well as requirements, that consequently expand the candidate-set of security countermeasures. As a result, discovering the most cost-effective plan from numerous combinations of countermeasures is computationally very challenging.

  To address the challenge, this dissertation presents two heuristic approaches that CyberARM leverages for model reduction and decomposition. These heuristic functions improve the computation complexity by pruning the problem space significantly. In order to increase cost-effectiveness of recommended planning,

CyberARM applies an incremental approach that continues trying to improve the previously discovered plan through toughening minimum cost-effectiveness (i.e., ROI) requirement, until it fails to find a better solution.

- Effective defense planning to impede I-DDoS requires predicting attack behavior by inferring currently adopted attack strategies. However, this is very challenging as attackers adapt their strategies according to their observations about the current network condition and previous attack consequences. Besides, sophisticated attackers may also execute random actions to deceive the attack prediction model. As attack strategies largely depend on the domain or environment behavior, domain-specific data is required to train the prediction model, which is hard to get.

  To address the challenge, this dissertation applies time series forecasting [36] to predict the next attack behavior probabilistically, based on previous attack actions observed by agents across the network. To implement time series forecasting, it uses Recurrent Neural Network (RNN) that can infer dynamic temporal behavior based on sequences in observed attack actions. This approach follows never ending learning [37], that actively learns the attack behavior by updating the attack prediction model incrementally with new attack observations. Moreover, it can not only learn new attack strategies but also detect attacker's strategical adaptations and attack deceptions without requiring any preliminary domain-specific data.

- To understand the effectiveness of defense actions against I-DDoS, the agent needs to understand consequences of attack and defense interplays, that depend on the behavior of the environment. However, the environment consists of many correlated and uncertain factors such as the amount of benign traffic, physical link failures, unanticipated queuing delays, current attack behavior, and others.

Moreover, due to lack of domain-specific data or deep knowledge, integration of these factors implicitly into decision-making, even probabilistically, is infeasible. Besides, attack distinguishability that also regulates defense effectiveness is not always the same. Therefore, it is very hard to formulate the behavior of the environments considering all these correlations and uncertainties.

To address the challenge, this dissertation formulates consequences of attack and defense interplays using the Markovian property, which specifies that the probabilistic transitions from current state to next state depend on the combination of attack and defense actions. These state transitions are uncertain due to not considering other dynamic environmental factors explicitly for inferring next state. To determine these probabilistic state transitions, the agent leverages a Q-table that does not require any complex model of environment. This Q-table updates its values based on its observations throughout the passage of time. The evaluation empirically shows that using this approach, the agent converges to the optimal defense decision-making within reasonable time.

- Alongside the behavior of the network that can only be observed partially, I-DDoS defense optimization depends on attack behaviors. Hence, the I-DDoS defense decision-making needs to be formulated as a partially observable stochastic game (POSG). However, the POSG problem with multi-objective rewards (different payoffs for different players) can only be approximated for small or limited scenarios. Therefore, real-time defense optimization for upstream network points having many critical resources is computationally infeasible using POSG.

  To address this challenge, this dissertation presents a novel approach to integrate expected attack behavior at current context into the defense agent's decision-loop. Thus, it reduces the POSG problem into a single agent (defense

agent) decision-making problem solvable by POMDP. Many existing algorithms can approximate POMDP solution with closer proximity to the optimal solution. Alongside improving the computation complexity of the problem, this approach enables the incorporation of different payoffs of different players. To ensure dynamic defense optimization while not losing granularity due to problem reduction (i.e., from POSG to POMDP), the POMDP model is dynamically updated based on recent experience and learning of attack behaviors and critical factors.

- The emergence of optimal aggregated/global defense policy with multi-agent defense planning is very challenging, because all agents' defense planning must orchestrate in an effective way. Besides, all agents need to perform asynchronously to avoid delays of synchronous decision-making. However, integrating all agents' decision-models *explicitly* into an agent's decision-optimization is infeasible due to high computational and modeling complexities. Besides, it arises another challenge – *which agent's model will determine first and what should be the correct modeling sequence?* Additionally, due to a lack of domain-specific data, integrating the expected behavior of other agents into an agent's decision process is not generally possible.

  To address this challenge, this dissertation divides the problem of computing global optimal policy into multiple sub-problems, where an agent solves its associated subproblem without concerning other agents' behavior. To do so, this dissertation decouples the agents' collaboration model (aims to improve other agents' conditions) from its decision model (aims to optimize its own local defense planning). The presented collaboration model establishes agents' interactions regarding bandwidth sharing with the guarantee of no conflict and wastage of spare bandwidth. Because of the decoupling, an agent's decision model only focuses on maximizing benign traffic serving through its link.

1.5 Related Work

This section describes the current state-of-the-art relevant to research problems of this dissertation.

1.5.1 Automated Cyber Risk Mitigation Optimization

Economic models of cybersecurity risk management focus on critical assets, vulnerabilities, and threat likelihood to mitigate the risks while maintaining the balance between cost and benefit of security investment [38], [39], [40]. However, most of these frameworks are very coarse-grained and rely only on qualitative values to compute and assess cyber risk mitigation plans. As these frameworks are not scalable or flexible enough to work with continuous quantitative values, they may fail to assess a defense plan in mitigating risk with fine-grained reasonings [41]. As a result, these works cannot distinguish among defense plans regarding benefits and costs appropriately. In general, organizations quantify ROI to evaluate the benefit of the security portfolio compared to its installment, deployment, and usability cost [42], [39]. Yet determining asset values and threat or threat action rates properly is itself a challenging problem, and uncertainties of these values also introduce uncertainties in the effectiveness of security countermeasures. It is observed in [43] that such uncertain values cause deviations from optimal plannings. This work used a multi-objective multiple-choice knapsack for its security planning optimization. Some researchers applied fuzzy approaches to address uncertainties such as [41] that considers fuzzy value for threat rates to estimate the risk. This approach has used a genetic algorithm to compute the near-optimal security portfolio.

Researchers have solved zero-sum control games to define how security controls need to be implemented and how frequently (e.g., daily scanning, monthly scanning) these proactive security controls need to act [44]. However, these frameworks do not consider correlations among threat and threat actions, which made defense plan

assessment very hard due to failing to address sensitivity related to correlations. Some researchers tried to minimize average expected loss by quantifying Value at Risk (VaR) that specifies how much risk of cyber attacks is still existing [45]. Some researches aimed to minimize the worst case loss by quantifying Conditional Value at Risk (CVaR) that specifies the maximum possible loss considering potential cyber attacks [46].

Security portfolio is also assessed by quantifying the lowest expected profit (PaR) that anticipates the lowest expected return due to deploying considered security portfolio [47]. This research also considered customers' willingness to pay for service into the cost function. On the other hand, researchers have proposed Tailored Tabu Search (TS)-based heuristic approach [48] and heuristic genetic algorithm [49] to maximize the mitigation of the existing vulnerabilities in the network or system. However, most of these frameworks consider same likelihoods of threat actions for all enterprises, which may vary with asset types and enterprises in real-world scenarios. Though attacks may exploit multiple vulnerabilities [50], their success likelihoods of exploitation do not remain the same. Hence, the checklist based mitigation cannot evaluate the security portfolio. In contrast, the presented CyberARM in this dissertation not only considers correlations among threat, threat actions, and vulnerabilities but also correlates vulnerability exploitation with asset types and domains. Besides, CyberARM addresses their probabilities based on both cyber incident reports and vulnerability reports while addressing the likelihood of missing reports. Importantly, these probabilities vary across the enterprise, asset types, and domains.

1.5.2 Autonomous Cyber Defense Against Adaptive I-DDoS

Over the years, researchers have proposed many automated techniques to defend DDoS attacks [51]. Though their applications have several limitations, they provide valuable hints about mitigating DDoS attacks.

1. ***Detections or Deterrence Based Frameworks:*** Researchers endeavor to cre-

ate profiles of legitimate sources, in order to distinguish attack flows [52] or malicious sources [53], based on packet rate statistics [54], wavelet analysis [55], time-series behavioral deviations [56], and packet properties [57, 58, 59]. To detect attack bots, SPIFFY [60] leverages virtual bandwidth expansion, and Defense by Offense [61] encourages sources to increase traffic rate. Both these works assume full upload bandwidth consumption by these attack bots. However, such assumption is not necessarily true due to cheap price of bots, and also, it may deteriorate the congested conditions further. There are frameworks that ask sources to solve puzzles [62], [63] to deter attack bots, which demand extra computations from benign users too. Some frameworks apply Pushback [64], packet marking [65, 24], history [66], score [25], or hop-count [26] based filtering approach to minimize or deter attack traffic transition. However, one of the severe limitations of these filtering and puzzle or offense based deterrence is the computation overhead induced to process massive traffic.

2. ***Bandwidth or Route Isolation Based Frameworks:*** Bandwidth isolation approaches based on destination provided capabilities [67, 68], trust-domain [69], and inter-domain bandwidth isolation [70] seek to guarantee services for top privileged users. Some approaches deter attackers by packet rerouting [71], topology obfuscating [72], or Virtual Network placement and migration [73]. Firstly, these approaches demand dedicated spare bandwidth which is expensive and can only protect top prioritized users. Secondly, sources with capabilities or trust may be compromised and flood the network easily. Nyx leveraged BGP poisonings to avoid the congested link and setup a minimized detour to ensure service to prioritized ASes [74], but it cannot guarantee the desired path isolation due to the visibility of BGP messages to other ASes [75].

3. ***Defense Composition Frameworks:*** Game theory based defense frameworks such as DDos traffic injection game [76], optimizing firewall setting against bandwidth

depletion [77, 78], fair bandwidth allocation among all genuine IP addresses [79], and signaling game based service hoping strategy [80] are presented, but these works cannot offer required defense adaptations at real-time against adaptive I-DDoS attackers. Game-theoretic frameworks have been proposed to mitigate DoS and DDoS attacks on SDN networks [81]. Researchers aim to infer attack strategies through modeling the attack intents and objectives to choose the appropriate game between attackers and defenders [82], which is only feasible for known or static domains. Moreover, these works are rigid in modeling attack behaviors while also failing to integrate uncertainties of many critical factors. A bayesian network-based semi-form game integrates strategic thinkings [83] of attackers and defenders from the same sophistication levels, but it relies on static attack utility function which is hard to approximate under environment uncertainties. Researchers also applied sequential planning to mitigate DDoS attacks effectively assuming deterministic behavior of the network that is not realistic [84]. Some researchers have applied Monte Carlo Process to find game-theoretic equilibria for Moving Target Defense planning considering few static attack strategies [85]. SDN-based Bohatei [86] is scalable and dynamic but does not work against I-DDoS attacks.

1.6 Contributions

This dissertation presents following majors contributions:

- Formal models of an automated multi-dimensional decision-optimization framework to compute cost-effective and resilient portfolio for any large enterprise. The computed portfolio minimizes cyber risk within the affordable budget while guaranteeing to satisfy any cybersecurity investment requirements.

- A pragmatic data-driven methodology to prioritize the menacing attack surface by quantifying their risks using a novel probabilistic risk metric. This metric addresses non-homogeneous relationships among threat, threat actions, and

vulnerabilities against an asset.

- An autonomous multi-agent architecture to employ dynamic and optimal defense strategy compositions for protecting critical network links against I-DDoS attackers. This architecture ensures scalability in enacting real-time I-DDoS defense optimization at upstream network points.

- A hybrid Reinforcement Learning (RL) approach to optimize defense composition through integrating the stochastic network behavior and expected attack behavior into the decision-loop. It fastens the convergence of an agent's decision-optimization towards optimal actions by ignoring the exploration of irrelevant actions. It actively learns the environment and adaptive attack behavior based on interactive experience without requiring deep-domain knowledge or human intervention.

- An effective attack prediction approach using time series forecasting. This approach trains a Deep Neural Network (DNN) model that learns attack traits through inferring dynamic temporal dependencies in observed attack action sequences. It can detect attacker's strategical adaptations and shows robustness against adversarial machine learning.

- A model, BRITE loop, defining capabilities of an autonomous defense agent. BRITE model enables agents to not only *observe* and *actuate* but also to *understand* and *investigate* in order to accurately estimate the environment state under uncertainty, and *evolve* to learn about the system dynamics and parameters for optimizing the decision-making.

CHAPTER 2: Automated Planning of Risk Mitigation using Cyber Defense Matrix

In the past few years, cyber attacks have been tremendously increasing in their sophistication, magnitude, and impact demanding a genuine effort from academic, industry, and government agencies to create guidelines for security controls. These guidelines are used as a defacto standard for measuring and mitigating risk against evolving cyber attacks. An example of this effort is the CIS Critical Security Control [19] enlisting a comprehensive set of defense approaches as security controls. However, the deployment of all security controls is irrational and economically infeasible due to constraints such as limited budget and business policies. Hence, risk mitigation planning is an essential process for Chief Security Officers (CISOs) and cybersecurity analysts to determine the appropriate countermeasures to deploy in order to create a cost-effective cybersecurity portfolio. However, composing such a portfolio to optimize the cyber defense Return on Investment (ROI) is an arduous and error-prone task for the following reasons:

- First, this decision-making is a multi-dimensional process that requires to consider multiple critical factors: threat surface, previous attack incidents, security controls effectiveness, risk appetite, and resource constraints.

- Second, the computation complexity to discover such portfolio grows non-linearly with the increase of assets of different types and asset values (regarding Confidentiality, Integrity, and Availability).

- Third, there is a lack of mechanisms to cope with security control failures which are unavoidable. Therefore, enterprises usually use ad hoc and manual techniques to choose security controls, which does not systematically yield measur-

able risk mitigation and ROI values.

Though some researchers aim to optimize the decision-making for cyber risk[48, 47, 41, 44], they are too general and do not address the problem of risk mitigation through selecting the optimal set of security controls and correlating attack tactics, techniques, and vulnerabilities.

As three pivotal factors of cybersecurity decision-making, CISOs seek to determine "what" security controls (and security functions), "where" to deploy effectively, and "why" or for what purposes. In this research, I consider the widely used Critical Security Controls (CSC) of Center of Cybersecurity (CIS) [19] (other security controls can also be used) and Security functions of NIST Cybersecurity Framework [87] to answer "What". CIS CSC is the prioritized set of defense actions, that accumulates globally accepted security best practices to defend and to minimize the consequences of cyber attacks [19]. Additionally, I consider five deployment options (Network, Device, People, Application, and Data) for enforcing CSC (and their corresponding products) to answer "where". To answer "why", we use text mining techniques to extract a large number of *threat actions* from MITRE Attacks Tactics, Techniques, and Procedures (TTP) [6], and thousands of Symantec cyber threat intelligence (CTI) reports [88]. Thereupon, these threat actions are associated with security controls (CSC sub-controls) in the context of the kill chain of threat lifecycles. Ultimately, threat actions defensible by CSCs at kill chain phases answer "Why" the enterprise needs a particular set of security controls.

2.1 Problem Statement

The goal of this chapter is to develop an automated decision-making framework called **CyberARM** (Cyber Automated Risk Mitigation) to assess and manage cyber risks effectively by computing the desired cost-effective risk mitigation plan [89, 90]. CyberARM guarantees the satisfaction of business policies regarding risk appetite, ROI, and budget and cybersecurity policies regarding resiliency, defense-in-depth,

and others. To satisfy policies regarding defense phases and understanding the risk plan acutely, this dissertation presents a multi-dimensional model named Cyber Defense Matrix (CDM) (Fig. 2.3). Previously, a concept of Cyber Defense Matrix was presented in [91, 92] that aligns security countermeasures across NIST Cybersecurity Framework (first dimension) to protect assets such as devices, applications (second dimension).

This dissertation extends the existing concept of CDM by integrating three frameworks: (1) NIST Cybersecurity Framework [87], (2) CIS CSC [19], (3) Cyber Kill Chain into a single structure. In this dissertation, this model categorizes the security controls based on three fundamental properties: Security Function, Enforcement Level, and Kill Chain Phase, which are three dimensions of the extended CDM. However, the concept of this extended CDM is different than the existing CDM. The two-dimensional CDM categorizes security countermeasures based on their security function and types (e.g., application, data, and others) of assets these defend. In contrast, the extended three-dimensional CDM categorizes security countermeasures based on: (1) what security functions these offer, (2) where these operate (i.e., enforcement level), and (3) at which kill chain phases of a particular threat action these confront. Markedly, the second dimension of this extended CDM is different from the other CDM, because it defines where security controls operate instead of which type of assets these defend. To clarify, for instance, a security control may protect an application or desktop from malware by blocking malicious domain; hence, this security control in the three-dimensional domain resides in the network layer.

The extended multi-dimensional Cyber Defense Matrix (CDM) divides risk mitigation against the continuum of sophisticated cyber attacks into many phases, that enables the framework to meet up strict resiliency requirements (e.g., multi-layer and multi-control, k-resiliency, and others). Fig. 2.2 illustrates an example of cybersecurity planning by CyberARM, where SF (Security Function), EL (Enforcement Level),

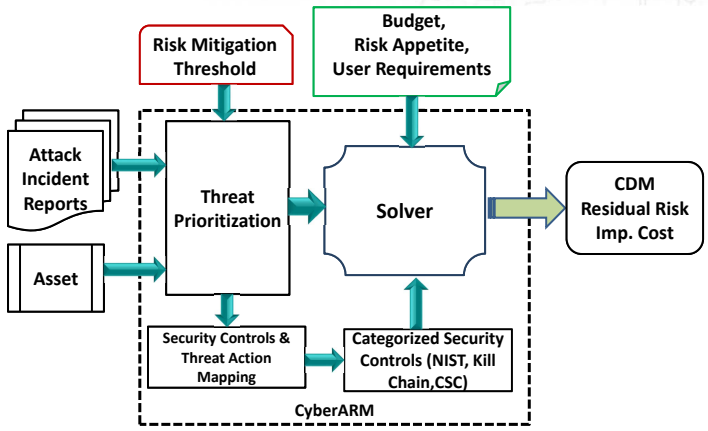


Figure 2.1: CyberARM System Overview

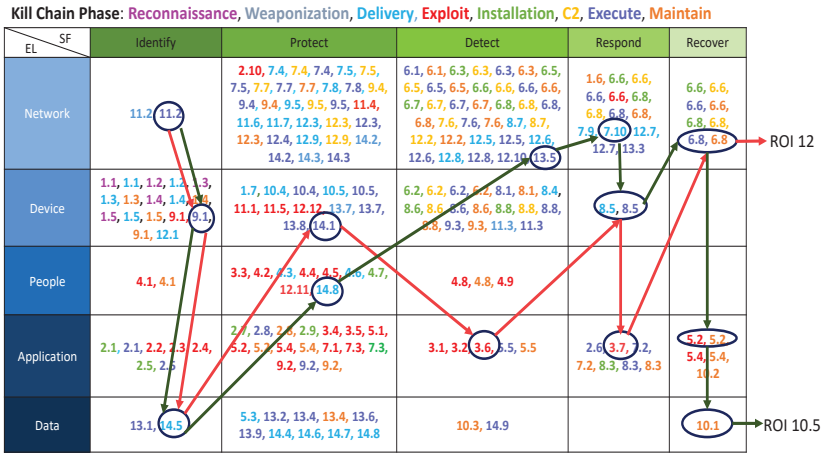


Figure 2.2: Cyber Defense Matrix in Action - An Example

and Kill Chain Phase depict three dimensions of CDM. To clarify, a security control 3.5 represent the 5^{th} sub-control "Deploy Automated Software Patch Management Tools" of CSC 3 [19]. In Fig. 2.2, the red pattern shows the most cost-effective portfolio composed by CyberARM with higher ROI. However, CyberARM deviates its planning (as shown by the green pattern) from the red pattern when the user has specific preferences for recovering ("Recover") from attacks.

The CyberARM decision-making is a computationally complex Constraint Satisfaction Problem (CSP). Therefore, I developed an algorithm to decompose the CSP to accommodate a large number of distinct cyber assets, that significantly improves the scalability of CyberARM. I compute and verify the generated planning to check

its compliance with the constraints using a SMT verification solver, Z3Prover/z3 [93].

This chapter contains the following novel capabilities:

- The main contribution of this chapter is the development of an automated multi-dimensional resilient decision-making approach to recommend the cost-effective set of security controls and the corresponding countermeasures (cybersecurity products). The presented approach can solve the problem for a large enterprise of 15000 assets and shows robustness in the presence of uncertainties and noise.

- This dissertation presents a pragmatic data-driven threat prioritization framework to quantify the risks based on a probabilistic model developed using threat incident reports and vulnerability scanning reports. Moreover, this framework considers a realistic attack model where an adversary can exert several attack actions (threat actions) exploiting different vulnerabilities to execute a successful attack (threat) and vice versa.

- This dissertation presents a heuristic approach applying model reduction and decomposition to approximate a cost-effective risk mitigation plan with reduced computational complexity. These heuristic approaches enable the framework to employ a cost-effective portfolio for large enterprises. This dissertation empirically shows that these heuristic models reduce the computational complexity significantly while approximating the plan closer to the optimal plan.

2.1.1 Architecture Overview of CyberARM

Figure 2.1 show CyberARM architecture with user inputs, and outputs. It takes the following user inputs: (1) *Cyber assets information* that includes services, and the value of the asset (in $$) (*i.e.*, the inflicted loss due to a compromise of confidentiality, integrity, or availability (CIA) of a particular asset), (2) *Vulnerability scanning reports which includes the existing vulnerabilities, and the CVSS score of these vulnerabilities*, (3) *Enterprise business requirements* specifying the maximum target budget, risk

appetite and RoI, (4) *Attack incident history (optional)* which includes the previous cyber incident reports, and (5) *Enterprise security policies (optional), if any*, such as the priority of certain security function, enforcement level or guaranteed resiliency for mission-critical assets. The output of CyberARM includes a recommendation of the cost-effective cybersecurity planning with installation cost, global residual risk, risk statistics per threat action and assets, and RoI.

CyberARM instigates the risk management through uncovering the most menacing threat surface that the tool prioritizes to alleviate using "Attack Incident Reports" (if provided), "Vulnerability Reports", and asset values. Threat surface amalgamates the distinct pairs of threat (e.g.,Hacking, Malware, Social Engineering) and threat action. However, threat actions types vary as some threat actions are fine-grained (e.g. overwrite ferite script files), whereas, some are very generic like Sql Injection. In figure 2.1, "Threat Prioritization" module prioritizes the combinations of threat and threat actions against "Assets" using the estimated quantitative risk based on "Risk Mitigation Threshold".

The following step is the identification of the set of security controls enforceable against at least one of the prioritized threat actions. For this research, we create a comprehensive mapping from CIS security controls to the threat actions shown by "Security Controls & Threat Action Mapping" in figure 2.1. Accordingly, the module named "Security Controls & Threat Action Mapping" picks the appropriate set of security controls (candidate set of security controls) for the prioritized threat actions (output of "Threat Prioritization"). Moreover, the framework prunes the set based on our model reduction and includes the "Indirect" threat action (not prioritized before but defendable using the pruned set) into the threat surface. "Categorized Security Controls" module contains all CSC security controls categorized based on Security Function (SF), Enforcement Level (EL), and Kill Chain Phase (KC Phase) which are three dimensions of CDM structure.

The next step is the formation of "Candidate-set of Security Control Products" where each product implements one or more security controls of the candidate-set of controls. In this research, we assume that CyberARM contains a list of currently available "Security Control Products" and their association with CIS security controls. To clarify, an association between a product and security control defines that the product implements the security control. In general, such an association is easily obtainable from the products' descriptions. However, the users (CISOs) can also provide this association of products and security controls as input.

Thereupon, CyberARM engine ("Solver" in the fig. 2.1) infers the appropriate subset from the candidate-set of products for each quadrant of CDM structure to compose the desired cybersecurity planning ensuring less residual risk than risk appetite and greater ROI than user-defined minimum ROI within the bounded resource (Budget). Moreover, the users can tweak the planning by providing their business-oriented policies (User Requirements) which enhances the usability of the proposed tool across diversified enterprises.

As an example, figure 2.2 illustrates a case where the user wants to know the cost-effective set of security controls. CyberARM discovers planning (green) with ROI 11.6, but it includes CSC 1.2 when the user emphasizes more on detecting ("Detect") attacks and generates planning with ROI 10.5.

2.2 CyberARM System Primitives

This section describes basic components of CyberARM residing in the framework internally.

2.2.1 Cyber Defense Matrix (CDM)

Cyber Defense Matrix structure in Fig. 2.3 resemblances the deployable cyber defense strategy using three dimensions. These three dimensions represent three fundamental properties of a security control: Security Function (SF), Enforcement

Level (EL), and Kill Chain Phase (KC Phase). It segregates cyber defense into many stages or phases, and as a result, the tool can employ divide and conquer against the sophisticated and advanced threat actions. This multi-dimensional CDM structure constitutes of $8 \times 5 \times 5$ quadrants, where each quadrant represents a distinct defense phase. Moreover, a threat action propagates through different stages (quadrants) of CDM structure in its lifespan and achieves its objectives when it defeats all the deployed countermeasures at those stages.

The first dimension of the CDM structure, kill-chain-phase (KC phase), has eight different values: Reconnaissance, Weaponization, Delivery, Exploit, Installation, C2, Execute, and Maintain. There is a plane (2D matrix) in the CDM structure for each distinct KC phase, which depicts the defense strategy at that particular KC phase. In general, kill chain phase defines the current attack stage; hence, the KC phase dimension illustrates which security controls encounter the threat actions during a particular stage of their life-cycles. Therefore, this dimension delineates "Why" the security controls require to be enforced. Besides, the framework can consider any requirement related to the life phase of the threat actions because of this dimension. For example, the framework can satisfy a user preference in preventing threat actions during the "Exploit" phase.

The second dimension (Enforcement Level) outlines the layers where the chosen security controls confront threat actions. Specifically, the enforcement level of a security control depends on the attributes on which it operates. As an instance, a control named "Blacklist IP Address (12.3)" resides in the "Network" layer as it operates on IP address, whereas "Establish Secure Configurations (5.1)" control belongs in the "Application" layer due to its operations on the configurations of applications. Therefore, this dimension answers "Where" the security controls terminate the progression of the attackers and consequently, enables the articulation of CyberARM's decision based on defense layers.

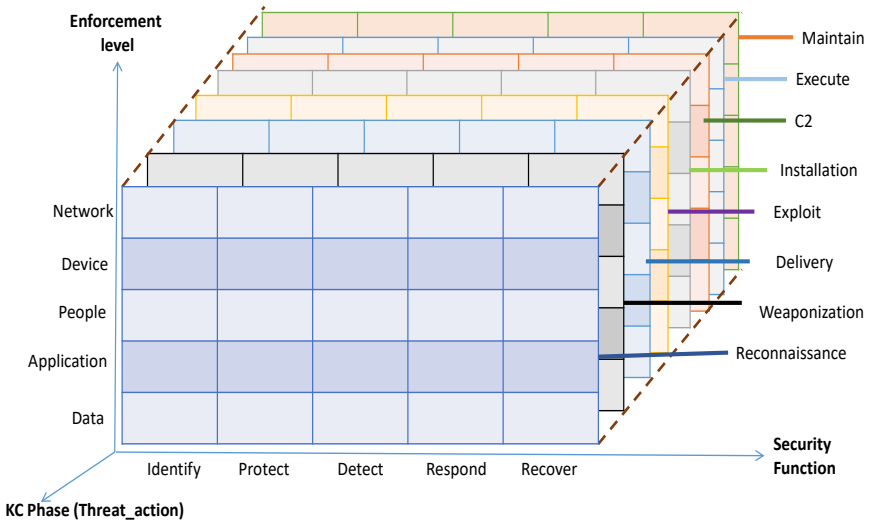


Figure 2.3: Cyber Defense Matrix (CDM) Structure

Security function, third and last dimension, is the exact type of defense actions executed by the security controls. NIST framework has categorized these defense actions into five groups: Identify, Protect, Detect, Respond, and Recover [87]. In fact, security controls at a particular enforcement level of CDM structure exercise these security functions sequentially as drawn in Fig. 2.3. In essence, CyberARM depicts "What" terminates the attackers and meets the user requirements (e.g., cost distribution among security functions, guaranteed resiliency at specific security function) by leveraging this dimension.

CyberARM evaluates possible defense strategies for each phase of the CDM structure individually to form the recommended planning. Moreover, CDM structure enables the tool to meet the diversified security requirements (e.g., multi-layer and multi-control resiliency, specific security portfolio preference) of various enterprises emerged due to the heterogeneity in their business objectives and policies. CyberARM investigates across SF, EL, and KC Phase dimensions and also across the defense phases of CDM structure to do so. Therefore, this structure strengthens CyberARM to leverage specialized humans decisions into automation by integrating CISOs' knowledge and experiences as policies. Finally, CyberARM chooses the rel-

evant subset of security controls and the associated products for each quadrant in a way that it guarantees the orchestration (e.g., considerations of correlations among the products in defending threat surface) with other quadrants to assure resiliency and cost-effective plan. Therefore, CyberARM fills all quadrants of CDM structure with the optimal set of security controls to compute and recommend a cost-effective **CDM** as optimal cybersecurity planning.

2.2.2 Security Controls Categorization

To determine the locus of a security control in CDM structure, I categorize all considered security controls based on their fundamental defense properties that are represented by three dimensions of CDM structure. Moreover, security control and its associated security products can belong to more than one quadrant, where an association between security product and security control specifies that the product has implemented the security control with non-trivial effectiveness. Aforementioned, CyberARM considers the widely recognized 20 Critical Security Controls (CSCs) of CIS Controls [19] as a use case, that categorizes all security controls based on their defense objectives. Each CSC consists of more than one security control such as, for example, CSC 7 aiming to protect email and web browsers grouped 10 security controls. In CSC 7, the second security control (CSC 7.2) blocks unauthorized scripts, whereas the fourth security control (CSC 7.4) performs network-based URL filtering. I collaborate with CIS to map security controls to "Security Function", "Enforcement level (EL)", and "Kill Chain Phase (KC Phase)" [19] according to NIST framework [87]. Though CyberARM considers security controls of CIS Critical Security Controls (CSCs), this framework is flexible to incorporate any newly discovered security controls or security controls from other frameworks.

Fig. 2.2 contains the categorizations of all security controls of first 14 CIS CSCs, where a security control 3.5 in $(Application, Protect)$ quadrant with red color (the same color as "Exploit" in Kill Chain Phase) defines that it provides security function

Table 2.1: Security Controls to Threat Action Mapping

Security Control	Threat Action
Network Based URL Filters	Malicious C2 Connection
Limit Unauthorized Scripts	Cross Site Scripting (XSS)
Password Constraints	Brute Force Attack
Limit External Devices	Unapproved Hardware
Block Malicious Email Attachment	Phishing

"Protect" at enforcement level "Application" against threat actions during "Exploit" kill chain phase. Understandably, all security control products having same defense properties (security function, enforcement level, and kill-chain phase of encountering threat actions) like its associated security controls belong to same quadrants. Hence, CDM structure encloses similar security controls and products into a particular defense group based on their defense traits, where a defense group encounters attack actions during a specific attack phase.

2.2.3 Threat Action to Security Control Mapping

In order to identify the appropriate set of security controls against specific threat actions, CyberMirror needs a mapping from CSCs to threat actions. CyberARM contains a comprehensive manual mapping from CSCs to Threat Actions. As a use case study, threat actions are extracted from Symantec reports [88], Mitre Attacks [6], and VERIS using TTPDrill [94]. Thereupon, I manually associate each CSC security control to appropriate threat actions by analyzing their course of mitigation actions, and these associations have been validated using experts' opinions. Table 2.1 shows some examples of security controls to threat actions mapping, where, for example, the first association depicts that *Network Based URL Filters* can defend *Malicious C2 Connection* with a considerable effectiveness.

Understandably, a security control product has non-zero or significant effectiveness against all threat actions associated with its implemented security controls. To clarify, products offering *Limit Unauthorized Scripts* can defend *Cross Site Scripting (XSS)*

with an effectiveness defining the defense accuracy regarding true and false positive, and true and false negative. Importantly, two products offering same security control not only have different effectiveness but also have different deployment costs. Notably, CyberARM does not consider a security product against a specific security control if its effectiveness is lower than another product while having a higher deployment cost. Though I assume the effectiveness of these products for this research, frameworks like [95, 96] can assess the effectiveness or defense accuracy of these products in terms of performance score (recall, precision, F1 score). Moreover, CyberARM supports both qualitative and quantitative effectiveness measurements.

2.3 CyberARM Data Model

CyberARM data model in Fig. 2.4 illustrates the approach of determining the candidate-set of security control products from the user-given list of asset entities using Entity Relationship Diagram (ERD). Most importantly, this data model correlates objects of the framework to translate into logic, and allows it to increase CyberARM's capabilities without significant alternation. In the figure, rectangular, diamond and elliptical shapes represent entities (objects), actions, and attributes respectively. An action describes the relation between two entities, and an attribute defines the property of an entity or an action. Moreover, any underlined attribute in Fig. 2.4 is a primary key that is used to designate the specific entity or action.

The *Contains* action between *Asset* and *Vulnerability* entities enlists the existing vulnerabilities on the assets discovered by vulnerability scanning tools. Each vulnerability has two properties represented as attributes: a unique CVE ID and CVSS score that defines the severity of the vulnerability. Notably, severity of a vulnerability specifies how exploitable the vulnerability is to launch a successful attack; hence, risk of an attack exploiting a particular vulnerability increases with the increase of CVSS score. Moreover, each asset has a value as a property that defines the impact induced due to compromising the asset. Importantly, the value is a function of Confidentiality,

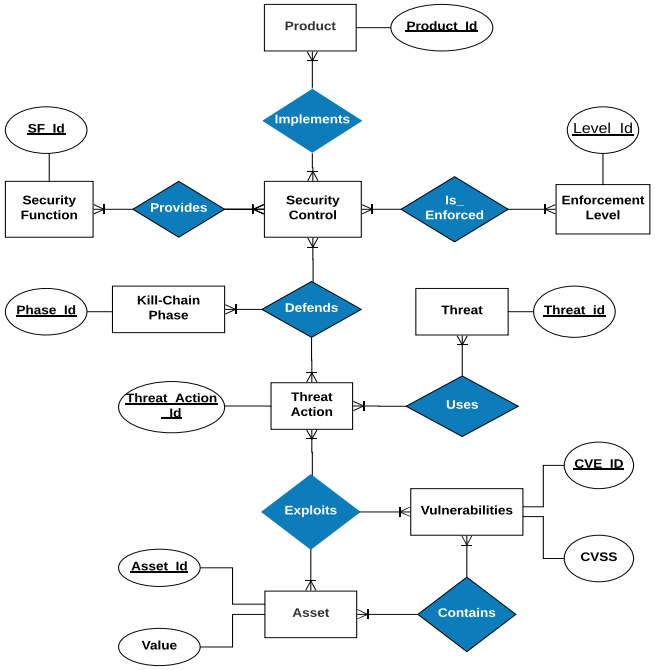


Figure 2.4: CyberARM Data Model

Integrity, and Availability that are generally dependent on the domain of the asset. To clarify, the asset of the advertising domain does not have large value regarding Confidentiality but has significant value considering Integrity and Availability. On the other hand, the asset containing historical data of an enterprise comparatively has higher value regarding Confidentiality and Integrity while it can afford the loss of availability for a certain time.

