12-2-2019

Cybersecurity Games: Mathematical Approaches for Cyber Attack and Defense Modeling

Mohammad Hadi Valizadeh
University of Connecticut, mohammad.valizadeh@uconn.edu

Cybersecurity Games: Mathematical Approaches for Cyber Attack and Defense Modeling

Mohammad H. Valizadeh, Ph.D.

University of Connecticut, 2019

ABSTRACT

Cyber-attacks targeting individuals and enterprises have become a predominant part of the computer/information age. Such attacks are becoming more sophisticated and prevalent on a day-to-day basis. The exponential growth of cyber plays and cyber players necessitate the inauguration of new methods and research for better understanding the "cyber kill chain," particularly with the rise of advanced and novel malware and the extraordinary growth in the population of Internet residents, especially connected Internet of Things (IoT) devices.

Mathematical modeling could be used to represent real-world cyber-attack situations. Such models play a beneficial role when it comes to the *secure* design and evaluation of systems/infrastructures by providing a better understanding of the threat itself and the attacker's conduct during the lifetime of a cyber attack. Therefore, the main goal of this dissertation is to construct a proper theoretical framework to be able to model and thus evaluate the defensive strategies/technologies' effectiveness from a security standpoint.

To this end, we first present a Markov-based general framework to model the interactions between the two famous players of (network) security games, i.e., a system defender and an attacker taking actions to reach its attack objective(s) in the game. We mainly focus on the most significant and tangible aspects of sophisticated cyber attacks: (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objective(s) by translating attacker-defender interactions into well-defined games and providing rigorous cryptographic security guarantees for a system given both players' tactics and strategies.

We study various attack-defense scenarios, including Moving Target Defense (MTD) strategies, multi-stage attacks, and Advanced Persistent Threats (APT). We provide general theorems about how the probability of a successful adversary defeating a defender's strategy is related to the amount of time (or any measure of cost) spent by the adversary in such scenarios. We also introduce the notion of learning in cyber-security games and describe a general "game of consequences" meaning that each player's chances of making a progressive move in the game depend on its previous actions.

Finally, we walk through a malware propagation and botnet construction game in which we investigate the importance of defense systems' learning rates to fight against the self-propagating class of malware such as worms and bots. We introduce a new propagation modeling and containment strategy called the "learning-based model" and study the containment criterion for the propagation of the malware based on theoretical and simulation analysis.

Cybersecurity Games: Mathematical Approaches for Cyber Attack and Defense Modeling

Mohammad H. Valizadeh

M.S., University of Connecticut

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2019

APPROVAL PAGE

Doctor of Philosophy Dissertation

Cybersecurity Games: Mathematical Approaches for Cyber Attack and Defense Modeling

Presented by

Mohammad H. Valizadeh, M.S.

Major Advisor	Marten van Dijk
Associate Advisor	Benjamin Fuller
Associate Advisor	Walter O. Krawec

University of Connecticut

2019

DEDICATION

To Fatima who is the beacon of light in the darkness of my world.

ACKNOWLEDGMENTS

As the saying goes, "sometimes science is so difficult it makes me sad."[1] However, I was blessed[2] to be surrounded by people who made the science less "difficult," and therefore, made me less "sad." Here is a non-exhaustive list of people who helped me in that sense, and I would like to express my appreciation to them.

I want to start with my advisor, Prof. Marten van Dijk, who was more like a fantastic friend and a great mentor to me rather than merely an academic advisor. Thanks for your continuous support of mine from both research and personal perspectives. I would also like to thank my cordial and thoughtful committee members Dr. Benjamin Fuller and Dr. Walter Krawec. Thank you both for letting my defense be a memorable experience, and thanks for your support, your encouragement, and your insightful comments! My express appreciation goes to my best friend and beloved wife, Fatima. Thanks for your unparalleled love and support. I genuinely believe that you are an angel on Earth. I want to thank my family members as well, including my parents, brother, and sister, for their spiritual and emotional support. You all have been always there for me no matter what. Love you so much. Last but not least, my special thanks go to all my friends and also my fellow labmates during my stay at UConn. The past four years were fun, enjoyable, and memorable all because of you. What an adventure this has been. Thanks all.

[1]I prefer the word "confused" instead of "sad," and it is truly one of the most pleasant feelings one can experience in life, "the scientific confusion/amusement."

[2]My preferred name, "Saeed," literally means "blessed" in Persian!

Contents

Chapter 1

Introduction

Cyber-attacks targeting individuals or enterprises have become a predominant part of the computer/information age. Such attacks are becoming more sophisticated (qualitative aspects) and prevalent (quantitative aspects) on a day-to-day basis [18]. The exponential growth of cyber plays and cyber players necessitate the inauguration of new methods and research for better understanding the "cyber kill chain," particularly with the rise of advanced and novel malware (e.g., Stuxnet [35], WannaCry ransomware crypto worm [18], the Mirai and its variants [6]) and the extraordinary growth in the population of Internet residents, especially connected Internet of Things (IoT) devices.

Mathematical models can represent real-world cyber-attack situations. Such models can play a beneficial role when it comes to the design and evaluation of systems/infrastructures, especially from a *security* perspective. Utilizing mathematical modeling, we can have a better understanding of system vulnerabilities, attack vectors, points of failure, and in general, the life cycle of a cyber attack. Moreover, one can analyze various security defense mechanisms/strategies, provide rigorous security guarantees for a system, and perform a cost-benefit analysis by finding the best attack-response strategies in different attack scenarios. Besides, we can define what it means for a system/infrastructure/strategy to be *secure* using mathematical approaches.

We believe that measuring the security of a system, or in other words, how much *security capacity* a system has by design, is an interesting problem that requires further investigation as it can provide a better understanding of the system under the attack, and the actions that must be taken by a defender protecting the system. Through this presented dissertation (especially in Chapters 2-3), we show how one can *measure* security and study under what circumstances (e.g., defense strategies, attacker's (time) budget) a system can be called *secure* by defining the notion of "security capacity," and "capacity region" of a system and relating the most significant and tangible aspects of sophisticated cyber attacks, i.e., the amount of time it takes for the adversary to accomplish its mission and the success probabilities of fulfilling the attack objective(s). The presented *worst-case* analysis is beneficial when it comes to bootstrapping cryptographic protocols and having a proper reduction to common assumptions such as the rate of play or the learning rate of a defender. We also present an *average-case* analysis of a particular cyber-attack scenario, namely a malware propagation and botnet construction game, which shows how a robust learning mechanism from the defender's side can immensely restrict the adversary's progress in the game.

1.1 Motivation

Overall, mathematical modeling can help the security community to understand the threat better, to analyze the attacker's conduct during the lifetime of a cyber-attack, and, therefore, to be able to provide authentic responses to adversarial actions in different phases of an attack. The sparse amount of research on modeling and evaluating defensive systems' efficiency (especially from a *security* perspective), however, warrants the need for constructing a proper theoretical framework. Such a framework allows the community to be able to evaluate the defensive strategies/technologies' effectiveness from a *security* standpoint, and this is the primary objective of this dissertation, i.e., to utilize mathematical approaches to model and study various cyber scenarios prevalent today.

1.2 Outline of the Dissertation

We start this dissertation with the main focus on Moving Target Defense (MTD) games as the state of the art defense strategy. MTD seems to be a promising defense mechanism in that it makes fulfilling the attack objective more difficult for an adversary by changing the attack surface dynamically. To get a better understanding of MTD's effectiveness, we propose a general Markov-model-based framework for Moving Target Defense analysis. The presented framework allows modeling of a broad range of MTD strategies. We further provide general theorems about how the probability of a successful adversary defeating an MTD strategy is related to the amount of time or cost spent by the adversary and show how a multi-level composition of MTD strategies can be analyzed by a straightforward combination of the analysis for each one of these strategies. Within the proposed framework, we define the concept of *security capacity*, which measures the strength and effectiveness of an MTD strategy. The security capacity depends on MTD specific parameters and general system parameters. We apply our framework to two concrete MTD examples.

We then focus on a more general game to extend our analysis and presented language to a broader range of cyber attack and defense scenarios, especially multi-stage and APT-like attacks. Using a Markov-based general framework, we model the interactions between the two famous players of network security games, i.e., a defender (taking advantage of common security tools and technologies such as Intrusion Detection and Prevention Systems (IDPSes), Firewalls, and Honeypots (HPs)) and an attacker (and possibly its agents) who takes actions to reach its attack objective(s) in the game. Again, we mainly focus on the most significant and tangible aspects of sophisticated cyber-attacks: (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objectives. Therefore, our goal is to translate attacker-defender interactions into a well-defined game so that we can provide rigorous cryptographic security guarantees for a system given both players' tactics and strategies. In this game, to have a more comprehensive and realistic model, we consider an intelligent defender (in contrast to a 'dummy' one) where it can learn regarding the adversarial actions and strategies. As time elapses, the defender's learning process can potentially lead to smaller chances of making a progressive move for the attacker. We provide general theorems about how the learning rate of the defender plays a role in the adversary's success probability of winning the introduced game. Within this framework, we can extend the previously defined notion of "security capacity" to a *capacity region* concept where we relate attack success probabilities to the time or cost spent by the attacker. The presented *worst-case* analysis in this chapter is beneficial when it comes to bootstrapping cryptographic protocols and having a proper reduction to common assumptions such as the learning rate of a defender.

As the last part of this dissertation, we walk through a malware propagation and botnet construction game in which we investigate the importance of defense systems' learning rates to fight against the self-propagating class of malware such as worms and bots. To this end, we introduce a new propagation modeling and containment strategy called the "learning-based model" based on the interactions between an adversary (and its agents) who wishes to construct a zombie army of a specific size, and a defender taking advantage of standard security tools and technologies in the network environment. We present an *average-case* analysis which is valuable from practical perspective for understanding how the adversary advances in its mission (without being able to bootstrap cryptographic protocols based on this analysis).

1.3 Organization and Summary of Contributions

This dissertation is organized as follows: In Chapter 2, we propose a Markov-model-based framework for Moving Target Defense analysis. The presented framework allows modeling of a broad range of MTD strategies. We further provide general theorems about how the probability of a successful adversary defeating an MTD strategy is related to the amount of time or cost spent by the adversary and show how a multi-level composition of MTD strategies can be analyzed by a straightforward combination of the analysis for each one of these strategies. Within the proposed framework, we define the concept of security capacity, which measures the strength and effectiveness of an MTD strategy. The security capacity depends on MTD specific parameters and more general system parameters. We apply our framework to two concrete MTD examples. Our contributions and results can be summarized as follows:

- We present a formal definition of MTD as the interaction between a defender and an attacker modeled by probabilistic algorithms and we show that such an interaction corresponds to a defender-attacker game characterized by a Markov chain. Our framework is general in that it formally defines (and can be used to analyze) a large class of MTD strategies as games based on Markov chains.

- We prove a first theorem analyzing the number of time steps needed for an adversary to "win" an MTD game. We measure the effectiveness of an MTD strategy by analyzing *when the attacker will be in a winning state for the first time*. Additional analysis of *how long an attacker will be in a winning state once it arrives at a winning state* and *when the attacker will again be in a winning state once it exits a winning state* is required to understand the efficacy of an MTD strategy in more detail. Since reaching a winning state allows an adversary to launch a successful attack, we think that *when the attacker will be in a winning state for the first time* is the most crucial question to analyze first. Based on this theorem, we introduce the concept of *security capacity* as a measure of the effectiveness of an MTD strategy; an MTD strategy with security capacity c has the property that, for all s and t such that $c = s + t$, the probability that the adversary wins the MTD game within 2^t time steps is at most 2^{-s}. We show that time steps can be converted into any measure of cost for the adversary.

- We further analyze two concrete examples of MTD schemes (IP hopping MTD mechanism) to demonstrate the applicability of our framework beyond a mere theoretical exercise.

- Motivated by the idea of Defense in Depth (DiD), we notice that a single MTD may not be beneficial in that it provides a small security capacity. Therefore, it may be possible to obtain a large security capacity, only if multiple MTDs are deployed together. We call this approach a "Layered MTD" or "MTD^*." In this regard, we expand our framework by introducing compositions of MTD games for which we extend our theorem and show how one can analyze a multi-layer MTD game.

In Chapter 3, we present a Markov-based framework and methodology for modeling various computer security scenarios prevailing today, including opportunistic, targeted and multi-stage attacks. Motivated by the rise of Advanced Persistent Threats (APT), we are particularly interested in situations in which each players' progress in the game can be viewed and modeled as an incremental process. From the viewpoint of the attacker, a progressive adversarial move associated with a probability distribution can bring it one step closer to its desired winning state (i.e., the attack objective). Similarly, by monitoring the system and capturing the adversarial moves, the defender can also get one step closer to its objective which could be different based on various attack/defense scenarios. For instance, the defender's goal could be generating a detection signal that indicates the presence of a coordinated set of activities that are part of an APT campaign [50], developing a signature for a malware attack to halt next adversarial moves [55, 85], or shuffling its resources after enough number of samples (evidence) is obtained as in a Moving Target Defense strategy (MTD) [47, 76].

Our contributions and results are summarized as follows:

- First, we propose a general network security game based on the interactions between an adversary, (and possibly its agents, e.g., bots controlled by a botmaster), and a defender who is equipped with a logically centralized defense system implemented in the network. We explain how this game can be directly mapped to an equivalent Markov model and we provide the security analysis for this Markov model as well as presenting compelling interpretations of the role of both players' strategies in their probability of winning the introduced game.

- We introduce the notion of learning in cybersecurity games and describe a general "game of consequences" meaning that each player's (mainly the attacker) chances of making a progressive move in the game depends on its previous actions. More specifically, as a consequence of the adversarial imperfect moves in the game, the other party, i.e., the defender has the opportunity to incrementally learn more and more about the attack technology. This learning enables the defender to be able to seize and halt following adversarial moves in the game, in other words, containing the attackers' progress in the game. We argue that such learning is possible since the actions that need to be taken by an adversary in each cycle of a cyber attack's life inevitably entails abnormal and suspicious events and activities on both host and network levels.

- Unlike game-theoretic methods which commonly focus on finding the best attack-response strategies for the players, we, however, with a cryptographic mindset, mainly focus on the most significant and tangible aspects of sophisticated cyber attacks: (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objectives.

Therefore, our goal is to translate attacker-defender interactions into a well-defined game so that we can provide rigorous cryptographic security guarantees for a system given both players' tactics and strategies. We generalize the basic notion of computational security of a cryptographic scheme [45] to a *system/infrastructure* which must be defended and introduce a *computationally secure system* in which a "given *system* is (T_{bud}, k, ϵ)-*secure* (for $T_{bud}, k > 0$, and $\epsilon \leq 1$), if any adversary limited with a (time) budget at most T_{bud} succeeds in reaching the attack objective k, with probability at most ϵ." We study under which circumstances the defense system can provide an effective response that makes $\epsilon = negl(k)poly(T_{bud})$.

- By modeling the learning rate of the defender as a function $f(l)$ (which is the probability of detecting and halting a next adversarial move) where l represents the number of adversarial moves and actions so far observed and collected, we present the following results:
  - If $f(l)$ does not converge to 1, then we call such a learning rate stagnating, and the adversary can reach its winning state (attack objective) k within $poly(k)$ time budget.
  - If $f(l) \rightarrow 1$ and more specifically $1 - f(l) = O(l^{-2})$, then the probability of the adversary winning the game is proven to be negligible in k and polynomial in the attacker's time budget, i.e., $\epsilon = negl(k)poly(T_{bud})$.
  - The above results hold for time budget measured in logical Markov model transitions or physical time (seconds), and also hold for learning rates that remain equal to $f(l) = 0$ until a certain number $l = L^*$ adversarial moves have been collected after which learning starts.

  - The adversary may not reach its desired winning state of k, but will reach a much smaller k_c (called containment parameter) of the order $O(L^* + \log T_{bud})$, where T_{bud} is the attacker's time budget, for $1 - f(l) = O(l^{-2})$.
  - We conclude that at a metalevel the adversary needs to find one attack exploit/vector for which the defender cannot find a fast enough increasing learning rate and the defender needs to have a learning system/methodology in place which should be able to reach fast enough learning rates for any possible attack exploit/vector. In practice, infrastructures are being compromised and this means that effort is needed in order to understand a defense system's associated learning rates and why these are $\leq 1 - \Omega(l^{-a})$ for some 'a' too small or have $L^* = O(k)$ (i.e., learning starts too late).

- The proposed adversarial/defender game in terms of a Markov model is general and fits most practical scenarios as our extensive overview shows. To show how the presented modeling can be applied to various cyber attack and defense scenarios, we walk through two examples and provide the security analysis for them: (1) a malware propagation and botnet construction game (due to the increase in the number of bots found in the wild), and (2) a moving target defense game (as the state of the art defense strategy).

Based on our theoretical analysis and its applicability to practical scenarios, we understand when an adversary with a time budget T_{bud} can be contained and prevented from reaching its attack objective k with probability $\geq 1 - negl(k)poly(T_{bud})$. As a consequence, if such containment is in place, then this allows one to trust cryptographic protocols/systems which assume (for proving their security) attackers that have not reached k (and have a key renewal / refresh operation that can be called every T_{bud} seconds in order to push adversaries back to their initial state). Moreover, we believe that the introduced notion of *learning* is an exciting approach to model cybersecurity games and worth further investigations since it enables one to consider a more intelligent defender instead of a "dummy" one. Besides, through this presented framework, we are able to extend the idea of security capacity introduced in Chapter 2 and provide the definition of a *security capacity region* as a metric for gauging a defensive system's efficiency from a security perspective (by presenting rigorous cryptographic security guarantees in terms of the security capacity region).

In Chapter 4, we study the effectiveness of content-based filtering strategies[1] based on automatic signature generation schemes. Instead of presenting yet another malware detection/prevention methodology, we revisit the spread of malware phenomenon, especially the self-propagating ones such as worms and bots, from a new perspective. We introduce a new model for capturing the interactions of an adversary and its agents with the defensive system during the early phases of a zombie army construction. Unlike a malware propagator who usually follows a passive attack strategy for malware distribution to a broad audience, meaning that the attack technology usually remains the same after unleashing the first few samples of the malware, the defender can incrementally learn regarding the attack technology (specifically, from observed malware payloads). As time elapses, and the defender captures more attack data, it can reach better detection rates and accuracy. This learning rate indeed reflects into higher quality malware signatures which can be used for online malicious traffic filtering and hence malware containment purposes. Our contributions

[1]As one of the most effective and feasible solutions to tackle the spreading of worm-like malware [54].

and results are summarized as follows:

- Motivated by the advances in networking infrastructure and technologies such as the introduction of Software-Defined Networking (SDN), and also the widespread embrace of Automated Machine Learning (AutoML) techniques in different realms of networking and security fields, we introduce a novel malware propagation model called the "learning-based model" suitable for today's technologies and more advanced malware.

- The previous outdated models are mainly focused on the attacker's strategies (specifically target discovery and scanning rates [16, 71, 75]), or in best cases, an ill-defined cleaning of the infection or patching processes [15, 21, 103]. However, we show how the increased knowledge of the defender can be taken into consideration by modeling the learning rate of the defense system as a function $f(l)$ (which is the probability of detecting and filtering a next malicious payload) where l represents the number of malicious payloads for a specific exploit so far collected.

- The model leads to the development of an automatic self-replicating malware containment strategy that prevents the spread of the malware beyond its early stages of propagation.

- We provide precise conditions on the convergence rates of a learning-based signature generation algorithm that determines whether the malware spread will ultimately settle or not.

- We derive tight upper and lower bounds on the total number of hosts the malware propagator can infect.

- The model is general enough and enables us to evaluate the effectiveness of learning-based signature generation schemes and containment strategies. It is especially suitable for "Cloud environments" where the defender (in this case the Cloud service provider) has more control, and better visibility over its network/infrastracture and the environment exposes more homogeneity by having high-level similarities in the different layers of the software stack (e.g., hypervisors, OS).

- We study learning functions of the form $1 - f(l) = (\frac{A}{A+l})^{\alpha}$ in which A is considered to slow down the learning process, and α is the amplification factor in the learning. This enables us to capture characteristics of both yesteryears malware (e.g., a monomorphic worm) and also today's more advanced ones such as polymorphic malware.

- Our numerical analysis and simulation results show that regardless of attacker's scanning rates, with a proper learning function that converges fast enough to 1, the propagation will be contained and only a negligible number of susceptible population will be infected.

- The attacker needs to find just one exploit for which the defender is too slow to react or too slow in learning. Instead of focusing on higher scan rates, the attacker must invest in more stealthier and robust target discovery schemes and malware delivery techniques which would not raise suspicions, and provides fewer attack samples to the defense system.

We conclude and summarize this dissertation in Chapter 5.

1.4 Publications

Note: The preferred name "Saeed" is used on the publications.

Conference papers that are accepted and published with *primary authorship* include [47, 84]:

1. **S. Valizadeh**, and M. van Dijk, "MalPro: A Learning-based Malware Propagation and Containment Modeling," In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop (pp. 45-56). ACM.*

   The e-print version of this document is available at:

2. **S. Valizadeh** and M. van Dijk, "On the Convergence Rates of Learning-based Signature Generation Schemes to Contain Self-propagating Malware," *arXiv preprint arXiv:1905.00154 (2019).*

3. H. Maleki, **S. Valizadeh**,[2] W. Koch, A. Bestavros, and M. van Dijk, "Markov Modeling of Moving Target Defense Games," In *Proceedings of the 2016 ACM Workshop on Moving Target Defense (pp. 81-92). ACM.*

Conference papers that are accepted and published with *co-authorship* include [35]:

1. C. Jin, **S. Valizadeh**, and M. van Dijk, "Snapshotter: Lightweight Intrusion Detection and Prevention System for Industrial Control Systems," In *2018 IEEE Industrial Cyber-Physical Systems (ICPS) (pp. 824-829). IEEE.*

Currently submitted and under review conference papers with *primary authorship* include [86]:

[2] Corresponding author. The first two authors are in alphabetical order and share lead authorship.

1. **S. Valizadeh** and M. van Dijk, “Tick Tock: a Tale of Advanced Persistent Threats,” *The 5th IEEE European Symposium on Security and Privacy 2020 (EuroS&P)*, 2020.

   The preprint of this work is available at:

2. **S. Valizadeh** and M. van Dijk, “Toward a Theory of Cyber Attacks based on Markov Models,” *arXiv preprint arXiv:1901.01598 (2019).*

Chapter 2

Markov Modeling of Moving Target Defense Games

2.1 Introduction

In spite of the fact that today's computer systems are designed and implemented with security in mind and take advantage of secure services, security primitives and mechanisms, and crypto protocols, the history of computer systems has patently revealed that perfect security is unattainable [1], i.e., all systems have vulnerabilities that can be exploited to launch successful attacks.

For a static system in which configuration parameters and system settings remain relatively unchanged over relatively long periods, attackers have enough time to infer system's configurations, vulnerabilities, and the corresponding attack vectors which can lead to the attacker's success in compromising the system in any number of ways [2]. In order to mitigate such potential risks, Moving Target Defense (MTD) has been introduced as the notion of controlling change across multiple system dimensions to intensify uncertainty and complexity for attackers, by reducing their window of opportunity and increasing the costs of their probes and attack efforts [60]. Research in MTD focuses on enabling the continued safe operation of a system by dynamically hiding vulnerabilities or attack vectors (which we call targets in our framework).

To date, various MTD schemes have been introduced as a proactive defensive solution to make a system less stationary and less deterministic. In general terms, these techniques can be categorized into five different domains based on their placement within the execution stack [62]: (1) *Dynamic Data* which mostly involves changing data representation or its format [10, 17, 57], (2) *Dynamic Software* which includes changing the application code [70, 72], (3) *Dynamic Runtime Environment* which is about altering the execution environment [36, 63, 77], (4) *Dynamic Platform* which deals with modifying platform properties [20,29,73] and finally (5) *Dynamic Network* which is about changing network configurations and properties [5, 34]. A thorough study of most of these methods, including their advantages and weaknesses, can be found in [61].

Our Approach. Even though the use of MTD seems to be a promising defense strategy and many efforts have been made to introduce different practical MTD schemes, most research only focuses on introducing a new MTD scheme in one of the five categories mentioned above and then assessing the effectiveness of the introduced method using a simulation-based approach. However, there has been very few research that shows the effectiveness of these methods from a theoretical standpoint, especially when multiple MTD schemes at different layers of the software stack are combined.

Therefore, in the first part of this dissertation, we introduce a Markov-based approach for modeling different MTD schemes/strategies. Then, we show how the introduced framework can be used to gauge the effectiveness of any given MTD scheme/strategy.

Contributions, Results & Organization. Our contributions are summarized as follows:

- In Section 2.3.1, we define MTD as the interaction between a defender and an attacker modeled by probabilistic algorithms and show that such an interaction corresponds to a defender-attacker game characterized by a Markov chain. Our framework is general in that it formally defines (and can be used to analyze) a large class of MTD strategies as games based on Markov chains. We walk through two concrete examples of MTD games in Section 2.3.2, and later we provide their analysis to demonstrate the applicability of our framework beyond a mere theoretical exercise.

- Section 2.4 proves a first theorem analyzing the number of time steps needed for an adversary to "win" an MTD game. Based on this theorem we introduce *security capacity* as a measure of the effectiveness of an MTD strategy; an MTD strategy with security capacity c has the property that, for all s and t such that $c = s + t$, the probability that the adversary wins the MTD game within 2^t time steps is at most 2^{-s}. We show that time steps can be converted into any measure of cost for the adversary.

- Although a single MTD may not be very effective in that it has a small security capacity, it may be possible to obtain a large security capacity if multiple MTDs are deployed together. We call this approach a "Layered MTD" or "MTD^*." In this regard, Section 2.4.3 is an expansion of our framework by introducing compositions of MTD games for which we extend our theorem.

2.2 Background and Related Work

A central theme in MTD research is first to present a new feasible technique and then estimate its effectiveness. In this regard, many MTD techniques have been proposed, e.g., Address Space Layout Randomization (ASLR) [36], Software Diversity [59, 66], Dynamic Data [4, 57], etc. Existing methods for estimating the effectiveness of such techniques are mainly divided in two categories: a) simulation based [77, 79, 94, 101] and b) based on mathematical models [24, 28, 33, 99].

Simulation-based: Two examples of simulation-based methods for evaluating MTD's effectiveness are [94, 101]. In [94], the authors introduce a quantitative framework for evaluating the effectiveness of MTD strategies (in the network layer) using simulation experiments. Because of the limited scope of experimental conditions, it is difficult to compare the effectiveness of an MTD strategy for different system settings. Zhuang *et al.* [101] present a preliminary design schema of MTD in enterprise networks. The authors investigate an adversary's chance of success in a network with dynamic configurations and parameters using a simulation-based approach. One of the problems with simulation-based MTD evaluation is that these techniques are case-dependent meaning that they only stand for their specific introduced example.

Based on mathematical models: The work described in [24] considers the effectiveness of "Diversity defenses" (including Address Space, Instruction Set, and Data Randomization) as an MTD method. The authors present a model for analyzing the effect of MTD, focusing on scenarios where only the undefined semantics of the underlying programming language is changed. The study is worthwhile as the authors show that the effectiveness of MTD depends on multiple aspects: the type of execution properties that are changed, address randomization entropy, etc. However, the study does not lead to a generally applicable framework.

Of specific interest to us are game theoretic models that assist in the design and analysis of MTD mechanisms [58]. They can be used to maximize the defender's "security" while minimizing/constraining the "diversity rate," defined as the time interval between each system adaptation (in our terminology, we use λ which is the rate at which the defender moves in its MTD strategy), to a usable setting (i.e., to a setting giving acceptable performance overhead): Lye and Wing [46] use a two-player general-sum stochastic game to model the interaction between an attacker and a defender (system administrator) for analyzing the security of computer networks. Manadhata [48] looks into the MTD problem as a game of perfect and complete information for attack surface shifting. Zhu *et al.* [96] propose a game theoretical framework for multi-layer attack surface shifting with an adaptive defender who changes defense strategy based on real-time feedback information and risk analysis. Carter *et al.* [12] study dynamic platform MTD through a game theoretical approach for modeling an attacker who can monitor the defender's moves in the game.

Rather than using game theory, we propose a general framework where the moves of the attacker and defender are represented in a Markov model allowing an immediate and straightforward characterization of the effectiveness of the used MTD strategy by the defender (with respect to the modeled attacker).

Zhuang [97, 100] made a first attempt using Markov models to analyze MTD strategies; each state in the proposed Markov model represents the complete configuration of the whole system which means that the number of states in the Markov model grows exponentially in the number of machines in the system. This attempt being too complex, Zhuang [97, 99] considers an attack graph as a Markov model where transitions are labeled with a probability density and where nodes keep track of the machine's configuration together with necessary overall system parameters of which the probability densities are a function. Nodes can represent physical machines, software stacks, or software objects such as Virtual Machines (VMs). A transition from one node to another node means that the adversary extends its footprint to the other node. MTD can now be modeled as being part of the overall system parameters. From this attack graph perspective, precise formulas can be derived with respect to the time needed for an adversary reaching its objective as a function of these probability density functions, which in turn are functions of the MTD and system parameters [97, 98]. Such formulas cannot become concrete as the whole system architecture with MTD and adversarial behavior needs to be modeled – the proposed Markov model requires too much detail. As a result only simple relationships can be proven such as a larger diversity rate λ implies a larger expected time till successful attack, and faster expected transition times imply a smaller expected time till successful attack.

We realize that using Markov models in which each state represents a complete system configuration is useful only if we could compress the Markov model to one where states simply represent the projection of the necessary system and MTD parameters. This can lead to precise formulas and relations which are easy to interpret by plugging in the known parameters.

2.3 Defining MTD as a Two-player Game

2.3.1 MTD Games

First, by expressing MTD as an abstract game between a defender and an attacker, we show how a large class of MTD strategies can be modeled via this representation. After providing an algorithmic definition of MTD games, we explain how this interpretation of players' interactions within the system, naturally translates into an equivalent definition based on a Markov modeling description of MTD games.

Definition 1. *An MTD game is defined by a set of possible defender moves* $\mathcal{D} = \{\epsilon, d_1, d_2, \ldots\}$*, attacker moves* $\mathcal{A} = \{\epsilon, a_1, a_2, \ldots\}$ *(where* ϵ *indicates "no move") and a probabilistic algorithm* $\mathcal{M}$ *which outputs a stream of triples* $\{(D_i, A_i, w_i)\}_{i\geq 1}$ *with* $D_i \subseteq \mathcal{D}$*,* $A_i \subseteq \mathcal{A}$*, and* $w_i \in \{0, 1\}$ *indicating whether the attacker has beaten the defender's moving target strategy; we say the attacker is in a winning state at time step* i *if* $w_i = 1$*.*

The above definition as an interplay of a defender and adversarial strategy is very general: Algorithm $\mathcal{M}$ is probabilistic and uses coin tosses which, together with the current state of $\mathcal{M}$, decides at each time step i which collection of defender and adversarial moves are being played and in what order. The changes result in a change in the state of the algorithm $\mathcal{M}$ from time step to time step. $\mathcal{M}$'s state represents the combination of the adversary's and defender's state together with the state of the system which is attacked/defended. This means that $\mathcal{M}$ also models adaptive strategies.

Algorithm 1 shows how $\mathcal{M}$ can be simulated; *simState* keeps track of the system state φ, the view of the defender $view^D$, the view of the attacker $view^A$, and whether the attacker is in a winning position; $NextState(.,.,.)$ simulates how the moves by the attacker and defender change the system state; the defender and attacker are represented by algorithms D and A where $D.update(.,.)$ and $A.update(.,.)$ keep track of how their views change because of the new system state; notice that even if only one party moves, both views may be affected.

Algorithm 1 MTD Game Simulator

```
1: function SIMULATOR($\mathcal{M}$)
2:     $\varphi$ = initial system state;
3:     $view^A$ = adversary's initial state of the system used in its adversarial strategy;
4:     $view^D$ = defender's initial state of the system used in its MTD strategy;
5:     Initialize $simState = (\varphi, view^D, view^A, 0)$;
6:     while true do
7:         $(moveD, moveA, w) \leftarrow \mathcal{M}(simState)$;
8:         $\varphi = NextState(\varphi, moveD, moveA)$;
9:         $view^D \leftarrow D.update(view^D, \varphi)$;
10:        $(view^A, w) \leftarrow A.update(view^A, \varphi)$;
11:        $simState = (\varphi, view^D, view^A, w)$;
12:        if $w == 1$ then
13:            Return "Attacker is in a winning position";
14:        end if
15:    end while
16: end function
```

We measure the effectiveness of an MTD strategy by analyzing *when the attacker will be in a winning state for the first time.*[1] For this purpose, we do not need to keep track of the exact moves being played by the defender and attacker; we only need to keep track of how the state of $\mathcal{M}$ changes from time step to time step and whether the new state corresponds to a winning state for the adversary. This defines

[1] Additional analysis of *how long an attacker will be in a winning state once it arrives at a winning state* and *when the attacker will again be in a winning state once it exits a winning state* is required to understand the effect of an MTD strategy in more detail. Since reaching a winning state allows an adversary to launch a successful attack, we think that *when the attacker will be in a winning state for the first time* is the most crucial question to analyze first.

a Markov chain[2] where state transitions are a function of coin tosses. This leads to the following definition:

Definition 2. *An M-MTD game is defined by a $(k+1)\times(k+1)$ transition matrix M which describes a Markov chain of state transitions that reflect both defender and attacker moves. Initially the game starts in the state 0. For next time steps, the game transitions from its current state i to a new state j with probability $M_{i,j}$. We say the attacker wins the M-MTD game within T time slots if (the winning) state k has been entered during at least one of the first T time slots of the MTD game.*

Note that state k represents the winning state from the adversary's perspective. In other words, it depicts full control by the adversary; once the adversary reaches state k, it is possible for the attacker to launch an attack that will be successful against the system which the defender attempts to protect using its MTD strategy. From the above discussion we obtain the following lemma.