The framework recognizes *Threat Action* (attack technique) entities who can exploit these vulnerabilities against the *Asset* by *Exploits* entities that represent correlations among threat actions and existing vulnerabilities. Such correlations can be defined based on attack incident reports, NVD CVE list [97], and vulnerability and weakness repositories [97, 98, 99]. Thus, the framework recognizes the set of *Threat Action* entities likely to be successfully exercised against the *Asset* entities. Besides, the framework extracts *Threat* (attack tactic) entities capable of exerting these *Threat Action* to violate CIA (Confidentiality, Integrity, and Availability) of the assets. Attack incident reports, and ATT&CK Mitre framework [6] generally de-

fine such relationships. Hence, CyberARM determines the threat surface subjected to eliminate using correlations among threat, threat actions, vulnerabilities, and assets.

The *Defends* action delineates the *Security Control* entities applicable against *Threat Action* entities at the specific *Kill Chain Phase*. Security controls to threat actions mapping (described in Section 2.2.3) draw correlations among threat actions and security controls, whereas MITRE ATT&CK framework describes associations among threat actions and kill-chain-phase. Therefore, the framework determines the relevant set of security controls as candidate-set of security controls that can defend the prioritized threat surface with a non-trivial effectiveness. Additionally, *Provides* defines the *Security Function* of these security controls; whereas, *Is_Enforced* describes the *Enforcement Level* obtained by security control categorization 2.2.2. Therefore, *Provides*, *Is_Enforced*, and *Defends* fill the quadrant of CDM structure with the appropriate set of security controls.

Finally, CyberARM selects the security control products that implement security controls of the candidate-set by *Implements* action that correlates *Product* with *Security Control* entities. In summary, the model illustrates the approach for deciding candidate-set of security products spread throughout quadrants of CDM structure.

2.4 Threat Prioritization

It is almost impossible to design a bulletproof defense strategy that stops the execution of all cyber attacks. Besides, it may not be worthwhile to invest resources in eliminating threat actions with lower impacts or small exertion likelihoods. This is why, appropriate and worthwhile cybersecurity investment instigates with threat prioritization that aims to realize the most menacing threat surface as *prioritized* threat surface for the given asset list. Importantly, elimination of such prioritized threat surface aids CyberARM not only to maximize Return On Investment (ROI) but also to improve its scalability by pruning irrelevant search space. In this research, the prioritized threat surface comprises of distinct combinations of threat and threat ac-

tions as distinct *Attack Vector*, and CyberARM obtains relational associations among threats and threat actions from *Uses* relationships in Fig. 2.4. This module has three major parts: (1) Determining Risk of all Threat Actions, (2) Determining Global Risk, and (3) Composing Prioritized Threat Action Set.

2.4.1 Determining Risk of All Threat Actions

The risk due to a threat action depends on its likelihood of exertion by adversaries, and also on its capability in exploiting existing vulnerabilities. In order to understand the exploitability or severity of the vulnerabilities, I adopt Common Vulnerability Scoring System (CVSS) [97]. In general, the likelihood of threat action exertion depends on the sophistication of executing the threat action, and it generally decreases with the increase of sophistication according to available threat incident reports. For instance, *Brute Force* action is much easier to execute than *Cross Site Scripting (XSS)* due to less technical difficulties. Therefore, rational cybersecurity investment must not defend sophisticated threat action ignoring the most frequently appeared threat actions. However, a threat action needs to exploit a vulnerability to execute a successful attack, and hence, the sophistication of a threat action may increase due to small CVSS value of a vulnerability that it can exploit. For example, Brute force action against a password-constrained system is hard than executing XSS. Importantly, if a threat action can exploit multiple existing vulnerabilities, CyberARM considers the maximum CVSS score out of CVSS scores of those vulnerabilities.

According to the *Exploits* relationship in Fig. 2.4, CyberARM picks the set of threat actions against an asset that can exploit any of the existing vulnerabilities (CVEs) of that asset. Notably, though vulnerability scanning across the network enlists all existing vulnerabilities of a network, it cannot reveal the zero-day vulnerability. This research focuses on enlisted vulnerability, and aims to address zero-day vulnerabilities as our future tasks. Thereupon, CyberARM follows two different approaches to estimate the likelihood of those associated threat actions. In first approach, when

previous security incident reports are available, CyberARM initially distinguishes distinct attack vectors by extracting combinations of threat t, threat action ta, asset type a, and domain d from these reports to calculate probabilities $p(ta, t, a, d)$. In another approach, when no extracted attack vector from given reports can exploit an existing vulnerability, CyberARM identifies all the associated threat actions for that vulnerability to create distinct attack vectors. Then, it assigns a minimum probability value m_δ using Eqn. 2.1 that specifies the assignment of probabilities for attack vectors comprising a distinct combination of threat and identified threat actions (i.e., not extracted from available incident reports). Importantly, CyberARM applies Eqn. 2.1 when no threat action from given reports can exploit an existing vulnerability or there is no incident report given by the user.

In Eqn. 2.1, $p(ta_j, t_i, a_k, d_k)$ describes the likelihood of exerting threat action ta_j that imposes threat t_i against asset $k(a_k, d_k)$ that is of type a_k (e.g., mail server, database) in business domain d_k (e.g., infrastructure, finance, public information). Besides, V_k is the set of existing vulnerabilities that no threat action of given reports can exploit. Notably, ta^{V_k} is the set of threat actions that can exploit vulnerability of V_k but have zero probability based on given reports.

$$\forall(ta_j \in ta^{V_k}), \forall(t_i \in Th_j), p(ta_j, t_i, a_k, d_k) = m_\delta \tag{2.1}$$

where, Th_j is the set of threats which use threat action ta_j.

Then, CyberARM determine the conditional probability $p(ta_j|t_i, a_k, d_k)$ that specifies the probability of exerting threat action ta_j when the attacker wants to transform threat t_i into successful attack against asset k, using following equation:

$$p(ta_j|t_i, a_k, d_k) = \frac{p(ta_j, t_i, a_k, d_k)}{\sum_w p(ta_w, t_i, a_k, d_k)} \tag{2.2}$$

Similarly, CyberARM determines the probability $p(t_i|ta_j, a_k, d_k)$ that specifies the

probability of transforming threat t_i into successful attack when the attacker exerts threat action ta_j against asset k, using following equation:

$$p(t_i|ta_j, a_k, d_k) = \frac{p(ta_j, t_i, a_k, d_k)}{\sum_w p(ta_j, t_w, a_k, d_k)} \tag{2.3}$$

Thereafter, CyberARM determines the likelihood of exerting threat action ta_j against asset k, using the following equation:

$$p(ta_j|a_k, d_k) = \frac{\sum_x p(ta_j, t_x, a_k, d_k)}{\sum_w \sum_x p(ta_w, t_x, a_k, d_k)} \tag{2.4}$$

The framework computes the conditional probability $p(t_i|a_k, d_k)$ that defines the likelihood of transforming threat t_i into successful attack against asset k, using following equation:

$$p(t_i|a_k, d_k) = \sum_w p(t_i|ta_w, a_k, d_k) \times p(ta_w|a_k, d_k) \tag{2.5}$$

where, CyberARM determines $p(t_i|ta_w, a_k, d_k)$ and $p(ta_w|a_k, d_k)$ using Eqn. 2.3 and Eqn. 2.4 respectively.

Finally, CyberARM determines the imposed risk R_j^k by threat action ta_j on asset k, using the following equation:

$$R_j^k = \sum_{y \in \{C,I,A\}} \sum_i S_k^j \times p(ta_j|t_i, a_k, d_k) \times p(t_i|a_k, d_k) \times I_i^k(y) \tag{2.6}$$

where, S_k^j defines the maximum CVSS score among all existing vulnerabilities of asset k that are exploitable by threat action ta_j. Besides, $I_i^k(y)$ specifies the impact of executing threat t_i successfully against asset k due to losing of Confidentiality (C), Integrity (I), or Availability (A), where impact defines the expected loss due to transforming the threat into successful attack against the asset.

In risk metric (defined in Eqn. 2.6), CyberARM considers the impact I_k^i of threat i

separately for C, I, and A to integrate effects of t_i more precisely, that also aids to consider the impact according to the domain of the asset. For instance, database storing public information has zero impact regarding confidentiality, whereas, database storing social security information of employees/clients has significant impact regarding confidentiality.

2.4.2 Determining Global Risk

The global risk G_R of the enterprise is the summation of all risks imposed by all relevant threat actions against all assets of the enterprise, which is formulated in the following equation:

$$G_R = \sum_{k \in E_A} \sum_{j \in ta_k} R_j^k \tag{2.7}$$

where, E_A is the asset list of the enterprise, and ta_k is the set of threat actions against asset k.

2.4.3 Composing Prioritized Threat Action Set

CyberARM chooses the minimum subset of threat actions that, if mitigated, will lower the risk than given risk appetite A_R. However, the computed cybersecurity planning may not completely eradicate the likelihood of some threat actions due to the failure probability of security controls or requiring more investment than mitigated risk (will reduce ROI). Therefore, CyberARM considers *Expected* residual risk percentage, $(1-\delta_e)$, of prioritized threat action set despite deploying computed cybersecurity planning. Here, δ_e is the weighted average of effectiveness of security control products, that can be formulated through iterating over all possible threat actions and assets using the following equation:

$$\delta_e = \frac{\sum_{ta_j} (\sum_k R_j^k) e_w^j}{\sum_{ta_x} (\sum_y R_x^y) e_w^x} \tag{2.8}$$

where, e_w^j is the effectiveness of a security control product w in mitigating threat

action ta_j.

To address the residual risk induced due to imperfect security effectiveness, CyberARM chooses the minimum subset of threat action in a way that its aggregated risk, m_R, is greater than $(G_R - A_R)$ and satisfies the following constraint:

$$m_R \geq (G_R - A_R)(1 + (1 - \delta_e) + (1 - \delta_e)^2 + ... + (1 - \delta_e)^n)$$
$$m_R \geq (G_R - A_R) \times \frac{1 - (1 - \delta_e)^n}{\delta_e}$$
$$m_R \geq \frac{G_R - A_R}{\delta_e} \quad [if \quad n \longrightarrow \infty]$$

The above constraint compels CyberARM to enhance prioritized list to cover up for the increased likelihood of unmitigated risks (lower δ_e). Presumably, with the decrease of weighted effectiveness δ_e (ranging [0,1]), CyberARM has to expand the chosen threat action set due to increase of m_R. CyberARM composes the prioritized threat action set to defend based on m_R by the approach of Alg. 1.

In Alg. 1, CyberARM creates a list of objects comprised of threat action id (j), its imposed risk (R_j^k), asset id (k) at line 3-6 followed by prioritization in *descending* order based on imposed risk R_j^k at line 7. Importantly, CyberARM considers two types of threat action set: (1) Direct, and (2) Indirect. Direct threat action set imposes at least m_R risk cumulatively, and CyberARM prioritizes the set to deploy security control products against it. Whereas, Indirect threat action set are threat actions that CyberARM does not aim to mitigate, but chosen products against direct threat action set can mitigate these. The inclusion of Indirect threat action set aids CyberARM to evaluate products more precisely. In line 9-14, CyberARM composes Direct threat action set imposing at least m_R and includes into a list T_P. In line 16, CyberARM composes candidate set of security control products, cd_SP, to defend Direct threat action set (described in Section 2.5.2). In line 18-22, CyberARM selects those threat actions into T_P that can be defended by $candSP$ but not included in

Direct threat action set to compose Indirect threat action set. Finally, the algorithm returns the threat action set T_P as *Prioritized* Threat Action Set that CyberARM aims to mitigate.

Algorithm 1: prioritizeThreatActionSet

Input : Minimum Imposed Risk m_R, Global Risk G_R, Threat Action Set T_A, Enterprise Asset List E_A
Output: Prioritized Threat Action Set T_P

```
// This function prioritizes Threat Action Set to identify the
   cost-effective set of Security Controls of an Enterprise.
ta_risk_list=[ ]
// Creating list of threat action and its imposed risk
for k in E_A do
    // T_A(k) is the threat action set of asset k
    for j in T_A(k) do
        ta_risk_list.append([j,R^k_j,k])
ta_risk_list = Sorted(ta_risk_list)// Sort based on 2nd Index (risk)
// Selecting Direct Threat Actions
T_P = [ ], i_r = 0
for t_a in ta_risk_list do
    if i_r ≥ m_r then
        break
    i_r += t_a[1]
    T_P.append([t_a[0],t_a[1],k,True])//True defines direct threat action
// Select N number of security products against each threat
   action of T_P
candSP = chooseSecurityControlProducts(T_P,N)
// Selecting Indirect Threat Actions
for sc,k in candSP do
    // ta^k_sc is the threat action set defensible by a security
       control product sc for asset k
    for ta in ta^k_sc do
        if R^k_ta > 0 then
            T_P.append([ta,R^k_ta,k,False])//False defines indirect threat action
return T_P
```

2.5 Composing Candidate Security Control Product Set

Candidate Security Control Product Set consists of security control products/technologies that can defend the prioritized threat surface effectively. This section describes the composition of security control products and their reduction.

2.5.1 Selecting Security Control Products

Composing Candidate Security Control Product Set, *candSP*, instigates with extracting security controls to defend *Direct* prioritized threat action set, T_P^D. Aforementioned, there exists a relationship table *Defends* (shown in Fig. 2.4) that specify *which* security controls can defend *which* threat actions at *which* kill-chain-phases. Based on this, CyberARM extracts security controls that can defend at least one of T_P^D at a particular kill-chain-phase. Besides, based on *Provides* and *Is_Enforced* relationship tables, CyberARM knows *what* types of security functions these extracted security controls offer at *which* layers. After extracting the relevant security controls, CyberARM prunes any security control if it fails to satisfy any user requirement. For example, an enterprise hosting a Web application to offer information about its business model to clients or potential clients, may not want to deploy CSC 7.4 that obstructs any unauthorized URL. Because, this contradicts the purpose of the web application as enterprise may not know potential clients or future clients beforehand for authorizing their URLs. Therefore, CyberARM prunes any security control relying on URL-based filtering for such assets (e.g., advertising site, news sites) if the user provides it as a requirement.

Using associations defined by *Implements* in Fig. 2.4, CyberARM extracts products that implement any of the chosen security control to compose initial *candSP*. However, CyberARM prunes a product w for a specific asset k if it requires more investment than the offered benefit, where the benefit of w depends on the impact (imposed risk) of threat actions that w can defend. CyberARM formulates the benefit B_w^k of w for k by the following equation:

$$B_w^k = \sum_{j \in w_{ta}^k} (1 - e_w^j) R_j^k \tag{2.9}$$

where, w_{ta} is the threat actions set that w can defend, e_w^j is the effectiveness of

w in defending threat action ta_j, and R_j^k is the imposed risk by ta_j on k. Hence, CyberARM ignores w for k if $B_w^k < (C_w + \delta_w)$, where C_w is the implementation cost of w, and δ_w is a benefit threshold that can be determined based on weighted benefits of other products.

2.5.2 Model Reduction

This step prunes the candidate set of security products, *candSP* (obtained from previous Section 2.5.1), for reducing the problem space to improve scalability. Importantly, scalability is a critical factor for any cybersecurity planning automation due to the exponential growth of product combinations of *candSP* with the increase of assets. Hence, CyberARM applies a heuristic function that removes less cost-effective products from *candSP* to prune the irrelevant search space. The basis of the heuristic function is that cost-effective products maximize ROI due to following constraint:

$$\begin{aligned} Ce_{w_i} \geq Ce_{w_j} &\implies Ce_{w_i} - 1 \geq Ce_{w_j} - 1 \\ &\implies \frac{B_{w_i}}{C_{w_i}} - 1 \geq \frac{B_{w_i}}{C_{w_i}} - 1 \\ &\implies ROI_{w_i} \geq ROI_{w_j} \end{aligned}$$

where, B_{w_i}, C_{w_i}, Ce_{w_i}, and ROI_{w_i} are benefit (determined by Eqn. 2.9), cost, cost-effectiveness, and Return on Investment (ROI) of a product w_i respectively.

However, the above constraint will restrict CyberARM to select products only against threat action with high risk, that will be redundant for those threat actions. This is why, CyberARM always ensures the selection of N number of products against any threat action in the pruned *candSP*. Understandably, one of critical parameter of this step is to select the optimal N^*, for which, CyberARM performs linear search (discussed in Section 2.6.1.8) based on residual risk of computed CDM.

In general, increasing N enhances resiliency against threat actions, and experimental results show that if a security configuration with a larger N cannot improve

the solution regarding cost-effectiveness, increasing N further will not have any added benefit. This happens when the added resiliency against a threat action is redundant, because the residual risk is already too low that benefits of new countermeasures become trivial compared to deployment cost. However, this may also happen due to failing to find a satisfiable solution within finite time because of the exponential growth of problem space due to larger N.

Alg. 2 describes the approach/function of composing and pruning $candSP$, that takes threat action set T_P and N as inputs. Importantly, CyberARM endeavors to minimize the overall risk of the enterprise, and does not emphasize in mitigating specific threat action unless specified by user explicitly as requirement. Hence, CyberARM considers overall benefit of a product considering its effects against all associated threat actions to defend the subjected asset. Notably, CyberARM does not consider a product if its alternative can defend same threat action set with same effectiveness but less deployment cost. While iterating over threat actions of T_P (line 2), it ignores any indirect threat action, and selects N number of security control and products only for direct threat action (line 4). At line 5-7, CyberARM selects products of each security control that can defend threat action ta while not violating any given user-requirement. At line 8, CyberARM calls the function *getSortedProducts* to sort all selected products in descending order based on their cost-effectiveness considering all prioritized threat actions for the asset of ta. At line 10-15, CyberARM inserts N number of distinct products against each ta. Thus, CyberARM composes a concise set of candidate products, $candSP$, based on cost-effectiveness to address the computation growth of problem space. After this, CyberARM temporally removes indirect threat actions that cannot be defended with any of the security products in $candSP$ to avoid unnecessary complexity.

Algorithm 2: chooseSecurityControlProducts

Input : Threat Action Set T_P, Maximum Number of Security Products N
Output: Candidate Set of Security Products *candSP*

```
// This function composes set of security control products that
   can defend direct threat actions of T_P cost effectively based
   on given N.
candSP=[ ]
for ta in T_P do
    prod_for_ta = [ ]
    // ta contains (threat action id, imposed risk, asset id,
       Flag), Flag is True iff Direct Threat Action
    if ta[3] then
        // sc_ta is the security control set that can defend ta.
        for s_c in sc_ta do
            if s_c does not violate any constraint of asset ta[2] then
                // W_{s_c} is the product set that implement s_c.
                prod_for_ta.append(W_{s_c})
        // The function "getSortedProducts" sorts its argument
           "prod_for_ta" in descending order of their
           cost-effectiveness.
        sorted_Prod = getSortedProducts(prod_for_ta)
        num_Prod = 0
        for k in range(len(sorted_Prod)) do
            if (sorted_Prod[k], ta[2]) not in candSP then
                candSP.append([sorted_Prod[k], ta[2]])
            num_Prod + = 1
            if num_Prod == N then
                break
return candSP
```

2.6 Composing Resilient CDM

The objective of the CyberARM is to fill each quadrant of CDM structure with the appropriate set of security controls, so that, all deployed security control across all quadrants will collaborate cost-effectively to offer an optimal CDM as resilient cybersecurity planning. This section describes how CyberARM computes the planning:

2.6.1 Risk Mitigation Formalization

Residual risk is the remaining risk due to success probabilities of threats due to prevailing success likelihood against particular assets despite the deployment of CDM. Therefore, CyberARM aims to compute the cost-effective CDM as cybersecurity planning that minimizes the residual risk towards a tolerance level after satisfying all given user-requirements. Moreover, the deployment cost of the computed CDM must not exceed the user-given budget. This section explains how we ascertain the cost-effectiveness of cybersecurity planning regarding ROI for an enterprise.

In all following formulas, KC, EL, and SF are sets or lists representing kill-chain phase, Enforcement Level, and Security Function dimensions of CDM structure respectively, and index of all these list starts from 1 while increasing by one along the sequence. To clarify, for instance, $SF(1)$ specifies the first value of the list SF which indicates the *Identify* security function, whereas, $SF(2)$ indicates the *Protect* security function.

$KC = \{Reconnaissance, Weaponization, Delivery, Exploit,$
$Installation, C2, Execute, Maintain\}.$

$EL = \{Network, Device, People, Application, Data\}.$

$SF = \{Identify, Protect, Detect, Respond, Recover\}.$

Importantly, the likelihood of *successful exertion* of any threat or threat action specifies the probability that an adversary will enact the threat or threat action which will be successful in executing an attack. The likelihood of *successful exertion*

depends on two different probabilities: (1) Exertion likelihood (represented by smaller p) that is the probability of applying it by the adversary determined during threat prioritization (Section 2.4), and (2) Success likelihood (represented by capital P) that is the prevailing likelihood of it despite deploying CDM (determined in following subsections).

2.6.1.1 Success likelihood of threat

As illustrated by many-to-many relationship in Fig. 2.4, an adversary may exert one or more threat actions to transform a threat into successful attack against particular assets. Hence, the success of a threat is contingent on the success of its associated threat action set. Importantly, the threat becomes a successful attack if any of its associated threat action defeats all deployed products across CDM structure. Therefore, the success likelihood, P_i^k, of transforming a threat t_i into a successful attack against asset k depends on the probability that any threat action ta_j of associated threat action set Ta_i becomes successful. This intuition is formulated using the following equation:

$$P_i^k = 1 - \prod_{ta_j \in Ta_i} (1 - Pr_j^k) \tag{2.10}$$

where, Pr_j^k is the likelihood of successfully exerting a threat action ta_j against asset k.

The likelihood Pr_j^k depends on two factors: (1) exertion likelihood, $p(ta_j|t_i, a_k, d_k)$, of threat action ta_j (determined in Eqn. 2.2), and (2) success/prevailing likelihood, P_j^k, of ta_j at asset k. Hence, CyberARM formulates Pr_j^k using the following equation:

$$Pr_j^k = P_j^k \times p(ta_j|t_i, a_k, d_k) \tag{2.11}$$

where, a_k and d_k are asset type and domain of asset k respectively.

2.6.1.2 Success Likelihood of Threat Action

This sub-section describes how CyberARM determines the success likelihood P_j^k of threat action ta_j to integrate into Eqn. 2.11. A threat action becomes successful if and only if it defeats all deployed countermeasures (security control products) confronted while propagating through different quadrants of CDM structure. Therefore, the success likelihood of a threat action depends on its success probability across all quadrants. Notably, the success likelihood of a threat action at a particular quadrant depends on the effectiveness of deployed products operating in the quadrant, that is formulated using the following equation:

$$P_{jk}^{zyx} = \prod_{w \in T_{zyx}^j} (1 - e_w^j \times S_w) \tag{2.12}$$

where, zyx represents the quadrant of CDM structure with kc phase z, enforcement level y, and security function x, and P_{jk}^{zyx} specifies the success likelihood of threat action ta_j against deployed products at zyx quadrant. Besides, T_{zyx}^j represents the candidate-set of products that can defend threat action ta_j during its kc phase z by security function x at enforcement level y, S_w specifies whether the product w is deployed ($S_w = 1$) or not ($S_w = 0$), and e_w^j specifies the effectiveness of w against ta_j.

Therefore, to become successful at a particular enforcement level y during a specific kc phase z, the threat action ta_j needs to defeat all products offering any security function through operating on properties of y against its phase z. For instance, the success of a threat action at *Network* (enforcement level) during *Delivery* kc phase delineates that all products operating on the attributes of the network fail to prevent the threat action during its delivery by the adversary. CyberARM formalizes this

logic in the following equation:

$$P_{jk}^{zy} = \prod_{x \in SF} \prod_{w \in T_{zyx}^{j}} (1 - e_w^j \times S_w) \tag{2.13}$$

Understandably, the success likelihood of the threat action is actually the same to the probability that all deployed products fail against the considered threat action. Hence, considering all quadrants of CDM structure, CyberARM formulates the success likelihood, P_j^k, of ta_j against asset k, using the following equation:

$$P_j^k = S_k^j \times \prod_{z \in KC} \prod_{y \in EL} \prod_{x \in SF} \prod_{w \in T_{zyx}^{j}} (1 - e_w^j \times S_w) \tag{2.14}$$

where, S_k^j is the maximum CVSS score considering all existing and exploitable vulnerabilities of ta_j. This factor is important because no threat action can be successful without having a exploitable vulnerability, and higher S_k^j means higher success rate of the subjected threat action.

Notably, a product w may reduce risks of multiple threats by inhibiting any of their mutual threat actions, while w can also defend multiple threat actions of a specific threat. Thus, CyberARM incorporates correlations among products, threats, and threat actions into planning evaluation.

2.6.1.3 Global Residual Risk

Global residual risk, Gr_R, is the aggregation of all remaining risks of all threats against all assets of the enterprise. If Rr_i^k is the residual risk of a threat t_i against asset k, CyberARM formulates Gr_R using the following equation:

$$Gr_R = \sum_{k \in EA} \sum_{t_i \in T_h} Rr_i^k \tag{2.15}$$

where, EA is the asset list of the enterprise, and T_h is the list of existing threats

across all assets of the enterprise.

Residual risk, Rr_i^k, of a threat t_i at asset k depends on (1) the successful exertion likelihood, Pr_i^k, of t_i against asset k, and (2) the impact of the threat I_i^k on k regarding loss of CIA. CyberARM formulates Rr_i^k using the following equation:

$$Rr_i^k = \sum_{l \in C,I,A} Pr_i^k \times I_i^k(l) \tag{2.16}$$

where, $I_i^k(l)$ is the loss of l (C, I, or A) due to successful exertion of t_i.

The successful exertion likelihood, Pr_i^k, of Eqn. 2.16 depends on two factors: (1) Exertion likelihood $p(t_i|a_k, d_k)$ (determined in Eqn. 2.5), and (2) Success likelihood P_i^k (determined in Eqn. 2.10). Based on these two values, CyberARM formulates Pr_i^k using the following equation:

$$Pr_i^k = P_i^k \times p(t_i|a_k, d_k) \tag{2.17}$$

2.6.1.4 Budget and Risk Appetite Constraint

The recommended CDM must guarantee less remaining residual risk Gr_R (from Eqn. 2.15) than risk appetite A_R, that CyberARM formulates using constraint U_{gr}^c.

$$U_{gr}^c : Gr_R \leq A_R \tag{2.18}$$

Besides, the installation/deployment cost I_C of CDM must not exceed the affordable user-given budget B, that CyberARM formulates using constraint U_B^c.

$$U_B^c : I_C = \sum_{w \in T} C_w \times S_w \leq B \tag{2.19}$$

where, T contains all the products of candidate-set. Notably, both U_{gr}^c and U_B^c are mandatory constraint that the user must provide to CyberARM.

2.6.1.5 Return on Investment Constraint

Restrained Return On Investment (ROI) restricts the feasible solution space to mandate CyberARM to recommend a better solution regarding cost-effectiveness, while satisfying all given user-requirements (constraints). In Eqn. 2.20, ROI_U is the user-given minimum ROI requirement, and ROI_D is the ROI of the recommended cybersecurity planning (CDM) D.

$$ROI_D = \frac{(G_R - Gr_R^D) - I_C}{I_C}$$
$$U^c_{ROI_D} : ROI_D \geq ROI_U \tag{2.20}$$

where, Gr_R^D is the remaining residual risk if D is deployed as CDM. Therefore, increasing ROI_U compels CyberARM to recommends better solution if any, but reduces the number of satisfiable solutions. Notably, ROI_U is a mandatory constraint, but CyberARM searches for optimal ROI_U if not given by the user.

2.6.1.6 Enterprise Oriented User Requirements

Alongside the mandatory constraints (risk appetite, budget, and ROI), the user can provide optional diversified mission-oriented policies. This subsection describes some of enterprise oriented and non-mandatory user requirements that CyberARM currently can formulate.

- ***Multi-Layer and Multi-Control Resiliency User Requirements:*** CyberARM can satisfy users' heterogeneous multi-layer (vertical) and multi-control (horizontal) resiliency (diversified security control products with different security functions) configurations. Multi-layer resiliency (vertical) is the capability of defending a threat action at different enforcement levels to enable defense-in-depth and also to avoid the single point of failure. For example, the following constraint $U^c_{m_l}$ describes the enforcement of a minimum one product with se-

curity function of Protect ($SF(2)$), Detect ($SF(3)$), or Response ($SF(4)$) in at least three different enforcement levels.

$$F_{ta_k} = \{y | y \in EL \wedge \exists w (S_w \wedge w \in \bigcup_{x \in Rs_d} (\bigcup_{z \in KC} T_{zyx}^{ta_k}))\}$$

$$U_{m_l}^c : |F_{ta_k}| \geq 3$$

where, F_{ta_k} is the set of deployed security products to defend asset k against threat action set ta_k, and $Rs_d = \{SF(2), SF(3), SF(4)\}$.

- ***Cost Distribution:*** The enterprise may have a security portfolio preferring particular security functions, enforcement levels, and kill chain phases. In such cases, CyberARM introduces constraints across the dimensions of CDM structure. Even, it is possible to define such preference for a specific quadrant of CDM structure. To clarify, three constraints named Cost Distribution constraints are described below.

  The first requirement, $U_{d(2)}^c$, imposes the constraint that the rate of investment in deploying products offering *Protect* with respect to the total deployment cost cannot be less than $c_f \in [0, 1]$.

$$I_{C_{d(2)}} = \sum_z \sum_y \sum_{w \in T_{zySF(2)}} C_w \times S_w$$

$$U_d^c(2) : I_{C_{d(2)}} \geq c_f \times I_C$$

  where, $I_{C_{d(2)}}$ is the total deployment cost of considered CDM to ensure *Protect*.

The first requirement imposes minimum $c1\%$ investment of total investment in protecting assets depicted by a constraint on “Protect" of SF axis. $IC_{SF(x)}$ is the installment cost at xth security function of SF axis hence, $IC_{SF(2)}$ represents the

cost at “Protect". Similarly, I have described other two constraints $Cons_{EL(2)}$ and $Cons_{KC(3)}$ across EL and KC Phase axis respectively where $IC_{EL(2)}$ and $IC_{KC(3)}$ are the cost at “Network” of EL and “Delivery” of KC phase respectively.

$$IC_{SF(2)} = \sum_{z \in KC} \sum_{y \in EL} \sum_{w \in (z \cap y \cap SF(2))} C_w \times S_w$$

$$Cons_{SF(2)} : IC^{zy}_{SF(2)} \geq \frac{c1}{100} \times IC$$

$$Cons_{EL(2)} : IC_{EL(2)} \geq \frac{c2}{100} \times IC$$

$$Cons_{KC(3))} : IC_{KC(3)} \geq \frac{c3}{100} \times IC$$

2.6.1.7 K-Resiliency Constraint

The users can enable K-Resiliency (K is the maximum number of failed security products) to aspire guaranteed resiliency against a group of threat actions (RS_{ta}) for some assets or may desire more strict resilient configuration for some crucial assets. The framework can support such K-Resiliency requirements, and in the following formula, we introduce a constraint defining K-resiliency for some assets, RS_{EA} against specific threat actions. Here, ta_k is a threat action against asset (k) and F_{ta_k} is the number of selected products applicable against it. Similarly, we can build constraints for other user-defined resiliency configurations to fed to CyberARM.

$$Cons_{K_{Res}} : \forall_{ta_k}((k \in RS_{EA}) \wedge (ta_k \in RS_{ta})) \implies (F_{ta_k} > K)$$

2.6.1.8 **Discovering Cyber Defense Matrix (CDM)**

As the ultimate goal, CyberARM fills each quadrant of the CDM structure with the subset of the candidate-set applicable at that quadrant. Thereupon, CyberARM aggregates all chosen subsets of deployable products across all the quadrants of the CDM structure to construct **CDM** which satisfies all given constraints ($Cons$). CDM

is the desired cost-effective cybersecurity planning.

$$Cons = \bigwedge_{r} Cons_r, \quad where, r \in \{B, ARisk, ROI, UC\}$$

$$CDM = \bigcup_{k} \bigcup_{ta_k} \bigcup_{D} T_D^{ta_k} \models Cons, D = KC \times EL \times SF$$

CyberARM follows the approach described in algorithm 3 to compute the desired

Algorithm 3: Discover Cybersecurity Portfolio

Input : Threat Action Set T_P, Budget B, Risk Appetite A_R, ROI and User Constraints $Cons$

Output: Cybersecurity Portfolio CDM

`// This function computes the cost-effective cybersecurity portfolio.`

m_t = findMeanEffectiveness() `// Compute the mean effectiveness of security control products.`

$N_x = \frac{\log(0.005)}{\log(1-m_t)}$ `// Find the number of maximum number of security products that will keep the success rate of a threat action less than 1%.`

$Gr_R = -1$ `// Residual Risk`

$CDM = none$

`// Incrementally improving CDM through iterating from` $N = 1$ `to` $N = N_x$.

for N *in* $range(1, N_x + 1)$ **do**

  `// Calling Algorithm 2.`

  $candSP$ = chooseSecurityControlProducts(T_P,N)

  `// Minimum global residual risk if whole` $candSP$ `is deployed.`

  Gr_{min} = minResidualRisk($candSP$)

  if $A_R >= Gr_{min}$ **then**

    `// Call SMT Solver to compute the` CDM.