Lemma 1. *Each MTD game can be described as an M-MTD game for some transition matrix M.*

A subclass of MTD games with a more natural and intuitive correspondence to MTD strategies is given in the next definition:

Definition 3. *A (M^D, λ, M^A, μ)-MTD Game is defined by*

1. *parameters λ and μ satisfying $0 \leq \lambda + \mu \leq 1$ which represent the rate of play of the defender and attacker, respectively.*

[2]We avoid unnecessary state explosion by representing states by system and MTD parameters needed for describing the MTD game between adversary and defender with sufficient precision.

2. *$(k+1) \times (k+1)$ transition matrices M^D and M^A; for $i, j \in \{0, \ldots, k\}$, $M^D_{i,j}$, and $M^A_{i,j}$ respectively represent the probability of transitioning from an state i to an state j when the defender, respectively the attacker, issues a move in the state i.*

Initially the joint defender-attacker state is 0*. At each next time step either with probability λ the defender makes a move according to M^D, or with probability μ the attacker makes a move according to M^A, or with probability $1 - \lambda - \mu$ neither the defender nor the attacker makes a move. We say the attacker wins the MTD game within T time slots if it enters state k during one of its moves in the first T time slots of the MTD game.*

In the above definition, we exclude the possibility of both the defender and attacker playing at the same time: a (three-valued) random coin flip decides with probability λ that the defender can move, with probability μ that the attacker can move, and with probability $1 - \lambda - \mu$ that no one moves.

Notice that a (M^D, λ, M^A, μ)-MTD game is fully described by the transition matrix

$$M = \lambda M^D + \mu M^A + (1 - \lambda - \mu) I_{k+1}, \tag{2.1}$$

where I_{k+1} is the $(k+1) \times (k+1)$ identity matrix. At each time slot the joint defender-attacker state transitions to a new state according to M; M represents the MTD game as a Markov chain as in Definition 2.

A couple questions concerning the limitations of this abstract definitional framework immediately arise: The above definitions relate to a logical discrete time axis where at each fixed time step or "logical clock cycle" the system state changes according to the transition matrix M. How can this logical time be translated to real time (which is what we are interested in) or cost of playing (in some cases we may be able to assign a cost to adversarial moves)? The above definitions are very general and may lead to very complex huge Markov chains in order to model real systems. How much detail is needed in order to model a complex MTD game, is it possible to decompose games in multiple layers simplifying the analysis? Finally, how can we measure the security offered by an MTD strategy? The remainder of this chapter answers such questions demonstrating the richness of the language provided by our definitional framework.

2.3.2 MTD Game Examples

In order to show how our framework can be used to model different MTD techniques, we consider a system taking advantage of IP hopping (also known as IP address randomization) as an example of an MTD scheme.

The central intuition behind IP hopping is to frequently change the IP addresses of the hosts in a network to make reconnaissance difficult for an adversary who is trying to locate a host by IP addresses so that the attacker is forced to continually rescan the network to locate a specific target(s). Implementations of this technique have been proposed for OpenFlow [34] and DHCP [5].

From the adversary's perspective, a spatial search for the target host in a pool of N IP addresses is needed, which is performed by sending network probes to specific IP addresses. The probe may be a simple SYN packet to locate a host with a particular open port or an operating system/software fingerprinting to locate a host running a particular system. If the adversary receives an acknowledgment, the probe found a target, which means the adversary moved successfully and is now in control of the target. The adversary continues the search until all targets are found.

We consider two different scenarios in which the adversary's goal is to locate and control one, respectively k, valuable/vulnerable hosts in the network. This is referred to as Single-Target Hiding (STH) and Multiple-Target Hiding (MTH). Our analysis of STH assumes an optimal adaptive adversary while our analysis of MTH assumes an almost optimal adaptive adversary (being almost optimal is shown in Section 2.4). As explained earlier in the model development section, for both scenarios, we assume independence of action for players and exclude the possibility of both players issuing a move at the same time.

Single-Target Hiding

In IP hopping techniques, new IP addresses from a pool of unused addresses in the network subnet will be selected and assigned to hosts at some (average) rate λ. For instance, the defender plays a memoryless MTD strategy such that every time step it randomizes the IP addresses with probability λ (and with probability $1-\lambda$ there would be no randomization). Meanwhile, the adversary sends a probe at an average rate μ in an attempt to identify a target. It may play an adaptive strategy: If the attacker has tried a set of say q IP addresses after the defender's last IP address randomization, then the $(q+1)$-st trial should not be one of the q IP addresses it already tried, instead it should guess a new one. Even though the attacker does not learn when the defender randomizes the IP addresses, if it tries each IP address in sequential round robin order, then it knows that its next guess will never be of an IP address it already tried since the defender's last IP address randomization. As the defender plays memoryless, it does not matter whether the attacker tries each IP address in

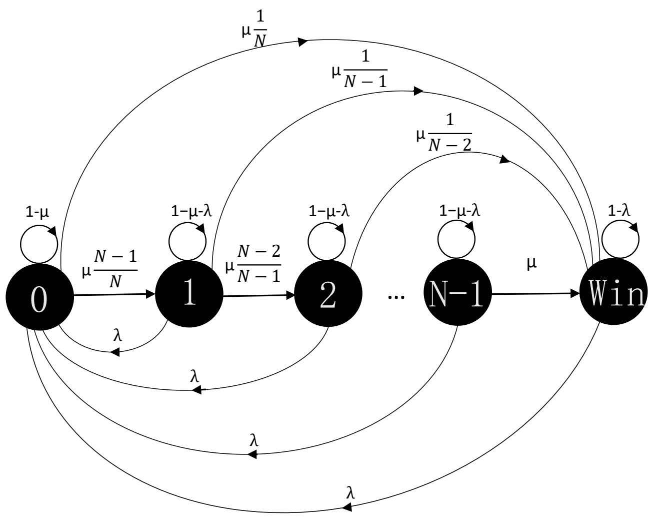


FIGURE 2.1: Markov chain of Single-Target Hiding

fixed known sequential round robin order. This optimal adaptive adversarial strategy is described by the Markov chain in Figure 2.1.

The Markov chain has $N + 1$ states where N equals the total number of possible IP addresses. Let state N (in the figure denoted by "Win") represent the winning state for the adversary and let states $0, 1, \ldots, N - 1$ correspond to when the defender randomized the IP address for the last time, i.e., state q means that exactly q time steps ago the IP address was randomized (by the defender). For $0 \leq q \leq N - 1$,

- $M_{q,0} = \lambda$; with probability λ the defender randomizes the IP address, as a result the Markov chain transitions back to the state 0,
- $M_{q,q+1} = \mu(N - q - 1)/(N - q)$; with probability μ the attacker tries the next IP address which with probability $1 - 1/(N - q)$ is not the right target,
- $M_{q,N} = \mu/(N - q)$; with probability μ the attacker tries the next IP address which with probability $1/(N - q)$ is correct (i.e., the target host) meaning that

the Markov chain transitions to the winning state for the adversary,

- $M_{q,q} = 1 - \lambda - \mu$; with probability $1 - \lambda - \mu$ both the defender and attacker do not issue a move,
- $M_{N,0} = \lambda$ and $M_{N,N} = 1 - \lambda$; the adversary does not need to try IP addresses, it will only be kicked out of the winning state if the defender randomizes the IP addresses.

Multiple-Target Hiding

The IP hopping game described above considers the case where the adversary only needs to find one single target. Another possible scenario one can think of is the case in which the adversary needs to find/penetrate more than one host/target in a network taking advantage of IP hopping defense strategy. Let N be the total number of "locations" where K targets (i.e., attack vectors or vulnerabilities) may hide; only if the combination of a smaller subset of k locations of these K target attack vectors are known by the attacker, the attacker is capable of launching a successful attack by exploiting the combination of the k associated vulnerabilities, i.e., the attacker wins the MTD game. The defender moves by re-allocating a target causing the attacker to renew its search for the locations. The attacker moves by selecting one of the N possible locations and testing whether it corresponds to one of the targets. For example, consider a hit-list worm targeting specific hosts on the network or servers part of a redundant system an adversary is attempting to disrupt. The MTH defense limits the success of an adversary attempting to compromise and persist on a set of servers in a network.

Both the defender and attacker interface with the defender's "system," the system implements the defender's MTD strategy and can also control the rate at which each party may access the system. Each time step allows three possibilities for defender-attacker moves (giving an example of the more abstract (M^D, λ, M^A, μ)-MTD game):

[**Defender moves**] *If the defender issues a move, it will take priority over any other player's request.* We assume a system which controls its own MTD: if the system/defender thinks it is time to move (i.e., re-allocate a target), then it will stall any other requests (in particular, those made by an adversary who wishes to figure out where targets are hiding by getting in touch with locations and testing them).

In our analysis we only consider a defender who plays a memoryless strategy: at every time step, the defender plays (issues a move) with probability λ regardless of any learned information about the attacker's strategy. Also, when the defender plays, it will choose one of the K targets at random and re-allocate the selected target to a random location (among one of the other $N - K$ possible locations).

[**Attacker moves**] The defender has not issued a move, but the attacker has: We consider *adaptive adversarial strategies*, i.e., the adversary can be modeled by a probabilistic algorithm $\mathcal{A}$ which uses previously learned information in order to select a new future time step at which the adversary will issue its next move. A few examples of previously learned information are the times at which the defender issued a move, and the attacker's discovery of when it gains/loses the location of a target. In our analysis we give the adversary more detailed information (for free): we assume that as soon as the defender re-allocates a target (which its location was known to the attacker before re-allocation), the attacker is told that this location does not any more correspond to one of the K targets.

[**No moves**] No moves are issued by either the defender or the adversary. We consider

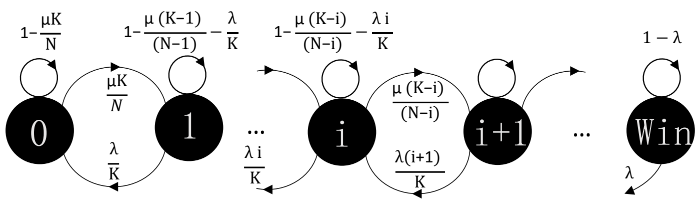


FIGURE 2.2: Markov chain of Multiple-Target Hiding

an adaptive adversarial strategy which, given some number of time steps T, maximizes the probability of winning the game (by finding a subset of k locations) within T time steps: suppose the attacker moves and decides to wait by not issuing a move the very next time step. Then the attacker runs the extra risk to lose one of the found targets during the next time step as the defender plays a memoryless strategy and will play/move with probability λ. This extra risk translates into a higher probability of not being able to learn a sufficient number of k locations within T time steps. To reduce this unnecessary probability of losing a location while waiting, the attacker should not wait in the first place. This means an optimal adaptive adversarial strategy is to issue moves at every time step as much as the system allows access. We assume a system which limits the rate r at which a party is allowed to access the system (in more detail: whenever a party requests to access the system, the request is put in a FIFO queue and the system will serve requests according to a memoryless Poisson process corresponding to r). Since the defender moves take priority, this implies that the attacker only plays when the defender does not play. Therefore, if the defender plays memoryless with probability λ, then either the defender moves with probability λ, or the attacker moves with probability $\mu = \min\{r, 1 - \lambda\}$, or with probability $1 - \lambda - \mu$ no one moves.

Figure 2.2 depicts the Markov chain which describes a close approximation of the above MTD game (the winning state "k" is denoted by state "Win"); as in the Single-Target Hiding game, an optimal adversary should select new locations according to a round robin strategy – we analyze a suboptimal adversary who selects random new locations: If the game is in state i and if the attacker plays, which happens with probability μ, then the attacker will hit one of the $K-i$ unknown locations with probability $(K-i)/(N-i)$ (since the defender chooses new locations uniformly at random): this explains the transition probability $M_{i,i+1}$ from state i to state $i+1$. If the game is in state $i+1$ and if the defender plays, which happens with probability λ, then the defender will randomize one of the $i+1$ locations known to the adversary with probability $(i+1)/K$: this explains the transition probability $M_{i+1,i}$ from state $i+1$ to state i. Hence, transition matrix M is given by

$$\begin{aligned} M_{i,i+1} &= \mu(K-i)/(N-i) \quad \text{for } 0 \leq i \leq k-1, \\ M_{i+1,i} &= \lambda(i+1)/K \quad \text{for } 0 \leq i \leq k-1, \end{aligned} \tag{2.2}$$

$M_{i,i} = 1 - M_{i,i+1} - M_{i,i-1}$ for $1 \leq i \leq k-1$, $M_{0,0} = 1 - M_{0,1}$, and $M_{k,k} = 1 - M_{k,k-1}$.

2.4 Security Analysis of MTD Games

We prove the following theorem which provides an upper bound on the winning probability of an attacker playing an M-MTD game within T time steps. The theorem's interpretation lead to the definition of "security capacity" of MTD games. The proofs of theorems in this chapter are presented in Section A.1.

Theorem 1 (Adversary's Winning Probability of an M-MTD Game)**.** *For an M-*

MTD game with stationary distribution[3] π *(i.e.,* $\pi M = \pi$*) and the initial state* 0 *representing the worst state to start the game from the adversary's perspective,*[4] *the probability that the attacker wins in the first* T *time steps is* $\leq T \cdot \pi(k)$*, where* k *is the winning state.*

In order to apply this or next theorems, an *upper bound* on $\pi(k)$ for a stationary distribution π must be computed/estimated in an efficient way. We refer the reader to the works presented in [7,40], which offer promising algorithms for efficiently computing this probability. Therefore, even if the Markov chain of an MTD game remains complex/large (after compressing the chain to one where states only represent the projection to necessary system and MTD parameters), the theorems can be used for security analysis of the MTD game. *The main challenge is to find an appropriate Markov chain in the first place.*

We can interpret Theorem 1 as follows: suppose $\pi(k) = 2^{-c}$. Then, the probability that the attacker wins in the first $T = 2^t$ time steps is $\leq 2^{-(c-t)}$. We may interpret $s = c - t$ as a security parameter and interpret c as the "security capacity" of the MTD game. The capacity $c = s + t$ is split into "a probability 2^{-s} of successful attack" and "a considered time window of 2^t logical time steps."

Definition 4. *An* M*-MTD game has at least security capacity* c *if the probability that the attacker wins in the first* $T = 2^t$ *time steps is* $\leq 2^{-s}$ *for all* $t + s = c$*. Notice that an* M*-MTD game with stationary distribution* π *satisfying the conditions of Theorem*

[3]This implicitly assumes the existence of a stationary distribution; this distribution may not be unique, however, the theorem applies to any such distribution as long as the other condition of the theorem is met.

[4]This excludes a scenario where initially the adversary has a leg up – e.g., because the initial state of the system is a "default" state where the adversary can assume specific "default" settings before the defender had the opportunity to move "targets" around.

1 has at least a security capacity[5] $c = -\log \pi(k)$.

If we know the cost of the real resources (i.e. money, time, etc.) an adversary needs to spend in order to move, then the next theorem shows how to compute the adversarial cost needed for a successful attack.

Theorem 2 (Cost Analysis). *For an M-MTD game with an expected adversarial cost per logical moves equal to α, stationary distribution π, and the initial state* 0 *representing the worst state to start the game from the adversary's perspective, the probability that the attacker wins the M-MTD game by paying at most C is $\leq C \cdot \pi(k)/\alpha$, where k is the winning state.*

Notice that C/α estimates the number of logical time slots T within which the adversary wins the M-MTD game. Applying Theorem 1 with $T = C/\alpha$ immediately gives the above result. A precise proof turns out to be more complex and is given in the appendix A.1.2.