    M_d = Solver($Cons$,CDM)

    if M_d *is UNSAT* **then**

      break

    $CDM = M_d$.getModel()

    $Gr_R = CDM$.getResidualRisk()

    $ROI_U = ROI_U + 1$

    `// Change the constraint of ROI (cost-effectiveness).`

    changeRiskAppetiteConstraint($Cons$)

CDM of an enterprise. This algorithm takes the prioritized threat action set T_P,

budget B, risk appetite A_R, and all requirements $Cons$ as *Input*, and it provides the cost-effective CDM as output. In line 1, it finds the mean effectiveness of security products using Eqn. 2.8 that is used to find the maximum number of security products, N_x at line 2. This N_x approximates how many security countermeasures needs to be deployed against a threat action to keep its remaining risk or success likelihood at a negligible level (0.5% at the algorithm) considering the mean effectiveness of security products.

To find optimal N^*, this algorithm starts from 1 and continues incrementing by 1 until reaching N_x. CyberARM uses linear search instead of binary search because binary search confronts more unsatisfiable scenarios than linear search. According to experiments in evaluation, on average, figuring out unsatisfiability takes more time than discovering a satisfiable solution; as a result, linear search is faster in approximating a better solution. Most importantly, binary search is not suitable for the problem as it may quit without searching a lower N that may have better solution but within the range of the previous two values of N with satisfiable scenarios.

In line 6, the algorithm chooses the set of security control products that can defend threat action set T_P, where N is the number of maximum security control products against a specific threat action. In line 7, the algorithm finds the minimum risk Gr_{min} that will remain despite deploying all security products of $candSP$. In line 8, it checks whether the risk appetite A_R is less than Gr_{min}. If it does not satisfy, the algorithm tries with another N as there is no way to find a satisfiable solution. Otherwise, at line 9, it calls SMT-solver to compute a CDM with $Cons$ and CDM (computed at previous iteration) as arguments. This SMT-solver tries to improve the previously computed CDM by introducing increased ROI_U (line 14) in $Cons$, and the computed CDM must satisfy all requirements in $Cons$. If the solver finds a satisfiable solution for increased ROI_U, it saves the model and residual risk. In line 15, the minimum ROI_U is increased in $Cons$ that is used for the next model discovering, which compels

the SMT-solver to find a better solution concerning ROI_U.

2.6.2 Model Decomposition

Though the candidate set pruning improves the performance of CyberARM significantly, an enterprise with a massive number of critical assets may still suffer for the exponential growth of the search space. Therefore, an algorithm is developed to divide the problem into a finite number of independent subproblems, and the aggregated solution of all the subproblems still satisfies the user-defined requirements of $Cons$. To do so, the problem space for each of the sub-problem requires to remain equal to achieve better performance. The complexity of a problem largely depends on the number of assets, risk tolerance (difference rate between the risk appetite and minimum residual risk), budget deficiency (difference between the given budget and maximum cost to deploy all products of candidate-set), which is also observed in the experiment (shown in 2.7.3.1). Th risk tolerance R_T and budget deficiency B_d are defined using the following equations:

$$R_T = \frac{A_R - Gr_{min}}{Gr_{min}}, \quad B_d = \frac{max(0, M_c - B)}{M_c}$$

where, A_R and B are the given risk appetite and budget respectively, and Gr_{min} and M_c are the minimum residual risk and deployment cost respectively if all security products of candidate-set are deployed.

Therefore, this module aims to keep all three properties (i.e., number of assets, risk tolerance, budget deficiency) the same for all sub-problems to distribute the computational hardness equally among all sub-problems. To achieve that, CyberARM distributes assets among D sub-problems in a way that the risk variance, $VarR$, of sub-problem will be minimized, while the difference in the number of assets between two sub-problems cannot be greater than one. Variance $VarR$ is defined using the

following equations:

$$R_k = \sum_{j \in T_P} R_j^k$$

$$VarR = \sum_{n \in D} \sum_{m \in D} | \sum_{k \in E_A^n} R_k - \sum_{l \in E_A^m} R_l | \tag{2.21}$$

where, R_k is the total imposed risk on asset k, T_P is the threat action set, and E_A^n is the asset list in sub-problem n.

Algorithm 4: Model Decomposition

Input : Candidate Security Control Set $candSP$, Sorted asset list on descending order of their risks E'_A, Risk of sorted asset list R', Number of sub-problems D, Budget B

Output: List of assets of sub-problems Sb_A, List of Imposed Risk of sub-problems Sb_R, List of Budget Sb_B

Sb_A =[ ], Sb_R =[ ], Sb_B =[ ] `// Initialize array of asset list, risk, and budget of` D `sub-problems.`

for *i in range(D)* **do**

  `// Iterating loop from 0 to D-1`

  Sb_A.append([ ])

  Sb_R.append([0,i])

  Sb_B.append(0)

$i = 0$, $dir = 1$

for k *in* E'_A **do**

  $Sb_A[i]$.append(k) `// append the asset`

  $Sb_R[i]$ += $R'[k]$ `// add the risk of the asset`

  if *(*$dir == 1$ *and* $i == D-1$*) or (*$dir == -1$ *and* $i == 0$*)* **then**

    dir *= -1 `// changing the rotation`

  i += $dir * 1$

mC = [ ]

for *i in range(D)* **do**

  mC.append(findExpectedCost($candSP$,Sb_A)) `// Find the expected implementation cost of security products to defend assets of a sub-problem.`

`// Budget Distribution among sub-problems`

for *i in range(D)* **do**

  $Sb_B[i] = \frac{Sb_B[i]}{sum(Sb_B)} \times B$

Algorithm 4 describes the approach that decomposes the problem into D sub-

problems by minimizing $VarR$. In order to accomplish these objectives, CyberARM initially sorts all the assets in descending order based on their imposed risk, which is considered as input alongside other variables in Algorithm 4. In line 1-5, this algorithm initializes three lists to return distributed asset list in sub-problems Sb_A, imposed risk of distributed assets in sub-problems Sb_R, and allotted budget for sub-problems Sb_B.

To distribute the assets, this algorithm applies a modified version of round-robin. This modified version has two directions: right (incrementing sub-problem index by 1) and left (decreasing sub-problem index by 1). Therefore, for the first D assets (i.e., right direction), each sub-problem gets one in sequence from 0 to D-1 by increasing index by 1. However, for the next D assets (i.e., left direction), each sub-problem gets one in sequence from D-1 to 0 by decreasing index by 1. This approach minimizes $varR$ while satisfying the constraint regarding the different number of assets in any two sub-problems. This approach is implemented at line 6-12. Afterward, in line 13-15, the framework determines the expected cost of the products in candidate-sets for each sub-problem, which depends on both the deployment cost and benefits of products. In line 16-17, this algorithm allocates the budget for the sub-problem based on its expected cost requirement, in order to maintain similar budget gaps among sub-problems.

2.7 Evaluations

This section discusses the experiment setup and the performance of CyberARM in mitigating risk. This section also describes the sensitivity of CyberARM with respect to many critical factors. Moreover, it provides a detailed analysis on the scalability of the tool.

2.7.1 Experiment Setup

This experiment considers an enterprise network that has 10% diversity regarding asset types (e..g., application, server, desktop) in two different domains (e.g., financial, public information). This experiment distributes the value of the assets using Normal distribution, where an asset value x defines that the asset is x times more expensive compared to the lowest asset value. In all experiments (if not mentioned explicitly), ROI is always greater than 10, and risk appetite is 30% of the total imposed risk when the budget is 60% of the required budget to eliminate 90% of the risk. In all experiments, CyberARM must consider three mandatory requirements: risk appetite, budget, and ROI.

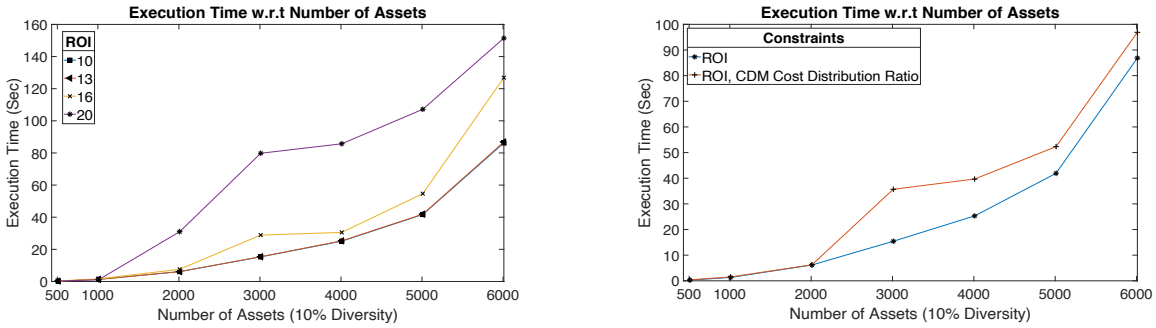
The framework is developed using Python 2.7, and the tool is available in [100]. This web-application provides the feature of CyberARM as software-as-service that can be used to compute cybersecurity portfolio for an enterprise considering its requirements. This web-application is developed using Django framework and Python 2.7. Besides, this web application contains CSC categorization and a comprehensive mapping of security controls and threat actions. As SMT-Solver, this framework uses z3-prover [93].

2.7.2 Scalability

The evaluation of this chapter assesses the scalability of the tool with respect to the number of assets and number of constraints. It is also analyzed how computational complexity changes due to constricting requirements. The evaluation also shows the benefits of applying model decomposition.

2.7.2.1 Impact of Asset Size and ROI

The average execution time to discover the satisfiable model stretches in Fig. 2.5a, because the prioritized threat surface, as well as the candidate-set, widens due to the increase in the number of critical assets. The performance of the tool regarding



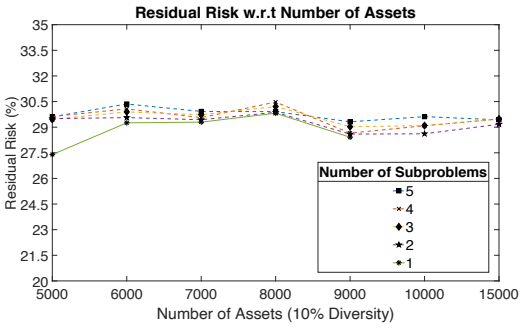
(a) (b)

Figure 2.5: Impact on Execution Time due to (a) Varying number of assets and ROI, and (b) Increasing number of constraints.

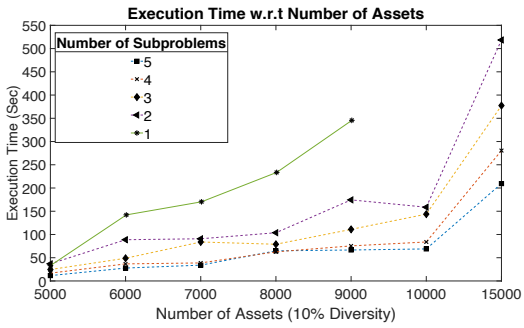
scalability is evaluated by increasing the number of assets from 510 to 6010. Moreover, four different user-defined minimum ROI constraints: 20, 16, 13, and 10 have been introduced. In all cases, I keep 25% budget deficiency and maximum 25% risk tolerance. I observe that the size of the candidate set swelled up approximately 8.5 times due to the increase of assets from 510 to 6010. In the figure, the required time for 6010 assets is 30 times than the time of 1010 assets, whereas it is more than 9 times than the time of asset-set 2010 for ROI 10. Hence, the computational complexity of CyberARM increases approximately in a quadratic fashion. Additionally, though the required time is similar for ROI 10 and 13, the comparison of the required time between ROI 20 and ROI 10 shows that the complexity of the tool changes non-linearly for the increase of minimum ROI due to the shrinking of solution space. However, discovering the solution for more than 6000 assets within 2 minutes shows CyberARM's applicability for large problem.

2.7.2.2 Impact of User Constraints

Different cost distribution constraints (optional constraints) are introduced to understand the growth of complexity due to increasing the number of constraints. In Fig. 2.5b, the blue curve illustrates the execution time when there is only mandatory constraints (i.e., budget, risk appetite, and ROI), and the red curve illustrates the



(a)



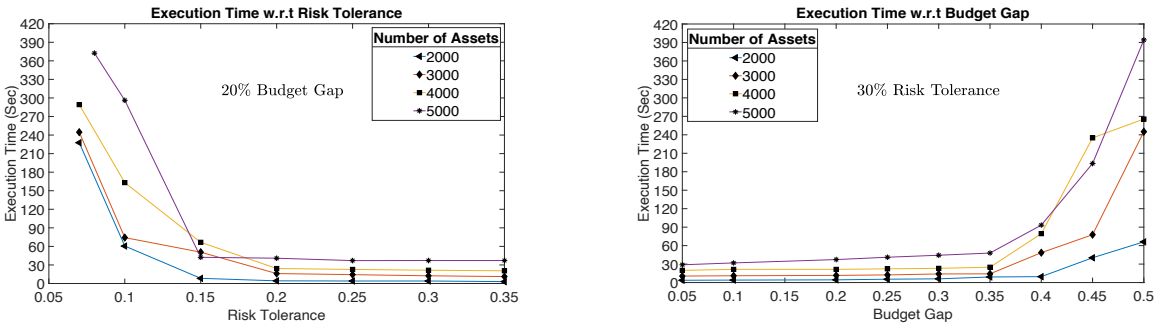
(b) .

Figure 2.6: (a) Impact on risk mitigation due to increasing number of sub-problems, and (2) Impact on execution time due to increasing number of sub-problems.

execution time when there are both optional and mandatory constraints. Two curves are almost similar until 2000 assets, but it deviates for further increase of assets. The deviation shows that the computational complexity enhances with the increase of number of constraints. Though the execution time increases with the increase of constraints, the deviation between these two curves remain same despite the increase of number of assets. This shows that the increasing of number of assets does not drastically change the computational complexity when the set of constraints is same.

2.7.2.3 Impact of Model Decomposition

In this experiment, I evaluate the performance of the presented decomposition algorithm that divides the problem into D number of similar sub-problems. Experiments have been run for several D: 1, 2, 3, 4, and 5, where $D = 1$ represents the global problem. In Fig. 2.6b, the required time of discovering a model is minimum for $D = 5$ and maximum for $D = 1$. Markedly, CyberARM could not find the solution within 600 secs for 10K and 15K assets for $D = 1$. Though the complexity of the problem is decreasing due to the increase of D, there is a payoff regarding residual risk due to subdivisions. As seen in the figure 2.6a, the divergence from the cost-effective solution (residual risk when $D = 1$) increments with the increase of D. However, the maximum deviation is approximately 3% which is very low compared to the gain



(a) (b)

Figure 2.7: Impact on Execution Time of (a) Risk Tolerance, (b) Budget Deficiency.

in the required time. This shows that CyberARM's model decomposition improves computational complexity significantly while not compromising too much regarding effectiveness of security portfolio.

2.7.3 Performance Analysis

This section describes the performance of CyberARM in mitigating cyber risk with respect to the given budget. It also analyzes the impact of N (i.e., number of security control products against a specific threat action) on risk mitigation for varying budgets. This section also discusses the sensitivity of the tool to critical factors. In this section, risk mitigation threshold, MT, defines the ratio of threat actions subjected for elimination.

2.7.3.1 Impact of Complexity Index

The running time of CyberARM is directly proportional to the Complexity Index CI (defined in Eqn. 2.22), where B_G is the budget deficiency, R_T is the risk tolerance, and N is the number of security control products chosen against a threat action. CI' represents the complexity index of the previous time.

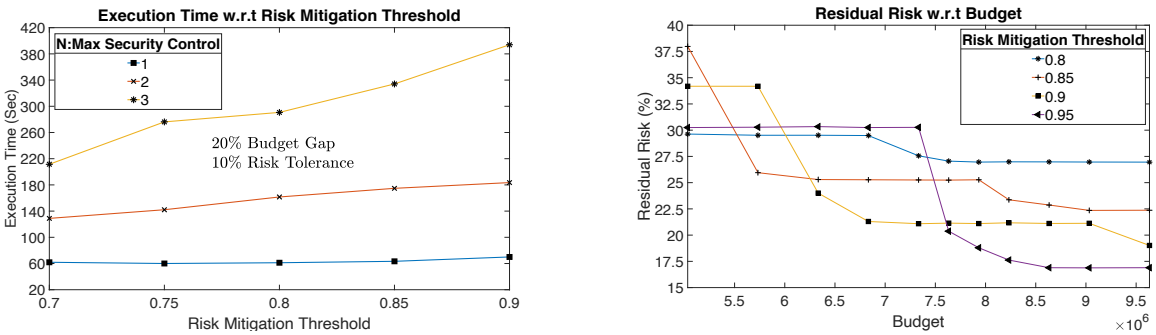
$$CI \propto CI' \times \delta N \times \frac{(\frac{\delta B_G}{2})^2}{\delta R_T^2} \tag{2.22}$$

- **Decreasing Risk Tolerance:** The performance of CyberARM is evaluated

for varying the risk tolerance R_T from 5% to 35% for the fixed budget deficit B_G and N. This experiment considers 20% B_G for all four asset-sets E_A of sizes: 2000, 3000, 4000, and 5000. However, at 5% risk tolerance, a satisfiable model is found only for 2000 asset-set. The minimum risk tolerance for other ascending 3000, 4000 and 5000 asset sizes are 6.5%, 7%, and 8% respectively. The progression of the execution time for rising R_T is plotted in Fig. 2.7a, and it shows that there are quadratic declines in the execution times from 10% to 15% risk tolerance for all asset-sets, and the slope for 5000 asset set is much steeper. The increment of R_T widens the solution space by increasing the inter-distance between minimum risk and risk appetite. This is why, in Eqn. 2.22, CI is inversely proportional to the square of R_T increment. However, the required time becomes constant after reaching 20% risk tolerance as the solution space becomes sufficiently wide.

- **Increasing Budget Gap:** With the amplification of B_G, the performance of CyberARM exacerbates as the solution space erodes because of inflation of budget deficiency. Figure 2.7b illustrates the performance in terms of execution times for same asset-sets: 2000, 3000, 4000, and 5000 at fixed R_T ($R_T = 30\%$) while increasing B_G from 5% to 50%. The increase rate of execution time is approximately zero until 35% budget gap followed by quadratic escalations of all four plots. Therefore, these patterns in Fig. 2.7b prove that the performance of the tool quadratically degrades with an increase of B_G for fixed R_T and N, which satisfies the square proportionality in CI.

- **Fixed $\frac{BG}{RT}$, Varying N:** The performance of the tool for various N is evaluated while keeping the ratio of B_G and R_T fixed for the same asset-set 5000. It is observed in Fig. 2.8a that the performance for $N = 1$ and $N = 2$ are not changing much, though the candidate-set of products has grown twice of



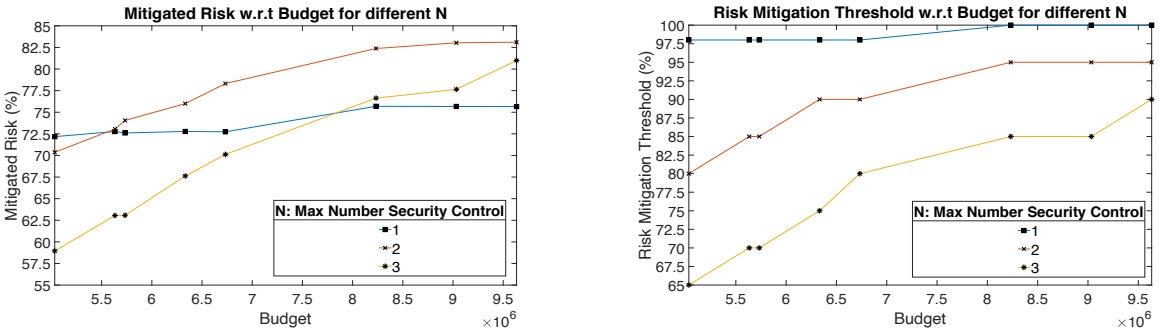
(a) (b)

Figure 2.8: (a) Impact on Execution Time for same Complexity Index, and (b) Impact on Residual Risk of Risk Mitigation Threshold (ratio of threat actions prioritized for mitigation).

its initial size due to increase of M_T from 0.7 to 0.90. Though the execution time for $N = 3$ requires 50 seconds more for 5% increase of M_T, it is still small compared to the expansion of problem space. So, the computational complexity is dependent on N, while the other factors of CI remain fixed. In this experiment, CI is same for same N, and therefore, the increase rate is negligibly changing despite the increase of MT.

2.7.4 Impact of Risk Mitigation Threshold

In this experiment, I try to analyze how the threat prioritization regulates the performance of CyberARM and how its behavior changes with the increase of budget. This experiment keeps minimum ROI requirement same and aims to minimize residual risk. With the increase of MT, the threat surface includes the less impactful threat action, which increases the problem space. Therefore, the increase of budget deficiency BG is high compared to risk tolerance RT, that increases CI due to the increase of MT. As a result, due to the bounded rationality, irrational MT considering current budget degrades the performance due to finite amount of time for solving. Figure 2.8b illustrates the mininum Residual Risk achieved by CyberARM at various MT for increasing budget. As can be seen, residual risk is minimum for $MT = 0.85$ during the budget ranging from 5.5 unit to 6.5 unit (1 unit = 10^6) which



(a) (b)

Figure 2.9: (a) Impact on Mitigated Risk with respect to Budget for varying Maximum Number of Security Control Product (N), and (b) Optimal Risk Mitigation Threshold with respect to Budget for varying Maximum Number of Security Control Product.

$MT = 0.90$ surpasses with the enhanced budget. For further addition of budget after $Budget = 7.5unit$, the residual risk is minimum at $MT = 0.95$. Therefore, CyberARM's performance in computing cybersecurity portfolio also largely depends on minimum imposed risk m_R of threat prioritization module.

2.7.4.1 Impact of N on Risk Mitigation Threshold

In Fig. 2.9a, the performance of $N = 3$ is worst whenever $N = 1$ is better than $N = 2$, where N is the number of security controls selected against each threat action. However, after $Budget = 8$ unit, the curve of $N = 3$ surpasses the curve of $N = 1$. The reasoning for such behavior is clear from Fig. 2.9b where optimal MT is gradually increasing for each N and is high for smaller N. The difference in optimal MT for $N = 1$ and $N = 2$ is 17.5% initially which decreases gradually due to the increased budget. Therefore, with the increase of budget, resilient configuration offers more benefit and thereby, $N = 2$ offers less residual risk than $N = 1$ eventually in Fig. 2.9b.

2.7.5 Use Case Study

In order to demonstrate the validity and the robustness (sensitivity to noise in asset-set input) of CyberARM, a synthetic dataset is created for an enterprise asset list E_A

Table 2.2: Reports on Use Case Study

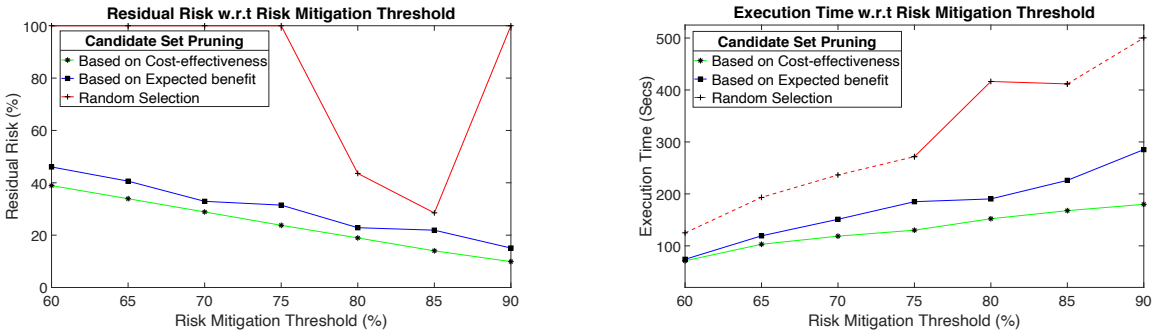
Top Threat Action (TA)				Cyber Defense Matrix (CDM)			
Asset (Type,Domain)	TA Name	Vul. (CVE #)	Residual Risk (%)	Product (CSC)	SF	EL	KC
A5 (Database, F)	Privilege Abuse	2008-3979 2004-1338	8.5	P1 (4.3)	2	3	3
				P3 (4.6)	2	3	3
A21 (System Admin, F)	Phishing	2017-12290 2012-1862	4.95	P12 (17.1)	2	3	3
				P17 (17.3)	2	3	7
				P21 (12.3)	3	1	6
A31 (Web Server, F)	Brute Force	2017-12129 2003-1363	3.4	P4 (4.2)	2	3	4
				P27 (4.5)	2	3	4
A24 (CMS, F)	XSS	2018-9173	5.13	P15 (7.3)	2	4	4
				P19 (12.5)	3	1	3
A33 (Router, F)	Backdoor	2004-1921 2014-0659	5.13	P37 (3.6)	3	4	4
				P15 (12.4)	2	1	7
				P7 (8.1)	3	2	8

Summary	
Properties	Value
User Input (Constraints)	
Risk Appetite	30%
Min. ROI	15
Budget	$ 505.7K
Mitigation Threshold	0.8
Output (CDM)	
Global Residual Risk	24.9%
ROI	20.5
CDM Cost	$ 505.3K
Execution Time	189s

with 5000 critical assets. I have produced vulnerability reports from the descriptions of CVE repositories and generated a collection of threat incident reports using VERIS, Symantec, and incident reports obtained from our industry collaborators. T

- **Report on Use Case:** Table 2.2 contains a partial snapshot of the problem, where *Top Threat Actions (TAs)* enlisted five most menacing threat actions among the prioritized threat actions. The most deadly threat action for the given E_A is "Privilege Abuse" which exploits the existing CVE-2008-3979 and CVE-2004-1338 for transforming threats "Hacking" and "Misuse" into successful attacks against A5. Here, A5 is a "Database" server in "Financial (F)" domain. Besides Budget, Risk Appetite, and Minimum ROI constraints in the table, three user requirements are introduced, which specify minimum investment in "Protect" (SF axis), "Application" (EL axis), and "Exploit" (KC axis) require to be 40%, 15%, and 25% respectively. Table 2.2 also shows the CDM computed by CyberARM to defend these threat actions. To clarify, it chose P1 (4.3) and P3 (4.6) as the effective countermeasures to prevent "Privilege Abuse", where P3 (4.6) implements CSC 4.6 that resides in $CDMQuadrant(Delivery, People, Protect)$. From table 2.2, we can see that CDM satisfies all mandatory requirements and have invested 51%, 19%, and 27% across "Protect", "Application", and "Exploit" respectively.

- **Impact of Model Reduction:** In this experiment, I evaluate the model re-



(a) (b) .

Figure 2.10: Comparing the performance of CyberARM with two different approaches of pruning in terms of (a) Residual Risk, and (b) Execution Time.

duction approach based on cost-effectiveness (CyberARM's approach) with two different approaches. In one approach, I prune the candidate-set based only on expected benefit, whereas, in another approach, I randomly select a specific number of products. Fig. 2.10a and Fig. 2.10b represent the performance due to pruning regarding residual risk and the execution time. CyberARM's approach has the minimum residual risk for all different risk mitigation thresholds MT (Fig. 2.10a), where the random approach has satisfiable models only for two MT: 80% and 85%. The pruning based on expected benefit is less effective than CyberARM's approach as some products although having high benefits may be very expensive. The dotted line represents the time required to prove the unsatisfiability. As can be observed, both the residual risk and the execution time is better if the framework prunes based on cost-effectiveness.

- **Impact of Noise in Asset Values:** The user-given asset value in the enterprise asset-set E_A concerning CIA may contain errors or noise. I discover CDMs for three different distributions of asset values; Normal, Uniform, and Power Law, while injecting noise in new 5% asset value incrementally until reaching 30% gross noise ratio. The deviation rate of planning from the ideal CDM (no noise) w.r.t. the total noise ratio is illustrated in Fig. 2.11. The figure shows

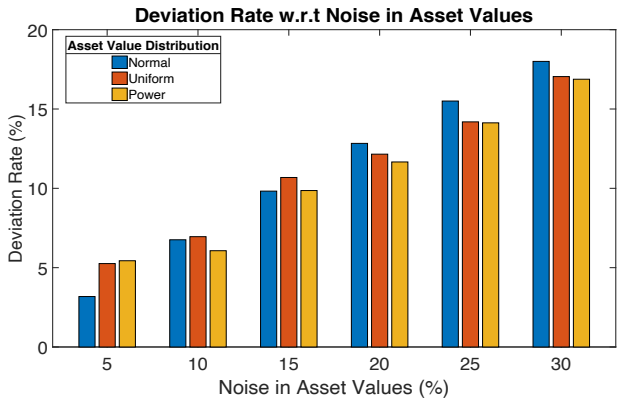


Figure 2.11: CyberARM's robustness against noise in asset values.

that the divergence due to noise is not drastic, instead approximately equal to the noise percentage. Therefore, noise cannot incur the avalanche effect in our decision-making, that proves the robustness of CyberARM to noise.

2.8 Summary

This chapter presents models, frameworks, and implementation of an automated tool named CyberARM that computes the cyber risk mitigation portfolio of an enterprise to minimize cyber risk after satisfying all enterprise-oriented requirements. Alongside minimizing risk, CyberARM also verifies its conformity with the given requirements. Even for large enterprise, CyberARM requires few minutes to compute its portfolio applying its model decomposition and reduction approach. According to evaluation, it computes a cost-effective security portfolio for an enterprise with 15k assets within 100 seconds, which proves its applicability for both offline and online decision-optimization.

In the evaluation, CyberARM's model reduction approach is compared with other possible reduction approaches, and evidently, CyberARM's model reduction based on cost-effectiveness defeats other approaches regarding mitigated risk, as well as computational complexity. Moreover, CyberARM's model decomposition approach does not deviate significantly from the optimal planning while reducing the complexity significantly. These experimental observations prove the rationality of CyberARM's

heuristic assumptions. This chapter also analyzes the sensitivity of CyberARM towards many critical factors, that can provide valuable insights for further improvement of defense planning concerning bounded rationality. However, further analysis of sensitivity to many other critical factors is required to improve the model decomposition approach, that I plan to conduct in future extensions. As per evaluation, CyberARM's planning does not deviate drastically from the optimal plan in cases of noise in asset values, which shows its robustness.

This chapter focuses on selecting an optimal subset of security controls to defeat the menacing threat surface against the enterprise. However, it does not concern how to design such security controls that define fine-grained defense actions to be enforced against specific attack types (e.g., DDoS, malware propagation). Designing such security controls requires an understanding of the attack scope and behavior of the environment where it will be deployed. Therefore, the next chapter focuses on composing fine-grained and dynamic defense compositions against I-DDoS attacks considering the dynamic environment and attack behavior.

CHAPTER 3: Autonomous Cyber Defense Against Adaptive Multi-strategy Attacks

Infrastructural Distributed Denial of Service (I-DDoS) attacks continue to be one of the most devastating cybersecurity attacks today [101]. In I-DDoS attacks, attackers target core backhaul links to impede the availability of critical networks or servers, in order to block legitimate traffic from reaching to victim servers without confronting any resistance from end-system defenses [24, 25, 26]. Moreover, I-DDoS attackers are highly sophisticated because they are constantly learning about network conditions and adapting their strategies dynamically to evade existing defense mechanisms. Therefore, attackers adopt mixed strategies with varying target links, traffic rate (aggressive or low), bots' location distribution (sparse or dense), and distribution of decoys servers to maximize attack stealthiness.

Existing DDoS mitigation approaches generally leverage the following two main techniques: (1) *traffic limiting/filtering* that aims to maximize the attack traffic drop [60, 24, 25, 26], or (2) *traffic diversion* that aims to guarantee the delivery of top prioritized users [74, 71, 54]. Traffic filtering approaches usually use static policies based on traffic flow features such as traffic rate, packet size, source IP, and others [102, 54, 103, 55, 56]. However, sophisticated I-DDoS attackers constantly evaluate the rate of attack traffic drop and network links' conditions, and adapt their attack strategies (e.g., traffic rate of malicious bots, bot distributions, target servers, etc.) to reduce the distinguishability of attack traffic [60, 2]. In addition, traffic filtering at the end-system is ineffective against *indirect* I-DDoS as attack bots send traffic towards decoy servers rather than the end-host victims.

For traffic diversion, the defender reroutes traffic of Autonomous Systems (AS) [74] or whitelisted sources [71] through alternative links with spare bandwidth to avoid the

congested links. However, the attacker possesses a cost-asymmetry advantage over the defenders due to the higher price of link bandwidth but the lower price of bots [60]. To clarify, the cost of bandwidth of a transit link is 7-80 times higher than the attackers' cost of flooding the links [60]. Therefore, in response to traffic rerouting, the attacker may increase the attack traffic volume or extend the target link-set exploiting this cost-asymmetry advantage [2]. In contrast, the defender cannot provide extensive bandwidth to confront such attackers due to high bandwidth price, which exacerbates further due to the special support required in the network infrastructure for traffic rerouting [73, 75]. In addition, defining complete whitelisting is often infeasible, particularly for widely accessed public domain services (*e.g.*, Wikipedia, Github).

Static defense strategies fail against such sophisticated attackers who dynamically evolve to overcome the deployed defenses. Hence, effective I-DDoS mitigation solutions must incorporate a range of defense actions, that are dynamically configurable and composable based on the experience and learning of attack behaviors and defense effectiveness. The state-of-the-art solutions are simply not there yet to provide such adaptive capability for autonomous DDoS defense. Though several static [78, 76, 79, 104, 81] and dynamic game [82, 80, 77] based approaches aim to compose multi-strategy DDoS defense planning, these analyses cannot yield effective solution at real-time from the stance of computing the equilibrium points due to the imposed bounded rationality [83]. On the other hand, some works [83, 85] aiming to enforce real-time defense planning offer coarse-grained adaptations while considering very restrictive attack models that are far from reality. Besides, these works are not applicable (or scalable) for real-time defense optimization against I-DDoS.

3.1 Problem Statement

This dissertation presents a novel approach (framework, model, and implementation) offering a fully autonomous cyber defense mechanism against highly sophisticated and dynamic I-DDoS attackers, called *Horde*. This chapter presents a dis-

tributed multi-agent architecture that protects critical network links collaboratively without human involvement using Reinforcement Leaning (RL) based decision models [105]. This architecture enables the generation of fine-grained multi-strategy compositions of *traffic limiting* and *diversion* at real-time based on learning of dynamic adversary behavior and network conditions. *Horde* computes the optimal sequence of defense actions to force attackers to be bankrupted (e.g., install more bots), detected (e.g., withdraw shrew attacks), or non-lethal, by exploiting the phenomena that an attacker may not be able to achieve high impact, high stealthiness, and low cost simultaneously. Although *Horde* utilizes the traffic flow risk score provided by an Intrusion Detection System (IDS) to quantify the potential of a source being a malicious bot, its decision-model copes with the IDS inaccuracies (false negative and false positive).