Now we study the security analysis of the two MTD examples we introduced previously to show how the theorems can be used for gauging the effectiveness of an MTD scheme.

[5]See the previous footnote, the security capacity will not be much larger than $-\log \pi(k)$.

2.4.1 Security Analysis of Single-Target Hiding Game

We compute the stationary distribution for the Single-Target Hiding scenario as follows: Equation $\pi M = \pi$ gives for $1 \leq q \leq N-1$

$$\pi(0) = (1-\mu)\pi(0) + \lambda\pi(N) + \lambda\sum_{q=1}^{N-1}\pi(q), \tag{2.3}$$

$$\pi(q) = (1-\lambda-\mu)\pi(q) + \mu\frac{(N-q)\pi(q-1)}{(N-q+1)}, \tag{2.4}$$

$$\pi(N) = (1-\lambda)\pi(N) + \sum_{q=0}^{N-1}\mu\pi(q)/(N-q). \tag{2.5}$$

Equation (2.4) yields the recurrence relation

$$\pi(q) = \frac{\mu}{\mu+\lambda}\frac{N-q}{N-q+1}\pi(q-1)$$

which is solved by

$$\pi(q) = \left(\frac{\mu}{\mu+\lambda}\right)^q \frac{N-q}{N}\pi(0).$$

Equation (2.3) combined with $\sum_{q=1}^{N}\pi(q) = 1-\pi(0)$ is solved by

$$\pi(0) = \frac{\lambda}{\mu+\lambda}.$$

Now we are able to derive

$$\sum_{q=0}^{N-1}\pi(q)/(N-q) = \left\{1-\left(\frac{\mu}{\lambda+\mu}\right)^N\right\}/N.$$

Substituting this in (2.5) gives the equilibrium probability of the winning state for the adversary:

$$\pi(N) = \left\{1 - \left(\frac{\mu}{\lambda+\mu}\right)^N\right\} \cdot \frac{\mu}{\lambda N}. \tag{2.6}$$

In practice, the security capacity $-\log \pi(N) \approx \log(\lambda N/\mu)$ is rather small as N is restricted to the number of unused addresses in the network subnet.

2.4.2 Security Analysis of Multiple-Target Hiding Game

In order to find the stationary distribution π of M in (2.2), we observe that $\sum_{j=0}^{i} \pi(j)$ and $\sum_{j=i+1}^{k} \pi(j)$ do not change after multiplying with M. This means that the probability of leaving state $i+1$ to i must be equal the probability of leaving state i to $i+1$ as a result of one multiplication by M: For $0 \leq i \leq k-1$,

$$\pi(i) \cdot M_{i,i+1} = \pi(i+1) \cdot M_{i+1,i}.$$

Rewriting the recurrence relation yields

$$\begin{aligned}
\pi(k) &= \pi(0) \prod_{i=0}^{k-1} \frac{\pi(i+1)}{\pi(i)} = \pi(0) \prod_{i=0}^{k-1} \frac{M_{i,i+1}}{M_{i+1,i}} \\
&= \pi(0) \prod_{i=0}^{k-1} \frac{\mu(K-i)/(N-i)}{\lambda(i+1)/K} \\
&= \pi(0) \left(\frac{\mu K}{\lambda}\right)^k \prod_{i=0}^{k-1} \frac{(K-k+i+1)}{(i+1)(N-i)} \\
&\leq \left(\frac{\mu K(K-k+1)}{\lambda(N-k+1)}\right)^k.
\end{aligned}$$

Notice that if $K = k = 1$, which describes the Single-Target Hiding game, the upper bound is $\mu/(\lambda N)$ which is close to (2.6) implying (by extrapolation) that the suboptimal adversary considered in the multiple-target hiding game is close to optimal.

The security capacity $-\log \pi(k) \geq k(\log(\lambda(N-k+1))-\log(\mu K(K-k+1)))$ scales linearly with k showing that multiple-target hiding is a good MTD design principle.

2.4.3 Composition of MTD Games

In order to design and maintain secure systems, the system architect or operator must be aware of the systems' attack vectors and vulnerabilities. This can be achieved only if a higher-level view of the system and these attack vectors is presented. A Cyber Attack Life Cycle (CALC), also known as "Cyber Kill Chain," provides this higher-level view of an attack in multiple stages where each stage accomplishes an intermediate goal. A more holistic approach for analytics and sensing can be taken into consideration by using the knowledge of the entire CALC [102]. The research summarized in [61] considers a large class of MTD schemes and establishes various CALC stages for which each of these schemes can be effective. Rather than using CALC, a more detailed representation of threats is through attack graphs [74], where each node represents a system state with attacker capabilities and a (labeled) directed edge represents an adversarial behavior causing a change in the system state. An attack graph teaches how individual (local) attacks can be combined in order to produce larger attack paths (reaching a global goal).

Composition of MTD games seems to be a natural defense mechanism meaning that although a single MTD may not have sufficient security capacity, a composition of multiple MTDs deployed together may have sufficient security capacity. Therefore, the system's defender, by having a higher-level view of the system through CALC or attack graphs, might be able to get higher security by composing multiple MTDs (along attack paths).

In this section, we analyze the composition of several levels/layers of MTD where the attacker can only play at a higher level if it is in the winning state of the previous level. Equivalently, the game at the higher level does not allow the attacker to move if it is not in the winning state of the previous level. The main idea is that the attacker must slowly penetrate the defense mechanism as in an Advanced Persistent Threat (APT) [87]. We call this approach a Layered MTD or MTD^*:

Definition 5. *A Layered MTD game (or MTD^*) is a composition of a sequence of MTD games M_j, $0 \leq j \leq f$, defined by the following rules: The defender plays all the games at each timeslot; the adversary plays MTD game M_0 each timeslot; for $0 \leq j \leq f-1$, the adversary only plays MTD game M_{j+1} in a timeslot if the adversary is in the winning position of MTD game M_j at the beginning of that timeslot.*[6] *The MTD^* game is won by the adversary, if the winning state in M_f has been reached.*

In the above definition, we assume that for all layers/levels M_j, the transitions occur with the same frequency (we analyze different frequencies in Section 2.4.4). The next theorem analyzes the security capacity of a layered MTD game.[7]

Theorem 3 (Upper Bound on the Winning Probability of a Layered MTD Game)**.** *Let M_j with the winning state k_j, stationary distribution π_j, and at least a security capacity $c_j = -\log \pi_j(k_j)$, $0 \leq j \leq f$, represent a sequence of MTD games. Assume that for each M_j-MTD game the initial state 0 represents the worst state to start the*

[6]If the adversary is not in the winning position of MTD game M_j, then these rules effectively change M_{j+1} temporarily into an MTD game M'_{j+1} where only the defender plays.

[7]For the thorough analysis and examples of analyzing composed games see A.1.3.

game from the adversary's perspective. Furthermore, assume that in each M_j-MTD game the defender implements moves which can transition from state k_j (the losing state from the defender's perspective) to a state $\neq k_j$, hence, $\lambda_j = 1 - (M_j)_{k_j,k_j} > 0$.

Then, the probability that the attacker wins the layered MTD game based on $M_0, \ldots, M_f$ in the first T time steps is

$$\leq T \cdot \pi_0(k_0) \prod_{j=1}^{f} \pi_j(k_j)/\lambda_j.$$

i.e., the layered MTD game has a security capacity at least

$$c = \sum_{j=0}^{f} c_j + \log \prod_{j=1}^{f} \lambda_j. \tag{2.7}$$

2.4.4 Time Conversion

How do we define a composition game made of levels with different time scales? Let us measure the time t_j per logical transition at level j in a universal time unit (e.g. seconds) such that $t_j \geq 1$ for all j. In reality defender and adversarial moves complete in very little time and corresponding transitions are close to immediate; the rate of play, however, measured in universal time units scales with $1/t_j$. The corresponding new transition matrix for level j is equal to

$$M_j^{new} = \frac{1}{t_j} M_j + (1 - \frac{1}{t_j}) I.$$

Since the time per transition in M_j^{new} is now 1 universal time unit regardless of the level, we can apply the composition theorem to these new transition matrices.

Clearly the transformation to the new transition matrices does not have any effect on the stationary distribution π_j since for the new transition matrix we also have $\pi_j = \pi_j M_j^{new}$. This proves that $\pi_j(k_j)$ stays the same for M_j^{new}. However we notice that

$$\begin{aligned} \lambda_j^{new} &= 1 - (M_j^{new})_{k_j,k_j} = 1 - [\frac{1}{t_j}(M_j)_{k_j,k_j} + (1 - \frac{1}{t_j})] \\ &= \frac{1}{t_j}(1 - (M_j)_{k_j,k_j}) = \frac{\lambda_j}{t_j} \end{aligned}$$

Moreover, $T^{new} = T \cdot t_0$ since it is measured in time steps for the lowest level game. This yields the following corollary.

Corollary 1. *Assume that for each M_j-MTD game the transitions take on average $t_j \geq 1$ time units. Then, the probability that the attacker wins the layered MTD game based on $M_0, \ldots, M_f$ in the first T time units is*

$$\leq T \cdot t_0 \pi_0(k_0) \prod_{j=1}^{f} t_j \pi_j(k_j)/\lambda_j.$$

Corollary 2. *The security capacity of the layered MTD game measured in time units is at least (2.7) minus a time conversion factor $\log \prod_{j=0}^{f} t_j$ which corresponds to how much time transitions take in each game.*

2.5 Concluding Remarks and Future Directions

In this Chapter, we introduced a Markov-model-based framework for MTD analysis. The framework allows modeling of concrete MTD strategies, provides general theorems about how the probability of a successful adversary defeating an MTD strategy is related to the amount of time/cost spend by the adversary, and shows how a multi-level composition of MTD strategies/schemes can be analyzed by a straightforward combination of each MTD strategy's analysis.

Central in the proposed framework is the concept of security capacity, which measures the strength of an MTD strategy: the security capacity depends on MTD specific parameters and in general system parameters. As the first direction for future research, the security capacity versus system performance can be analyzed as a function of system and MTD parameters. This allows the defender to understand the price paid for deploying an MTD strategy from the performance perspective, and how MTD and system parameters should be set given system constraints and parameters. Also, various MTD strategies can be compared, and the most effective one concerning the defender's cost in terms of performance can be selected.

MTD strategies hide targets, which is a combinatorial game leading to information theoretic guarantees expressed by the security capacity. As demonstrated in the analysis of the Multiple-Target Hiding MTD game, the security capacity is the largest if an adversary needs to find multiple targets (and not just one single target) which are each being hidden by the MTD strategy. The analysis shows that the probability of finding each target scales with the product of the probabilities of finding each specific target. This exponential dependency on the number of hidden targets is also demonstrated by the analysis of Layered MTD strategies.

A second direction for future research is to analyze the Layered MTD composition in more detail and improve the lower bound on the security capacity proved in this work (as demonstrated for the toy example (see A.1.3), even though the current proven bound shows an exponential dependency on the number of layers, it is much weaker when compared to simulation results). Additional future research regarding the composition is to investigate into what extent MTD schemes and strategies can be composed according to a more effective and adaptive mechanism where the state in each MTD game supplies feedback to each other MTD game in their composition. Feedback from one MTD game can be used to adapt the MTD parameters of another MTD game, and this could lead to an overall larger security capacity of the composition.

In the next chapter, we focus on a more general game to extend our analysis and presented language to a broader range of cyber attack and defense games, especially APT-like attacks. We study how the probability of a successful adversary defeating a defender's strategy is related to the amount of time/cost spent by the adversary in such scenarios. We present a Markov-based general framework to model the interactions between the two famous players of network security games, i.e., a defender (taking advantage of common security tools and technologies such as Intrusion Detection and Prevention Systems (IDPSes), Firewalls, and Honeypots (HPs)) and an attacker (and possibly its agents) who takes actions to reach its attack objective(s) in the game. Similar to our previous line of thinking in Chapter 2, we mainly focus on the most significant and tangible aspects of sophisticated cyber-attacks: (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objectives. Therefore, our goal is to translate attacker-defender interactions into a well-defined game so that we can provide rigorous cryptographic security guarantees for a system given both players' tactics and strategies.

We introduce a notion of "learning" in cybersecurity games and study its impact on players chances of prosperity in the game. The presented framework extends the idea of security capacity introduced in Chapter 2 and provides the definition of a *security capacity region* as a metric for gauging a defensive system's efficiency from a security perspective (by presenting rigorous cryptographic security guarantees in terms of the security capacity region).

Chapter 3

Toward a Theory of Cyber Attacks based on Markov Models

3.1 Introduction

Cyber attacks targeting individuals or enterprises have become a predominant part of the computer/information age. Such attacks are becoming more sophisticated (qualitative aspects) and prevalent (quantitative aspects) on a daily basis [18]. The exponential growth of cyber plays and cyber players necessitate the inauguration of new methods and research for better understanding the "*cyber kill chain*," particularly with the rise of advanced and novel malware (e.g., Stuxnet [35], WannaCry ransomware crypto worm [52], the Mirai and its variants [6,8]) and the extraordinary growth in the population of Internet residents, especially connected Internet of Things (IoT) devices.

Mathematical modeling could be used to represent real-world cyber-attack situations. Such models play a beneficial role when it comes to the *secure* design and evaluation of systems/infrastructures by providing a better understanding of the threat itself and the attacker's conduct during the lifetime of a cyber attack. The sparse amount of research on modeling and evaluating a defensive systems' efficiency (especially from a security perspective), however, warrants the need for constructing a proper *theoretical framework*. Such a framework allows the community to be able to evaluate the defensive technologies' effectiveness from a security standpoint. In this regard, a proper model is needed to capture the interactions between the two famous players of network security games i.e., a defender (taking advantage of common security tools and technologies such as Intrusion Detection and Prevention Systems (IDPSes), Firewalls, and Honeypots (HPs)) and an attacker (and possibly its agents) who takes action to reach its attack objective(s) in the game. Modeling only one player by itself (e.g., opportunistic/targeted attacks, or a defensive system such as an IDPS) without acknowledging the other party's capabilities, set of actions, and strategies would be unjustifiable. Hence, a realistic model should take both parties' characteristics, objectives, and actions into consideration, as they are typically oppositional.

Game-theoretic methods have been proposed to model the interactions between an attacker and a defender in a network environment [3,11,13,46,87]. Although essential and insightful, as figuring out the best attack-response strategy is in the focal point of a game-theoretic analysis, such models suffer from a lack of general applicability due to the case-specific assumptions made to reduce the complexity of the problem leading to a case-specific game with a corresponding set of players' actions and payoffs. Therefore, the developed models become ineffective in the representation of general settings.

Our Approach. We introduce a Markov-based framework and methodology for modeling various computer security scenarios prevailing today, including opportunistic, targeted and multi-stage attacks. Motivated by the rise of Advanced Persistent Threats (APT), we are particularly interested in situations in which each players' progress in the game can be viewed and modeled as an incremental process. For an adversary limited with a specific time budget (representing the "dwell-time" in an APT-like attack [26]), a progressive, adversarial move associated with a probability distribution can bring it one step closer to its desired winning state. In our framework, we call this winning state the attack objective, which can represent individual goals in any stage of an APT (e.g., data exfiltration) or the attack itself as a whole. Similarly, by monitoring the environment and capturing the adversarial moves, the defender tries to put all the gained information together so that it can eventually prevent the adversary from fulfilling its mission. For instance, the defender's goal could be generating a detection signal that indicates the presence of a coordinated set of activities that are part of an APT campaign [50], developing a signature for a malware attack to halt next adversarial moves [55, 84], or shuffling its resources after enough number of samples (evidence) is obtained as in a Moving Target Defense strategy (MTD) [47, 76].

Contributions & Results. Our contributions are summarized as follows:

- We introduce the notion of learning in cybersecurity games and describe a general "game of consequences" meaning that each player's (mainly the attacker) chances of making a progressive move in the game depends on its previous actions. More specifically, as a consequence of the adversarial imperfect moves in the game, the other party, i.e., the defender has the opportunity to incrementally learn more and more about the attack technology. This learning enables the defender to be able to seize and halt following adversarial moves in the game, in other words, containing the attackers' progress in the game. We argue that such learning is possible since the actions that need to be taken by an adversary in each cycle of a cyber attack's life inevitably entails abnormal and suspicious events and activities on both host and network levels [23, 31, 32, 38, 68, 93].

- Unlike game-theoretic methods which commonly focus on finding the best attack-response strategies for the players, we, however, with a cryptographic mindset, mainly focus on the most significant and tangible aspects of sophisticated cyber attacks: (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objectives. Therefore, our goal is to translate attacker-defender interactions into a well-defined game so that we can provide rigorous cryptographic security guarantees for a system given both players' tactics and strategies. We generalize the basic notion of computational security of a cryptographic scheme [45] to a *system/infrastructure* which must be defended and introduce a *computationally secure system* in which a "given *system* is (T_{bud}, k, ϵ)-*secure* (for $T_{bud}, k > 0$, and $\epsilon \leq 1$), if any adversary limited with a (time) budget at most T_{bud} succeeds in reaching the attack objective k, with probability at most ϵ." We study under which circumstances the defense system can provide an effective response that makes $\epsilon = negl(k)poly(T_{bud})$.

- By modeling the learning rate of the defender as a function $f(l)$ (which is the probability of detecting and halting a next adversarial move) where l represents the number of adversarial moves and actions so far observed and collected, we present the following results:

- If $f(l)$ does not converge to 1, then we call such a learning rate stagnating, and the adversary can reach its winning state (attack objective) k within $poly(k)$ time budget. This result answers the question of why fighting cyberattacks (especially APT-like attacks) remains a challenging quest for cyber defenders. Even in ideal scenarios (especially where there is no false positive in the learning process), to 'win' the game as the defender, i.e., preventing the attacker from accomplishing its mission, the true-positive rates must converge (fast enough) to 1, which is a remaining challenge for any defense system.
- If $f(l) \to 1$ and more specifically $1 - f(l) = O(l^{-2})$, then the probability of the adversary winning the game is proven to be negligible in k and polynomial in the attacker's time budget, i.e., $\epsilon = negl(k)poly(T_{bud})$.
- The above results hold for time budget measured in logical Markov model transitions or physical time (seconds), and also hold for learning rates that remain equal to $f(l) = 0$ until a certain number $l = L^*$ adversarial moves have been collected after which learning starts.
- The adversary may not reach its desired winning state of k, but will reach a much smaller k_c (called containment parameter) of the order $O(L^* + \log T_{bud})$, where T_{bud} is the attacker's time budget, for $1 - f(l) = O(l^{-2})$.
- We conclude that at a metalevel the adversary needs to find one attack exploit/vector for which the defender cannot find a fast enough increasing learning rate and the defender needs to have a learning system/methodology in place which should be able to reach fast enough learn-

ing rates for any possible attack exploit/vector. In practice, infrastructures are being compromised and this means that, as given by our general framework, and presented theoretical and numerical analysis, effort is needed in order to understand a defense system's associated learning rates and why these are $\leq 1 - \Omega(l^{-a})$ for some 'a' too small or have $L^* = O(k)$ (i.e., learning starts too late).

- The presented adversarial/defender game in terms of a Markov model is general and fits most practical scenarios as our extensive overview shows. To show how the presented modeling can be applied to various cyber attack and defense scenarios, we walk through two examples and provide the security analysis for them: (1) a malware propagation and botnet construction game (due to the increase in the number of bots found in the wild), and (2) a moving target defense game (as the state of the art defense strategy).

Based on our theoretical analysis and its applicability to practical scenarios, we understand when an adversary with a time budget T_{bud} can be contained and prevented from reaching its attack objective k with probability $\geq 1 - negl(k)poly(T_{bud})$. As a consequence, if such containment is in place, then this allows one to trust cryptographic protocols/systems which assume (for proving their security) attackers that have not reached k (and have a key renewal / refresh operation that can be called every T_{bud} seconds in order to push adversaries back to their initial state). Moreover, we believe that the introduced notion of *learning* is an exciting approach to model cybersecurity games and worth further investigations since it enables one to consider a more intelligent defender instead of a "dummy" one. Besides, the formal definitions of *capacity region* could be in particular interest as a metric for gauging a defensive system's efficiency from a security perspective (by presenting rigorous cryptographic security guarantees in terms of the security capacity region).

Limitations and Future Research. We explain why fighting against advanced and persistent threats remains a challenging task for system defenders. In particular, we show that even in ideal situations, where there are no false positives in the defender's learning process, the learning must not stagnate, and it must tend to 1 with a fast-converging rate satisfying our presented criterion. Adding false positives to this picture makes the defenders' job assuredly more laborious, as such incidents can disrupt the learning process, and therefore trouble the normal operation of the system. In extreme cases, such as dealing with a delusive attacker [56, 89] who can maliciously engineer the training data (capable of conducting noise injection attacks), the learning may not always be positive, and the learning engine's false-positive rates must be taken into consideration. We recognize that in the current modeling, false positives can happen on two levels: first, when it comes to learning from adversarial actions and second, while using the gained information for blocking future adversarial moves. We notice that if the attack samples collected by the defender contain many false positives, the learning could become arduous (or even impossible), and the assumption of enhancement in the "defender's detection rates by witnessing more samples" becomes less realistic. Therefore, a remaining challenge for future work is taking false positives into consideration to have a more robust and realistic model. Providing bounds on the total number of false positives that the defender can tolerate in its learning process is the next essential step in extending this presented framework, and we leave this problem for future studies. Also, the presented model can be fine-tuned to investigate various scenarios for a defender with a more extensive defensive action set, such as host-level cleaning and patching of network nodes. Although this can potentially lead to a more comprehensive model, we left the analysis for this case for future studies since assuming a universal host-level defense system deployed on all the network hosts is an unrealistic assumption in most practical scenarios.

Organization. The rest of this chapter is organized as follows: Section 3.2 briefly reviews related works in modeling cyber attacks as a game between an attacker and a defender. The interactions of the defender and the adversary are described as a stochastic network security game in Section 3.3. The security analysis of the introduced game is presented in Section 3.4 followed by the corresponding parametric analysis of the game for various learning function and system's parameters to explore the effects of such variables on attack-defense objectives. In Section 3.5, we study two attack-defense scenarios to show how our presented framework can be used to capture attacker-defender interactions in real-world settings, in particular, we show how a proper learning mechanism can bring the attacker's progress in the game to a standstill. We finally conclude this chapter and discuss future directions in Section 3.6.

3.2 Background and Related Work

Modeling cyber attacks as a game between a defender and an attacker is a classical but challenging research problem. Here, we review some of the research that has been done in this area which we find the most relevant (current state-of-the-art) to this presented work.

The intrusion detection problem in heterogeneous networks consisting of nodes with various security assets is studied in [13]. The expected behaviors of rational attackers in addition to the optimal defender strategy are derived by formulating the attacker-defender interaction as a non-cooperative game. The paper concludes that sufficient resources for monitoring the environment and proper system configuration at the defender side are two necessary conditions of efficiently protecting the network. Similarly, the interaction of players is modeled as a general-sum stochastic game in [46] in which Lye and Wing studied three different attack-response scenarios including defacing a website, stealing confidential data, and launching a Denial of Service (DoS) attack. The authors, though, left richer and more complex scenarios for future works. These plots include a more capable defender with a more extensive action set for attack detection and prevention purposes. Such a stronger defender could potentially lure the attacker and learn the attack technology by setting up defensive agents such as honeypots within the environment. Carroll and Grosu [11] consider such stronger defense strategies, i.e., taking advantage of camouflage techniques (e.g., disguising a regular system as a honeypot or vice versa) by investigating the effects of deception on players' interaction using a signaling game.

Motivated by the rise of advanced persistent threats, van Dijk *et al.* [87] introduced the FlipIt game to model the interaction of two players competing in a race of maximizing the amount of time each is in control of a shared computing resource while minimizing their total cost (associated with each player move). Strongly dominant strategies for both players (if there exist such strategies) are determined based on different employed attack strategies. Also, they provided general guidance on how and when to implement a cost-effective defense strategy.

For a review of existing game-theory-based solutions for network security problems, we refer the reader to the surveys provided in [43, 49]. In brief, Liang *et al.* [43] summarized the presented game models' application scenarios (both cooperative and non-cooperative games) under two categories: attack-defense analysis, and security measurement. Manshaei *et al.* [49] however, surveyed the use of game theory in addressing diverse forms of privacy and security problems in mobile and network applications. The studied works are organized in six main categories: physical and MAC layer security, security of self-organizing networks, intrusion detection systems, anonymity and privacy, network security economics, and cryptography.

Although essential and insightful, the aforementioned game-theoretic modelings and methods are mainly focused on finding the best attack-response strategies for the players, and therefore, they deal with the difficulties of defining a proper set of players' actions, strategies, and payoffs in the game. Our work differs from such studies, in the sense that we are looking into the problem of cyber attack-defense modeling from a different perspective, in that we are seeking answers to different questions such as "*is it possible to provide rigorous cryptographic security guarantees for a system/infrastructure given the players' set of actions, strategies, and system's parameters?*". We believe answering such questions can lead to a new metric for security and a means to be able to measure security and therefore define what it means for a system/infrastructure/strategy to be secure. This is especially beneficial when it comes to bootstrapping cryptographic protocols and having a proper reduction to common assumptions on, for instance, the learning rate of a defender. To answer such questions, we mainly focus on the most meaningful and tangible aspects of sophisticated cyber attacks: (1) the amount of time it takes for an adversary to accomplish its mission (called an attack objective in this manuscript) and (2) the success probabilities of fulfilling the attack objective(s).

In the previous chapter, we made an initial effort in relating the number of adversarial moves (i.e., logical time steps) and attack success probability in an MTD game using Markov modeling. We modeled the interaction of players in dynamic environments[1] by probabilistic algorithms characterized by a Markov chain which led to the concept of "security capacity" as a metric for gauging the effectiveness of an MTD scheme [47]. In particular, the relationship between the attack success probability and the time it takes to reach the attack objectives (i.e., the winning state for the adversary in the studied game scenarios) is then translated into the security capacity concept: "the security capacity of an MTD game (a defense system deploying an MTD strategy) is at least c if the probability that the attacker wins in the first $T = 2^t$ time steps is $\leq 2^{-s}$ for all $t + s = c$." In this chapter, however, we introduce and study a general game which can capture not only MTD games but also various cybersecurity game scenarios. Besides, we realize that the introduced "security capacity" concept can be generalized to a broader and more precise definition which we call security "capacity region" in our framework, and show that the studied MTD games in the previous chapter are only specific examples of the general capacity region concept.

3.3 Game Modeling

In this Section, we introduce a general network security game based on the interactions between an adversary, its agents (e.g., bots controlled by a botmaster), and a defender who is equipped with a logically centralized defense system[2] implemented in the environment. We explain how this game can be directly mapped to an equivalent Markov model and in the next Section, we provide the security analysis for this

[1]Those with changing system configurations and parameters, and therefore attack surfaces

[2]For instance, network/host-based intrusion detection and prevention systems, honeypots, etc. One motivation for our work is the advent of Software-Defined Networking (SDN) in which the entire network infrastructure can be controlled from a centralized software controller.

Markov model as well as presenting compelling interpretations of the role of both players' strategies in their probability of winning the introduced game.

3.3.1 The Game of Consequences

[**Game Setup**] In almost any sophisticated and persistent cyber attack, the adversary continues the attack until its attack objective is satisfied. However, due to the incomplete and imperfect information of the players (in this case the attacker), and the probabilistic nature of an attack's success rates, the attacker's objective can not usually be achieved in only a few numbers of moves/attempts. Adding a defender to this picture leads to an unceasing game unless the attacker decides not to play anymore (i.e., dropping out of the game, whether the attack objective is satisfied or the chances of making a progressive move become overwhelmingly small). We model the interactions of players in such scenarios as a stochastic game.

To develop a mathematical model, we make reasonable simplified assumptions from both attack and defense perspectives. This is due to the significant level of freedom in attack design and technology and the complexity and diversity of defense mechanisms and systems. For instance, when it comes to evaluating a defense system's efficiency (especially an IDPS), the accuracy, performance, completeness, fault tolerance, and timeliness properties should be taken into considerations as the top five criteria [22]. However, for this study, we believe considering all the playing factors in modeling such systems makes the model excessively complicated (and possibly inaccurate). For this reason, we mainly focus on *security-related* concerns and specifications, i.e., the accuracy and completeness (dealing with false alarms and detection rates) of the system. This means that we only care about the ability of the system in detecting malicious behaviors and parties in the environment, and its capabilities in foiling such incidents, regardless of the system architecture, the used methods and their impact on the system performance. Therefore, assuming that the non-security related traits are ideal (e.g., no latency, high performance, unlimited bandwidth, and fault tolerance), we are dealing with a defense system which is neither entirely accurate nor complete, as every information technology system suffers from security shortcomings and deficiencies.

In this regard, we consider a general system state (i, l) where i, and l represent the attacker's and defender's view in the game respectively. Notice that the defined state of the system at any instance of time projects the players' progress in the game based on merely the security-related parameters. The players start the game at the state $(i = 0, l = 0)$ in which it represents the inauguration of the attack and the zero-knowledge of the defender regarding the attack technology at time zero. From the adversary's perspective, its view of the system state i will change only if it makes a progressive move in the game. On the other hand, by noticing that any adversarial move (whether fruitful or fruitless) can potentially be observed and captured by the defender (for instance via the defense system's agents and sensors implemented in the environment), we model the defender as an incremental online learning process, meaning that the defender's view l gets updated as a consequence of discovering an adversarial move (i.e., an attack sample).[3] This increasing knowledge of the attack technology enables the defender to correctly detect and halt a new incoming malicious action with some probability $f(\cdot)$ (true positive). It is also possible that the system fails in detecting an adversarial move (or it falsely labels it as benign) with probability

[3]Another motivation for this work is the introduction and widespread embrace of Automated machine learning (AutoML) which provides the opportunity for automatic (and possibly distributed) attack learning, detection, and prevention [30, 55, 81, 83].

$1 - f(\,\cdot\,)$ (false negative). We later briefly discuss the impact of false positives in the defender's description part.

[**Attacker**] The attacker's objective is to reach a winning state of the form $(k, *)$ within a limited time budget T_{bud}. We emphasize that k or in other words the attack objective differs from scenario to scenario. It can represent the attacker's goal in each stage of an APT (e.g., reconnaissance) or the final objective of the adversary in the attack as a whole. For instance, a malware propagator or a bot herder desires to push k (in this case, the total number of infected nodes) as high as possible.[4] On the contrary, in an advanced persistent threat, the attacker may prefer to stop playing after reaching a much smaller k to minimize the attack detection probability since once the target machine has been identified, the attacker should use as few infections as possible to decrease the chance of exposing the operation. An adversary trying to access distributed information on a set of nodes within the network terminates the attack as soon as its mission is accomplished. In a similar fashion, to construct a hitlist, the attacker will conclude its reconnaissance after a compiled list of vulnerable nodes is constructed.

To reach the winning state, at any time step, the attacker issues a move associated with a success probability p (purely depends on attack strategy and not the defender's maneuvering) that can potentially lead to incremental progress in the game, that is getting one step closer to the final attack objective ($i = i + 1$). This enables us to represent the adversarial move in a single transition in the Markov model which will be explained in Section 3.3.2. For generality, we consider a state-dependent success probability, i.e., p_i, since, intuitively, there could be cases in which the chances of

[4] Or at least 5% of the total number of vulnerable hosts, as [67] shows via simulation that in order to have hopes that no more than 50% of the total vulnerable hosts ever become infected, patching process must begin before 5% of such population become ever infected.

making a progressive move depends on the current state of the attacker in the game. For instance, consider an attacker who is trying to locate vulnerable nodes within an address space of size N, while i out of K vulnerable hosts are already found, this means that the probability of hitting a new vulnerable node is $p_i = (K - i)/(N - i)$. Also, we assume that the adversary is aware of its current state in the game i in (i, l) for some $l \geq 0$ but it does not know the defender's knowledge of the attack[5] expressed as l. This assumption basically enables the adversary to decide when to terminate the attack and stop playing the game.

Moreover, we assume that at each time step, the interaction of the attacker (or an attacker agent) with the system which is encapsulated as an adversarial move, can potentially leak some information to the defender (if observed) regarding the attack technology and methodology. This is because regardless of the mastery and skillfulness of an attacker, its taken actions during any phase of the attack, inevitably lead to abnormal and suspicious incidents and events on both host and network levels. Hence, almost all security tools and technologies rely on the existence of such attack signal indicators for detection and prevention purposes. For instance, unusual port usage [32], irregular system call sequences [38], and the occurrence of pointer value corruption on a host's process memory [44] are amongst a few events that can happen as a result of adversarial moves on host levels. On the network levels, such conducts include but are not limited to an increase in the network latency, high bandwidth usage, suspicious traffic on exotic ports, irregular scan activity, simultaneous iden-

[5]An adaptive adversary, however, can infer information regarding the defender's state in the game. For instance, if the attacker observes that it cannot make progressive moves in the game and its actions/moves are being halted in the environment, it can deduce that the defender has already gained enough knowledge of the attack technology. Hence, an adaptive adversary may decide to stop playing as soon as it observes no progress in the game, minimizing the cost of playing or investing in new and enhanced attack methodologies to have a fresh start in the game.

tical Domain Name System (DNS) requests, and Internet Relay Chat (IRC) traffic generated by specific ports.