The presented approach advances the area of autonomous defense by addressing three key challenges through: (1) *improving* an agent's RL exploration by pruning irrelevant search space, (2) *developing* models to integrate any attack behavior dynamically into agent's decision-process, and (3) *optimizing* defense strategy at real-time. To address the first challenge, *Horde* applies a hybrid approach of RL learning that formulates the RL-agent's environment with two factors: system dynamics and expected attack behavior using the Bayesian chain rule. Such decomposition of the environment with fine-grained information aids the agent to avoid exploring irrelevant actions that may not be possible with traditional RL learning approaches. It aids not only to converge faster but also to avoid executing irrelevant disastrous actions. To address the second challenge, *Horde* extends the POMDP model used to solve RL-model by integrating the expected attack behavior into agent's decision model. This extended model enables decision-making optimization based on not only the experience about defense consequences but also the adaptive attack strategies, without struggling with the complexity and rigidity of stochastic games. With the dynamic

approach of POMDP solving, *Horde* can incorporate changes in network and attack behavior into the decision-model. To address the third challenge, a novel model for autonomous defense, BRITE Loop (Observe-Understand-Investigate-Evolve-Actuate), is developed and integrated into the agents' decision-making process. BRITE model enables agents to not only *observe* and *actuate* but also to *understand* and *investigate* to accurately estimate the environment state under uncertainty, in order to optimize decision-making. It also enables the agent to *evolve* to cope with the dynamic properties of system dynamics and critical parameters. This distributed multi-agent architecture with BRITE loop address the real-time defense optimization challenge.

In summary, the presented approach formulates the agent's decision-making problem as RL-model and applies POMDP to solve it, to compute optimal defense composition through interacting synchronously with the environment and addressing the imperfect attack detectability and network observations [31]. This approach reduces the agent's exploration space and aids in converging towards the optimal solutions fast in a new domain or after a domain shift. The effectiveness of this approach is evaluated against numerous adaptive attack strategies in various network environments. These experiments show that the agent can deliver approximately 97% benign traffic despite diversified attack sophistication and IDS uncertainties.

This chapter offers the following main contributions:

- An effective approach that integrates the attack behaviors, system dynamics, and IDS inaccuracies into the agent's RL decision-model and balances the exploration-exploitation trade-off more optimally.

- A distributed multi-agent autonomous architecture to employ dynamic and optimal defense strategy compositions for protecting critical network links against I-DDoS attackers, where the agent learns to optimize decision-making based on interactive experience instead of relying on static human configurations.

- A model named BRITE loop to define the capabilities of an autonomous cyber defense agent that aims to optimize decision making in a stochastic and dynamic environment.

3.2 Attack Model

The objective of the attacker is to disrupt the availability of critical servers by congesting critical or backbone links that carry significant portion of traffic of target area/network. This section gives an overview of attack types, attack actions, and attack workflow.

I-DDoS Attack Type: The attacker sends traffic from distributed bots (botnet) to execute I-DDoS that are of two types based on attack destinations: 1) Indirect, and 2) Direct. The difference between these is that in *Indirect I-DDoS* attack, instead of sending traffic directly to the target, the attacker sends traffic to selected decoy bots (Coremelt [3]) or decoy servers (Crossfire [2]) which are neighbours of the target area. As a result, critical links of a target area shared with those destination decoys or bots get congested. In Fig. 3.2, there are three critical links: L_1, L_2, and L_3 for target network W. Attack bots send traffic to bots and decoys to congest links L_1 and L_3 respectively, in order to induce delay and drop for benign traffic transmitting through these to reach W.

Attack Actions: At a specific time-sequence t, the attacker may perform same or different actions against different links. This section describes different attack actions, that are mainly classified into three major actions.

1. **Reconnaissance**: The attacker's *Reconnaissance* aims to identify critical links that, if flooded, maximize target area's flow disruptions. To do so, the attacker sends traceroute or pathping commands from attack botnet to target areas to find *flow density* of a targeted link [2, 3], which is high for a critical link [2]. Therefore, many traceroute packets transmit through the link when the attacker performs Reconnaissance across it. As links' flow densities follow power-law distribution, only attacking

few links with high densities satisfies attacker's objectives [2]. However, he periodically performs Reconnaissance to identify new critical links appeared (or disappeared) due to ISP's load balancing approach or traffic engineering [2].

2. **Launch Attack**: If the Reconnaissance identifies the subjected link as critical, the attacker may launch or continue sending traffic with a specific attack tactic through it to induce congestion. Notably, properties such as traffic protocol, ingress ports, and others vary across attack flows. He composes diversified attack tactics by tuning following parameters:

- **Adjusting Botset**: According to attacks in [2, 3], the attacker *generally* adjusts attack botnet by executing one of following: (1) Use Existing Bots (i.e., continuing with previous botset) that compromises stealthiness but causes less attack cost, (2) Extend Botnet (i.e., recruiting new bots or reactivating silent bots) that increases attack volume but causes more cost, or (3) Change Bot Distribution (i.e., replacing a portion of the previous botset) that enhances stealthiness but causes more cost. Therefore, the attacker increases stealthiness by executing *Change Bot Distribution* more [2]. Whereas, he executes *Extends Botnet* to increase attack volume or lower bots' traffic rate without compromising the attack intensity.

- **Adjusting Traffic Rate**: The attacker strategically adjusts his bots' traffic rate by executing one of 1) Use Existing Traffic Rate, 2) Decrease, or 3) Increase Traffic Rate [2]. Generally, decreasing traffic rate increases stealthiness but shrinks attack volume, which is opposite for increasing traffic rate [60]. To avoid early congestion while dynamically assigning attack flow rate, he uses bandwidth estimation tools (e.g., Pathneck [106]).

3. **Inactive**: The attacker may pause or stop sending traffic through a specific link due to strategical reasonings or others.

Attack Workflow: This dissertation assumes that a sophisticated and rational

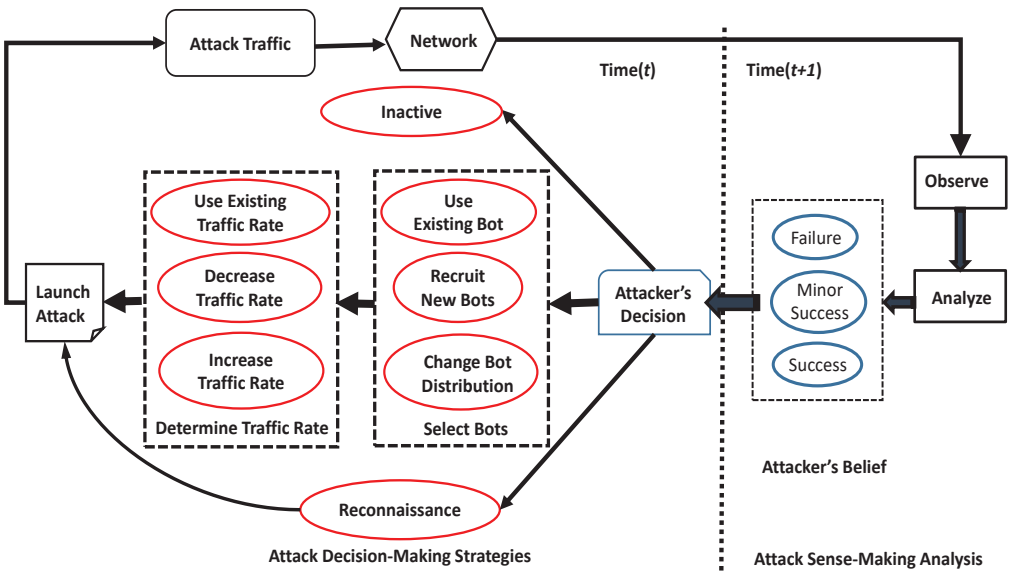


Figure 3.1: Attack Model

I-DDoS attacker is adaptive who observes the network condition and previous attack consequences to decide his next attack action. Therefore, as illustrated in Fig. 3.1, attack workflow has two parts: sense-making and decision-making. In this research, the attacker's has 11 actions including 9 ways of launching attack (i.e., 9 attack tactics) considering possible combinations of adjusting botset and traffic rate. The attacker executes same/different actions for different critical links, and Fig 3.1 depicts the workflow against such a targeted critical link at t.

1. **Attacker's Sense-making**: At the start of t, the attacker receives observations about the link, based on which, he analyzes link utilization, number of bots blocked or rate limited, Quality of Service (QoS), and response ratio of his connection-oriented traffic (e.g., TCP, ICMP). For bots sending connection-less traffic (e.g., UDP), he may send connection-oriented probe packets to reduce uncertainties. However, attacker's observations may not reveal all these information certainly. Hence, through addressing uncertainties, he infers previous attack actions' outcomes (e.g., Failure, Minor Success, and Success) *probabilistically*.

2. **Attacker's Decision-making**: Based on sense-making, the attacker decides his next action that defines whether to send traceroute packets (Reconnaissance), or

attack traffic (Launch Attack with particular tactic), or no traffic (Inactive) through a specific critical link at any time t. This dissertation assumes that the attacker follows a specific *strategy* that dictates his next optimal action based on current environment condition and enacted defense plan. For example, an attack strategy recommends "Decrease traffic rate if most of attack traffic are dropped"; whereas, another strategy recommends "Inactive" for the same case. Importantly, attack strategy considers two sensitive attack parameters: aggressiveness and stealthiness. Stealthy attacker puts more emphasis on stealthiness and prefers stealthy attack tactic. For instance, low detectability of shrewd attack flow motivates *stealthy* attacker to lower traffic rate or change bot distribution frequently. In contrast, aggressive attacker (e.g., time or cost-constrained) puts more emphasis on aggressiveness and prefers aggressive attack actions. For example, they keep traffic rate high or use all bots simultaneously (increases detectability due to repeated use) to achieve his goals within a short time or budget. Moreover, an attacker can change his attack characteristic (e.g., from stealthy to aggressive), that is called *Attacker's Characteristic Type Adaptation*.

3.3 *Horde* Deployment and Architecture

This section gives an overview about the deployment of *Horde* and its components, that is illustrated in Fig. 3.2.

3.3.1 *Horde* Deployment

An upstream entity such as ISP, AS, or large enterprise network deploys the framework, *Horde*, to protect its or clients' critical links, which have several advantages:

- Defending I-DDoS at upstream network point is essential not only due to its effectiveness and flexibility with a view over aggregated traffic behavior, but also due to defense infeasibility at target area [107, 108].

- The upstream *Horde* can offer cost-effective and real-time *defense-as-service* to its clients. In contrast, enterprises unwilling to pay to upstream have to buy and

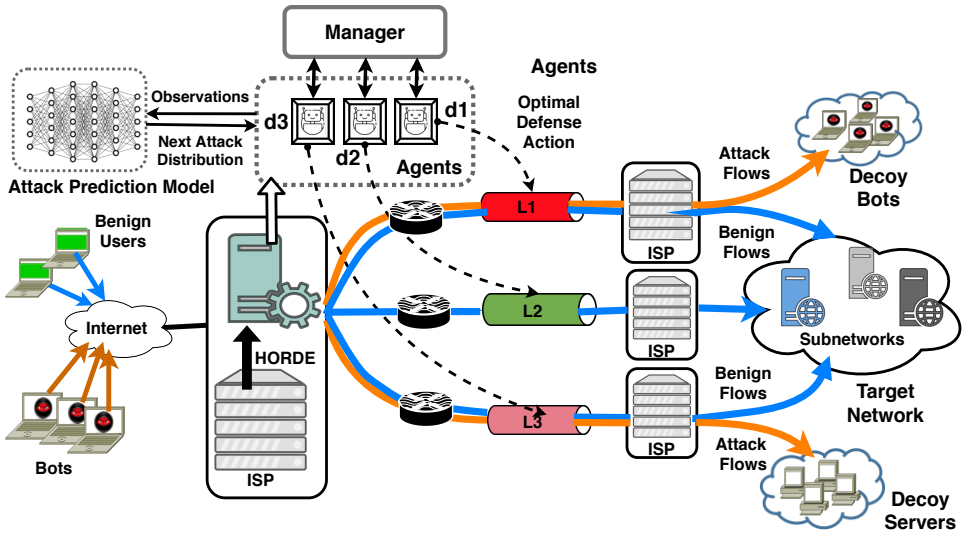


Figure 3.2: Architectural Overview of a Horde deployed at an upstream ISP to protect Critical Links. An agent d_i is responsible to optimize defense composition to protect benign traffic transmitting through link L_i to Horde's customers.

integrate more hardware appliances into their infrastructure to defend diversified I-DDoS approaches, which is tedious, ineffective, and expensive [86].

- Filtering malicious traffic at upstream link is also beneficial for an ISP due to significant reduction in its network usage [109].

These advantages have encouraged commercial efforts from many ISPs [110]. Notably, a downstream ISP can pay to upstream ISP, for instance, tier-3 ISP pays to tier-2 ISP for getting I-DDoS defense service. Moreover, such *defense-as-service* model may induce inter-ISP (among ISPs of the same tier) economic relations, where an ISP collaborates to reroute its traffic through links of other ISPs to employ fast and efficient recovery from failures [70]. Importantly, inter-ISP collaboration is beneficial for a participating ISP because it (1) gets paid for carrying others' traffic through its currently spare bandwidth, and (2) can reroute its prioritized traffic through others in cases of congestions at its links. Therefore, researchers develop and recommend client-to-ISPs (or ASes) and inter-ISP (or inter-AS) collaborations models such as CoDef [111], SIBRA [70], and SENSS [108] to enable efficient DDoS defense. Moreover,

SENSS offers an automated and secured approach to establish such collaborations with low deployment cost on ISPs' existing infrastructures. Though *Horde* does not necessarily require inter-ISP collaboration, *Horde*'s efficacy boosts with more participation of ISPs into the collaboration due to more traffic rerouting options and significant deviations among mutated and previous routes. However, defining economic models to establish such collaborations is out of this paper's scope, but *Horde* can leverage the payment model described in SIBRA.

3.3.2 *Horde* Architecture

Horde mainly consists of the following components:

1) Autonomous Defense Agent: *Horde* employs intelligent and autonomous entities as *Autonomous Agents* which are responsible to enact optimal defense plan against diversified I-DDoS without any human intervention. *Horde* assigns a dedicated agent for each critical link of a *Horde* customer (pay incentives), that *independently* optimizes defense planning at real-time to protect its assigned link. In Fig. 3.2, *Horde* at ISP assigns agents d_1, d_2, and d_3 to protect critical links L_1, L_2, and L_3 respectively for a customer W. Importantly, *Horde* activates an agent of a critical link if it is transmitting an abnormal volume of traffic or traceroute packets, and inactivates if traffic volume is normal for many time-sequences.

Scalability: Generally, a small set of links (i.e., critical links) carries most traffic of a target area (i.e., network or domain) as internet's data connectivity follows power-law distribution [2, 73]. Moreover, according to experiments (using PlanetLab) on numerous enterprises, only four links carry 60-90% flow densities [112]. Hence, the approach of assigning one agent for a critical link is scalable, for which, the owner of *Horde* also gets paid. Besides, an agent is neither computationally nor financially expensive, and *Horde*'s collaboration model ensures no conflict between any two agents.

2) Agents' Collaborations: *Horde* agents mainly collaborate for traffic rerouting with its *artificial manager*, which is responsible for both intra-Horde and inter-Horde

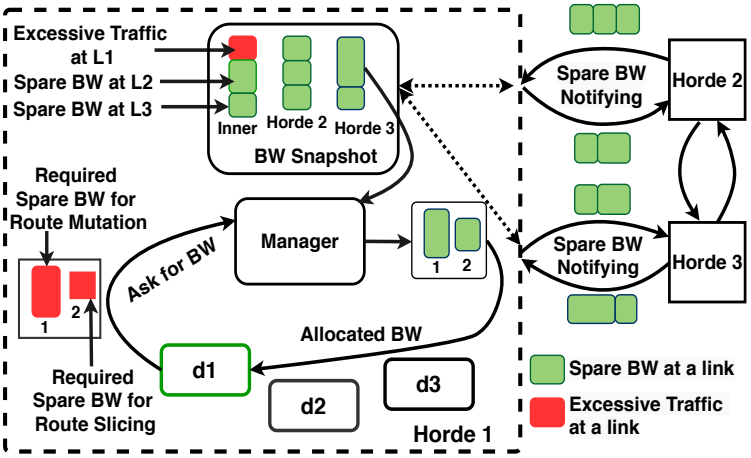


Figure 3.3: Bandwidth (BW) Sharing Among Internal Agents (d_{1-3}) and External *Hordes* through *artificial* manager. #BW box size depends on required or spare BW, and each box represents particular link.

(among Hordes of different ISPs/ASes) collaborations. Fig. 3.3 illustrates intra-Horde collaborations for $Horde_1$, and inter-Horde collaborations of $Horde_1$ with outer $Horde_2$ and $Horde_3$. In $Horde_1$, agent d_i is responsible for critical link L_i, and only d_1 is active due to excessive traffic at its link L_1. To reroute traffic through alternative links with spare bandwidth (i.e., leftover link bandwidth after consumption by current traffic), the manager keeps traces of two bandwidth types: 1) *Inner* bandwidth (spare or required to mitigate excessive traffic) inside the owner ISP, and 2) *Outer* spare bandwidth of other *Hordes*. In Fig. 3.3, the *Inner* stack has one red BW box for L_1 specifying required bandwidth, and two green boxes for L_2 and L_3 specifying their spare bandwidth. Besides, $Horde_1$ is notifying other *Hordes* about its spare bandwidth while receiving notifications about their spare bandwidth.

In Fig. 3.3, agent d_1 asks for spare bandwidth, where red boxes at index 1 and 2 define bandwidth required for route mutation and slicing (details in Section 3.6) respectively. The manager replies with allocated spare bandwidth, where green boxes at index 1 and 2 define spare bandwidth for route mutation and slicing respectively. At a specific time, the agent executes either route mutation or route slicing; hence, the same spare bandwidth can be allocated both at index 1 and 2. Importantly, an ISP

knows routing policies, BGP path information, and customer IP ranges of neighbor ISPs [113], that the manager can access alongside routing configurations of its own ISP. Hence, the manager is responsible to determine whose spare bandwidth to allot for which agents. Besides, Routing Information Base (RIB) storing multiple routes for a destination AS can be used for alternative route selections [114].

The manager temporarily locks (not accessible anymore) the allotted bandwidth if *Inner*, or asks corresponding managers to lock otherwise. However, after computing the optimal defense action by the requesting agent (d_1), the manager unlocks or asks to unlock the non-required bandwidth portion. Thus, Horde avoids conflicts among agents and ensures less traffic rerouting than available spare bandwidth.

3) Attack Prediction Model: As shown in Fig. 3.2, all agents send their observations regarding attack actions to an *Attack Prediction Model*. The *artificial manager* of the *Horde* retrains the model based on these observations to learn the currently adopted attack strategy *actively*. This model may or may not be trained with initial domain-specific data, but it never stops learning that helps it to learn any new attack strategy based on agents' interactive experience. This model not only detects the attacker's strategical characteristic adaptations (e.g., from aggressive to stealthy) but also shows robustness against attack deceptions that aim to deceive the prediction model by executing random actions ignoring his strategy. However, the model assumes that he cannot execute random actions frequently in order to not deviate too much from his goals.

4) Traffic Classifiers to Distinguish Attack Flows: *Horde* accommodates both *signature* and *anomaly* based classifiers/IDS to distinguish attack flows; where, the signature, knowledge, or rule based detector matches a traffic pattern with the previously identified attack signature profiles [115, 116]. In contrary, anomaly based classifiers quantify the the deviations of traffic flows' behaviors from normal behaviors [117]. It is hard, if not infeasible, for an attacker to mimic the aggregated benign

behavior from all aspects. Moreover, some traffic properties/features need to be deviated from normal behavior for successful attack execution. For instance, even the stealthiest attacker has to send abnormal amount of HTTP GET requests (e.g., click an image) to decoy servers to execute Crossfire, that may induce attack alerts [118]. However, considering all features in a single classifier will lead to overfitting due to the curse of dimensionality [119], while identifying the optimal set of features *on-the-fly* against a specific attack strategy is hard. Therefore, *Horde* deploys multiple lightweight *anomaly-based* traffic classifiers, where a classifier is trained with a distinct set of *coarse-grained* traffic features assuming that particular feature combinations may expose specific attack approaches (e.g., DNS amplification, low-rate TCP, and others) better.

Technologies for detecting network anomalies have spanned from time-series forecasting [56] and signal processing [55] to network-wide approaches considering traffic rate [60], entropy [103], packet properties [54, 53], puzzle-game [63], and others [107]. There are tremendous efforts ongoing, boosted by recent advancements of Graphics Processing Unit (GPU), to develop lightweight Network-IDS (NIDS) such as LADS [107], FADM [120], Zeek [121], and SNORT [122]. These IDSs aim to balance the trade-off between scalability and accuracy to detect attack at real-time with *low-latency* at upstream points. However, *Horde* considers the likelihood of poor performance by all classifiers. Each distinct classifier *independently* quantifies the maliciousness of existing flows based on its feature set to provide risk scores. The maximum value among all IDS scores is considered as risk score of a source. For anomaly based classifier, risk score ranges between 0 and 1, but it is either 0 (benign) or 1 (malicious) for signature based classifier.

3.4 Overview and Reasoning of Defense Agent's Decision-making

This section gives an overview of defense objectives, and decision formalization of an agent.

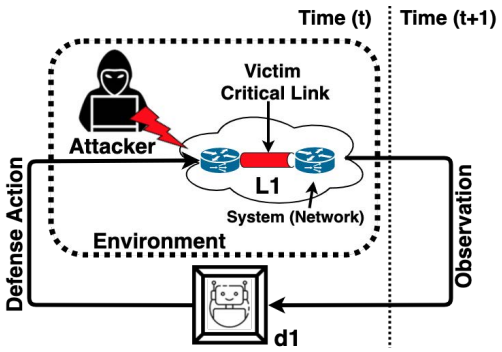


Figure 3.4: Defense Agent Sequential Decision Process

• ***Defense Objective*** The agent mainly applies two defense techniques: (1) *Traffic Limiting* that aims to maximize attack traffic drop, and (2) *Traffic Diversion* that aims to exhaust attack resources or enhance attack distinguishability while ensuring transmission of prioritized flows. The agent considers different compositions of these two techniques, that vary regarding traffic ratios and approaches. Presumably, traffic limiting depends on IDS performance that be jeopardized by attack indistinguishability and induces non-tolerable benign traffic drop. In contrary, traffic diversion imposes risk of delaying benign traffic for rerouting through non-optimal routes, and induces operational costs for route reconfigurations that complicate further while rerouting through multiple alternative links. Moreover, traffic diversion may be infeasible due to scarce of available spare bandwidth.

Therefore, the main objective of an agent is to compute context-aware and optimal composition of traffic limiting and diversion dynamically at real-time to protect its assigned link, where the context depends not only the current environment and expected attack behavior but also the defense deployability, cost, and expected effectiveness. Consequently, the collective defense plan emerged due to all agents' actions maximizes benign traffic serving with minimized cost.

• ***Defense Planning Formulation*** An autonomous agent's defense workflow has two parts: 1) Sense-making that aims to understand the effectiveness and deployabil-

ity of available defense strategies for the current environment, and 2) Decision-making that aims to compute optimal defense strategy based on sense-making. As shown in Fig. 3.4, the agent formulates the environment with two components: (1) current attack behavior that will be the reaction to current defense plan, and (2) system (environment) dynamics that regulate changes of its assigned link due to attack and defense interactions. The environment exhibits stochastic behavior because of (1) adaptive attack behavior specifying that the attacker adapts its attack approaches, and (2) stochastic system dynamics specifying that the consequence of a specific attack and defense interplay is not always same. Therefore, the agent formulates the decision-optimization problem as Sequential Decision Process (SDP) [31] to optimize planning at a such stochastic environment.

Understandably, to optimizing decision-making, the agent needs to know both the attack behavior and system dynamics. However, initially, the agent has no or limited knowledge of these parameters due to lack of deep domain knowledge or data. Hence, the agent's SDP is formulated using RL, where the agent's sense-making understands possible consequences of defense strategies by *actively* (i.e., never stops learning) learning system (environment) dynamics and attack strategy (to predict the next attack reaction) [13].

• ***Solving Agent's Reinforcement Learning (RL) Model*** *Horde* solves the RL-model using dynamic POMDP planning, in order to exploit available knowledge on system dynamics and current attack strategy more specifically. This approach integrates fine-grained information about attacker's behavior into agent's decision-model. In contrast, using traditional model-free RL learning, the agent has to either ignore attack behavior that reduces the accuracy of defense plan or embed the expected attack behavior into state space that makes real-time defense optimization computationally infeasible. As the attack behavior can only be learnt probabilistically, considering probabilities of possible attack actions into state space, even qualitatively,

incurs state-space explosion. Moreover, this approach confiscates agent's exploration to avoid exploring irrelevant defense plan at current environment condition, that aids the agent not only to converge faster towards optimal planning but also to minimize negavtie impact of exploration.

At each time t, the agent creates a new POMDP model, which is a tuple of 7 parameters $(S, A, T, \Omega_S, M_O, R, \gamma_F)$. S is state space, A is defense space, Ω_S is observation space, and O represents observation matrix, which are primitives of agent's POMDP model. T is state transition matrix defining probable effectiveness of defense actions, R is reward function quantifying possible payoffs of defense actions, and γ_f is the discount factor that defines how far into future the agent looks to understand current defense consequences into future. T, R, and γ_f changes with time that the agent always needs to update. By solving the POMDP model, the agent computes an optimal *policy* that recommends the *optimal action* for the current belief, where belief probabilistically define the current environmental condition to address imperfect and incomplete observability of the environment.

3.5 Capabilities of Autonomous Defense Agent

This dissertation presents BRITE loop (at Fig. 3.5) that divides the agent's workflow from observation to action execution into five different phases to generate the POMDP model. This section gives an overview of BRITE loop phases, where first four phases involves sense-making followed by decision-making at last phase.

1) Observe: An agent's sense-making initiates with observing the network symptoms and IDS risk scores for existing flows as feedback from the environment, that may also include benign users' feedback for losing or delaying their packets.

2) Understand: Based on recent observations, the agent analyzes its link condition and attack intensity (using IDS risk scores) to understand the current environment. Based on the analysis, the agent computes *belief* about current state of the environment. This information is critical, because, for instance, traffic limiting

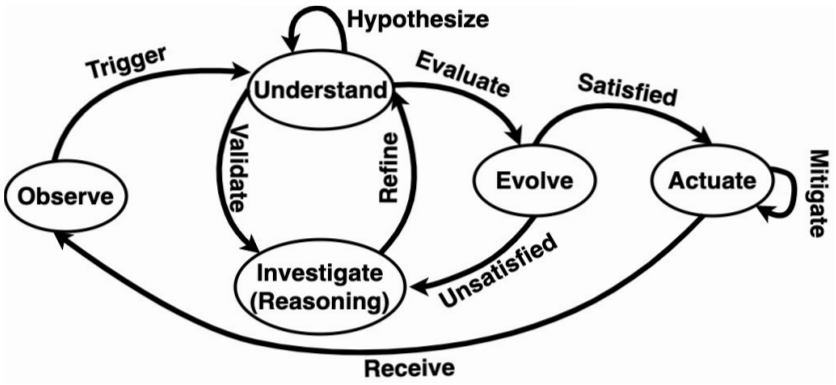


Figure 3.5: BRITE Loop

is ineffective when the congestion is due to abnormal amount of benign traffic. At this phase, the agent characterizes the last attack behavior/action probabilistically and sends it to the Attack Prediction Model that is retrained using all agents' recent attack observations. Then, for all possible defense scenarios at current link condition, the agent learns the next attack action *probabilistically* against its link using the prediction model.

3) Investigate: The agent's understanding about the link condition may sometimes fail to address uncertainties, that necessitates *Investigation* to integrate additional measurements in such cases. Moreover, it aims to tackle adversarial machine learning [123] by investigating whether the recently observed attack action is deceptive or not.

4) Evolve: An agent evolves to cope with the change of environment such as, for instance, though traffic filtering was effective against stealthy attack actions, it is currently failing. Therefore, it has to *actively* refine the knowledge about its link's system dynamics through weighing recent observations and previous knowledge appropriately. By considering the updated system dynamics and predicted attack behavior during *Understand*, the agent creates state transition matrix T. At this phase, the agent also learns discount factor γ_f of POMDP, and updates its expectation about IDS error rate.

5) Actuate: At this phase, the agent computes *expected* rewards R of POMDP for all possible attack and defense scenarios based on its sense-making at previous phases. Using model primitives and recently learnt T, γ_F, and R, the agent creates POMDP and solves it at real-time to get the optimal policy. The policy recommends dynamic and optimal composition of defense approaches based on current environmental condition that is inferred at *Understand* phase.

The next Section 3.6 describes agent's defense approaches that the agent combines to create defense space A consisting of composite defense actions/strategies.

3.6 Defense Approaches of An Agent

The agent applies two approaches of traffic diversion: (1) Route Mutation, and (2) Route Slicing, and two approaches of traffic limiting: (1) Selective, and (2) Sampled.

1. ***Route Mutation:*** Route mutation reroutes traffic of a specific victim destination (i.e., *Horde*'s customer) through an alternative link with spare bandwidth, that makes previous critical link non-critical [71]. *Horde* chooses the alternative link exploiting the property that well-crafted path diversity either *increases* distinguishability or *exhausts* resources of indirect attackers. Because, significant route deviations force the attacker to change decoy servers, decoy bots, or source bots.

The *manager* of *Horde* selects alternative links satisfying constraints such as Reachability (i.e., traffic must reach to the destination), Load satisfaction (i.e., chosen link must be able to carry traffic load), QoS (i.e., tolerable network latency), and Loop avoidance (i.e., no loop at the new route), according to the approach in [73]. Additionally, it tries to minimize the set of common destinations of traffic transmitting through previous and alternative links (if such information is available).

Presumably, to attack the new critical links, attack bots have to exhibit one of the following properties:

(a) **Bots Changing Destination Decoys:** Bots that change destinations showing

high correlations with route mutations are more likely to be detected, because benign users hardly show such correlated behaviors [111, 124]. It also breaks the property of bot indistinguishability of Crossfire attack due to associations with multiple decoy servers [2].

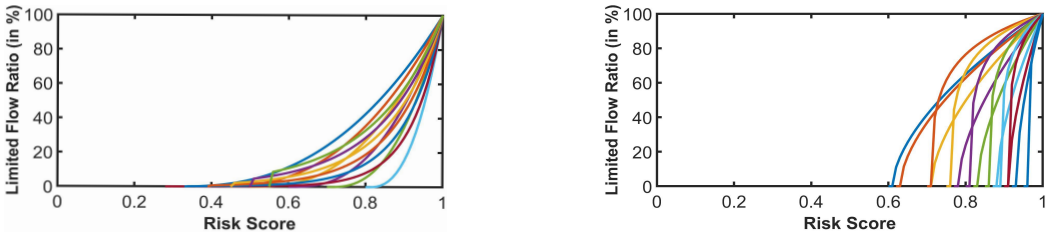
(b) **Bot Disappearance:** These bots stop sending traffic, because their target links are not critical anymore. Besides, persistent traffic sending to specific destination incurs indications as attack contributors [2]. Though these bots seem suspicious, *Horde* cannot detect these only based on disappearance.

(c) **Bot Reappearance:** Bot that disappears with route mutation but reappears again if a particular link becomes critical again, are high likely to be detected. For instance, if source s_i appears when l_j becomes critical and disappears otherwise, then well-crafted traffic diversion reveals attack bot like s_i.

(d) **Newcomers:** Bots appearing for the first time to send traffic to new decoys cannot be suspected with high confidence. However, to avoid properties (a) and (c), these should not be used after route mutation assuming that bots avoid non-critical links. Therefore, frequent change of critical links exhausts attack resources in buying/renting such newcomers.

On principle, attack bots send multiple traceroutes to detect destination decoys and early congestion [2] for indirect I-DDoS, that significantly increases with route mutation. Besides, analysis on Mirai [125] and Conficker [126] reveal that most bots came from small set of ASes [74, 111]. Hence, correlating traceroutes and ASes with suspicious sources can also reduce uncertainties regarding maliciousness. Thus, route mutation enhances detectability of stealthy indirect attack bots. Though route mutation carries direct I-DDoS traffic to newly chosen links, it enhances attack sophistications regarding early congestion avoidance, more attack flows, and others.

2. ***Route Slicing:*** This approach transmits prioritized flows of the agent's link



(a) Timid Limit Function (b) Aggressive Limit Function

Figure 3.6: Rate Limiting Functions. A composite action with timid or aggressive traffic limiting chooses one limit function based on current context and amount of traffic it wants to drop.

through *multiple* alternative links, whose volume cannot exceed excessive traffic unlike route mutation. It actually offers extra *virtual* bandwidth to remove congestions at the critical link. The agent reroutes top prioritized flows for ensuring their reachability. In response, attackers have to recruit more bots (exhaust attack resources) or increase bots' traffic rate (increase detectability). Moreover, indirect attackers have to attack more critical links simultaneously that demands more efforts for selecting decoys, estimating bandwidth across routes, tracing routes, and others.

3. ***Selective Traffic Limiting:*** It drops or restricts traffic rate of suspicious flows by dynamically defining *maximum allowable traffic rate* as threshold that differs among existing sources based on link condition, current traffic patterns, sources' risk scores, and predicted attack strategy. This dissertation presents a new approach of dynamic traffic limiting that is more effective than traditional rate limiting approaches due to not blindly restricting all sources within the same limit without concerning risk scores and other critical environmental factors [83]. Blind restriction induces huge benign traffic drop during large volume attacks due to preference of traffic policing (dropping excessive traffic) over shaping (queuing excessive non-prioritized traffic) [127]. The agent considers following function genres to define thresholds:

- **Timid Limiting:** Timid approach initiates traffic dropping from comparatively lower ranges to find stealthy attack flows. It drops few flows from lower ranges while

increasing drop non-linearly towards higher ranges. This approach is beneficial when attack is comparatively stealthy and IDS has non-negligible false positive rate.