However, a skillful attacker leveraging different anti-malware and IDPS evasion techniques[6] is capable of minimizing the detection possibilities and therefore reducing the chances of leaking attack information at each adversarial move by sneaking through the defensive system. This skillfulness can provide the adversary more time to play the game[7] as each attack sample has a lower probability of being discovered and in extreme cases, it can ideally be different for each adversarial move making the defender's job surely difficult. For instance, an attacker taking advantage of a polymorphic or metamorphic malware (in contrast with a monomorphic one) will disclose minimum information by encrypting the content and obfuscating the instruction sequences for each connection respectively leading to more propagation time available to the attacker. Therefore, dealing with a skillful adversary, if the defense system captures an attack sample, this property intuitively leads to maximizing the time and effort to push out a solution to stop the attack and generate an attack signature. More importantly, it means that only a few samples are not enough for attack signature generation.

[**Defender**] The defender's objective is to learn more and more regarding the attack technology and methodologies to be able to cease a next incoming adversarial move. In this regard, it monitors the environment for attack detection and prevention purposes via common security tools and technologies. Intrusion detection and prevention systems are amongst the most common type of defense technologies used

[6]Obfuscation methods, fragmentation and session splicing, application/protocol violations, and DDoS attacks [51] are among the most common techniques used by the adversaries to decrease the attack detection probability or to increase the chances of attack prosperity.

[7]By minimizing the information leaked to the defender for the suspicious traffic flow detection, signature generation, malware analysis, and reverse engineering processes.

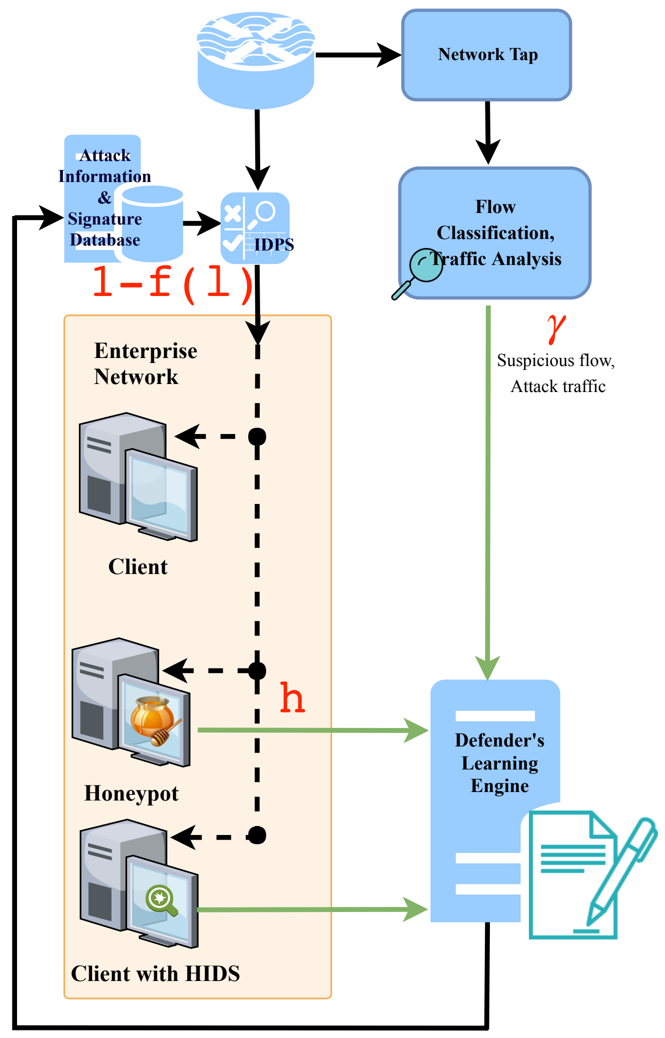

Network Tap
Attack Information & Signature Database
IDPS
1-f(l)
Flow Classification, Traffic Analysis
γ
Suspicious flow, Attack traffic
Enterprise Network
Client
h
Honeypot
Defender's Learning Engine
Client with HIDS

FIGURE 3.1: Defense system architecture; There exist both host-level defensive agents and network-level traffic analysis and flow classification tools. The defender uses the gained knowledge of the attack in bringing future adversarial actions to a standstill.

for monitoring, incident identification, attack detection, and prevention on both host and network levels.[8] Virtual and physical sensors are a common component of such systems used for data collection and analysis. Also, the defender can take advantage of electronic decoys known as honeypots for the attack information assembly.

We give the defender the opportunity to learn from observed adversarial moves and actions. The idea is that in the early phases of the game, the defender's knowledge of the attack is limited (almost non-existent). As the game proceeds, the defender's detection rates will be enhanced by witnessing enough number of attack samples. In order to study the learning rate of the defender or the time that the system is finally trained and is able to detect the adversary's actions with probability almost 1, we consider a probabilistic function $0 \leq f(\,\cdot\,) \leq 1$ as the detection rate, which is a function of the total number of times an adversary agent's activity is captured or in other words, an attack sample is given to the defender. The function f indicates the probability that given l samples so far, the system halts a new incoming adversarial move. The idea is the more samples are given to the defender, the more accurate would be its attack signatures which immediately reflects in the detection rate which is used by the system to filter future adversarial moves.

In this regard, we assume that all incoming traffic passes through an inline NIDS implementation[9] which gives the defender the ability to be able to block an adversarial move with some probability $f(l)$ in which l represents the current realization of the attack by the defender (i.e., based on so far observed and collected adversarial moves and attack information). Moreover, we assume that a copy of all incoming traffic is

[8]The first one is known as Host-based Intrusion Detection System (HIDS), and the latter is known as Network Intrusion Detection System (NIDS).

[9]Meaning that network traffic directly passes through the IDPS sensors, and the system is capable of session snipping, dropping/rejecting suspicious network activities and altering and sanitizing the malicious content within a packet.

given to the defender's traffic analysis and classification engine via a network tap or a spanning port which provides the opportunity of detecting suspicious flows with some probability. For simplicity and generality, however, we model this offline analysis as a probabilistic sampling process [80] in which each incoming adversarial move can be correctly sampled/labeled as suspicious with probability γ (therefore $l = l + 1$), and with $1 - \gamma$, the defender misses the adversarial move or it falsely labels it as benign on this level.[10] If the defender fails in bringing an adversarial move to a halt with probability $1 - f(l)$, and therefore the attacker proceeds within the system, the defender still has the opportunity to learn (i.e., $l = l + 1$) regarding the attack technology with probability h via the defense agents (e.g., HPs, HIDS) implemented in the environment. Hence, if the adversary deals with an electronic decoy, or as a consequence of attack activities, on an endpoint device equipped with a HIDS, the defender can learn regarding the attack that is $l = l + 1$ (see Figure 3.1).

We recognize that false positives can happen on two levels: first, when it comes to learning from adversarial actions and second, while using the gained information for blocking future adversarial moves. However, we consider the ideal situation where there are no false positives in the learning, or at least the false positive rate is low enough so that it does not disturb the defender's learning process (i.e., we assume the learning is always positive). Therefore, in the current modeling, false positives do not change the security state of the system (neither the attacker's nor the defender's view). Providing bounds on the total number of false positives that the defender can tolerate in its learning process is the next essential step in extending this presented framework, and we leave this problem for future studies.

[10]Note that γ (which represents the offline network-level analysis) reflects the classifier's accuracy in which it is the probability that an incoming packet will be marked as suspicious traffic conditioned on the fact that it is indeed malicious.

In summary, a progressive move by the defender ($l = l + 1$) leads to an increase in its future detection rates f. This function gets updated in two different manners: (1) if the attack traffic is correctly labeled as suspicious on the network levels with probability γ due to the defender's (offline) traffic analysis and classification; or (2) if the attack activity leaks information on host levels with probability h (whether via a honeypot or a host-based intrusion detection system installed on a fraction of defender systems).[11] This incremental learning of the attack enables the defender to be able to bring a new incoming adversarial move to a halt with probability $f(l+1)$. Notice that the adversary's chance of making a progressive move in the future time steps decreases as the f function's value increases over time.

3.3.2 Markov Model of the Game

The above description of the adversary-defender interactions can immediately be translated into a Markov model. In summary, at each time step, the state of the system can be identified with a tuple (i, l) in which i represents the adversary's progress in the game, and l delineates the defender's knowledge of the attack methodologies and technology. The adversary (or one of its agents) makes a move associated with a success probability p_i. Note that p_i is indeed the attack success probability while there exists no opponent player in the environment (i.e., the defender). In the meantime, the defender can block an incoming adversarial move with probability $f(l)$, it

[11]Notice that, in case of a honeypot, any adversarial move will lead to an increase in the defender's knowledge of the attack l since all incoming communications with an HP is suspicious, while for a HIDS, not all the adversarial moves might be observed by the defense system. For simplicity, we use a single parameter h as the probability of gaining information on host levels (mainly HPs). However, one can easily separate the learning from HPs and the HIDS agents by considering another parameter for HIDS agents h' or in general, a cumulative function $f_{h'}$ for the host-level attack information and signatures (observed and gained from HIDS agents).

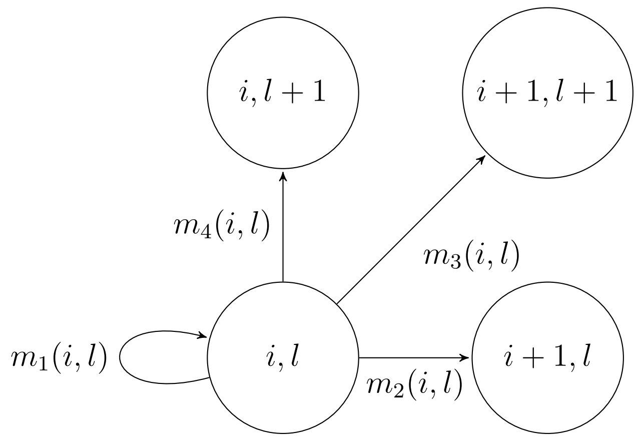


FIGURE 3.2: Possible transitions at each state (i, l) in the Markov model of the game

can also learn regarding the attack technology if the attack is observed on network or host levels with probability γ, and h, respectively.

Let F denote the occurrence of the event that an adversarial move comes to a halt by the defense system (with probability $f(l)$), and S denote the occurrence of a network-level event that it gets sampled (or labeled as suspicious) by the defense system (with probability γ). Notice that these two events are independent, as we are assuming that the straining and filtering happens online for each detected incident whereas sampling deals with a copy of the actual traffic for attack analysis and classification. Therefore the following outcomes are possible at each timestep:

- $(F \wedge \bar{S})$: The adversary's move comes to a standstill by the defender (based on the current knowledge of the attack technology) but not sampled which happens with probability $f(l)(1-\gamma)$, as a result, the Markov chain stays at state (i, l).
- $(\bar{F} \wedge \bar{S})$: The adversary's move gets neither filtered nor sampled, there exist three possible scenarios for this case:
  - If the adversary is dealing with an electronic decoy (e.g., a honeypot), the

defender has the opportunity to learn from this interaction meaning that the Markov model transits from state (i, l) to $(i, l+1)$ with probability $(1 - f(l))(1 - \gamma)h$.

  - If the attacker makes a progressive move with probability p_i then the Markov chain transits to state $(i+1, l)$ with probability $(1-f(l))(1-\gamma)p_i$. Note that this is a "perfect move" for the adversary as in which the attacker gets one step closer to the attack objective while the defensive system was not able to detect and locate the adversary's activities or it falsely labeled the activity as benign (false negative) meaning that no information and knowledge regarding the attack technology is leaked to the defender,
  - and if the attacker is not dealing with a defensive agent and also not making a progressive move with probability $(1 - p_i - h)$, the Markov chain stays at the current state with probability $(1 - f(l))(1 - \gamma)(1 - p_i - h)$.

- $(F \wedge S)$: If the attacker's move gets both filtered and sampled, the chain transits to the state $(i, l+1)$ with probability $\gamma f(l)$.

- $(\bar{F} \wedge S)$: If the attacker's action does not get filtered but it gets sampled,

  - for a successful move with probability p_i, the Markov chain transits to the state $(i+1, l+1)$ with probability $(1 - f(l))\gamma p_i$,
  - and for an unsuccessful move with probability $(1 - p_i)$, the Markov chain transits to the state $(i, l+1)$ with probability $(1 - f(l))\gamma(1 - p_i)$.

In summary, the transition probabilities can be expressed by:

$$\begin{aligned} m_1(i,l) &= Prob((i,l) \rightarrow (i,l)) = f(l)(1-\gamma) + \\ &\quad (1-f(l))(1-\gamma)(1-p_i-h), \quad (3.1) \\ m_2(i,l) &= Prob((i,l) \rightarrow (i+1,l)) = \\ &\quad (1-f(l))(1-\gamma)p_i, \quad (3.2) \\ m_3(i,l) &= Prob((i,l) \rightarrow (i+1,l+1)) = \\ &\quad (1-f(l))\gamma p_i, \quad (3.3) \\ m_4(i,l) &= Prob((i,l) \rightarrow (i,l+1)) = \\ &\quad (1-f(l))(1-\gamma)h + f(l)\gamma + \\ &\quad (1-f(l))\gamma(1-p_i). \quad (3.4) \end{aligned}$$

Figure 3.2 depicts the possible transitions at each state of the Markov model. The initial state of the game is $(0,0)$ which represents the attack initialization, and the defender's nonexistent knowledge/realization of the attack at the beginning (e.g., a zero-day vulnerability/exploit). As the game evolves, reaching to an state k is in favor of the adversary which can happen via horizontal m_2 or diagonal m_3 transitions. In the meantime, the defender's knowledge of the attack technology and signatures improves through transitions via m_3 and m_4 in which the number of attack samples provided to the defender, i.e., l increases and the function $f(\cdot)$ gets updated consequently.

3.4 Security Analysis

In this section, we provide the analysis of the presented Markov model governing the players' interactions defined in the previous section. We first discuss what it means to contain the attacker in the game by intuitively defining the desired asymptotic for the game. We then present the general formula for the attacker's winning probability $w(k, T_{bud})$, that is reaching a specific state in the game as the attack objective within a limited number of moves/transitions. We then show that a defender with a stagnating learning rate (i.e., $1 - f(l)$ remains larger than some constant $\tau > 0$) cannot contain the adversary, i.e., $w(k, T_{bud})$ does not follow the desired asymptotic. For a learning rate $f(l) \geq 1 - O(1/l^2)$ which approaches 1 fast enough, we are able to prove the desired asymptotic. We also develop tight lower and upper bounds on $w(k, T_{bud})$ and present an efficient algorithm which is able to compute these bounds with $O(k \cdot T_{bud})$ complexity. The impacts of the defender's delayed learning on the adversary's progress in the game, translating logical time steps to physical ones, and formal definition of a capacity region are presented in this section as well. We conclude this section by parametric analysis of the learning rate and its impact on the attacker's winning probability. For the proofs of theorems refer to the Section A.2.

3.4.1 Adversarial Containment

The goal of the adversary is to reach a winning state (k, l) for some $l \geq 0$ within a limited time budget T_{bud} in which k represents the attack objective which is defined based on various attack scenarios,[12] and time budget T_{bud} expresses the number of

[12]For instance, k could be the total number of nodes the adversary wishes to compromise before using the collective of these nodes in the next attack phases (e.g., botnet applications).

transitions in the Markov model, i.e., the total number of adversarial moves.

The objective of the defender is to learn more and more regarding the attack technologies (e.g., malicious attack payloads), in order to be able to recognize and halt future moves by the attacker. That is, the defender hopes to push level l – the number of so far discovered adversarial moves– higher and higher, leading to a probability $f(l)$ of successfully blocking a next adversarial action. The ultimate goal of the defender is to have a good enough learning mechanism in place such that the probability of reaching k is negligible in k (i.e., exponentially small in k) and only polynomial in T_{bud}. In other words, the adversary cannot reach a winning state in practice.

We define $w(k, T_{bud})$ as the probability of the adversary reaching to a state (k, l) for some $l \geq 0$ within time budget T_{bud}. In the next subsections we analyze this probability. We assume a parameter $p = p_i$ for all i, i.e., we simply give the adversary the advantage of using $p = \sup_i p_i$ instead of the smaller individual p_i values. We want to find a tight upper bound and determine for which model parameters γ, h, p, and learning rate $f(l)$ the upper bound proves the desired asymptotic

$$w(k, T_{bud}) = negl(k) poly(T_{bud}). \tag{3.5}$$

We notice that in this case, even though the adversary cannot reach a large k because $w(k, T_{bud})$ is negligible in k, the adversary may reach a (much) smaller k. We are interested in the order of magnitude of this value as a sub-objective the adversary is able to reach. For this reason, we define k_c as the solution of

$$w(k_c, T_{bud}) = 1/2.$$

For instance, k_c could reflect the order of magnitude of the number of compromised nodes to which the adversary is *contained.* By using $w(k, T_{bud}) = negl(k)poly(T_{bud})$, we can show that the '*containment parameter*' is $k_c = O(\log T_{bud})$.[13]

3.4.2 Formula for $w(k, T_{bud})$

The following theorem characterizes a closed-form expression for the probability that the attacker reaches a desired state k in the game within T_{bud} number of transitions/moves, i.e., the attacker's winning probability $w(k, T_{bud})$.

Theorem 4 (Closed Form Winning Probability)**.** *The adversary's probability of winning $w(k, T_{bud})$ is equal to*

$$\sum_{L=0}^{T_{bud}-1} \sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L) + m_3(L)}{1 - m_1(L)} \cdot A \cdot \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]} B$$

with A and B substituted by

[13]This implies that with significant probability the adversary can reach an attack objective at most $O(\log T_{bud})$ (e.g., compromising at most $O(\log T_{bud})$ nodes). Therefore, if the adversary would not have the advantage of using $p = \sup_i p_i$, then it would (compromise less nodes and) only use parameters $p_0, \ldots, p_{O(\log T_{bud})}$ in the Markov game. If these first few p_i values do not differ much, then a constant p becomes a realistic assumption which does not give the adversary that much advantage.

$$
\begin{aligned}
A &= \sum_{\substack{b_1,\ldots,b_L \in \{0,1\} s.t. \\ \sum_{l=1}^{L} b_l = B}} \prod_{l=0}^{L-1} \left(b_{l+1} \frac{m_3(l)}{1-m_1(l)} + (1-b_{l+1}) \frac{m_4(l)}{1-m_1(l)} \right) \\
&\cdot \sum_{\substack{g_0 \geq 1,\ldots,g_L \geq 1 s.t. \\ \sum_{l=0}^{L}(g_l-1)=(k-1)-B}} \prod_{l=0}^{L} \left(\frac{m_2(l)}{1-m_1(l)} \right)^{g_l-1}, \textit{ and} \\
B &= \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \prod_{l=0}^{L} \binom{t_l+g_l-1}{t_l} m_1(l)^{t_l} (1-m_1(l))^{g_l}.
\end{aligned}
$$

Note that the above expression simply corresponds to all the possible paths for which the adversary can reach to a desired state (k, l) for all $l \geq 0$ within at most T_{bud} transitions. We later show that neglecting the sum over T of B values leads to tight upper and lower bounds on this probability. The remaining sum over A values turns out to correspond to a simpler Markov model, which can be used to compute this sum in $O(k \cdot T_{bud})$ time using dynamic programming.

3.4.3 Stagnating Learning Rate

Suppose that the learning rate stagnates in that $1-f(l) \geq \tau$ for some constant $\tau > 0$. This means that $f(l) \leq 1-\tau$ and as a result at least a fraction τ goes undetected.

Theorem 5 (Stagnating learning rate). *The probability $w(k, T_{bud})$ of the attacker winning a game in which the defender's learning rate stagnates in that $1-f(l) \geq \tau$ is given by*

$$w(k, T_{bud}) \geq 1 - k(1-\tau p)^{T_{bud}/k}. \tag{3.6}$$

For a stagnating learning rate, the attacker almost always wins the security game, i.e., $w(k, T_{bud}) \approx 1$ even for relatively small time budgets $T_{bud} = c \cdot k$ for some multiplicative factor $c \geq 1$. This is because $1 - w(k, T_{bud}) \leq k(1 - \tau p)^{T_{bud}/k}$ is exponentially small in $T_{bud}/k = c$.

In order to have the desired asymptotics (3.5), the learning rate should not stagnate, i.e., we need $1 - f(l) \to 0$ for $l \to \infty$. Theorem 5 answers the question of why fighting cyberattacks (especially APT-like attacks) remains a challenging quest for cyber defenders. Even in ideal scenarios (especially where there is no false positive in the learning process), to 'win' the game as the defender, i.e., preventing the attacker from accomplishing its mission, the true-positive rates must converge (fast enough) to 1, which is a remaining challenge for any defense system.

3.4.4 Upper and Lower Bounds on $w(k, T_{bud})$

We present upper and lower bounds on the attacker's winning probability. In the notation of Theorem 4, we define $\bar{w}$ as

$$\bar{w}(k, T_{bud}) \quad = \quad \sum_{L=0}^{T_{bud}-1} \sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L) + m_3(L)}{1 - m_1(L)} \cdot A. \tag{3.7}$$

Theorem 6 (Upper and Lower Bounds on w). *For $v \geq k$, we have*

$$\bar{w}(k, T_{bud}/v) \cdot (1 - (k + \frac{T_{bud}}{v})(1-\gamma)^{\frac{v}{k}}) \leq w(k, T_{bud}) \leq \bar{w}(k, T_{bud}). \tag{3.8}$$

In particular, for $v = k \ln((T_{bud} + k)/\epsilon)/\ln(1/(1-\gamma))$, we have $w(k, T_{bud}) \geq \bar{w}(k, T_{bud}/v) \cdot (1 - \epsilon)$.

Upper bound (3.7) proves that if $\bar{w}(k, T_{bud})$ has the desired complexity meaning that $\bar{w}(k, T_{bud}) = negl(k)poly(T_{bud})$, then $w(k, T_{bud})$ inherits this desired asymptotic. And vice versa, the lower bound in (3.8) for the special choice of v proves that if $w(k, T_{bud})$ has desired asymptotic $w(k, T_{bud}) = negl(k)poly(T_{bud})$, then also $\bar{w}(k, T_{bud}) \leq w(k, T_{bud} \cdot v)/(1-\epsilon) = negl(k)poly(T_{bud} \cdot v) = negl(k)poly(T_{bud})$. We conclude that in order to find out which learning rates lead to the desired asymptotic for the defender, we only need to study which learning rates lead to $\bar{w}(k, T_{bud}) = negl(k)poly(T_{bud})$.

The advantage of studying $\bar{w}(k, T_{bud})$ is having eliminated expression B in Theorem 4 which represents the effect of self-loops (and turns out to be very inefficient to evaluate as an exact expression). Notice that there is no learning from the defender's perspective for the self-loops (m_1 transitions) in the Markov model. Hence, intuitively, as far as the attacker's move is not observed by the defender and no learning happens, for a fruitless move, we can ignore the adversarial move and act like it never happened, which makes sense from the practical perspective as well. Probability $\bar{w}(k, T_{bud})$ corresponds to a Markov model without any self-loops and with horizontal transition probabilities $m_2(l)/(1-m_1(l)) = (1-\gamma)\alpha(l)$, diagonal transition probabilities $m_3(l)/(1-m_1(l)) = \gamma\alpha(l)$, and vertical transition probabilities $m_4(l)/(1-m_1(l)) = 1-\alpha(l)$, where

$$\alpha(l) = \frac{p(1-f(l))}{\gamma + (1-\gamma)(p+h)(1-f(l))}. \tag{3.9}$$

Algorithm 2 depicts how $\bar{w}(k, T_{bud})$ can be computed in $O(k \cdot T_{bud})$ time using dynamic programming – and this is what we use in section 3.4.10 to evaluate $\bar{w}(k, T_{bud})$ for different learning rates.

Algorithm 2 $\bar{w}$ Computation

```
function $\bar{w}(k, T_{bud})$
    $pr \leftarrow zeros(k-1, T_{bud}-1)$
    $pr[0,0] \leftarrow 1$
    for $i \leftarrow 0, T_{bud}-1$ do
        if $i! = 0$ then
            $pr[0,i] \leftarrow (1-\alpha(i-1))pr[0,i-1]$
        end if
        for $j \leftarrow 1, k-1$ do
            if $i == 0$ then
                $pr[j,i] \leftarrow [(1-\gamma)\alpha(0)]^j$
            else
                $pr[j,i] \leftarrow (1-\gamma)\alpha(i)pr[j-1,i] + \gamma\alpha(i-1)pr[j-1,i-1] + (1-\alpha(i-1))pr[j,i-1]$
            end if
        end for
    end for
    $\bar{w} \leftarrow 0$
    for $i \leftarrow 0, T_{bud}-1$ do
        $\bar{w} \leftarrow \bar{w} + \alpha(i)pr[k-1,i]$
    end for
    return $\bar{w}$
end function
```

3.4.5 Analyzing Learning Rate $f(l) = 1 - \frac{d}{(l+2)^2}$

We further upper bound $\bar{w}(k, T_{bud})$ in the next theorem which shows that small enough learning rates $f(l) \geq 1 - d/(l+2)^2$ will attain our desired asymptotic (as explained as a consequence after the theorem). In Section 3.4.6 we will discuss what happens if there is an initial period during which the adversary already starts making progress in the game (e.g., compromising nodes) without the defender being aware or not having learned anything. For example, $f(l) = 0$ for $l \leq L^*$ and $f(l) \geq 1 - \frac{d}{(l-L^*)^2}$ for $l > L^*$. Here, L^* represents this initial period; only after a sufficient level L^* is reached, i.e., a sufficient number of attack samples (e.g., malware payloads) have been collected, the learning rate increases.

Theorem 7 (Upper Bound on $\bar{w}$)**.** *Assume that there exists a $d \geq 0$ such that $1 - f(l) \leq d$ for all $l \geq 0$, or equivalently, $\beta(l) = \sqrt{\frac{1-f(l)}{d}} \leq 1$ for all $l \geq 0$. Let $\theta \geq (1-\gamma)/\gamma$. Then,*

$$\bar{w}(k, T_{bud}) \;\; \leq \;\; (1+\theta^{-1}) \sum_{L=0}^{T_{bud}-1} [\theta p d]^k \prod_{l=0}^{L} \left(1 + (1+\theta^{-1}) \frac{\beta(l)}{1-\beta(l)}\right).$$

Let $c = \int_{l=0}^{T_{bud}-1} \frac{\beta(l)}{1-\beta(l)} dl + \frac{\beta(0)}{1-\beta(0)}$. Then, minimizing the above upper bound with respect to θ proves

$$c \leq \frac{1-\gamma}{\gamma} k \quad \Rightarrow \quad \bar{w}(k, T_{bud}) \leq (1-\gamma)^{-1} T_{bud} e^{(1-\gamma)^{-1} c} [\frac{1-\gamma}{\gamma} p d]^k.$$

Applying this statement for $1 - f(l) \leq \frac{d}{(l+2)^2}$ gives if $T_{bud} \leq e^{-1+k(1-\gamma)/\gamma}$, then

$$\bar{w}(k, T_{bud}) \leq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot T_{bud}^{1+1/(1-\gamma)} \cdot \left[\frac{1-\gamma}{\gamma} p d\right]^k.$$

For $f(l) \geq 1 - \frac{d}{(l+2)^2}$ with $d < \frac{\gamma}{p(1-\gamma)}$, the theorem shows that if $T_{bud} < exp(k)$, then $w(k, T_{bud}) \leq \bar{w}(k, T_{bud}) = poly(T_{bud})negl(k)$ satisfying the desired asymptotic. For analyzing other learning rates, the more general upper bound of the theorem can be used.

3.4.6 Delayed Learning

In practice, learning may only start after observing a sufficient number of attack samples. That is, $f(l) = 0$ for $l < L^*$ for some L^* after which $f(l)$ starts to converge to 1. What would be the attacker state in the game during this initial phase, where no learning takes place (for instance, how many nodes will be compromised)?

Theorem 8 (Delayed Learning). *Suppose $f(l) = 0$ for $l < L^*$. Define $u(k^*, L^*)$ as the probability that L^* is reached by a state (i, L^*) with $i \leq k^*$, where we give the adversary the advantage of an unlimited time budget. Then, for $k^* \geq L^* + 1$,*

$$u(k^*, L^*) \geq 1 - L^*[\frac{(1-\gamma)p}{1-(1-\gamma)(1-(p+h))}]^{\frac{k^*-1}{L^*}}.$$

Let $w'(k, T_{bud})$ correspond to learning rate $f(l + L^*)$ as a function of l (e.g., $1 - f(l + L^*) \leq \frac{d}{(l+2)^2}$). Then, for $k^* \geq L^* + 1$,

$$w(k, T_{bud}) \leq (1 - u(k^*, L^*)) + w'(k - k^*, T_{bud}), \tag{3.10}$$

where $1 - u(k^*, L^*)$ is the probability that $> k^*$ is achieved during the initial phase and where $w'(k - k^*, T_{bud})$ is the probability that another $k - k^*$ progressive adversarial moves happen after the initial phase with time budget T_{bud}.

The lower bound on $u(k^*, L^*)$ can be further simplified by using $p \leq 1 - h$ which implies

$$u(k^*, L^*) \geq 1 - L^*[(1-\gamma)(1-h)]^{\frac{k^*-1}{L^*}}.$$

Since $1 - u(k^*, L^*)$ is exponentially small in k^*/L^*, this shows that very likely $k^* = O(L^*)$. This makes common sense because initially, every adversarial move has a significant probability of success since the defender's knowledge of the attack is very limited in the beginning (the attack is still unknown to the defender, in other words, no valid attack signature is formed yet). If we achieve desired asymptotic (3.5) for $w'(k, T_{bud})$, then the resulting containment parameter for $w(k, T_{bud})$ (see Section 3.4.1) is $k_c = O(L^* + \log T_{bud})$. It is very important for the defender to keep L^* small. A larger L^* leads to a longer initial phase during which the adversary keeps on making progressive moves of the order of $O(L^*)$.

3.4.7 Time Budget Translation

We discuss the effect of translating the time budget T_{bud}, measured in Markov model transitions, to a time budget which is measured in real time (seconds) as this makes sense in some practical scenarios (e.g., malware propagation). For the sake of argument, let us assume that each state in the Markov model represents the number of nodes (e.g., infected machines or bots) in control of the attacker. If the adversary wins as soon as k nodes are compromised and if each Markov model transition takes Δ seconds, then at most k moves can be made by each of the compromised nodes *in parallel* in Δ seconds. With a time budget of T seconds, this implies $T_{bud} \leq Tk/\Delta$. This means that the desired asymptotic (3.5) translates into

$$w(k, T_{bud}) \leq poly(Tk/\Delta)negl(k).$$

This is $poly(T)negl(k)$ and again implies that in practice the adversary cannot reach k since the probability of reaching k is negligible in k and only polynomial in the physical time budget T.

In some scenarios, each adversarial move costs a certain amount, say C USD, for example due to having to rent a virtual machine in order to remotely launch or execute an attack move. In such cases, the time budget T_{bud}, measured in Markov model transitions, directly translates into $C \cdot T_{bud}$, the amount of USD the adversary is restricted to. Such conversion allows one to reason about economically constrained adversaries.

3.4.8 Capacity Region

In this subsection, we show that as a result of our analysis we are able to prove statements like the one in the following definition which defines a 'capacity region' of combinations of parameters defining the adversarial time budget, the probability of winning, and k which characterizes what it means to be in a winning state. The definition describes what was previously called 'desired asymptotics' since a *non-empty* capacity region shows that an exponentially small probability of winning $O(2^{-s})$ is achieved if $k = O(s + \log T_{bud})$ implying that the probability of winning is $poly(T_{bud})negl(k)$.