- **Aggressive Limiting:** It adopts comparatively aggressive approach in dropping traffic from higher risk score ranges assuming better detectability of attack flows. However, it still allows some traffic from sources with higher scores.

- **Traditional Blocking:** It drops all flows of a source if its risk score above than a threshold γ_s, that is very effective with highly accurate IDS.

Fig. 3.6a and 3.6b show some timid and aggressive limiting functions satisfying convex and concave properties respectively, where a value at Y-axis represents the percentage of traffic dropped having risk score x (at X-axis). Both timid and aggressive limiting allow some traffic of sources with higher risk scores, which may help to reduce uncertainties regarding their maliciousness. Because, unlike benign users, attack bots either continue sending traffic without concerning maximum allowed rates or disappear. Even if those bots send traffic following maximum allowed rate, the attacker needs to rent new bots to meet attack volume expectation. Notably, a limit function implements *blocking* by assigning allowed traffic rate to 0 and *rate limiting* by restricting traffic rate within [0,1).

4. ***Sampled Traffic Limiting:*** This approach samples sources to drop traffic for finding new attack signatures to boost IDSs' performance. This approach is expensive in terms of dropping benign traffic, and the agent generally executes this approach in cases when it fails with other limiting approaches while having no/less spare bandwidth for further traffic diversion. Generally, such cases may appear against those *indirect* attackers if and only if the attacker can always afford newcomers to defeat traffic diversion. However, executing such indirect attacks is realistically infeasible due to non-zero bot cost and are not considered in this research. In contrary, such cases may be induced by *direct* I-DDoS attackers who can afford traffic sending at

very low rate. However, such attacks, though direct, is very difficult to launch due to sending traffic to few destinations that increases detectability and requiring high number of bots to keep traffic rate low. Hence, it is rational to assume that such direct attacks, if happens, can only occur at victim premises instead of highly provisioned ISP links.

At this approach, the agent creates clusters based on traffic properties such as source and destination IP, source AS, traffic rate and protocol, and others. It drops traffic uniformly from all clusters, and the dropped flow from a cluster is chosen randomly. This approach assumes that the volume of attack traffic is higher than benign traffic; hence, the probability of dropping attack traffic is higher despite random limiting. To known whether sources of dropped traffic are malicious or not, their destinations send puzzles [62] to those sources assuming that benign sources will solve these puzzles. Thus, the agent may discover new attack traits that can probabilistically detect a portion of previously stealthy attack bots.

The next Section 3.7 discusses primitives of *Horde*'s POMDP models.

3.7 An Agent's POMDP Model Primitives

This section describes basic components of an agent's defense model.

3.7.1 Defense Action Space

Horde's defense space A consists of multiple composite defense actions/strategies that are created by combining traffic limiting approaches with traffic diversion approaches. Each composite action is distinct considering limit function type, traffic diversion type, and diversion/limiting ratio. Table 3.1 shows A, where $\mathcal{T}_M$ is the excessive traffic volume that needs to be mitigated to remove congestion from the agent's link, and $\mathcal{T}_C$ is the volume of traffic of victim clients. In the table, *Inactive* action defines that the agent does nothing.

In the table, each composite defense action has particular traffic diversion volume

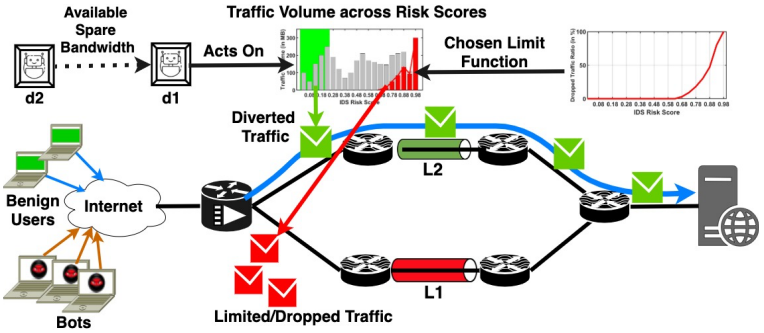


Figure 3.7: An Optimal Defense (Composite) Strategy that agent d_1 is executing on its link L_1 by borrowing bandwidth from d_2. The red portion in the plot of traffic volume specifies dropped traffic, green portion specifies rerouted traffic, and all other traffic are transmitted through L_1.

Table 3.1: Defense Action Space A

Defense Type	Limit Function Type	Traffic Diversion Type	Diverted Traffic Volume ($\mathcal{T}_D$)
Inactive	N/A	N/A	N/A
Composite Defense Action	Timid	Route Slicing	0
			$r_1 \times \mathcal{T}_M$
			$r_2 \times \mathcal{T}_M$
	Aggressive	Route Slicing	0
			$r_1 \times \mathcal{T}_M$
			$r_2 \times \mathcal{T}_M$
	Traditional	Route Slicing	0
			$r_1 \times \mathcal{T}_M$
			$r_2 \times \mathcal{T}_M$
	Sampled	Route Slicing	0
			$r_1 \times \mathcal{T}_M$
			$r_2 \times \mathcal{T}_M$
	N/A	Route Slicing	$\mathcal{T}_M$
	N/A	Route Mutation	$\mathcal{T}_C$

$\mathcal{T}_D$ and traffic limiting volume $\mathcal{T}_L$ that is equal to $max(\mathcal{T}_M - \mathcal{T}_D, 0)$. $\mathcal{T}_D$ depends on r_i that defines a ratio depending on the currently available spare bandwidth S_B, and $r_i < r_{i+1}$ (e.g., $r_1 = 33\%$ and $r_2 = 66\%$). Understandably, a particular defense strategy becomes more vigorous in dropping traffic with the increase of $\mathcal{T}_L$. Notably, at the second last row, there is no limiting function as the volume of diverted traffic volume is equal to the excessive traffic. In the last row, no limiting is required as all clients' traffic are rerouted through alternative links. Though repeated traffic rerouting can find *probable* target customer [124], *Horde* has to perform route mutation for all customers through different links to avoid congestion. Fig. 3.7 illustrates a composite defense action, where the *Green* and *Red* portions of plot "Traffic Volume across Risk Scores" represent $\mathcal{T}_D$ and $\mathcal{T}_L$ respectively, and the rest of the traffic (grey bars) use their regular routes through L_1.

At each time, for each composite action/strategy $a_d \in A$, the agent dynamically determines "one" context-aware limit function (timid function in Fig. 3.7) based on its limit function type, current traffic distribution across risk scores, and associated $\mathcal{T}_L$. The chosen function drops $\mathcal{T}_L$ through minimizing *expected* false positive f_p. The agent determines the expected false positive rate f_p and false negative rate f_n of limit function Y_d considered for a_d, using following equations:

$$\begin{aligned} f_p &= \frac{\sum_{x \in X}(1-x) \times Y_d(x) \times \mathcal{T}_x}{\sum_{k \in X} Y_d(k) \times \mathcal{T}_k} \\ f_n &= \frac{\sum_{x \in X} x \times (1 - Y_d(x)) \times \mathcal{T}_x}{\sum_{k \in X}(1 - Y_d(k)) \times \mathcal{T}_k} \end{aligned} \tag{3.1}$$

where, $\mathcal{T}_x$ is traffic volume with risk score x, $Y_d(x)$ is the ratio of $\mathcal{T}_x$ that Y_d will drop, and $\sum_k Y_d(k) \times \mathcal{T}_k \approx \mathcal{T}_L$ that is the amount of traffic a_d wants to drop.

3.7.2 Attack Action Space

Table 3.2 illustrates the attack action space required to formulate the attack model at Section 3.2. β represents the type of botset adjustment, and χ specifies qualitative

Table 3.2: Attack Action Space (V)

Action Type	Adjusting Botnet β	Botnet Adjusting Rate (χ)	Adjusting Traffic Rate (τ)	Attack Tactic Characteristics
Inactive	_	_	_	Silent
Reconnaissance	_	_	_	Discover Target Link
Launch Attack	Use Existing Botnet	_	Same	Same Attack
		_	Decrease	Y+1
		_	Increase	E+1,Y+(-1)
	Extend Botnet	Low	Same	E+1
			Decrease	Y+1
			Increase	E+2,Y+(-1)
		Medium	Same	E+2
			Decrease	E+1,Y+1
			Increase	E+3,Y+(-1)
		High	Same	E+3
			Decrease	E+2,Y+1
			Increase	E+4,Y+(-1)
	Change Bot Distribution	Low	Same	Y+1
			Decrease	E+(-1),Y+2
			Increase	E+1
		Medium	Same	Y+2
			Decrease	E+(-1),Y+3
			Increase	E+1,Y+1
		High	Same	Y+3
			Decrease	E+(-1),Y+2
			Increase	E+1,Y+2

levels (i.e., low, medium, and high) defining percentages (e.g., low means less or equal to 33%) of adjustment. χ specifies the botnet extension rate (percentage of new bots) when β is *Extend Botnet*, or botnet change rate (percentage of replaced bots) when β is *Change Bot Distribution*. The traffic adjustment is represented by τ that has three values: Same, Increase, and Decrease. Notably, though the attacker adjusts each bot's traffic rate separately, the agent only concerns about the mean traffic rate that can insinuate attack action's aggregated stealthiness or aggressiveness. Moreover, each attack action has a level of stealthiness $Y + n_i$ and expansion $E + n_j$ (column 5), where positive n_i or n_j defines the increase of stealthiness and expansion respectively.

Table 3.3: Observations Space (Ω_S)

Link Utilization		Packet Drops	
Utilization Value (u)	Symbol	Drop Ratio (l)	Symbol
$u \leq 0.33B$	u_1	$l \leq \gamma_1$	l_1
$0.33B < u \leq 0.67B$	u_2	$\gamma_1 < l \leq \gamma_2$	l_2
$0.67B < u \leq B$	u_3	$\gamma_2 < l \leq \gamma_3$	l_3

3.7.3 State Space of Agent's Link

State qualitatively represents the link condition considering attack intensity and traffic congestion. The following states give qualitative but comprehensive view over link condition.

- ***Normal (N):*** The link contains zero or negligible attack flows during this state.
- ***Abnormally Highly Utilized (H):*** At this state, attack flow intensity is not at its peak due to partial success of attackers (or defenders), attack initiation, or strategic attack reduction. Though the abnormal link utilization at this state is still tolerable, it forecasts the imminent high attack flow intensity.
- ***Flooded (F):*** The attack severity is at its peak engendering significant packet delays and drops.

3.7.4 Agent's Observations for Understanding Link Condition

The agent observes link symptoms, based on which, the agent tries to understand the current link state. Though deep analysis of flows can infer the current link state certainly, it is infeasible for a real-time system. Instead, inspecting packet drop rate, delays, utilization, and others and correlating these with previous attack history are more pragmatic. Therefore, our agents analyze Packet Drop Rate and Link Utilization as real-time observations, which can provide insight about the network state with comparatively less uncertainties [128, 129]. Table 3.3 shows the observations space Ω_s of our POMDP model, where B is the link bandwidth, and γ_i is defined threshold.

Table 3.4: Observation Table

Observation (Link Utilization)	p_g	p_i	$p_{\hat{g}}$	$p_{\hat{i}}$
Low	0.2	0.2	0.6	0.7
Medium	0.4	0.6	0.4	0.3
High	0.6	0.5	0.2	0.1

Composing Observation Matrix for Agent's POMDP Model: The environment is only partially observable with imperfect observations; hence, the agent cannot certainly know about the current state only based on observations. To clarify, higher packet drop rate generally occurs due to attack traffic (high correlation), but it might also happen due to abnormal amounts of benign traffic, data link layer failures, and others [129] (low correlation). Therefore, the agent composes observation matrix M_O (for POMDP) to define correlations among states and observations, which is required to address uncertainties in associating current observation to the underlying state. M_O contains $p(o|s)$ for all possible state $s \in S$ and observation $o \in \Omega_s$, where, $p(o|s)$ specifies the likelihood of observing $o \in \Omega_s$ at a state $s \in S$. It determines $P(o|s)$ assuming same probability $p(o)$ for all observations, using following equation:

$$p(o|s) = \frac{p(s|o)p(o)}{\sum_{i \in \Omega_S} p(s|i)p(i)} = \frac{p(s|o)}{\sum_{i \in \Omega_S} p(s|i)} \tag{3.2}$$

where, $p(s|o)$ defines the probability that the current state is s when the observation is o.

Determining State-Observation Correlations: This section describes how the network administrator updates $p(s|o)$ based on recent observations and experience. The conditional probabilities $p(s|o)$ of a state $s \in S$ after observing $o \in \Omega_s$ depend on both benign and attack behaviors and may not be static. To learn these, since a particular observation o at time t to time $t + x$, the network administrator monitors two properties: potential botset size and their flow growth rate, to find the aggregated change of these properties within that time window. Based on these properties for

incidents with observation o, he estimates (updates) four probabilities: (1) Probabilities of increasing flow growth rate, p_g, (2) Probabilities of increasing botset, p_i, (3) Probabilities of decreasing flow growth rate, $p_{\hat{g}}$, and (4) Probabilities of decreasing botset, $p_{\hat{i}}$. Similarly, he calculates these probabilities for all possible observations. Table 3.4 shows such an example, where each row is associated with a specific type of link utilization observations.

For each o, the agent considers the state as Normal (N) if both properties are decreasing, because attack flow intensity reduces from N. It considers it as Abnormally Highly Utilized (W) if one is increasing and other is non-decreasing, because attack flow intensity increases from W. It considers it as Flooded (F) if exactly one is increasing and the other is decreasing, because the attacker is diversifying attack vectors. Notably, when the state is F, the attacker is not trying to change the attack intensity as he already met the expected intensity. There is another case when both remain unchanged, which belongs to either N or F depending on the current link utilization u. The network administrator determines these conditional probabilities using following equation:

$$\begin{aligned} p(N|o) &= p_{\hat{g}}p_{\hat{i}} + (1-u)(1-p_{\hat{g}}-p_g)(1-p_{\hat{i}}-p_i) \\ p(H|o) &= (1-p_{\hat{g}})p_i + (1-p_{\hat{i}})p_g - p_g p_i \\ p(F|o) &= p_{\hat{g}}p_i + p_{\hat{i}}p_g + u(1-p_{\hat{g}}-p_g)(1-p_{\hat{i}}-p_i) \end{aligned} \tag{3.3}$$

The following Sections 3.8-3.9 describe phases of BRITE loop with details.

3.8 Understand & Investigate Phases of Agent

After observing the network symptoms at *Observe* phase, the agent enters into *Understand & Investigate* phase of BRITE loop (Fig. 3.5), and initiates its efforts to learn non-static POMDP parameters. At this phase, it mainly focus on understanding current environment and attack strategy based on observations.

3.8.1 Computing Agent's Belief

Based on current observation $o \in \Omega_S$ and last defense action $a_d \in A$, the agent computes Belief b_t that is the probabilistic distribution among possible states to deduce the current state of the link probabilistically. The agent determines b_t using Eqn. 3.4 that extends the traditional belief calculation to integrate *expected* attack behavior.

$$\begin{aligned} \forall s \in S, b_t(s) = \frac{1}{p(o|b_t, a_d)} \times \\ p(o|s) \sum_{s'' \in S} \sum_{a_v \in V} p_{t-1}(s|s'', a_d, a_v) q(a_v) b_{t-1}(s'') \\ p(o|b_t, a_d) = \sum_{s} p(o|s) \sum_{s''} \sum_{a_v} p_{t-1}(s|s'', a_d, a_v) q(a_v) b_{t-1}(s'') \end{aligned} \tag{3.4}$$

where, $b_t(s)$ is the probability of state as current state, $p(o|s)$ comes from Observation matrix M_O, $q(a_v)$ is the likelihood of a_v as previous attack action, and $p_{t-1}(s|s'', a_d, a_v)$ is the system dynamics at previous time.

3.8.2 Quantifying Defense Effectiveness

Defense effectiveness of a defense action $a_d \in A$ at current state $s \in S$ specifies a vector consisting of transitional probabilities $p_t(s'|s, a_d)$ for all possible next states $s' \in S$. Here, $p(s'|s, a_d)$ defines the probability of transiting to next state $s' \in S$ from current state $s \in S$ for a_d. Understanding defense effectiveness is essential to optimize decision-making for the current situation. As *Horde* formulates the defense agent's decision-making as POMDP problem, T of defense agent's POMDP is composed by computing defense effectiveness for all $a_d \in A$ and $s \in S$. However, in reality, such state transitions also depend on the attack actions that can only be known probabilistically. Therefore, the agent calculates *expected* defense effectiveness considering

expected attack behavior, using following equation:

$$p_t(s'|s, a_d) = \sum_{a_v \in V} p_t(s'|s, a_d, a_v) \times p_t(a_v|s, a_d) \tag{3.5}$$

where, $p_t(s'|s, a_d, a_v)$ represents system dynamics, and $p_t(a_v|s, a_d)$ defines the likelihood of a_v as attack reaction to a_d at s.

Eqn. 3.5 iterates over over all attack actions $a_v \in V$ to integrate expected attack behavior into agent's defense effectiveness [130]. Besides being stochastic, defense effectiveness changes with time due to non-stationary system dynamics and adaptive attack behavior. The agent predicts the next attack behavior at *Understand* phase (discussed in next section), and refines system dynamics at *Evolve* phase (discussed in section 4.2.1).

3.9 Actuate Phase of Agent

During *Actuate* phase, the agent completes sense-making by computing *expected* rewards followed by solving the POMDP model to generate optimal policy.

3.9.1 Quantifying Agent's Reward

The agent computes reward for all possible defense and attack scenarios to complete the parameter R of *Horde*'s POMDP models. Reward quantifies the payoff of a defense action/strategy against a specific attack action based on changes (improvement or degradation) of link condition. For all possible scenarios considering all current state $s \in S$, next state $s' \in S$, defense action $a_d \in A$, and attack action $a_v \in V$, the agent formulates reward using the following equation:

$$R(s', s, a_d, a_v) = (\Psi_s - \Psi_{s'}) - f_p(1 + \delta_p)\mathcal{T}_L I_l - \mathcal{T}_D I_d + C_v + \epsilon_d$$

where, Ψ_s is the average amount of benign traffic dropped previously at state s, f_p is the *expected* false positive rate, δ_p^v is the IDS error rate regarding f_p.

In the equation, Ψ_s is determined based on *historical* data or experience. The second and third terms compute *expected* loss due to benign traffic drop and delay, where I_l and I_d are cost of dropping and delaying per GB benign traffic respectively. Hence, reward depends on both historical or previous experience and expectation over benign traffic drop. Besides, it includes C_v representing the attack cost to incentivize those defense actions which increase attack costs.

In the equation, ϵ_d is the incentive for executing less-explored defense actions. Exploration is a critical criterion for a RL agent to avoid the local optima. However, over exploration slows down the policy convergence and induces regret (i.e., loss due to deviating from optimal solution). Hence, ϵ_d must be able to address the exploration-exploitation dilemma of the agent's RL approach. *Horde* applies UCB1-NORMAl policy for exploration [131], that achieves the logarithmic regret uniformly over time. ϵ_d of defense action a_d is determined using the following equation:

$$\epsilon_d = \sqrt{\frac{(w_d - n_d\overline{r_d}^2) \times ln(n-1)}{n_d \times (n_d - 1)}}$$

where, w_d is the aggregated squared rewards for executing a_d, n_d is the number of time a_d executed, $\overline{r_d}$ is the average reward, and n is the number of time passed.

The incentive for a defense action a_d increases due to deviations in rewards and less execution ($n >> n_d$) of a_d. Importantly, due to the exploration incentive, if any two actions have the same expected payoffs and effectiveness, the agent chooses the one that is less explored.

Finally, the agent determines the reward of a_d by integrating the expected attack behavior in Eqn. 3.6, where, $p(a_v|s, a_d)$ is the probability of next attack action a_v for current state s in reaction to a_d.

$$R(s', s, a_d) = \sum_{a_v \in V} R(s', s, a_d, a_v) \times p(a_v|s, a_d) \tag{3.6}$$

3.9.2 Dynamic Defense Planning Generation

After calculating rewards, the agent generates a new POMDP model based on all system primitives (S, A, Ω_s, and O) and learnt non-static parameters (T, R, and γ_f).

POMDP Solution Methods: The agent solves the POMDP model using Heuristic Search Value Iteration (HSVI) [32], to approximate the policy with a bounded regret rate J (user-defined) that defines the precision of our approximation algorithm. The agent cannot apply exact solution approach such as Policy iteration [132], Value Iteration [133] as the agent is Boundedly Rational [134] due to required real time optimization. Given an initial belief, HSVI prunes irrelevant (unreachable) belief space to ensure fast and real-time optimization [135]. It computes the optimal policy π^* to recommend optimal action a_d^* for current belief b_t ($\pi^*(b_t) \rightarrow a_d^*$), using the following approach:

$$V^{\pi}(b_t) = E\left[\sum_{t=0}^{\infty} \gamma_f^t R(s', s, a_d^t) | b_t, \pi\right]$$

$$\pi^* = \arg\max_{\pi} V^{\pi}(b_t)$$

where, $V^{\pi}(b_t)$ is the accumulated rewards that is maximized by applying policy π, considering the present and possible future payoffs $R(s', s, a_d^t)$ (of Eqn. 3.6) with discount factor γ_f.

3.10 Summary

This chapter presents models and framework named Horde to automate real-time and dynamic defense composition to protect critical network links against I-DDoS effectively. According to experiments in evaluation, *Horde* ensures the transmission of more than 97% of benign users in most of cases. *Horde* applies a hybrid RL learning approach that learns the environment by integrating the system dynamics and expected attack behavior into the environment. It will be interesting to analyze how

more fine-grained defense composition may aid to reduce benign traffic drop, while also ensuring that such extension will not deteriorate *Horde*'s scalability significantly concerning computational complexity. Moreover, there is scope to improve the sampled traffic limiting by improving the clustering and sampling approach. *Horde* reacts not only to the change of current link condition but also to the adapted attack behavior. Without requiring domain-knowledge and explicit attack strategies, *Horde*'s agent converges to the optimal defense composition within few time-sequences based on its interactive experience. This shows *Horde*'s applicability for real-world scenarios, where gathering deep domain-specific knowledge is generally infeasible without going to operations.

This chapter focuses on real-time defense optimization *autonomously* that requires some understanding and reasoning of the behavior of the environment and attackers. However, this chapter does not address how these agents evolve to cope with dynamic environment and attack behaviors. *Horde* needs to implement *Evolve* capabilities of BRITE loop to deploy autonomous agents in a environment whose behavior changes with time due to critical network events. Moreover, an agent's decision-model expects the expected attack behavior to be integrated, but such expected attack behavior changes with the attacker's strategical adaptations/evolvement. However, it is infeasible to have data or knowledge on all possible attack approaches, that necessitates the attack behavior learning on-the-fly. Therefore, the next chapter focuses on developing models to learn attack strategy on-the-fly and *Evolve* to cope with environmental changes, based on interactive experience. Moreover, the next chapter discusses the evaluation of *Horde* against diversified attack strategies in dynamic environment.

CHAPTER 4: Evolution of Defense Planning for Dynamic Defense Optimization Against Strategical I-DDoS Attacks

Understanding the network behavior is essential to optimize the I-DDoS (Infrastructural Denial of Service) defense planning. Most enterprise I-DDoS solutions rely on human inputs/configurations which are not only static but also ad-hoc. In contrast, the network is so dynamic with many correlated factors, that may make these inputs irrelevant. For example, newly appeared bug in a updated firmware induces huge traffic queuing delay for previously effective traffic rerouting. Defining such fine-grained network behavior manually or formulating the network considering its correlated and uncertain factors is very hard, if not impossible. This chapter presents methods and models, deployable by *Horde* (discussed in previous chapter 3), to learn the currently adopted attack strategy and consolidate the capability of *evolving* to cope with changes of the environment. These capabilities need to be integrated into *Horde* to make it applicable in a dynamic environment against strategical and adaptive attackers.

The easy affordability of bots with wide global distribution makes nowadays DDoS attackers more sophisticated and diversified. For example, indirect attacks like Crossfire defeat traffic load balancing or rerouting by congesting new critical links with different set of bots [2]. Hence, deploying effective defense is unachievable without understanding the attack reaction. Besides, failing to cope with the changes of critical environmental factors make these enterprise solutions non-optimal; as a result, these organizations easily fall victim to I-DDoS attacks despite adopting strong mitigation approaches [136]. For instance, enterprises either allow too much attack traffic or drop too many benign traffic due to being non-adaptive to IDS performance which changes

with stealthy attack behavior or abnormal benign traffic behavior. This chapter develops an attack prediction model to learn the currently adopted attack strategy in an *online* approach based on recent attack observations. It is necessary to predict the imminent attack behavior required for employing proactive, as well as, optimal reactive defense approach. With its incremental and interactive learning approach, the prediction model learns new attack strategy and their adaptations without being trained explicitly with deep domain knowledge and data.

To integrate the *evolve* capability of BRITE loop into *Horde*, this chapter presents models to tune critical system parameters: (1) system dynamics, (2) discount factor, and (3) IDS error rate incrementally, based on recent experience. System dynamics regulate the behavior of the link condition with respect to the enacted defense and attack approaches, understanding which, is indispensable for defense optimization. Discount factor (ranging between $[0, 1]$) defines the future-horizon until which current defense action will have impact. Without determining optimal discount factor, the decision-making converges to a non-optimal solution due to greedy decision-making or redundant overestimation. *Horde* determines IDS error rate as the *expected* deviation of IDS risk scores from real scores (i.e., 0 for benign users, and 1 for attack bots) to incorporate the likelihood of IDS inaccuracies into decision-optimization process. This chapters presents approaches that learn or refine these parameters in an *online* approach to evolve decision-optimization according to the dynamics of system.

The evaluation experiments show that *Horde* agents can cope with dynamic attack and network behavior effectively by integrating models for attack prediction and defense planning evolution. The experiments on diversified and adaptive attack strategies reveal that the attack prediction model can detect sophisticated attack strategies with high accuracy despite being unseen. It also shows that it can distinguish between attacker's strategical adaptations and attack deceptions.

In summary, this chapter offers following contributions:

- An online approach to train an attack prediction model that can learn any attack strategy without being explicitly trained. Alongside detecting attacker's strategical adaptations, this model is robust against attack deceptions.

- An autonomous approach to integrate the *evolve* capability into *Horde* agent's decision-making that can cope with the change of critical link behavior, and IDS inaccuracies. It also presents an incremental approach to train a Deep Neural Network model to predict the optimal discount factor.

The next Section 4.1 describes how *Horde* learns attack strategy on-the-fly, and the following Section 4.2 discusses how the agent evolves according to changes of the environment.

4.1 Characterization of Attack Behavior

An agent must reason over next attack behaviors to determine context-aware optimal defense action. For instance, deploying traffic filtering against stealthy attackers is expensive due to the risk of higher benign traffic drop, whereas, traffic rerouting against easily distinguishable attack flows seems unnecessary. Hence, a *Horde* agent determines state transition matrix of its POMDP through integrating the expected attack behavior. The *expected* next attack behavior is a probabilistic distribution across attack action space V, that specifies the probability of each action as next attack action. Determining it is essential to perform strategic reasoning required to deploy optimal, as well as proactive defense. However, predicting next attack behavior is challenging due to lack of domain knowledge on diversified attack approaches and their adaptations. A dynamic attacker cannot be expected to choose same attack action always at a particular network condition, because he adapts his attack strategy with the observations of previous attack consequences. For example, an attacker who previously became stealthier after flooding the network finds out that aggressive attack behavior at flooded condition achieves his objective faster. Thus,

the attacker can adapt his strategy according to his observations from the network. Hence, the agents need to deploy an *incremental* approach to learn attack strategy and adaptations continuously based on recent attack symptoms.

Another challenge related to attack prediction arises due to random attack execution instead of following current strategy to deceive the prediction model [123]. Therefore, the attack prediction model must not only learn new attack strategy without explicit pre-training but also be robust against attack deceptions. Additionally, due to imperfect observations, an agent cannot identify the last attack action certainly, that makes certain attack prediction infeasible. Moreover, attack observations are also partial with imperfect and incomplete information. Hence, attack and defense observations may not be identical, which may also be responsible to confront different attack approach than anticipated. Therefore, the agent aims to determine the *expected* next attack behavior through addressing all these uncertainties.

Let assume that $p(s'|s, a_d)$ is the probability of transiting from current state s to next state s' for defense action a_d. The *Horde* agent creates the state transition matrix, T, of its POMDP model with all $p(s'|s, a_d)$ considering all possible values of $s \in S$, $s' \in S$, and $a_d \in A$. It formulates $p(s'|s, a_d)$ using the following equation:

$$p(s'|s, a_d) = \sum_{a_v \in V} p(s'|s, a_d, a_v) \times p(a_v|s, a_d) \tag{4.1}$$

where, $p(s'|s, a_d, a_v)$ is the probability of transiting from current state s to next state s' for combination of attack a_v and defense a_d action. In Eqn. 4.1, $p(a_v|s, a_d)$ is the probability of confronting a_v as next attack reaction against the employed defense action a_d at a specific state s.

In this dissertation, *expected attack behavior* is a matrix defining $p(a_v|s, a_d)$ for all attack a_v at possible current state s against defense a_d. This section describes the *Horde*'s approach to determine $p(a_v|s, a_d)$ for all possible state $s \in S$ and defense

actions $a_d \in A$, where $p(a_v|s, a_d)$ is the probability of confronting a_v as next attack reaction against the employed defense action a_d at a specific state s.

Overview of the approach: From an defense agent's perspective, the rational attacker determines his next action considering the current context that mainly depends on (1) *what is the current network link condition?*, (2) *what is the consequence of last attack actions?*, and (3) *what is the currently deployed defense action?* Any rational attacker reasons over the first question, because the effectiveness of an attack action largely depends on the current link condition. The attacker tries to reason over the second question, because it mainly aids him to decide whether he needs to change the current attack strategy or not. For example, if the attacker finds out that the defender cannot detect attack bots despite having high traffic rate, then he is more likely to continue with that cheap approach. Concerning the third question, it is rational to assume that the attacker tries to infer the next defense action, for example, he decreases traffic rate in reaction to emphasized traffic filtering.

Both the first and third question are already integrated in $p(a_v|s, a_d)$; hence, the agent needs to expand $p(a_v|s, a_d)$ to make it more fine-grained to integrate the second question. The consequence of last attack action defines the improvement or degradation of the link condition due to the last attack and defense interaction. To understand it, the agent needs to know the last state s''. Hence, *Horde* expands $p_t(a_v|s, a_d)$ to incorporate state transition from previous state $s'' \in S$ to current state $s \in S$, using the following equation:

$$\begin{aligned} p_t(a_v|s, a_d) &= \sum_{s''} p_t(a_v|s, s'', a_d) \times p_{t-1}(s''|s, a_d) \\ &= \sum_{s''} p_t(a_v|s, s'', a_d) \times b_{t-1}(s'') \end{aligned} \tag{4.2}$$

where, $b_{t-1}(s'')$ is the previous belief about last state s''.

To integrate the last attack and defense reaction, *Horde* expands Eqn. 4.2 into

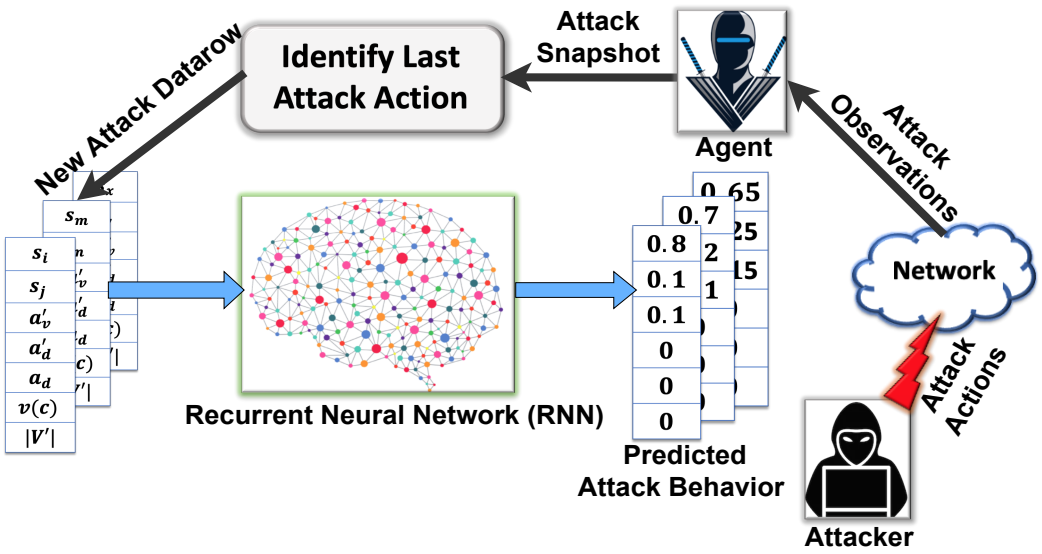


Figure 4.1: Learning Attack Strategy. An agent creates new datarows based on attack observations from the network, that is used to updated the RNN model.

Eqn. 4.3 to include previous attack action a'_v and previous defense action a'_d.

$$\begin{aligned} p_t(a_v|s, a_d) &= \sum_{a'_v} \sum_{s''} p_t(a_v|s, s'', a_d, a'_d, a'_v) \\ &\quad \times p_t(a'_v|s, s'', a'_d) \times b_{t-1}(s'') \\ &= \sum_{a'_v} \sum_{s''} p_t(a_v|s, s'', a_d, a'_d, a'_v) \times q(a'_v) \times b_{t-1}(s'') \end{aligned} \tag{4.3}$$

where, $p_t(a_v|s, s'', a_d, a'_d, a'_v)$ specifies the probability of a_v for the context $(s, s'', a_d, a'_d, a'_v)$, and $q(a'_v)$ represents the likelihood that the last attack action was a'_v.

In Eqn. 4.3, the given condition $(s, s'', a_d, a'_v, a'_d)$ is the context, where (s, s'', a'_d, a'_v) specifies last attack consequence, s specifies current link condition, and a_d is the currently considered defense plan. Therefore, the agent reasons over *what action the attacker is likely to do at a specific context*? to predict $p_t(a_v|s, a_d)$, which is regulated by the currently adopted *Attack Strategy*. Hence, to characterize the attack behavior (as shown in Eqn. 4.3), the agent has to do two things: (1) identify the last attack action for $q(a'_v)$, and (2) learn the attack strategy to predict next attack action.

Fig. 4.1 illustrates the *online* approach of characterizing the attack behavior, using

the approach of agent (d_1) as an example. At each time-sequence, the attacker executes attack actions on the network, and d_1 observes it from the network. From these observations, d_1 creates an attack snapshot, based on which, d_1 identifies the last attack action. Based on identified last attack action, the agent creates new datarows, where $|V'|$ is the likelihood distribution for the last attack action. Similarly, all other agents observe attack actions at their links and create datarows. Based on all these datarows, the RNN model is updated *incrementally*. While determining the effectiveness of a defense action, all agents ask the model to predict the expected attack behavior on their links for given context, which is shown by *Predicted Attack Behavior* in Fig. 4.1.

Section 4.1.1 describes how the agent identify previous attack action *probabilistically*, and section 4.1.2 describes how the agent learns attack strategy.

4.1.1 Identifying Previous Attack Action

Characterizing the last attack action is essential to retrain the attack prediction model and refines the system dynamics. Notably, this action has generated the currently existing attack traffic. To identify it, the agent creates a potential *Attack Snapshot* comprising the number of newly appeared and disappeared bots with mean traffic rate and risk scores, based on *given* benign traffic statistics. As shown in Fig. 4.1, the agent creates a new attack snapshot based on attack observations from the network, that is used to identify the last attack action. We assume that the network administrator knows the benign traffic behavior and can create benign traffic profiles and statistics containing information such as distributions of ratios of new benign sources appearance and disappearance with mean values and variance. These statistics can be time-sensitive which means that the administrator may follow different distributions at different time. For example, during a product release, a website may be hit by more users at a time with longer session interval.

This attack snapshot is created assuming that anything unusual is mostly due to the

attack action. However, the agent considers the uncertainties related to the abnormal benign traffic behavior. Based on this snapshot, the agent characterizes the attack action using Algorithm 5 which determines four types of parameters: (1) probability of different action types (i.e., Reconnaissance, Launch Attack, Inactive), (2) botset adjustment type β, (3) botset adjustment level χ, (3) traffic rate adjustment τ, and (4) likelihood of not identifying appropriate β. According to these parameters, the agent composes a likelihood matrix named "Characterized $|V'|$" quantifying the likelihood $q(a'_v)$ of each $a'_v \in V$ as last attack action as the output of this step.

The rest of the section describes the details of Algorithm 5 that mainly performs following actions:

• **Determining Action Type:** In line 1-4 of Algorithm 5, the agent determines: (1) $V'[I_v]$ that defines the likelihood of Inactive (I_v) attack action, (2) $V'[R]$ that defines the likelihood of Reconnaissance, and (3) l_a that defines the probability of launching attack. Here, r_s defines the increased amount of scanning traffic compared to mean scanning traffic, G is the distribution of benign traffic volume, and $\mathcal{T}$ is the current amount of benign traffic.

• **Determining Botset Adjustment Type (β) and Level (χ):** In line 5-23, Algorithm 5 determines two attack parameters: β and χ. At line 5-8, it determines the rate of bot appearance β_a and disappearance β_d assuming that the appearance/disappearance of extra sources is generally due to bots, where n'_b is number of bots at immediate past. However, it considers uncertainties related to appearing or disappearing more benign sources (discussed later).

It assigns (1) $\beta = 0$ (same botset) if negligible bot appearance, (2) $\beta = 2$ (changed bot distribution) if bot appearance is same as disappearance, and (3) $\beta = 1$ (extended botset) if bot appearance is non-negligibly higher than disappearance. For $\beta = 1$ or $\beta = 2$, it also assign the value for χ based on adjustment level (line 14-17 and 20-23),

Algorithm 5: Determine Previous Attack Action

Input : Changed Ratio of Scanning Traffic r_s.
Output: Characterized Attack Distribution V'.

$V'[I_v] = 1 - P(G \geq \mathcal{T})$ // Probability of Inactive.
$V'[R] = r_s$ // Probability of Reconnaissance.
$l_a = 1 - V[I_v] - V[R]$ // Probability of Launch Attack.
Normalize($V[I_v]$,$V[R]$,l_a)
number of newly appeared bots, $b_a = a_s - \overline{a_s}$
bot appearance rate, $\beta_a = \frac{b_a}{n'_b}$
number of disappeared bots, $b_d = d_s - \overline{d_s}$
bot disappearance rate, $\beta_d = \frac{b_d}{n'_b}$
if $\beta_a \approx \beta_d$ **then**
    if $\beta_a == 0$ **then**
        // Case: Use Existing Bots
        $\beta = 0$, $\chi = None$
    else
        // Case: Changed bot distribution
        $\beta = 2$
        for *i in range(3)* **do**
            if $\beta_a \leq (i+1)0.33$ **then**
                $\chi = i$
                break
if $\beta_a > \beta_d$ **then**
    // Case: Extended Botset
    $\beta = 1$
    for *i in range(3)* **do**
        if $(\beta_a - \beta_d) \leq (i+1)0.33$ **then**
            $\chi = i$
            break
// Assigning Traffic Rate
previous traffic rate $m'_t = m_t$
current traffic rate $m_t = \frac{\mathcal{T}}{n_s}$
if $m'_t \approx m_t$ **then**
    $\tau = 0$
if $m'_t > m_t$ **then**
    $\tau = 1$
else
    $\tau = 2$
V' = address_uncertainties(V',l_a,β,χ,τ)
return V'

whereas, χ is None for $\beta = 0$ due to no adjustment. Notably, this algorithm does not consider the case with higher bot disappearance rate, due to considering it with $V'[I_v]$.

• **Adjusting Traffic Rate (τ):** In line 24-31, the agent determines whether the mean traffic rate remain unchanged ($\tau = 0$), decreased ($\tau = 1$), or increased ($\tau = 2$), where $\mathcal{T}$ and n_s are the current traffic volume and number of sources respectively.

Algorithm 6: Addressing Attack Characterization Uncertainties

Input : Attack Vector V', Launch Attack Probability l_a, Botset Adjustment β, Bot Adjustment Level χ, Traffic Rate Adjustment τ
Output: Attack Vector V'
`// Consider uncertainities in attack characterization`
1 **if** $\beta == 0$ **then**
2 $\quad V'[(\beta, \chi, \tau)] = l_a$
3 $\quad V'[(\beta + 1, \chi, \tau)] = l_a \times P(X < \overline{a_s}) \times \mu_r(a_s)$
4 $\quad V'[(\beta + 2, \chi, \tau)] = l_a \times P(X < \overline{a_s}) \times P(Y < \overline{d_s})$
5 **if** $\beta == 1$ **then**
6 $\quad V'[(\beta - 1, \chi, \tau)] = l_a \times P(X > \overline{a_s}) \times (1 - \mu_r(a_s))$
7 $\quad V'[(\beta, \chi, \tau)] = l_a$
8 $\quad V'[(\beta + 1, \chi, \tau)] = l_a \times P(Y < \overline{d_s}) \times (1 - \mu_r(a_s))$
9 **if** $\beta == 2$ **then**
10 $\quad V'[(\beta, \chi, \tau)] = l_a \times P(X > \overline{a_s}) \times P(Y > \overline{d_s})$
11 $\quad V'[(\beta + 1, \chi, \tau)] = l_a \times P(X < \overline{a_s}) \times \mu_r(a_s)$
12 $\quad V'[(\beta + 2, \chi, \tau)] = l_a$
13 Normalize(V')
14 return V'

• **Addressing Uncertainties:** At line 32, Algorithm 5 calls Algorithm 6 to assign uncertainties in attack characterization induced due to abnormal appearance/disappearance of benign users. In Algorithm 6, $P(X < \overline{a_s})$ and $P(Y < \overline{d_s})$ respectively define probabilities of appearing and disappearing less benign sources than normal, and μ_r defines the mean risk score of newly appeared sources.

Algorithm 6 considers three mutually exclusive cases based on botset adjustment type β (determined in Algorithm 5). In all these cases, we consider the likelihood of

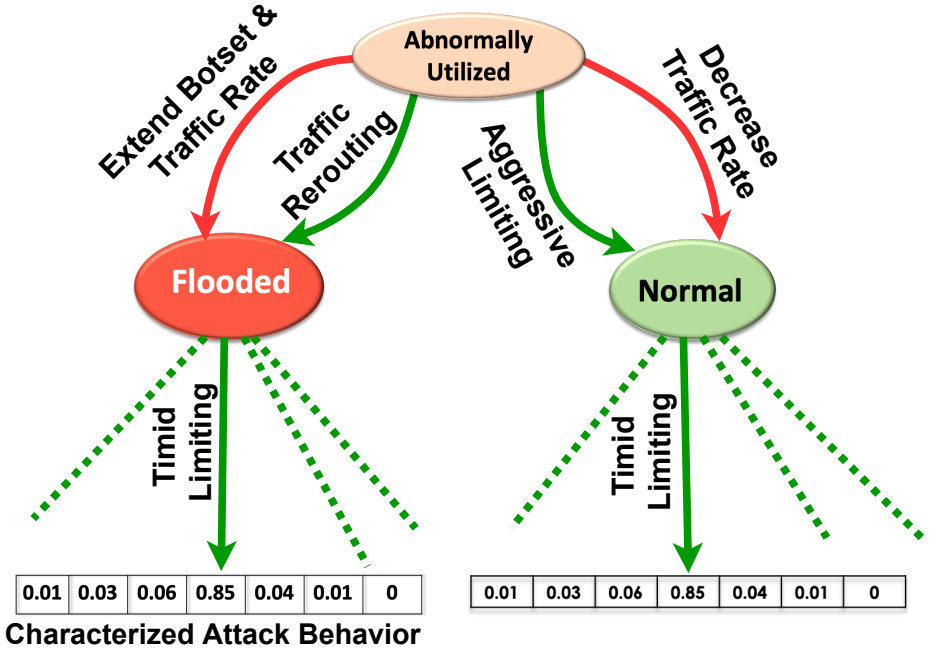


Figure 4.2: An Example showing reasoning over previous attack experience/observations. Characterized attack behavior is the probabilistic distribution across attack actions computed as last attack action. Each leaf represents a specific case, and red and green edges define attack and defense actions respectively.

not identifying (1) unchanged botset ($\beta = 0$) due to appearing and disappearing more benign sources, (2) botset extension ($\beta = 1$) due to appearing less benign sources, and (3) bots' changed distribution due to appearing and disappearing less benign sources.

4.1.2 Learning Attack Strategy

Though it is impossible to know the attack strategy certainly, previously observed attack sequences provide insights to learn attack strategy probabilistically. Fig. 4.2 illustrates an example of previously observed attack trends. In the figure, the link transits from *Abnormally Utilized* state to *Flooded* state when the defense action is *Traffic Rerouting* and attack action is *Extend Botset & Traffic Rate* for some past cases. When the defender previously executed *Timid Limiting* at such a scenario, the identified attack reaction followed the probabilistic distribution of *Characterized Attack Behavior* in one particular incident. Hence, previous attack observations can

reveal attack trends like the example.

Therefore, *Horde* deploys a *Deep Learning Model (DNN)* as Attack Prediction Model (shown in Fig. 3.2), that leverages previously observed attack traits under particular contexts to *actively* learn attack strategy in an *online* approach. For instance, an attack trait specifies that the attack action a_v followed a'_v with likelihood $p_{t'}(a_v|s_i, s_k, a_d, a'_d, a'_v)$ during a past time t' when the context was $(s_i, s_k, a_d, a'_d, a'_v)$. In the example at Fig. 4.2, s_i is *Abnormally Utilized* state, s_k is *Flooded* state, a'_d is *Traffic Rerouting*, and a'_v is *Extend Botset & Traffic Rate*. Through analyzing previously observed $p_{t'}(a_v|s_i, s_k, a_d, a'_d, a'_v)$ (t' represents past) under particular contexts, it is possible to identify attack traits. Such attack traits can be leveraged to predict $p_t(a_v|s_i, s_k, a_d, a'_d, a'_v)$ for all attack actions a_v and possible contexts, in order to reason over upcoming attack action.

– ***Approach of Learning:*** After identifying the previous attack action, each agent sends the probabilistic distribution $|V'|$ to the *Horde*'s manager as immediate past attack action at its link. Thereupon, *Horde* trains the DNN model in an online manner, using all agents' experience on observing attack actions sequentially throughout previous time sequences $(t-1, t-2, ...0)$. It enables the model to learn any new attack strategy and its adaptations without being explicitly trained. Deep learning has shown good results in strategical reasoning at complicated reinforcement settings such as Go, Dota where data is unavailable or less reliable [35]. Moreover, *Horde* can detect attacker's deceptions to confront adversarial machine learning. Importantly, before deploying the model in the production environment, *Horde* simulates numerous attack scenarios to train the model preemptively for days.

– ***Features and Labels:*** With the arrival of new observations, the agent creates multiple datarows based on identified last attack action as shown in Fig. 4.1. In these data, feature-set consists of context c ($c = (s, s', a_d, a'_d, a'_v)$) and $v(c)$, and the

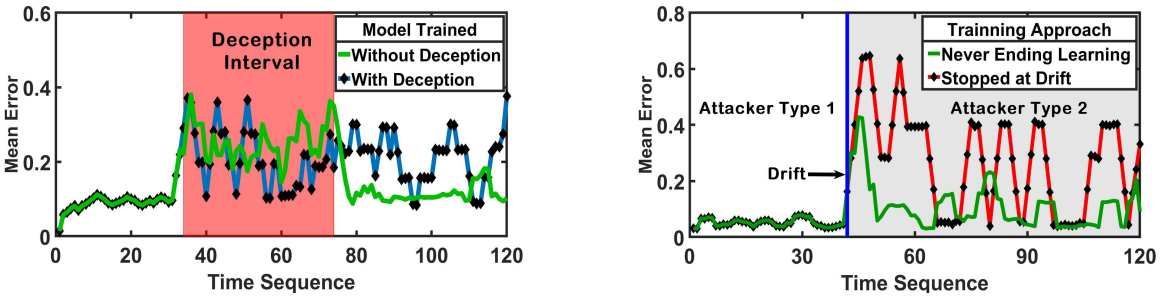
class or label is the characterized $|V'|$ for previous attack action. Here, $v(c)$ specifies the likelihood of c as context of previous event, that the agent computes using the following equation:

$$v(c) = b_{t-1}(s'') \times b_t(s) \times q(a'_v)$$

For each possible c with non-zero $v(c)$, the agent creates a new datarow by associating c with the characterized $|V'|$. A dataset named *Attack Strategy Knowledge Database* KD accumulates all the simulated and real attack scenarios.

– ***Prediction Model:*** Due to temporal dependence among attack observations, the problem of predicting next attack action is similar to time series forecasting that considers sequences in data. Though machine learning with sliding window can transform sequential supervised learning into traditional supervised learning, it cannot consider the temporal dependence outside of that window [36]. In contrast, Recurrent Neural Network (RNN) allows dynamic temporal behavior [137] and fits more with our problem. Moreover, *Horde*'s RNN model uses Long Short Term Memory (LSTM) network [137] that can memorize information over arbitrary time and address long or short term dependencies. *Horde*'s RNN model has two hidden-layers and applies Adam optimization [138] to minimize Mean Square Error (MSE). The model uses Dense Layer [139] to produce the output vector with the size of attack action space V, and leverages Cross-Entropy Softmax function [140] to distribute probabilities among V. At each time, the previously trained model is refined using the newly created data from recent observations. The model predicts the next attack behavior as "Predicted $|V|$" for all possible contexts, that the agent considers as $p_t(a_v|s, s'', a_d, a'_d, a'_v)$ in Eqn. 4.3.

Robustness Against Adversary Deceptions and Adaptations Fig. 4.3 shows the behavior of our prediction model in the presence of adversary deceptions and attacker's characteristic adaptations (e.g., from stealthy to aggressive). The attacker



(a) Attack Deception (b) Attack Type Adaptation

Figure 4.3: (a) Impact of ignoring attack deceptions, and (b) Benefit of continuous learning to cope with attacker's characteristic adaptation. **Note#** Type 1 means aggressive attacker, and Type 2 means stealthy attacker.

may leverage Adversarial Machine Learning [123] and executes random actions to deceive the prediction model. In Fig. 4.3a, the attacker executed deceptions during the interval between 30 to 70, where *Mean Error* at X-axis shows the average difference between real and predicted attack vectors for attack parameters β, χ, and τ. Understandably, error rate spikes during adversary deceptions, but, the model "Without Deception" (represented by green plot) that avoids training during deception intervals shows better performance afterward. However, error rate also spikes due to concept drifts (at time 40 of Fig. 4.3b) because of attacker's characteristic type adaptations. In that case, the model ("Stopped at Drift") which stopped training at drift, performed bad than other that learnt new attack strategy through retraining. This is because, that model is not learning new attack traits and is still biased to previous strategy.

To stop or continue training, *Horde* must understand whether the error is due to adversary deception or attack characteristic (type) adaptation. However, discriminating these two events is hard initially. So, *Horde* consider two models since those error spikes to a finite window; where only one is trained with new observations. After the window, the framework stores the model with better performance as the prediction model and discards the other. Currently, this research keeps the window size larger to ensure distinguishability, which slightly degrades performance due to

avoiding non-deceptive points after the deception interval. Moreover, the framework considers an error as spike if the mean error rate within a specific interval exceeds pre-defined threshold.

4.2 Agent's Evolving to Cope With Dynamic Environment

After *Understand* & *Investigate* of BRITE loop, the agent enters into *Evolve* phase. At this phase, the agent refines system dynamics, determines the optimal discount factor for the current context, and re-evaluates IDS performance.

4.2.1 Learning System Dynamics

System Dynamics regulate the changes of the link condition due to executed attack and defense actions, which depend on non-static and correlated factors such as benign and attack traffic volume, queuing and processing delay, and other links' congestions. Hence, at each time, the defense agent learns or refines these parameters based on previous observations to anticipate defense impact. It is defined by $p(s'|s, a_d, a_v)$ that specifies the probability of transition from current state s to next state s' for attack action a_v and defense action a_d. The *Horde* agent needs to determine it to compose state transition matrix, T, as defined in Eqn. 4.1.

Notably, system dynamics learnt by an agent is different than other agent as they protect different critical links. However, formulating system dynamics are hard for both the attacker and defender due to (1) complex correlations among these factors, and (2) lack of comprehensive knowledge on system dynamics in a new domain or after a domain shift. These challenges exacerbate further due to imperfect and incomplete observability of critical environmental factors. Hence, *Horde* applies a model-free approach using Q-table and gradient descent optimization to refine the knowledge about system dynamics.

Horde applies Q-table (used for Q-learning [141]) to infer the stochastic behaviors without formulating complex correlations. Each row of the Q-table represents a dis-

tinct pair of current state s_i and next state s_j, and each column represents a distinct pair of defense action a_d and attack action a_v. Importantly, Q-value of the quadrant indexed with (s_i, s_j) and (a_d, a_v) contains $p_t(s_j|s_i, a_d, a_v)$ that defines the expected probability of transiting from state s_i to s_j for attack a_v and defense a_d at time t. After observing the consequence of recent defense action a_d and attack action a_v, the agent updates the relevant Q-value using the following equation:

$$Q_t((s_j = s, s_i = s''), (a_d, a_v)) = p_t(s|s'', a_d, a_v) = \\ (1-\alpha)Q_{t-1}((s, s''), (a_d, a_v)) + \alpha b_t(s) b_{t-1}(s'') q(a_v) \quad (4.4)$$

where, s and s'' are current and previous state respectively, $q(a_v)$ is the likelihood of a_v as immediate past attack action that comes from recently characterized $|V'|$ (previous attack distribution), and α is the parameter that weighs the trade-off between past experience and recent observations.

To optimize α, the agent applies Gradient Descent Optimization [142] on observations within a window, where loss depends on deviations between observed and predicted effectiveness (Q_{t-1}). The agent determines the gradient descent $\frac{\partial j}{\partial \alpha}$ using the following equation:

$$\frac{\partial j}{\partial \alpha} = \frac{2}{n} \sum_{i=1}^{n} \sum_{(s, s'', a'_v, t') \in E} (z'_t(s, s'', a'_d, a'_v) - Q_{t'-1}((s, s''), (a'_d, a'_v))) \quad (4.5)$$

where, n is the number of time-sequences, E is the set of all possible events concerning all previous state $s'' \in S$, current state $s \in S$, attack action $a_v \in V$, and defense action a'_d (certainly executed at time t). Here, $z_t(s, s'', a_d, a_v)$ is the observed effectiveness, $Q_{t'-1}((s, s''), (a'_d, a'_v)))$ is the predicted effectiveness, and t' is the previous time within a finite past horizon.

The agent determines $z'_t(s, s'', a_d, a_v)$ using the following equation:

$$z'_t(s, s'', a_d, a_v) = b_{t'}(s) \times b_{t'-1}(s'') \times q(a_v)$$

Finally, the agent determines α using the following equation:

$$\alpha = \alpha_{old} - \theta \frac{\partial j}{\partial \alpha}$$

where, θ is the leaning factor. Notably, the agent optimizes α once for the window interval.

4.2.2 Tuning The Decision-horizon For Optimizing Reward

Discount factor γ_f ($\gamma_f \in [0, 1)$) introduces the future impact of a particular action as delayed reward that decays with the passage of time [31]. It is one of the critical decision parameters of *Horde* agent's POMDP model. Finding appropriate γ_f is essential to optimize policy generation, as a greedy approach without considering possible future leads to a bad condition. However, overthinking also leads to a non-optimal solution due to the dynamic environment and deteriorated computational complexity. Hence, the agent tunes γ_f to regulate the decision-horizon optimally based on current environmental condition. For instance, at a flooded state with an aggressive attacker, there is no option other than taking immediate remediation. Whereas, at a normal state with many tracroute packets, the agent has to think about near future. However, the tuned optimal γ_f^* must not induce too much complexity to ensure defense planning withing bounded time.

Model Properties: To determine optimal discount factor for current context, the agent trains a discount factor prediction model that is updated based on recent observations, once in a *episode* (consisting of 40 time-sequences). One of the critical factor for deciding discount factor is the attack behavior, because the optimal

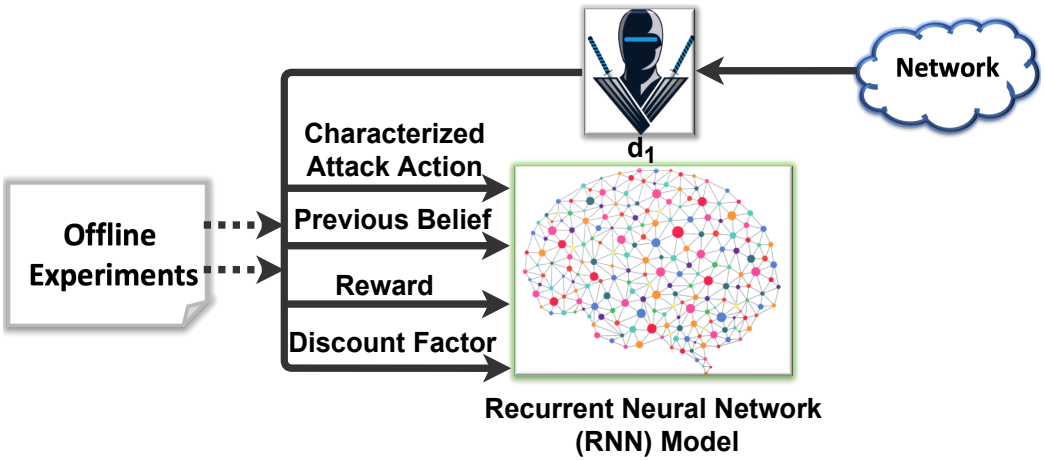


Figure 4.4: Training of the Discount Factor RNN model with new observations and offline experiments.

decision-horizon depends on whether the attacker is aggressive with high attack flows or stealthy with comparatively lower attack flows. Hence, this prediction model uses characterized ($|V'|$) (identified last attack action), and predicted $|V|$ (expected next attack action) as two of the features. Besides, the current link condition is essential to understand optimal discount factor. For example, the agent must weigh immediate payoffs highly when the attacker is aggressive at a flooded state. Hence, this prediction model considers current belief $b_{t'}$ (during past time t') as another feature. The last feature of the model is discount factor γ_f, and the label of the prediction model is the reward. Therefore, the model is trained to predict *probable* reward that may be achieved considering a particular discount factor at current link condition against the current attack behavior.

Training the Model: Let assume, the agent wants to update the prediction model at time t' which is one episode later of time t'' (i.e., last time of prediction model update). Based on all observations from t'' to t, the agent creates datarows, where *Reward* is the mean reward accumulated from t'' to t'. As shown in Fig. 4.4, offline experiments (optional) can also be conducted to enrich the data, where a simulated environment will emulate the real-world traffic distribution. It is observed that

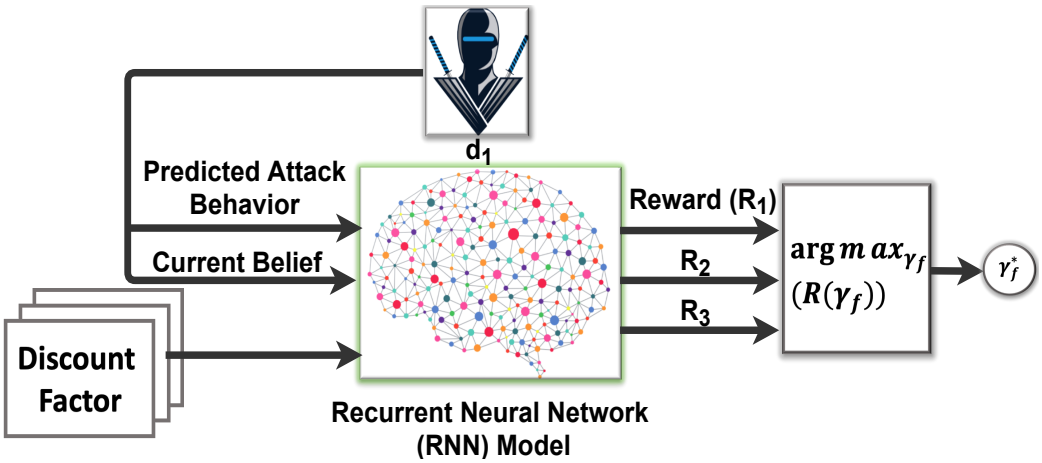
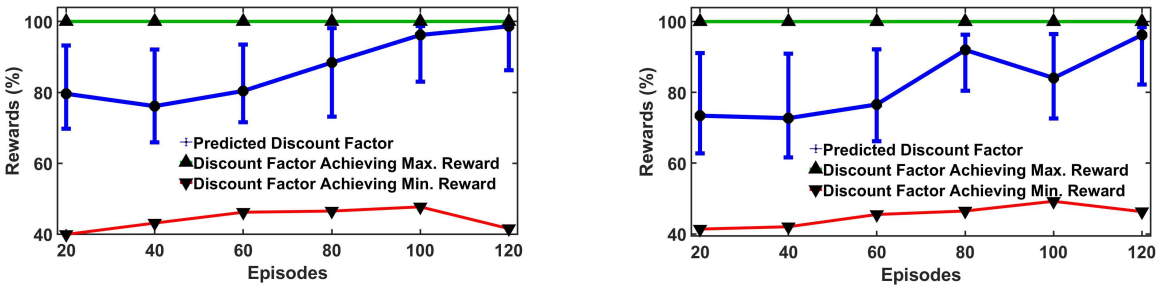


Figure 4.5: Determining the optimal Discount Factor (γ_f^*) for the current context.

such emulated offline experiments improve the performance of the prediction model. As these observations and experimental results are sequential, *Horde* leverages LSTM RNN with Rectified Linear Unit (ReLU) activation function [143] as the prediction model. In order to find the appropriate prediction model, multiple models are developed considering different combinations of traffic properties as features. These experiments reveal that adding more over considered features (i.e., $|V'|, |V|, b_t$) does not improve the performance, instead degrades the balance between overfitting and underfitting. One of the requirement of decision making is that the discount factor should not impose more required time than maximum tolerable time. However, computational time is not considered into feature set, because all previous decision-making was bounded by the maximum tolerable time.

Predicting Optimal Discount Factor: The objective of the prediction model is to predict the *probable* reward for a particular discount factor at the current context. As shown in Fig. 4.5, for various discount factors, the RNN model predicts the probable reward for the current feature set. This approach predict rewards for a finite set of discount factors ranging between 0.2 and 0.95. The discount factor with the maximum reward is chosen as the optimal discount factor for the next episode.

***Experimental Results*:** Fig. 4.6a and Fig. 4.6b compare reward achieved by



(a) Env. 1 (b) Env. 2 (More Dynamic)

Figure 4.6: Reward Achieved by Predicting Discount Factor.

Horde's prediction approach with approaches adopting state discount factor. The progression of maximum reward (green plot) and minimum reward (red plot) are plotted by running *offline* experiments for numerous static discount factors alongside *Horde*'s prediction approach. The plot *Predicted Discount Factor* shows average reward achieved by *Horde*'s approach within a period. For instance, the bar at episode 40 at Fig. 4.6a specifies that the reward achieved within episode 40-60 is approximately 80% of maximum reward with 15% positive deviation and 7% negative deviation. It is observed from these experiments that no single discount factor always achieves maximum or minimum rewards, and optimal discount factor depends on both link behavior and attack behavior (more oscillations in Fig. 4.6b). Apparently, *Horde*'s approach moves toward maximum rewards with more training with observations.

4.2.3 Refining IDS Error Rate:

IDS error rate specifies the deviation of IDS risk scores from ideal scores (i.e., 1 for attack bot and 0 for benign source). The *Horde* agent calculates IDS error rate to quantify the defense payoffs required for composing reward matrix of POMDP model. Due to dynamic network and attack behavior, *Horde*'s manager periodically refines IDS error rate to cope with probable IDS inaccuracies. *Horde* considers two types of IDS error rates: (1) error regarding false positive, δ_p, that defines the deviation

of *expected* false positive (based on IDS risk scores) from actual false positive rate (based on ground truth), and (2) error regarding false negative, δ_n, that defines the deviation of *expected* false negative from actual false negative rate. Though describing approaches to refine IDS error rate is not in the scope, this paper hints out some approaches for obtaining ground truth regarding attack and benign sources.

– Dropping benign traffic incurs negative user feedback, whereas, no feedback is received for dropping attack traffic.

– Sources that do not respect TCP congestion control can be considered malicious [60]. Therefore, sources that do not follow maximum traffic rate despite being restricted by agents' traffic limiting are considered to be malicious.

– Probe flows imitating the behavior in benign traffic profiles can be send to assess δ_p during uncongested condition.

– For direct I-DDoS traffic, destinations may send puzzles (e.g., captcha) assuming bots cannot solve puzzles [144].

– For a sample of traffic, *Horde* can perform time-series or deep behavioral analysis (available commercially) on flow behaviors in *offline*, to label benign and attack flow.

Based on feedback of these approaches, *Horde* can determine the *expected* IDS error rate to address IDS uncertainties in defense consequences quantification. The agent determines the IDS error rate regarding false positive δ_p and IDS error rate regarding false negative rate δ_n, using the following equation:

$$\delta_p = \frac{1}{n}\sqrt{\sum_{i \in X_B} x_i^2}$$

$$\delta_n = \frac{1}{m}\sqrt{\sum_{i \in X_A} (x_i - f_i)^2}$$

where, X_B is the set of benign sources, and n is the number of benign sources. X_A is the set of attack sources, and m is the number of attack sources. Notably, both X_A and X_B are determined based on feedback.

4.3 Implementation & Evaluations

This section givens a short overview of *Horde*'s implementation that is developed as a proof-of-concept to evaluate *Horde*. This section also describes the experiment setup and performance of *Horde*'s agent for varying scenarios.

4.3.1 Implementation

For *Traffic limiting*, *Horde* enforces traffic policing that drops any excessive traffic of a source surpassing its maximum allowed rate [145]. CISCO routers maintain Committed Access Rate (CAR) based on properties such as incoming interface, IP precedence or access list [146]. For *Traffic diversion*, researchers show the feasibility of traffic rerouting using the current traffic engineering techniques without making any infrastructural change [74, 147, 148], or using SDN-based techniques [60, 73, 71]. This research leverages Policy Based Routing (PBR) to reroute traffic through alternative links [149]. PBR enabled routers can route traffic not only based on destination address but also based on source address or port, protocol, and others. We assume that inter-Horde communications regarding spare bandwidth happen through establishing tunnels using techniques such as MPLS tunneling, Secure Shell (SSH) tunneling, or Secure Socket Tunneling. *Horde* may also use these tunnels to forward its traffic to other *Hordes* for traffic rerouting. The framework is developed using Python 3.6.8, and all RNN models are developed using Keras (version 2.2.4) [150] and TensorFlow (version 1.14.0) [151]. zmdp is used to solve the POMDP model with HSVI [152].

4.3.2 Experiment Setup

Horde's performance is evaluated regarding the minimization of benign traffic drop. As all benign traffic have same value and *Horde*'s collaboration model ensures no wastage of spare bandwidth, all agents' decision-optimization emerge optimal defense composition for the whole network. Therefore, feasibility and effectiveness of *Horde* can be assessed based on the performance of individual agents. I confront several challenges to design insightful experiments regarding how to (1) generate *rational and adaptive attack strategies*, (2) integrate *changes in IDS performance* due to diversified attack behaviors, and (3) design *botnet realistically*?

The attack decision model of this dissertation considers weight of aggressiveness w_v and stealthiness w_s that are tuned to generate various attack characteristic types (e.g., stealthy). In these experiments, Type-i adversary strategy type is more aggressive ($w_v(i) > w_v(j)$) and less stealthy ($w_s(i) < w_s(j)$) than Type-j if $i < j$. Various IDS performance regarding actual false positive and negative rate are considered, whose detection accuracy is reactive attack behavior. This research leverages many real-world dataset to design attack bots with properties similar to the real-world scenarios.

Hence, the agent's performance is empirically evaluated in protecting its link against varying attack distinguishability, IDS accuracy, and S_B. Notably, internet topologies affect agent's defense optimization through S_B that varies for different topologies. Hence, experiments with varying S_B also assess the agent's performance for varying topologies. To conduct experiments, a discrete events simulator is designed with entities for benign users and malicious bots. All experiments are simulated using Dell Alienware machine with 16-core 3GHz Intel Core i7-5960X processor, 64GB RAM, and three 4GB NVIDIA GM200 graphics cards. The following paragraphs describe setup of parameters of our experiments:

1. ***Bot Properties:*** Each bot has following properties: IP address, traffic type (e.g., TCP, UDP), AS number (ASN), generated traffic amount, limited traffic ratio (uncertain) at immediate past, duration, and time of its last use. This research does not concern IP spoofing assuming that there are advanced techniques to detect IP spoofing. To assign IP address of botnet, this dissertation uses Mirai-scanner data feed [153] that provides ideal distribution for modeling large-scale DDoS attacks [154]. It also uses IP address lookup to assign ASN, that provides IP's locations, ASN, ISP, and time zone [155].

This dissertation leverages available DDoS dataset attacks [156, 157, 158, 159] to emulate real-world DDoS scenarios. These datasets are used to assign initial traffic and flow rate, maximum duration for continuous flow sending, and bots' appearance. These datasets are used to send attack flows rationally from appropriate botset. The bot receives the command from the attack engine to generate flows at a specific rate (zero if not used). While satisfying all properties, the attacker decreases traffic rate of a bots based on its limited traffic ratio, whereas, he increases its traffic rate if at least 60% of its traffic transmits without any obstruction.

2. ***Attacker's Decision Making:*** The assumption about the attacker is that he observes the consequences of his previous action to decide the next action. Algorithm 8 describes the decision-making approach, where the attacker determines minimum required stealthiness r_s and aggressiveness r_v based on probabilistic inference on link utilization and filtered traffic ratio of previously used botset. The chosen attack action for current time t maximizes his goal based on w_v and w_s while satisfying r_s and r_v. The *Objective* column of Table 3.2 represents the qualitative stealthiness and aggressiveness of attack actions.

There are two type of adaptations: (1) attack characteristic adaptation, where the attacker changes his type (e.g., from Type-1 to Type-2), and (2) attack action adaptation, where the attacker picks different action for same scenario. He leverages

UCB1 policy [131] to explore comparatively less-executed actions despite satisfying r_s and r_v for attack action adaptations. Though the prediction model is pre-trained with numerous attack strategies for various w_v and w_s, different w_v and w_s are used to send attack traffic during evaluation to assess the performance against unseen attack strategies.

Algorithm 7: Attack Strategy

Input : Bot Fail Rate l_b, Delayed Response Rate d_b, Weight Stealthiness w_s, Weight Aggressiveness w_v.

1 attackerBelief q_V = [0,0,0]
2 $q_V[0] = (1 - l_b)d_b$ #0 defines Success
3 $q_V[1] = l_b(1 - d_b)$ #1 defines Failure
4 $q_V[2] = 1 - sum(q_V)$ #2 for Minor Success
5 Normalize q_V
6 $r_s = r_e = 0$
7 $h_b = \frac{1}{r_s}$ #Bot Failed Threshold
8 **for** *i in range(maxStealthiness)* **do**
9 | **if** $l_b \leq h_b$ **then**
10 | | $r_s += 1$
11 | $h_b += \frac{1}{r_s}$
12 $h_d = \frac{1}{r_v}$ #Delayed Response Threshold
13 **for** *i in range(maxAggresiveness)* **do**
14 | **if** $d_b \leq h_d$ **then**
15 | | $r_v += 1$
16 | $h_d += \frac{1}{r_v}$
17 # *candA* is the set of candiate action list
18 $candA$ = [at **for** at in attack space V **if** $at.Y \geq r_s$ or $at.E \geq r_v$]
19 $cAt = maxImpact = None$
20 **for** *at in candA* **do**
21 | $impact = r(at|q_V)(w_s at.Y + w_a t.E) + \sqrt{\frac{2lnn}{n_{at}}}$
22 | **if** *attack is None* ***or*** $maxImpact > impact$ **then**
23 | | $cAt = at, \quad maxImpact = impact$
24 return cAt

In Alg. 8, l_b defines the ratio of bots that could not transmit a minimum traffic (e.g., blocked, limited), and d_b defines the ratio of packet responses that arrived late (i.e., dependent on link utilization). Based on these properties, he computes belief at line 1-5. At line 8-11, he determines the required stealthiness r_s of next attack action,

and he determines required aggressiveness (attack volume expansion) r_e at line 12-16. At line 18, he creates candidate attack set $candA$ whose stealthiness $(Y+n_i)$ is greater or equal to r_s or aggressiveness $(E+n_j)$ is greater or equal to r_v. Notably, $at.Y$ and $at.E$ defines the attack stealthiness and expansion respectively for a particular attack action at.

In line 20-23, the attacker chooses the action cAt that maximizes his current objective. In line 21, $r(at|q_V)$ is the achieved reward by a_t previously when his belief was q_V. The first term in the line defines the expected reward of at based on previous experience (Exploitation), whereas the second term $(\sqrt{\frac{2ln(n)}{n_{at}}})$ gives weight to unexplored actions (Exploration) with the guarantee of logarithmic regret [131]. Here, n is the number of executed attack actions, and n_{at} is the number of times when at was chosen. Finally, this algorithm returns the optimal attack action for current time.

3. ***Traffic Flow Generation*** The bandwidth B of considered critical link is $6GB$. At any time, the aggregated benign traffic amount is $0.8B$ with deviation 10%, and attack traffic amount is $maxAV$ (ranges between $1B$ and $2.5B$) with deviation 20% at the link of the considered agent. The number of benign sources is approximately 10k-100K, and the number of malicious bots varies from 200K-600K.

4. ***Considering Various IDS Performance Parameters:*** To analyze the robustness of *Horde*'s decision-optimization, four different IDS types of Table 4.1 are considered, where *High Risk Score* column represents the ratio of Attack Flows (or bots) having risk score greater than 0.65, and *Low Risk Score* column represents the ratio of Benign Flows (or benign users) with score less than 0.35. To distribute risk scores among newly appeared botset (not seen for a certain time), this dissertation applies power-law distribution that satisfies its associated High and Low Risk Score (%) properties.

The risk score of a bot increases with the increase of its traffic rate, flow rate, flow

Table 4.1: Different IDS Risk Score Distribution

Distribution Type	Attack Flows		Benign Flows	
	False Negative	High Risk Score (%)	False Positive	Low Risk Score (%)
High Accuracy IDS	Low	60-80	Low	70
High Fall-out IDS	Low	60-80	High	30
High Miss Rate IDS	High	30-50	Low	70
Low Accuracy IDS	High	30-50	High	30

duration, and frequent reappearance. Moreover, IDS increases risk scores of those bots that show strong correlations with traffic diversion (descrined in Section 3.6). Besides, IDS is more suspicious about those bots who come from specific AS that previously accommodated known attack bots. This is due to the fact that most bots of Mirai and Conficker came from a small set of ASes, while 97% of ASes had less than 50 attack bots [74]. To update IDS error rate, we assume that 50% benign users send negative feedback for interruptions/delays in their services due to dropping their traffic, whereas, deep analysis reveals 50% attack bots.

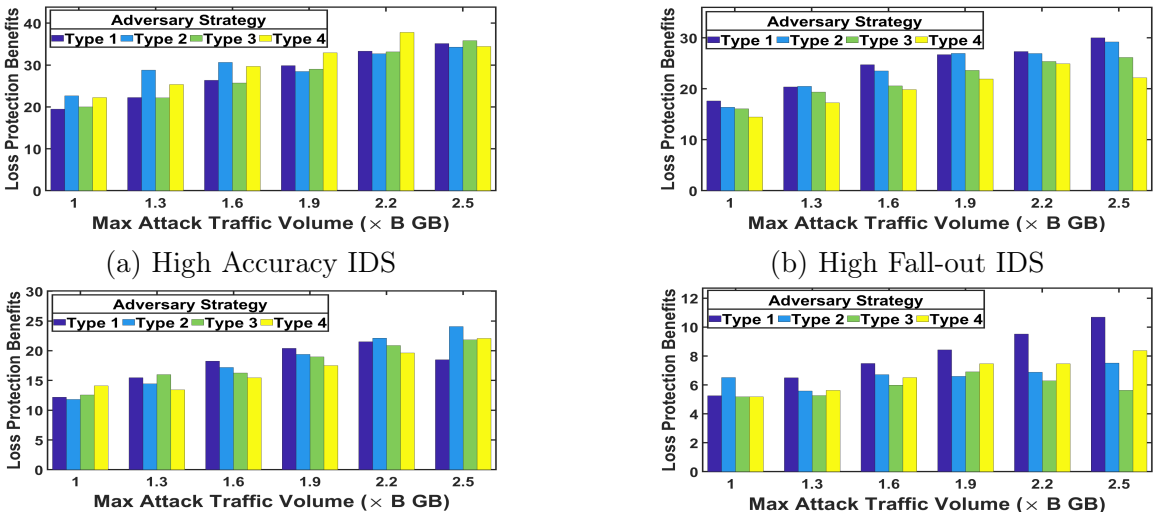
5. ***Evaluation Metrics:*** Effectiveness is assessed based on *Packet Loss Protection Benefits* $\mathcal{G}$ and *False Discovery Rate* F_P^t. Here, $\mathcal{G}$ quantifies the minimization of benign traffic drop due to our framework, and F_p^t quantifies the benign traffic ratio out of $\mathcal{T}_L^t$ that the defense action a_d drops at a time t.

$$\mathcal{G} = \frac{\overline{\mathcal{T}_B^l}}{\mathcal{T}_B^l}, \quad F_p^t = \frac{\mathcal{T}_B^{l_t}}{\mathcal{T}_{L_t}} \tag{4.6}$$

where, $\mathcal{T}_B^l$ and $\overline{\mathcal{T}_B^l}$ are total (considering all time) benign traffic drop *With* and *Without* framework respectively, and $\mathcal{T}_B^{l_t}$ is the dropped benign traffic by limit function l_t of a_d at t. The performance is also assessed by the ratio of *Benign Traffic Diversion* and *Drop* out of total benign traffic.

4.3.3 Performance Analysis

This section describes the agent's effectiveness in maximizing benign traffic serving and its feasibility in real-time optimizations. Besides, it analyzes the agent's



(a) High Accuracy IDS

(b) High Fall-out IDS

(c) High Miss Rate IDS

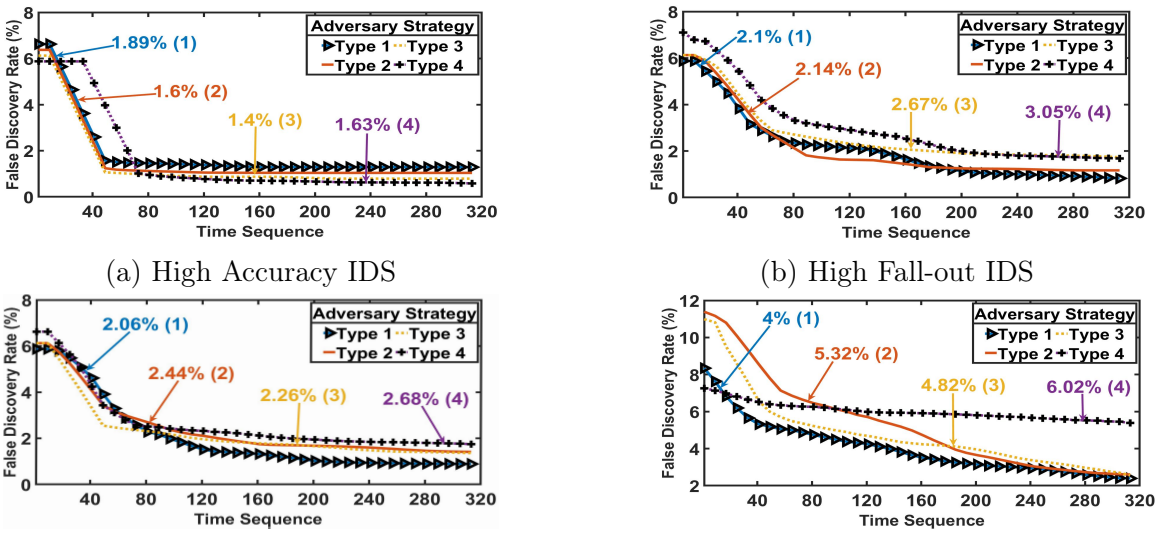
(d) Low Accuracy IDS

Figure 4.7: Analysis of Packet Loss Protection Benefit

capability in learning the environment incrementally based on observations.

1. ***Analysis of Packet Loss Protection benefits:*** Fig. 4.7 illustrates the *Loss Protection Benefit* $\mathcal{G}$ (Eqn. 4.6) with respect to the maximum attack volume $maxAV$ (x-axis). The performance is best with *high accuracy IDS* (in Fig. 4.7a) and worst with *low accuracy IDS* (in Fig. 4.7d). The agent performs 3 times better with high accuracy IDS than with low accuracy IDS. Moreover, with high accuracy IDS, the agent drops more than 80% attack traffic. Interestingly, benefit for *high fall-out IDS* in Fig. 4.7b is higher than benefit for *high miss rate IDS* in Fig. 4.7c, because the agent executes more traffic diversion for high fall-out IDS to minimize benign traffic drop. Notably, in all cases, benefit is significantly increasing with the increase of $maxAV$.

As per Fig. 4.7a, aggressive attackers induces more benign packet loss against high accuracy IDS due to higher attack volume as bots of attackers who impose more weight on stealthiness still get detected. Whereas, attack stealthiness induces more loss against high fall-out IDS (Fig. 4.7b) and high miss rate IDS (Fig. 4.7c). For high fall-out IDS, the maximum gain difference between most aggressive and most stealthy attacker is 10. Interestingly, against low accuracy IDS (Fig. 4.7d), the attacker with



(a) High Accuracy IDS

(b) High Fall-out IDS

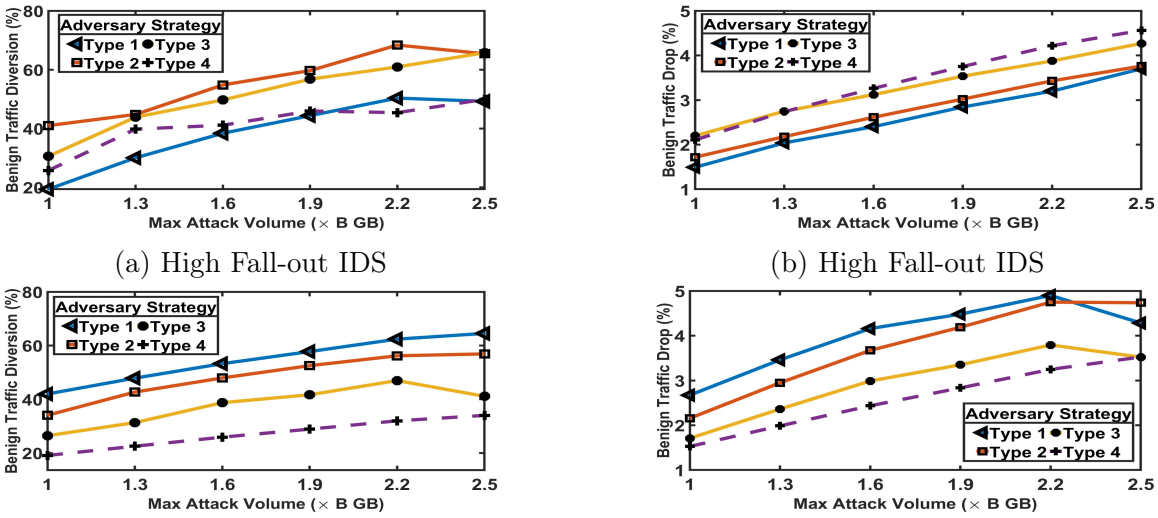
(c) High Miss Rate IDS

(d) Low Accuracy IDS

Figure 4.8: Analysis on Benign Traffic Drop By Traffic Limiting. Note# Text with arrow represents the mean traffic rate. For example, 1.89%(1) is the mean false discovery rate of adversary Type 1 for first 320 time-sequences.

balanced attack aggressiveness and stealthiness (*Type 2* and *Type 3*) causes serious disruptions. This is because such balanced attacks increase attack flow distributions at lower risk scores, that induces more benign traffic drop despite restrictive traffic limitings.

2. ***Analysis on False Discovery Rate:*** Fig. 4.10 illustrates the progression of *False Discovery Rate* F_p^t (Eqn. 4.6) with time. Initially, F_p^t is always high due to picking wrong actions because of insufficient domain-knowledge, but it reduces exponentially with time due to agent's learning. The learning is faster with *high accuracy IDS* (in Fig. 4.8a) that takes 40 time-sequences to reduce F_p^t by 75%, but worst with *low accuracy IDS* (in Fig. 4.8d) that takes 120 time-sequences to reduce 60%. For high accuracy IDS, the agent struggles only against aggressive attackers. For *high fall-out IDS* (in Fig. 4.8b), the agent reduces traffic limiting due to significant negative feedback. For *high miss rate IDS* (in Fig. 4.8c), the agent observes that only dropping from higher risk scores consequently brings more attack traffic due to allowing many attack bots, and starts emphasizing on timid limiting to restrict traffic



(a) High Fall-out IDS

(b) High Fall-out IDS

(c) High Miss Rate IDS

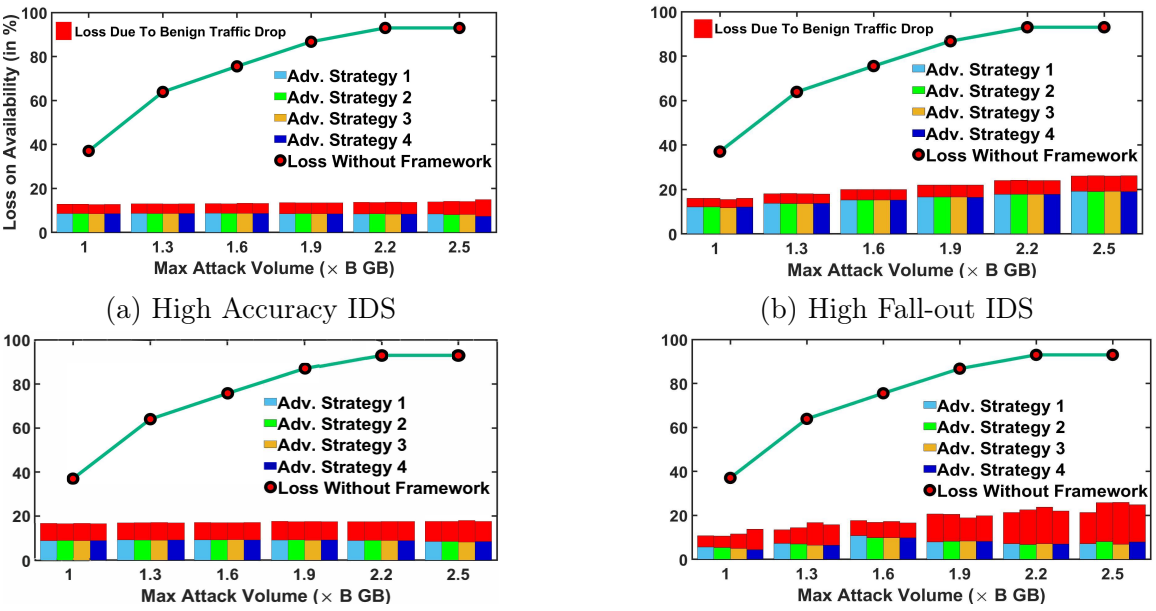
(d) High Miss Rate IDS

Figure 4.9: Analysis on Serving Benign Traffic

from comparatively lower ranges cautiously. However, in all cases, F_p^t reduces at least 50% within 50 time-sequences, that proves the effectiveness of agent's active learning.

3. ***Analysis on Benign Traffic Serving:*** According to Fig. 4.9a, attack stealthiness worsens the condition of *high fall-out IDS* as attack traffic starts mixing more with benign traffic that have comparatively higher scores. It compels the agent to enhance traffic diversion to minimize negative feedback of benign users. Interestingly, for *Type 4* attacker (super stealthy), benign traffic diversion is less than other stealthy attackers (*Type 2* and *3*), because diverted traffic contains too many attack traffic due to attack indistinguishability. For the same reason, benign traffic diversion for *high miss Rate IDS* (Fig. 4.9c) is also comparatively lower. For such IDS, benign traffic diversion for aggressive attacker is high due to better detectability. In both cases, the maximum benign traffic drop is less than 5% while the minimum is 1.5%.

4. **Loss Analysis:** Fig. 4.9 compares the loss of our defense framework with loss during "No Defense" (Loss Without Framework), which shows the maximization of availability of legitimate services by this framework. Firstly, it is apparent from the figure that the benefit of using this framework regarding loss reduction is quadratic



(a) High Accuracy IDS

(b) High Fall-out IDS

(c) High Miss Rate IDS

(d) Low Accuracy IDS

Figure 4.10: Analysis on Benign Traffic Drop By Traffic Limiting. Note# Text with arrow represents the mean traffic rate. For example, 1.89%(1) is the mean false discovery rate of adversary Type 1 for first 320 time-sequences.

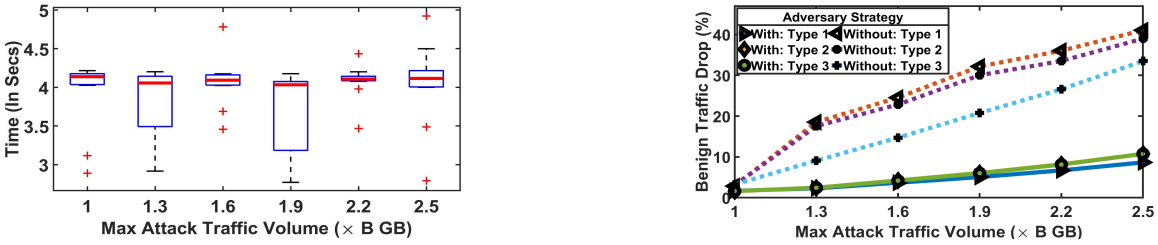
with the increase of attack volume. In Fig. 4.10a, the loss is almost constant despite the attack volume increase, which shows that enhanced attack volume cannot deteriorate the network condition against *High Accuracy IDS*. However, for *High Recall IDS* (in Fig. 4.10b), the defender had to reroute more traffic from higher risk score ranges due to enhanced attack volume and higher false positive rate. In contrast, for *High Specificity IDS* in Fig. 4.10c, higher loss of benign traffic drop occurs because of more traffic limiting while keeping benign traffic diversion same. Additionally, in such cases, the defender increased rerouting of attack traffic due to misidentifying these as benign traffic. Unsurprisingly, for *Low Accuracy IDS* in Fig. 4.10d, benign traffic drop and attack traffic diversion increased due to misidentification. In all cases except high accuracy IDS, stealthy attackers with balanced attack volume ("Type 2" and "Type 3") induces more loss.

5. ***Defense Reports:***

Table 4.2 contains the summary of consequences of executed defense planning. The

Table 4.2: Defense Reports

Dataset Source	Benign Traffic		Attack Traffic	
	Divert Ratio (%)	Drop Ratio (%)	Divert Ratio (%)	Drop Ratio (%)
CICDDoS2019 (1)	59	5	9.4	67.65
CICDDoS2019 (2)	60.5	3.6	8.75	71
CTU-13 (4)	57	7.5	8.4	73
CTU-13 (10,11)	61	3.1	8.9	70.75



(a) Scalability Analysis

(b) Performance Comparison

Figure 4.11: (a) Required time where the red line represents the mean time and box size represents deviations, (b) Benign Drop Reduction due to having Attack Prediction Model.

attack traffic drop ratio is approximately 70%, whereas benign traffic drop is near to 5%. Due to the higher false positive rate, the agent reroutes traffic from comparatively higher risk score ranges, that induces attack traffic diversion. Overall, our agent is effective against the considered real-world DDoS attacks.

6. ***Scalability Analysis:*** Fig. 4.11a shows boxplots of time required from *Understand* to *Policy Generation*. This plot is illustrated considering all experiments of evaluation, and POMDP policy generation generally requires majority of time. According to these experiments, the mean required time is less than 4.5 seconds and independent on attack volume, which shows the feasibility of *Horde* as real-time defense planner.

4.3.4 Performance Analysis of Attack Prediction Model

This section describes the importance of attack prediction by comparing the performance of *Horde* with the approach that considers static attack behavior. It also shows the performance of attack prediction model in learning attack strategies in

presence of attack deceptions and attacker's strategical adaptations.

1. ***Benefits of Attack Prediction:*** Fig. 4.11b illustrates the performance comparison in minimizing benign traffic drop when *without* attack prediction (dotted lines) with *Horde*'s approach *with* prediction model (solid line). Evidently, attack prediction always minimizes benign traffic drop, and its benefits become more vivid with attack volume increase. The deviation between drop ratios increases approximately from 15% to 30% for attack volume increase from $1.3B$ to $2.5B$. Because, picking action without predicting the attack reaction cannot be effective enough to avoid conditions that induce significant benign traffic drop. Whereas, prediction helps the agent to infer imminent attack intensity. This experiment also exhibits that aggressive attackers cause more benign traffic drop due to IDS inaccuracies when without prediction.

2. ***Performance of Attack Prediction Model:*** Fig. 4.12 shows the performance of our attack prediction in the cases of adversary deceptions and attack type adaptations; where, the attacker executes deceptive actions during the Deception interval, and changes his characteristic type (e.g., from aggressive to stealthy) at (near) the point of Drift (Fig. 4.12c). As in Fig. 4.12a, error rate becomes less than 10% within 140 time sequences for *Type 1* and *5* before deception 1, whereas it is less than 15% for other attack types. After the deception interval, errors reduce exponentially (approximately 64%), that shows robustness of our model. Interestingly, after each deception interval at Fig. 4.12a and 4.12c, error is high during the next interval with no deception due to not exactly determining the real deception window.

In Fig. 4.12c, all attackers adapt their characteristic types at Drift ($t = 240$); where, for instance, *Type (1,5)* specifies that the attacker becomes *Type 5* from *Type 1*. Notably, error growth rate is less than 1.3 times for *Type (3,1)*, whereas it is more than 4 times for other attack adaptation types. Interestingly, the model of *Type (5,3)* performs better for initial type but performs worst for the later, which means that

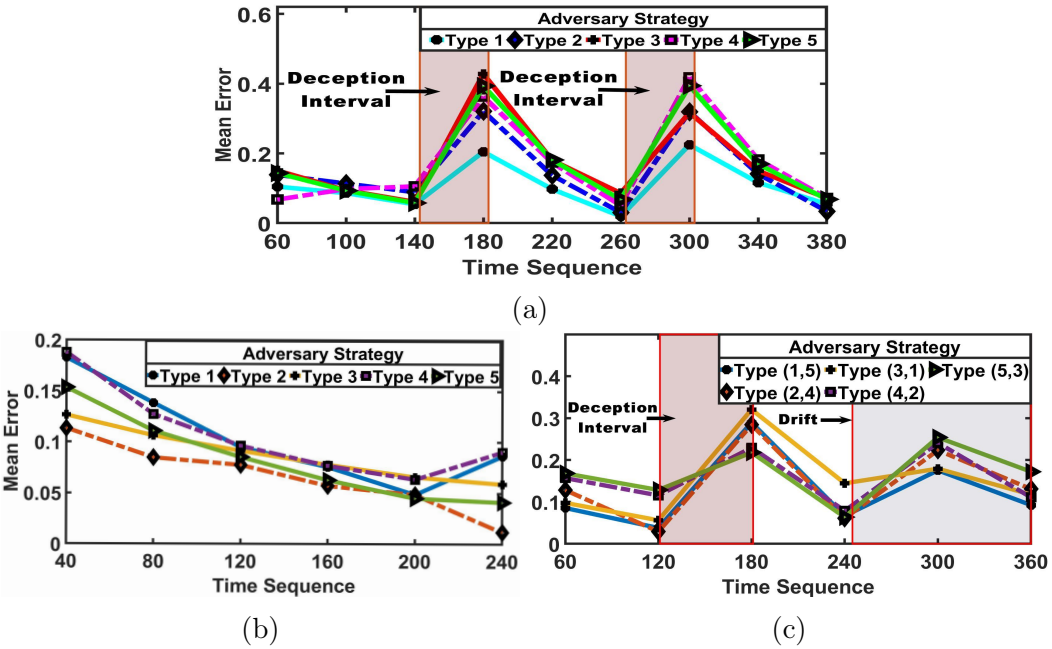


Figure 4.12: Performance of Attack Prediction Model. (a) Two Deception Intervals, (b) No Deception and No Attack Characteristics Type Adaptation, and One Deception and Attack Characteristics Type Adaptation (Drift).

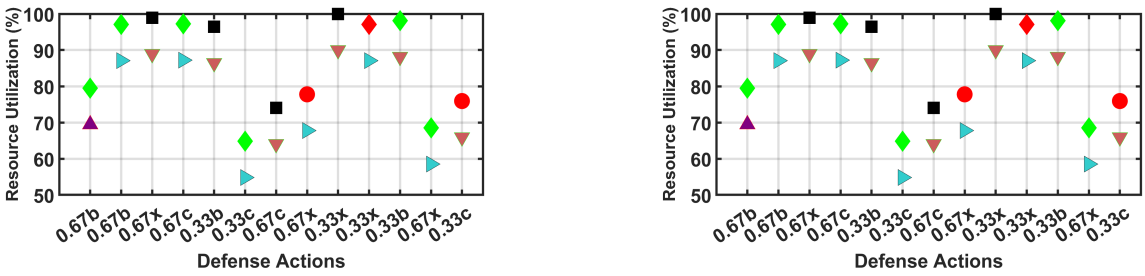
the model is still biased to initial adversary behavior. However, error rates gradually reduce with new attack observations in all cases, that shows the capability of learning new strategy incrementally.

4.3.5 Sensitivity Analysis of Agent's Decision-making

This section describes how the attacker and *Horde*'s agent react to the changes of link condition and opponent's behaviors.

1. ***Game Between Agent and Attacker:***

In Fig. 4.13, X-axis shows the defense actions sequentially; where, for instance, $0.33x$ specifies that the considered action reroutes 33% traffic and drops 67% of traffic by Timid limiting out of $\mathcal{T}_M$ (mitigated traffic). Here, x, c, and b represent timid, aggressive, and traditional limiting respectively, and the color of the shape (botset adjustment) is black when bonet adjust rate χ=None, blue when χ=0, green when χ=2, or red when χ=2. At a specific time, the y-coordinate of a shape represents

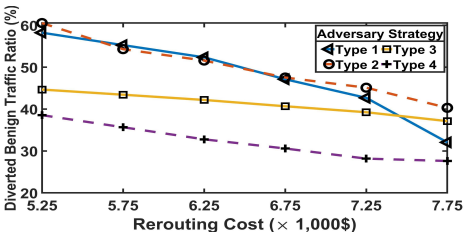


(a) High Miss Rate IDS (b) High Fall-out IDS

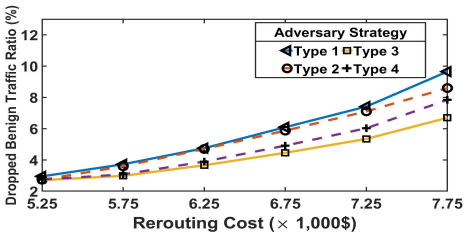
Figure 4.13: Analysis of Defense and Attack Game.
Shape (Botset Adjustment): *Box* for Unchanged, *Circle* for Expanded, *Diamond* for Changed Bot Distribution;
Triangle/Arrow (Traffic Rate Adjustment): *Right* for Unchanged, *Down* for Decreased, *Up* for Increased.

the utilization of the queue considering capacity $1.5B$. An emergent outcome of this analysis is that the attacker has to enhance botset (higher cost) to maintain high utilization while lowering traffic rate, and he has to change bot distribution or enhance botset (higher cost) to evade traffic filtering while increasing traffic rate. That also proves our phenomena that the attacker cannot achieve high impact, high stealthiness, and low cost simultaneously. Also, it is observed that too much traffic diversion motivates the attacker to enhance traffic rates, which causes exponential rise in attack traffic arrival. Hence, the agent with *high fall-out IDS* (at Fig. 4.13a) comparatively confronts more attack traffic due to high diversion. Interestingly, for *high miss rate IDS*, the attacker hardly increases his traffic rate to exploit the weakness (higher false negative) of IDS; instead, he enhances botset to increase attack volume. Moreover, for high fall-out IDS, the agent hardly leverages timid limiting unlike the other case, to minimize the risk of dropping benign traffic.

2. ***Impact of Rerouting Cost:*** Fig. 4.14a and 4.14b show impacts on benign traffic diversion and drop respectively for increasing per GB traffic rerouting cost. For aggressive *Type 1* attacker, benign traffic diversion reduces 50% while drop increases 8%, for 50% cost increase. However, against *Type 2*, traffic diversion reduces only 25% while the drop increases from 3% to 9%. Moreover, the situation is worst (7%

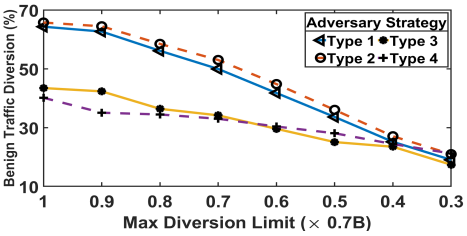


(a) Diverted Benign Traffic

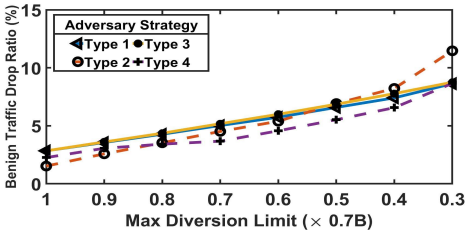


(b) Dropped Benign Traffic

Figure 4.14: Sensitivity to Rerouting Cost.



(a) Loss on Availability



(b) Benign Traffic Drop

Figure 4.15: Sensitivity to Max. Diversion Limit.

reduction) against *Type 3* attacker with balanced aggressiveness and stealthiness due to high attack density at lower risk scores. Hence, the decrease of diversion due to increased cost becomes slow with the increase of stealthiness. Notably, the growth rate of benign traffic drop depends on decrease rate of diversion.

3. ***Impact of Max Diversion Limit:*** Fig. 4.15a and 4.15b show impacts on benign traffic diversion and drop respectively with respect to *Max. Diversion Percentage* M_D that depends on overall conditions of all network links. Unsurprisingly, for aggressive *Type 1* attacker, the impact is less significant due to. higher attack detection accuracy. For stealthy *Type 2* attacker, the packet drop growth rate is 75% for 50% decrease of M_D from the point 0.6 (y-axis). For stealthier *Type 3* attacker, the drop growth rate is less than of *Type 2* attacker due to less attack traffic. Hence, stealthy attacker with non-compromised attack volume exploits the restriction on rerouting more, but such attack approach incurs huge attack cost.

4.4 Summary

This chapter focuses to develop models and methods to be integrated into *Horde* to enable agents' autonomous evolvement to cope with environmental changes. This chapter introduces new approaches to advance the state-of-the-art of autonomous decision-making in a dynamic cyber environment against adaptive and deceptive attackers. The presented approach of learning attack strategy *incrementally* not only detects attack adaptations but also shows robustness against attack deceptions. According to the evaluation, the attack prediction model shows impressive accuracy in understanding attack strategy, and detecting attacker's strategical adaptations and deceptions. The evaluation shows that predicting the next attack behavior using attack prediction model reduces benign traffic drop rate by at least twice, which empirically proves the benefit of integrating attack behavior into the decision-loop. It steers the agent's decision-making towards adversary-aware decision-optimization that significantly minimize the loss of dropping and delaying benign traffic serving while maximizing the attack traffic drop.

This chapter presents an *incremental* approach of refining/learning system dynamics. It aids an agent to learn the network behavior in a new domain or refine knowledge on network dynamics in the event of domain shift or other sensitive network events such as physical link/sensor failures, enhanced benign traffic, and others. this chapter presents an approach to predict the optimal discount factor based on current contexts. In average, defense execution considering predicted discount factor achieves more rewards than any static discount factor. The evaluation shows that this model converges towards the optimal context-aware discount factor within few time-sequences despite being in a new domain with limited or no existing knowledge. Both approaches of learning system dynamics and predicting discount factor based on recent observations improves an agent's decision-making in minimizing benign traffic

dropping with the passage of time. Apparently, it proves that integrating these learning models not only makes *Horde* applicable in a new domain without pre-knowledge but also maintains *Horde*'s efficacy despite the domain shift or critical network events. Moreover, as per evaluation, refining IDS error rate aids *Horde*'s agents to adapt its decision-making autonomously according to the current environment.

This chapter creates ample scope for future research. Though RNN based online attack learning approach have showed impressive potentials, it still struggles in dealing with uncertainties of previously identified attack actions. This is not a rare case when the agent finds previously identified attack action has non-trivial deviation from the ground-truth. This may be interesting to explore how to provide such corrected data/observations to the model to refine. This research only analyzes the robustness of prediction model against random attack actions as deceptive actions that generally deviate far from adopted attack strategy. However, sophisticated attackers might poison the learning in slower way with hardly noticeable deviation. Though it is not easy to execute such slow poisoning in a complex and dynamic cyber network, chances cannot be negated completely. Hence, in future, I plan to assess the performance of prediction model against sophisticated but realistic attack deceptions. The plan also includes determining optimal deception intervals to minimize deviations in predictions. One limitation of *Horde* is that it does not address the likelihood of delayed feedback which is not unusual considering user-perspective, network overhead of sending puzzles, or computational overhead of deep flow behavioral analysis. Though this research assumes timely arrival of feedback, one of the future extension will be understanding and integrating uncertainties regarding delayed feedback into decision-optimization.

CHAPTER 5: Conclusion

This dissertation focuses on the problems of optimizing cybersecurity portfolio and I-DDoS defense strategy dynamically. To address these challenges, it aims to satisfy three research objectives: (1) maximizing expected Return on Investment (RoI) by selecting optimal set of countermeasures, (2) developing distributed framework to optimize decision-making against I-DDoS attacks in real-time, and (3) developing methodologies to learn I-DDoS attack strategy and enable defense evolving in an online manner to ensure dynamic defense optimization. These objectives are discussed with details in previous chapters (Chapter 2-4). This section provides summary on contributions and finding of these chapters followed by limitations and future tasks.

The second chapter presents an automated tool, CyberARM, that selects an optimal set of security countermeasures to compose a cost-effective and resilient cybersecurity portfolio for an enterprise. This chapter extends and customizes the Cyber Defense Matrix (CDM) to increase the explainability of computed cybersecurity portfolio regarding defense mechanisms. Alongside the mapping from Critical Security Controls (CSC) to threat actions, CDM aids to satisfy fine-grained defense requirements. One of the contribution of this chapter is developing probabilistic models for prioritizing threat surface considering correlations among threat, threat actions, and vulnerabilities. Experiments on VERIS Community Database [160] reveal that these models can predict the likelihood of exerting particular threat actions using past threat incident reports with tolerable variance. CyberARM composes a cybersecurity portfolio that defends the prioritized threat surface effectively while satisfying budget, resiliency, and other business and mission oriented requirements. It formalizes the

problem of optimizing risk mitigation as Constraints Satisfiability Problems (CSPs), where all requirements are considered as SMT constraints.

To enhance the scalability, this chapter presents two heuristic approaches: model reduction, and model decomposition, that significantly prune the search space while still approximating the solution very close to the optimal solution. The evaluation results show that CyberARM can generate a correct-by-construction risk mitigation plan for a large enterprise with 15,000 assets and diversified requirements within 10 minutes. According to these experiments, the computational complexity of CyberARM is largely dependent on budget deficiency (i.e., difference between given and required budget) and risk tolerance margin (i.e., difference between minimum achievable risk and affordable risk). The evaluation also discusses the sensitivity and performance of the tool considering various sizes of asset lists and requirements. It shows that increased number of strict requirements hardens the problem more compared to increased number of assets. The discussed use case study shows the applicability and robustness of the tool in presence of noise in given inputs.

The third chapter presents a distributed multi-agent framework, *Horde*, that applies Reinforcement Learning (RL) based decision model to enact real-time defense optimization against multi-strategy and adaptive I-DDoS attacks. This chapter presents a hybrid approach of model-free and model-based RL learning. While *Horde* learns all environment system parameters without explicit models or formulations, it designs the environment with a particular model solvable with dynamic POMDP planning. This framework *autonomously* composes an optimal ratio of traffic filtering (i.e., dropping suspicious traffic) and diversion (i.e., rerouting through alternative links), in order to maximize benign traffic service while also maximizing attack traffic drop. With this decision model, an agents enables adaptive defense planning considering current condition of network link, traffic distribution across risk scores, and attack behavior. According to experiments in evaluation, the agent's learning of system

parameters based on observations and user-feedback accelerates the convergence of defense planning towards optimal solutions. This chapter also presents a BRITE loop describing five capabilities that must be integrated into decision process to develop an autonomous agent. Using the BRITE loop, this chapter discusses how agents ensure real-time defense optimization against I-DDoS attacks in a stochastic environment. All agents' defense plans orchestrate to compose a global optimal defense plan that aggregatedly maximizes the benign traffic serving.

The fourth chapter describes methodologies to integrate the *Evolve* capability into an agent's decision-process to cope with changes of dynamic environment and attack behavior. This chapter presents an online approach of training Recurrent Neural Network (RNN) model to learn the adversary behavior based on observations, in order to predict the imminent attack behavior required for both proactive and reactive defense strategies. Importantly, *Horde* learns new and adaptive attack behavior without being explicitly trained, that enables the agent's decision model to perform strategic reasoning against any attacker. According to experiments, this prediction model can not only detect sophisticated attack strategies impressively but can also differentiate the events of attack adaptations and deceptions. Besides, as per evaluation, this prediction model improves the performance of *Horde* significantly. This chapter presents a scalable online approach that can learn the sophisticated network behavior without requiring explicit formulations or deep-domain data.

The fourth chapter addresses the dilemma of short-term goals (i.e., small discount factor) and long term goals (i.e., large discount factor) through training a RNN model *incrementally* on previous observations. This RNN model predicts the optimal discount factor based on current link condition and attack behavior. In summary, despite having limited or no knowledge, *Horde* can learn the behavior of the environment, and can evolve according to changes of environment, as well as attack behavior. This chapter also presents several experiments assessing the performance of *Horde* against

diversified attack strategies considering different IDS inaccuracies. According to our experiment results, *Horde* can minimize benign traffic drop significantly against large-scale infrastructural attacks, and recovers from bad states within few time sequences. Moreover, *Horde* shows robustness in cases of unanticipated network or attack dynamics and attack deceptions. Additionally, this framework ensures scalability for real-time defense optimization through its distributed architecture and context-aware decision approximations. Besides, it is easily deployable without requiring any complex static or human configurations, that also ensures its applicability despite of domain shifts.

Limitations & Future Tasks: Though this dissertation addresses many key challenges related to optimizing cybersecurity portfolio and I-DDoS defense planning, it has limitations that I plan to solve in future extensions.

1. Though CyberARM addresses missing data of threat actions and vulnerabilities, these metrics do not consider critical factors such as attack sophistication, topological complexities, and others. Besides, there is likelihood that vulnerability scanning reports may fail to report vulnerabilities with high severity scores. In future, I will try to make threat prioritization metrics robust to missing data by leveraging attack graph that may provide hints of missing threat action/vulnerabilities.

2. One of the limitation of the work is that it cannot explicitly defend zero-day attacks. CyberARM selects the set of cybersecurity countermeasures by correlating threat actions and vulnerabilities. It assumes that either it knows threat actions that will exploit zero-day vulnerabilities, or the vulnerability exploitable by new attack approach can be found in vulnerability scanning reports. However, it cannot address the issue if the attacker exploits a new vulnerability with an approach that is not in previous threat incident reports. In future, I plan to extend CyberARM to address such cases of zero-day attacks, which will definitely attract more CISOs to explore

the tool.

3. One of the assumption of *Horde* is that it assigns risk score for each of the incoming flows at upstream points. However, it may struggle to handle such massive flows that can be gigabytes in volume at real-time. Hence, in future, I will try to develop a heuristic approach that will analyze only samples of flows instead of scrutinizing each of the flow.

4. *Horde* assumes that it receives feedback on benign and attack flows to refine IDS inaccuracies. However, in reality, it may come several days later. Therefore, one of the extension of the dissertation is to find a rational approach to incorporate the likelihood of delayed feedback into decision-making.

5. *Horde* assumes a sliding window with static size to distinguish between attack deceptions (i.e., random action execution) and strategical attack types adaptions (i.e., from aggressive to stealthy). However, the evaluation shows that the performance of the attack prediction model degrades due to considering large window size than the original deception window. Because, it ignores non-deceptive actions. One of the future goal is to determine the deception window dynamically.

6. Despite the distributed architecture, the scalability of *Horde* still struggles with the expansion of defense space. However, defense cost regarding benign traffic drop or delay can be further minimized by introducing more fine-grained defense compositions. Therefore, I plan to leverage sampling algorithm such as Monte Carlo Approximation to enhance scalability.

REFERENCES

[1] *4 Ways Cybersecurity Automation Should Be Used.* https://www.paloaltonetworks.com/cyberpedia/4-ways-cybersecurity-automation-should-be-used/.

[2] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy*, pp. 127–141, IEEE, 2013.

[3] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*, pp. 37–52, Springer, 2009.

[4] M. Mullane, *Why critical infrastructure is vulnerable to cyber-attacks.* https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f.

[5] *Become a hacker with $1.* https://www.cpomagazine.com/cyber-security/11-eye-opening-cyber-security-statistics-for-2019/.

[6] *Adversarial Tactics, Techniques & Common Knowledge.* https://attack.mitre.org/wiki/Main_Page.

[7] K. Sheridan, *Defense Evasion Dominated 2019 Attack Tactics.* https://www.darkreading.com/vulnerabilities—threats/defense-evasion-dominated-2019-attack-tactics/d/d-id/1337457.

[8] *Password Policy Discovery.* https://attack.mitre.org/techniques/T1201/.

[9] *Software Discovery.* https://attack.mitre.org/techniques/T1518/.

[10] A. Phillips, *The Asymmetric Nature of Cyber Warfare.* https://news.usni.org/2012/10/14/asymmetric-nature-cyber-warfare.

[11] W. Ng, *Defenders and Attackers - Economic Asymmetry in Cyber Security.* https://www.linkedin.com/pulse/defenders-attackers-economic-asymmetry-cyber-ng-cissp-ccnp/.

[12] *How much does it cost to launch a cyberattack?* https:/https://www.infoguardsecurity.com/how-much-does-it-cost-to-launch-a-cyberattack/.

[13] A. Dutta, E. Al-Shaer, and S. Chatterjee, "Constraints satisfiability driven reinforcement learning for autonomous cyber defense," *arXiv preprint arXiv:2104.08994*, 2021.

[14] C. T. Do et al., "Game theory for cyber security and privacy," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–37, 2017.

[15] Z. Hu, M. Zhu, and P. Liu, "Online algorithms for adaptive cyber defense on bayesian attack graphs," in *Proceedings of the 2017 Workshop on moving target defense*, pp. 99–109, 2017.

[16] E. Miehling, M. Rasouli, and D. Teneketzis, "A pomdp approach to the dynamic defense of large-scale cyber networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2490–2505, 2018.

[17] P. et al., "A game-theoretical approach to cyber-security of critical infrastructures based on multi-agent reinforcement learning," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, pp. 460–465, IEEE, 2018.

[18] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *arXiv preprint arXiv:1906.05799*, 2019.

[19] Center for Internet Security, *CIS Controls Version 7*, March 2018. https://www.cisecurity.org/controls/.

[20] L. Donnan, *Cybersecurity Venture Capital Investing Trends In 2020.* option3ventures. https://option3ventures.com/cybersecurity-venture-capital-trends/.

[21] T. Riley, *The Cybersecurity 202: Global losses from cybercrime skyrocketed to nearly $1 trillion in 2020, new report finds.* https://www.washingtonpost.com/politics/2020/12/07/cybersecurity-202-global-losses-cybercrime-skyrocketed-nearly-1-trillion-2020/.

[22] Neustar, *DDoS Attacks Increase by 151% in First Half of 2020.* https://www.home.neustar/about-us/news-room/press-releases/2020/ddos-attacks-increase-by-151-in-first-half-of-2020.

[23] Corero, *DDoS Protection for Critical Infrastructure.* https://www.corero.com/solutions/critical-infrastructure.html.

[24] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 1, pp. 338–347, IEEE, 2001.

[25] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "Packetscore: Statistics-based overload control against distributed denial-of-service attacks," in *IEEE INFOCOM 2004*, vol. 4, pp. 2594–2604, IEEE, 2004.

[26] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic," in *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 30–41, ACM, 2003.

[27] *Fortinet Announces the FortiDDoS E-Series with Two New Models â FortiDDoS 1500E and 2000E.* https://www.fortinet.com/blog/business-and-technology/fortinet-announces-fortiddos-e-series.

[28] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*, pp. 305–343, Springer, 2018.

[29] N. Bjørner and L. De Moura, "Z310: Applications, enablers, challenges and directions," in *Sixth international workshop on constraints in formal verification*, vol. 16, 2009.

[30] L. De Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Brazilian Symposium on Formal Methods*, pp. 23–36, Springer, 2009.

[31] D. Braziunas, "Pomdp solution methods," *University of Toronto*, 2003.

[32] T. Smith and R. Simmons, "Heuristic search value iteration for pomdps," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 520–527, AUAI Press, 2004.

[33] "Pomdp background." http://www.pomdp.org/tutorial/pomdp-background.html.

[34] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge, 1998.

[35] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[36] T. G. Dietterich, "Machine learning for sequential data: A review," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pp. 15–30, Springer, 2002.

[37] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[38] C. J. Alberts and A. Dorofee, *Managing information security risks: the OCTAVE approach*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[39] B. Jerman-Blažič *et al.*, "An economic modelling approach to information security risk management," *International Journal of Information Management*, vol. 28, no. 5, pp. 413–422, 2008.

[40] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 4, pp. 438–457, 2002.

[41] L. P. Rees, J. K. Deane, T. R. Rakes, and W. H. Baker, "Decision support for cybersecurity risk planning," *Decision Support Systems*, vol. 51, no. 3, pp. 493–505, 2011.

[42] W. Sonnenreich, J. Albanese, B. Stout, *et al.*, "Return on security investment (rosi)-a practical quantitative model," *Journal of Research and practice in Information Technology*, vol. 38, no. 1, p. 45, 2006.

[43] A. Fielder, S. Konig, E. Panaousis, S. Schauer, and S. Rass, "Uncertainty in cyber security investments," *arXiv preprint arXiv:1712.05893*, 2017.

[44] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Decision support approaches for cyber security investment," *Decision Support Systems*, vol. 86, pp. 13–23, 2016.

[45] T. R. Rakes, J. K. Deane, and L. P. Rees, "It security planning under uncertainty for high-impact events," *Omega*, vol. 40, no. 1, pp. 79–88, 2012.

[46] T. Sawik, "Selection of optimal countermeasure portfolio in it security planning," *Decision Support Systems*, vol. 55, no. 1, pp. 156–164, 2013.

[47] R. J. Kauffman and R. Sougstad, "Risk management of contract portfolios in it services: The profit-at-risk approach," *Journal of Management Information Systems*, vol. 25, no. 1, pp. 17–48, 2008.

[48] V. Viduto, C. Maple, W. Huang, and D. LóPez-PeréZ, "A novel risk assessment and optimisation model for a multi-objective network security countermeasure selection problem," *Decision Support Systems*, vol. 53, no. 3, pp. 599–610, 2012.

[49] M. Gupta, J. Rees, A. Chaturvedi, and J. Chi, "Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach," *Decision Support Systems*, vol. 41, no. 3, pp. 592–603, 2006.

[50] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 204–213, ACM, 2007.

[51] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[52] J. C.-Y. Chou, B. Lin, S. Sen, and O. Spatscheck, "Proactive surge protection: a defense mechanism for bandwidth-based attacks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 6, pp. 1711–1723, 2009.

[53] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 2, pp. 115–139, 2006.

[54] T. M. Gil and M. Poletto, "Multops: A data-structure for bandwidth attack detection.," in *USENIX Security Symposium*, pp. 23–38, 2001.

[55] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 71–82, ACM, 2002.

[56] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring.," in *LISA*, vol. 14, pp. 139–146, 2000.

[57] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM computer communication review*, vol. 35, pp. 217–228, ACM, 2005.

[58] Y. Chen and K. Hwang, "Collaborative detection and filtering of shrew ddos attacks using spectral analysis," *Journal of Parallel and Distributed Computing*, vol. 66, no. 9, pp. 1137–1151, 2006.

[59] F. Wang, H. Wang, X. Wang, and J. Su, "A new multistage approach to detect subtle ddos attacks," *Mathematical and Computer Modelling*, vol. 55, no. 1-2, pp. 198–213, 2012.

[60] M. S. Kang, V. D. Gligor, V. Sekar, *et al.*, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks.," in *NDSS*, 2016.

[61] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, D. Karger, and S. Shenker, "Ddos defense by offense," in *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 303–314, ACM, 2006.

[62] IETF, *TLS Client Puzzles Extension draft-nygren-tls-client-puzzles-00*. https://tools.ietf.org/html/draft-nygren-tls-client-puzzles-00.

[63] X. Wang and M. K. Reiter, "Defending against denial-of-service attacks with puzzle auctions," in *2003 Symposium on Security and Privacy, 2003.*, pp. 78–92, IEEE, 2003.

[64] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against ddos attacks," 2002.

[65] A. Yaar, A. Perrig, and D. Song, "Stackpi: New packet marking and filtering mechanisms for ddos and ip spoofing defense," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1853–1863, 2006.

[66] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based ip filtering," in *IEEE International Conference on Communications, 2003. ICC'03.*, vol. 1, pp. 482–486, IEEE, 2003.

[67] A. Yaar, A. Perrig, and D. Song, "Siff: A stateless internet flow filter to mitigate ddos flooding attacks," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 130–143, IEEE, 2004.

[68] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 39–44, 2004.

[69] H.-C. Hsiao, T. H.-J. Kim, S. Yoo, X. Zhang, S. B. Lee, V. Gligor, and A. Perrig, "Stride: sanctuary trail–refuge from internet ddos entrapment," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 415–426, ACM, 2013.

[70] C. Basescu, R. M. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, A. Kubota, and J. Urakawa, "Sibra: Scalable internet bandwidth reservation architecture," *arXiv preprint arXiv:1510.02696*, 2015.

[71] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *2013 IEEE Conference on Communications and Network Security (CNS)*, pp. 260–268, IEEE, 2013.

[72] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "Nethide: secure and practical network topology obfuscation," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 693–709, 2018.

[73] F. Gillani, E. Al-Shaer, S. Lo, Q. Duan, M. Ammar, and E. Zegura, "Agile virtualized infrastructure to proactively defend against cyber attacks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 729–737, IEEE, 2015.

[74] J. M. Smith and M. Schuchard, "Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 599–617, IEEE, 2018.

[75] M. Tran, M. S. Kang, H.-C. Hsiao, W.-H. Chiang, S.-P. Tung, and Y.-S. Wang, "On the feasibility of rerouting-based ddos defenses," in *To appear in Proceedings of IEEE Symposium on Security and Privacy (IEEE S&P)*, 2019.

[76] M. E. Snyder, R. Sundaram, and M. Thakur, "A game-theoretic framework for bandwidth attacks and statistical defenses," in *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, pp. 556–566, IEEE, 2007.

[77] Q. Wu, S. Shiva, S. Roy, C. Ellis, and V. Datla, "On modeling and simulation of game theory-based defense mechanisms against dos and ddos attacks," in *Proceedings of the 2010 spring simulation multiconference*, p. 159, Society for Computer Simulation International, 2010.

[78] H. S. Bedi, S. Roy, and S. Shiva, "Game theory-based defense mechanisms against ddos attacks on tcp/tcp-friendly flows," in *2011 IEEE symposium on computational intelligence in cyber security (CICS)*, pp. 129–136, IEEE, 2011.

[79] J. Xu and W. Lee, "Sustaining availability of web services under distributed denial of service attacks," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 195–208, 2003.

[80] X. Gao and Y.-F. Zhu, "Ddos defense mechanism analysis based on signaling game model," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, pp. 414–417, IEEE, 2013.

[81] M. V. De Assis, A. H. Hamamoto, T. Abrão, and M. L. Proença, "A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on sdn networks," *IEEE Access*, vol. 5, pp. 9485–9496, 2017.

[82] P. Liu, W. Zang, and M. Yu, "Incentive-based modeling and inference of attacker intent, objectives, and strategies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 1, pp. 78–118, 2005.

[83] G. Yan, R. Lee, A. Kent, and D. Wolpert, "Towards a bayesian network game framework for evaluating ddos attacks and defense," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 553–566, ACM, 2012.

[84] Y. Chen and K. Hwang, "Tcp flow analysis for defense against shrew ddos attacks," in *IEEE International Conference on Communications*, pp. 1–8, 2007.

[85] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman, "Moving target defense against ddos attacks: An empirical game-theoretic analysis," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, pp. 93–104, 2016.

[86] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 817–832, 2015.

[87] National Institue of Standards and Technology, *Framework for Improving Critical Infrastructure Security*. https:https://www.nist.gov/sites/.

[88] *Symantec Security Center*. https://www.symantec.com/security-center.

[89] A. Dutta and E. Al-Shaer, ""what","where", and "why" cybersecurity controls to enforce for optimal risk mitigation," in *2019 IEEE Conference on Communications and Network Security (CNS)*, pp. 160–168, IEEE, 2019.

[90] A. Dutta and E. Al-Shaer, "Cyber defense matrix: a new model for optimal composition of cybersecurity controls to construct resilient risk mitigation," in *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pp. 1–2, 2019.

[91] *OWASP Cyber Defense Matrix*. https://www.owasp.org/index.php/OWASP$_{C}yber_{D}efense_{M}a$

[92] S. Yu, *Cyber Defense Matrix*. https://cyberdefensematrix.com/.

[93] *The Z3 Theorem Prover*. https://github.com/Z3Prover/z3.

[94] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 103–115, 2017.

[95] *OWASP Benchmark Project*. https://www.owasp.org/index.php/Benchmark#tab=Main.

[96] Passmark Software, "2018 Consumer Security Products Performance Benchmarks," 2017.

[97] "Common Vulnerabilities and Exposures (CVE)," 2017.

[98] Mitre, *CVE Details*.

[99] R. A. Martin and S. Barnum, "Common weakness enumeration (cwe) status update," *ACM SIGAda Ada Letters*, vol. 28, no. 1, pp. 88–91, 2008.

[100] A. Dutta, *CyberARM*. Center for Cybersecurity Analytics and Automation. http://54.197.4.170/home/.

[101] "Corero ddos trends report," 2017.

[102] A. Kulkarni and S. Bush, "Detecting distributed denial-of-service attacks using kolmogorov complexity metrics," *Journal of Network and Systems Management*, vol. 14, no. 1, pp. 69–80, 2006.

[103] A. Navaz, V. Sangeetha, and C. Prabhadevi, "Entropy based anomaly detection system to prevent ddos attacks in cloud," *arXiv preprint arXiv:1308.6745*, 2013.

[104] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, 2002.

[105] A. Dutta, E. Al-Shaer, and B.-T. B. Chu, "A collaborative & distributed framework for defending distributed denial of service (ddos) attack," in *Proceedings of the 16th Annual Symposium on Information Assurance (ASIA '21)*, pp. 62–72, 2021.

[106] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 41–54, 2004.

[107] V. Sekar, N. G. Duffield, O. Spatscheck, J. E. van der Merwe, and H. Zhang, "Lads: Large-scale automated ddos detection system.," in *USENIX Annual Technical Conference, General Track*, pp. 171–184, 2006.

[108] S. Ramanathan, J. Mirkovic, M. Yu, and Y. Zhang, "Senss against volumetric ddos attacks," in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 266–277, ACM, 2018.

[109] D. Gong, M. Tran, S. Shinde, H. Jin, V. Sekar, P. Saxena, and M. S. Kang, "Practical verifiable in-network filtering for ddos defense," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1161–1174, IEEE, 2019.

[110] "Isps can help their customers defend against ddos attacks." https://www.corero.com/blog/isps-can-help-their-customers-defend-against-ddos-attacks/.

[111] S. B. Lee, M. S. Kang, and V. D. Gligor, "Codef: collaborative defense against large-scale link-flooding attacks," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pp. 417–428, ACM, 2013.

[112] Q. Duan, E. Al-Shaer, S. Chatterjee, M. Halappanavar, and C. Oehmen, "Proactive routing mutation against stealthy distributed denial of service attacks: metrics, modeling, and analysis," *The Journal of Defense Modeling and Simulation*, vol. 15, no. 2, pp. 219–230, 2018.

[113] S. Halabi, *Internet routing architectures*. Pearson Education India, 2008.

[114] "Routing information base info model." https://tools.ietf.org/id/draft-ietf-i2rs-rib-info-model-17.html.

[115] H. Wu, S. Schwab, and R. L. Peckham, "Signature based network intrusion detection system and method," Sept. 9 2008. US Patent 7,424,744.

[116] P. Kaur, M. Kumar, and A. Bhandari, "A review of detection approaches for distributed denial of service attacks," *Systems Science & Control Engineering*, vol. 5, no. 1, pp. 301–320, 2017.

[117] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *2006 IEEE International Conference on Communications*, vol. 5, pp. 2388–2393, IEEE, 2006.

[118] T. Yatagai, T. Isohara, and I. Sasase, "Detection of http-get flood attack based on analysis of page access behavior," in *2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 232–235, IEEE, 2007.

[119] "The curse of dimensionality." https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e.

[120] D. Hu, P. Hong, and Y. Chen, "Fadm: Ddos flooding attack detection and mitigation system in software-defined networking," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[121] "An open source network security monitoring tool." https://https://zeek.org/.

[122] "Snort." https://www.snort.org//.

[123] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, 2011.

[124] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 2, pp. 5–11, 2016.

[125] "Mirai scanner." http://data.netlab.360.com/mirai-scanner/.

[126] M. Thomas and A. Mohaisen, "Kindred domains: detecting and clustering botnet domains using dns traffic," in *Proceedings of the 23rd International Conference on World Wide Web*, pp. 707–712, 2014.

[127] CISCO, *Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting*. https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html.

[128] A. Hassidim, D. Raz, M. Segalov, and A. Shaqed, "Network utilization: The flow view," in *2013 Proceedings IEEE INFOCOM*, pp. 1429–1437, IEEE, 2013.

[129] S.-H. Chung, D. Agrawal, M.-S. Kim, J. W. Hong, and K. Park, "Analysis of bursty packet loss characteristics on underutilized links using snmp," *IEEE/IFIP E2EMON*, 2004.

[130] R. Tipireddy, S. Chatterjee, P. Paulson, M. Oster, and M. Halappanavar, "Agent-centric approach for cybersecurity decision-support with partial observability," in *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, IEEE, 2017.

[131] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[132] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *AAAI*, vol. 94, pp. 1023–1028, 1994.

[133] E. A. Hansen, "An improved policy iteration algorithm for partially observable mdps," in *Advances in Neural Information Processing Systems*, pp. 1015–1021, 1998.

[134] R. Selten, "Bounded rationality," *Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für die gesamte Staatswissenschaft*, vol. 146, no. 4, pp. 649–658, 1990.

[135] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[136] Mazebolt, *Why Your DDoS Mitigation Just Failed?* https://blog.mazebolt.com/why-ddos-mitigation-fails.

[137] *Understanding LSTM Networks.* https://colah.github.io/posts/2015-08-Understanding-LSTMs//.

[138] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[139] "Keras dense." https://www.tensorflow.org/api$_d ocs/python/tf/keras/layers/Dense$.

[140] "Softmax regression." https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html.

[141] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[142] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[143] "Rectified linear unit." https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[144] A. Juels, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proc. Networks and Distributed System Security Symposium (NDSS), 1999*, 1999.

[145] CISCO, *Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting.* https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.htmlpolicingvsshaping.

[146] CISCO, *Configuring Committed Access Rate.* https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos$_c lassn/configuration/15-mt/qos-classn-15-mt-book/qos-classn-car.pdf$.

[147] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional ip routing protocols," *IEEE communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[148] J. M. Smith, K. Birkeland, T. McDaniel, and M. Schuchard, "Withdrawing the bgp re-routing curtain," NDSS, 2020.

[149] CISCO, *Policy Based Routing.* https://www.cisco.com/c/en/us/td/docs/security/asa/asa94/config-guides/cli/general/asa-94-general-config/route-policy-based.pdf.

[150] "Keras: The python deep learning library." https://keras.io/.

[151] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[152] "Zmdp software for pomdp and mdp planning." https://github.com/trey0/zmdp.

[153] "Mirai scanner." http://data.netlab.360.com/mirai-scanner/.

[154] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, pp. 1093–1110, 2017.

[155] "Ip address lookup." https://www.whatismyip.com/ip-address-lookup/.

[156] "Ddos evaluation dataset."

[157] "Aposemat iot-23: A labeled dataset with malicious and benign iot network traffic." https://www.stratosphereips.org/blog/2020/1/22/aposemat-iot-23-a-labeled-dataset-with-malicious-and-benign-iot-network-traffic.

[158] "The bot-iot dataset." https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot$_{i}ot.php$.

[159] "Darpa_2009_ddos_attack-20091105." https://www.impactcybertrust.org/dataset$_{v}iew?idD$
742.

[160] *Vocabulary for Event Recording and Incident Sharing (VERIS)*. http://veriscommunity.net/index.html.

APPENDIX A: Background Knowledge

A.1 Algorithm For Attacker's Decision-Making

Algorithm 8: Attack Strategy

Input : Bot Fail Rate l_b, Delayed Response Rate d_b, Weight Stealthiness w_s, Weight Aggressiveness w_v.

1 attackerBelief q_V = [0,0,0]
2 $q_V[0] = (1 - l_b)d_b$ #0 defines Success
3 $q_V[1] = l_b(1 - d_b)$ #1 defines Failure
4 $q_V[2] = 1 - sum(q_V)$ #2 for Minor Success
5 Normalize q_V
6 $r_s = r_e = 0$
7 $h_b = \frac{1}{r_s}$ #Bot Failed Threshold
8 **for** *i in range(maxStealthiness)* **do**
9 | **if** $l_b \leq h_b$ **then**
10 | | $r_s += 1$
11 | $h_b += \frac{1}{r_s}$
12 $h_d = \frac{1}{r_v}$ #Delayed Response Threshold
13 **for** *i in range(maxAggresiveness)* **do**
14 | **if** $d_b \leq h_d$ **then**
15 | | $r_v += 1$
16 | $h_d += \frac{1}{r_v}$
17 # *candA* is the set of candiate action list
18 $candA$ = [at **for** at in attack space V **if** $at.Y \geq r_s$ or $at.E \geq r_v$]
19 $cAt = maxImpact = None$
20 **for** *at in candA* **do**
21 | $impact = r(at|q_V)(w_s at.Y + w_a t.E) + \sqrt{\frac{2lnn}{n_{at}}}$
22 | **if** *attack is None* ***or*** $maxImpact > impact$ **then**
23 | | $cAt = at, \quad maxImpact = impact$
24 return *cAt*

In Alg. 8, l_b defines the ratio of bots that could not transmit a minimum traffic (e.g., blocked, limited), and d_b defines the ratio of packet responses that arrived late (i.e., dependent on link utilization). Based on these properties, he computes belief at line 1-5. At line 8-11, he determines the required stealthiness r_s of next attack action, and he determines required aggressiveness (attack volume expansion) r_e at line 12-16. At line 18, he creates candidate attack set *candA* whose stealthiness ($Y + n_i$) is greater

or equal to r_s or aggressiveness $(E + n_j)$ is greater or equal to r_v. Notably, $at.Y$ and $at.E$ defines the attack stealthiness and expansion respectively for a particular attack action at.

In line 20-23, the attacker chooses the action cAt that maximizes his current objective. In line 21, $r(at|q_V)$ is the achieved reward by a_t previously when his belief was q_V. The first term in the line defines the expected reward of at based on previous experience (Exploitation), whereas the second term $(\sqrt{\frac{2ln(n)}{n_{at}}})$ gives weight to unexplored actions (Exploration) with the guarantee of logarithmic regret [131]. Here, n is the number of executed attack actions, and n_{at} is the number of times when at was chosen. Finally, this algorithm returns the optimal attack action for current time.