Definition 6. *Suppose there exist vectors $\boldsymbol{\delta}$, $\boldsymbol{\mu}$, $\boldsymbol{\xi}$, and the all-one vector $\mathbf{1}$, such that, for all $k \geq 0$ (characterizing a winning state for the adversary) and for all $s \geq 0$ and $t \geq 0$ satisfying*

$$s\boldsymbol{\delta} + t\boldsymbol{\mu} \leq k\mathbf{1} + \boldsymbol{\xi},$$

we have the following security guarantee: If the attacker has a time budget $\leq 2^t$*, then its probability of reaching a winning state is at most* $\leq 2^{-s}$*.*

We say that $(\boldsymbol{\delta}, \boldsymbol{\mu}, \boldsymbol{\xi})$ *describes a capacity region for* $\{(s,t) \ : \ s,t \geq 0\}$*. This definition implies that for a fixed* k *and time budget* 2^t*, the probability of winning is at most* $\leq 2^{-s(k,t)}$*, where*

$$s(k,t) = \max\{s \geq 0 \ : \ s\boldsymbol{\delta} + t\boldsymbol{\mu} \leq k\mathbf{1} + \boldsymbol{\xi}\}. \tag{3.11}$$

Similarly, for a time budget 2^t*, the probability of winning is* $\leq 2^{-s}$ *if a winning state is characterized by some* $k \geq k(s,t)$*, where*

$$k(s,t) = \min\{k \geq 0 \ : \ s\boldsymbol{\delta} + t\boldsymbol{\mu} \leq k\mathbf{1} + \boldsymbol{\xi}\}. \tag{3.12}$$

This definition generalizes the notion of security capacity we studied in the previous chapter for the MTD games [47], where only a single equation of the form $s + t \leq \delta \cdot k$ needs to be satisfied and where δ is defined as the "security capacity." The definition of capacity region may be of interest for other more general defender-adversarial games where a Markov model is used to characterize defender and adversarial moves.[14]

The definition of capacity region is of interest for a couple of reasons. First, it clearly describes the limitation of the adversary: The capacity region explicitly characterizes the probability of winning as a function of t and k, see (3.11). In practice, the attacker's time budget is $\leq 2^t$, limited to some "small" t, e.g., $t = 80$ or 128, and using such a fixed t makes the probability of winning only a function of k.

[14]A more general definition which captures winning states characterized by parameter vectors $\mathbf{k}$, rather than a single k, can also be formulated.

Second, the capacity region can also be used to compute the containment parameter k_c, defined as the solution $k(s,t)$ in (3.12) where s is set to 1 which corresponds to a probability of winning equal to 1/2. This allows us to understand how

$$k_c = k(1,t) = O(t) = O(\ln T_{bud})$$

depends on the parameters that describe the overall Markov model.

Third, the definition allows us to trivially compose capacity regions of Markov models corresponding to different learning rates. For instance, in practice, an attacker may use several exploits or attack vectors and use each attack vector to advance its footprint and increase the number of compromised nodes. For each attack vector i, the defender develops a learning rate $f_i(l)$. In essence, the defender plays individual games with each attack vector, and the adversary wins the overall game if the individual games lead to k_i compromised nodes such that $k = \sum_i k_i$. The following theorem makes this argument precise.

Theorem 9. *Let $f_i(l)$, $1 \leq i \leq a$, be learning rates corresponding to Markov models describing different defender-adversary games that are played simultaneously. Let $k_i(s,t)$ for the i-th game be defined as in (3.12). Let $T_{bud} \leq 2^t$ be the overall time budget of the adversary. Then the probability of reaching a state with $k_i(s + \ln a, t)$ compromised nodes in the i-th game is at most $\leq 2^{-(s+\ln a)} = 2^{-s}/a$. Since the transition probabilities in each of the Markov models are independent from the number of compromised nodes, the probability of reaching $k = \sum_{i=1}^{a} k_i(s + \ln a, t)$ is at most*

$$\leq 1 - \prod_{i=1}^{a}(1 - 2^{-s}/a) \leq 2^{-s}.$$

Notice that for 'a' simultaneous adversarial games, an exponentially small probability of winning $O(2^{-s})$ is achieved for $k = O(s + \ln a + \ln T_{bud})$. In this parallel game, if the time budget is measured in real time T, then (as before) $T_{bud} \leq Tk/\Delta$ for $k = \sum_{i=1}^{a} k_i(s+\ln a, t)$ and we require $T \leq \frac{\Delta}{k}2^t$. As a final note, $k = \sum_{i=1}^{a} k_i(s+\ln a, t)$ is dominated by $\max_{i=1}^{a} k_i(s + \ln a, t) \geq k/a$. In other words, the attacker keeps on searching for an attack vector which can be used to achieve a large fraction[15] of the desired k, and the defender tries to learn how to recognize attack vectors as fast as possible in order to contain these sufficiently.

3.4.9 Capacity Region Examples

As an example of a capacity region, we translate Theorem 7 for $1 - f(l) \leq \frac{d}{(l+2)^2}$ in terms of a capacity region:

Corollary 3. *For the learning rate* $1 - f(l) \leq \frac{d}{(l+2)^2}$, *we have a capacity region* $(\boldsymbol{\delta}, \boldsymbol{\mu}, \boldsymbol{\xi})$ *defined by 2-dimensional vectors*

$$\begin{aligned}
\boldsymbol{\delta} &= (0, q\ln 2), \text{ where } q = \left[\ln(\frac{\gamma}{pd(1-\gamma)})\right]^{-1}, \\
\boldsymbol{\mu} &= (\frac{\gamma \ln 2}{1-\gamma}, q(\ln 2)(1 + \frac{1}{1-\gamma})), \\
\boldsymbol{\xi} &= (-\frac{\gamma}{1-\gamma}, q(-\frac{1}{1-\gamma} + \ln\frac{1}{1-\gamma}).
\end{aligned}$$

As a second example of working with capacity regions, we translate (3.10) in Theorem 8 for the delayed learning:

[15]Due to the defender either starting learning after a too long initial period or, once learning starts, not learning fast enough (in order to attain the desired asymptotic).

Corollary 4. *Suppose $f(l) = 0$ for $l < L^*$ and let $(\boldsymbol{\delta}', \boldsymbol{\mu}', \boldsymbol{\xi}')$ be the capacity region corresponding to learning rate $f(l + L^*)$ as a function of l. Suppose that*

$$z = \left[\ln(\frac{1 - (1-\gamma)(1-(p+h))}{(1-\gamma)p})\right]^{-1} \geq 0.$$

Then, first, for all $s \geq \max\{0, 1/z \ln 2 - \ln(2L^*) \ln 2\}$ and $t \geq 0$ satisfying

$$s[\boldsymbol{\delta}' + L^*(\ln 2)z\mathbf{1}] + t\boldsymbol{\mu}' \leq k\mathbf{1} + [\boldsymbol{\xi}' - \boldsymbol{\delta}' + (L^*(\ln 2L^*)z - 1)\mathbf{1}]$$

and, second, for all[16] *$0 \leq s \leq 1/z \ln 2 - \ln(2L^*) \ln 2$ and $t \geq 0$ satisfying*

$$s\boldsymbol{\delta}' + t\boldsymbol{\mu}' \leq k\mathbf{1} + [\boldsymbol{\xi}' - \boldsymbol{\delta}' - (L^* + 1)\mathbf{1}],$$

we have the following security guarantee: If the attacker has a time budget $\leq 2^t$, then its probability of reaching a winning state is at most $\leq 2^{-s}$.

The corollary confirms that the containment parameter for delayed learning (for both cases) is $k_c = O(L^* + \log T_{bud})$.

3.4.10 Parametric Analysis of the Learning Rate

In this subsection, we use Algorithm 2 to evaluate $\bar{w}(k, T_{bud})$ (as an upper bound on the attacker's probability of winning i.e., making significant progress in the game) based on different learning rates corresponding to $\alpha(l)$ of the form $(l+1)^{-a}$, while $T_{bud} = 10,000$ logical timesteps is given to the adversary to play the game.

[16]Notice that $1/z \ln 2 - \ln(2L^*) \ln 2 \geq 0$ if and only if $L^* \leq \frac{1}{2}e^{1/z} = \frac{1-(1-\gamma)(1-(p+h))}{2(1-\gamma)p}$, which is large for small $(1-\gamma)p$.

Figure 3.3 depicts how $\bar{w}(k, T_{bud})$ decreases as the attack objective k increases for several values of γ, meaning that the chances of making progress in the game become overwhelmingly small for the attacker. The larger the γ, the more samples are provided to the defender's learning engine as γ is the expected rate that the defender observes adversarial moves on the network levels. As it can be seen from this plot, a proper defense mechanism (with sufficiently fast converging learning rates (e.g., $a > 1$) and large enough γ values) substantially decreases the adversary's chances of making progress in the game.

Figure 3.4 depicts how the containment parameter k_c (computed using $\bar{w}(k, T_{bud})$) depends on $\alpha(l) = (l+1)^{-a}$. For $a \geq 0.9$, $k_c \leq 20$ is very small making it impossible for the adversary to reach high-value attack objectives (e.g., total number of infected nodes in a malware propagation game).

The impact of delayed learning on the adversarial containment parameter k_c is shown in Figure 3.5 for two different cases. In the first scenario, the learning starts immediately (with no delay, i.e., $L^* = 0$), and in the second case, the learning commences after missing the first 100 attack samples (or not learning from the first 100 samples, i.e., $L^* = 100$). The defender must not wait for too long to start the learning in order to keep k_c as small as possible.

3.5 Case Studies

In this section, we walk through two examples of cyber attack-defense games as case studies to show how such games can be translated into our presented framework.

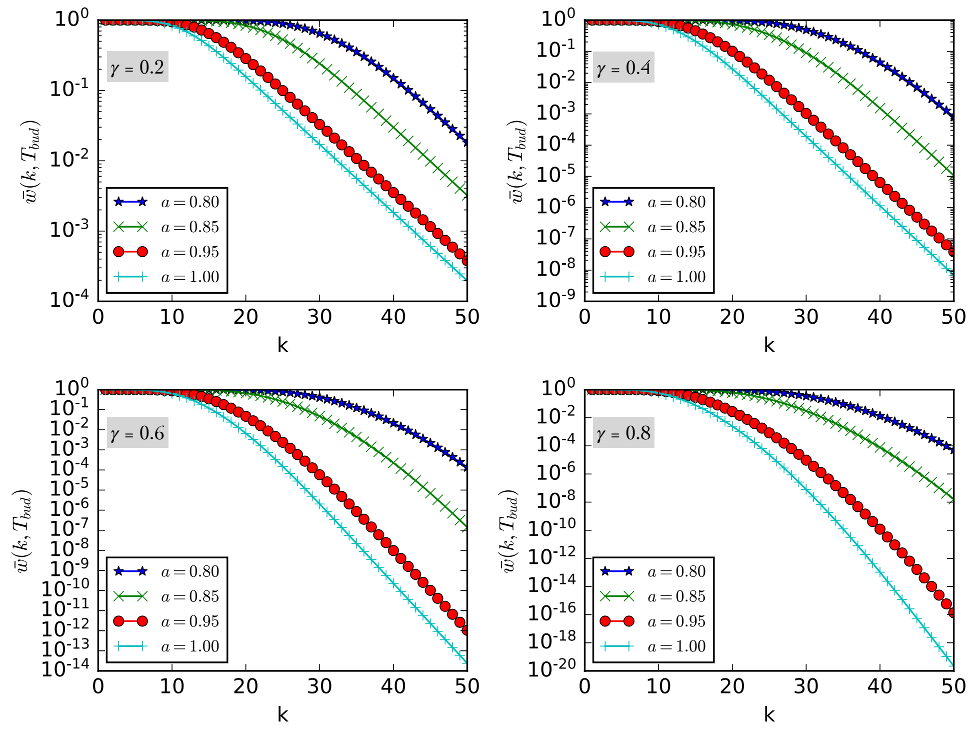


FIGURE 3.3: Upper bound on the attacker's probability of reaching state k for $\alpha(l) = (l+1)^{-a}$

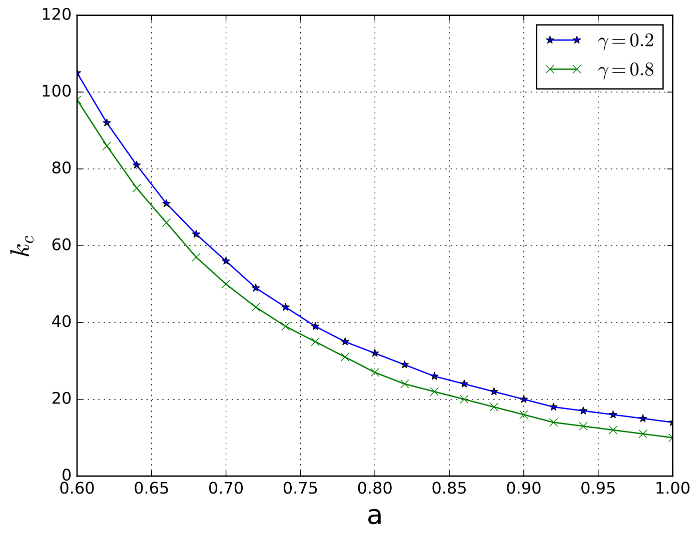


FIGURE 3.4: Containment parameter k_c based on $\alpha(l) = (l+1)^{-a}$

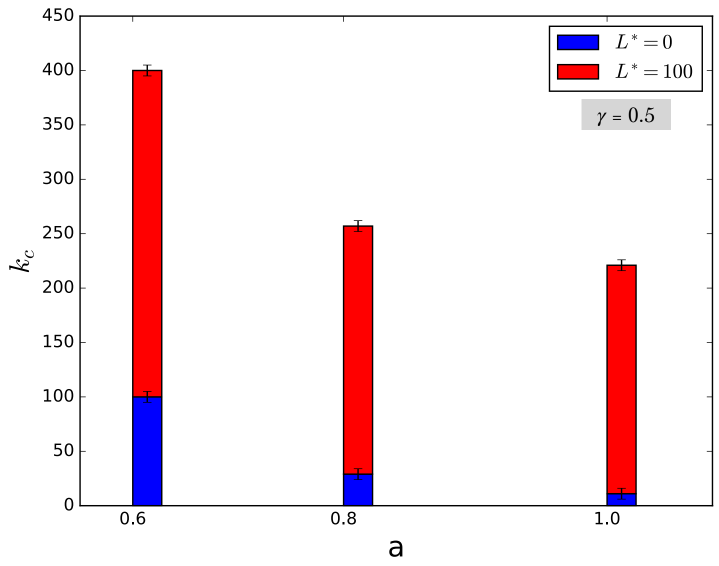


FIGURE 3.5: The impact of delayed learning on k_c

3.5.1 A Malware Propagation and Botnet Construction Game

Consider a statically addressed network of size $|\mathcal{N}|$ in which a fraction of the hosts $\mathcal{K}$ suffer from a zero-day vulnerability only known to an adversary. The attacker's objective is to locate such hosts in the network, based on a target discovery strategy σ (e.g., a random scan scheme, or a hitlist) associated with a probability p_i, and infect them during the Window of Vulnerability (WoV) time as the number of vulnerable systems is not yet shrunken to insignificance and the attacker's exploit is useful in this period. The game starts with one infected machine as the "Patient Zero." The attacker desires to take control of at least k out of $|\mathcal{K}|$ vulnerable hosts, meaning that the winning state for the adversary is defined as $(i, l) = (k, *)$. In the meantime, the defender's objective is to generate an attack signature to be able to filter next adversarial attack traffic (see Figure 3.1). The defender's knowledge of the attack can be increased in two manners: (1) if an adversary agent hits a honeypot with probability h (assuming that there exist in total $|\mathcal{H}|$ honeypots in the environment, therefore, $h = |\mathcal{H}|/|\mathcal{N}|$ for a memoryless blind scan strategy), or (2) if the attack traffic is correctly labeled as suspicious on the network (classifier) level with probability γ.[17] In either of these cases, the defender's detection rate $f(l)$ in which l is the number of so far collected attack samples will be enhanced. This function represents the probability that given l samples so far, the system blocks a new incoming malicious packet and filters it on the fly. We also consider a skillful attacker (for instance taking advantage of a polymorphic worm) as defined in previous sections, and therefore, having only a few attack samples is not enough for the purpose of signature generation.

The system state (i, l) represents the total number of infected machines and the

[17]This is a common architecture in automatic signature generation schemes (e.g., see [37,42,55,82]).

Algorithm 3 Malware Propagation Game Simulator

```
1: function SIMULATOR($\mathcal{N}, \mathcal{K}, \mathcal{H}, k, \sigma, \chi, \gamma, f(.), \vartheta$)
2:     $\mathcal{I} = \emptyset$; $l = 0$;
3:     droppingOut = False; winningState = False;
4:     while not droppingOut and not winningState do
5:         targetHost $\leftarrow$ $probe(\mathcal{N}, p_i(\sigma))$;
6:         filtered,sampled $\xleftarrow{f(l),\gamma}$ $transmit(\chi$,targetHost);
7:         if (not filtered and targetHost $\in \mathcal{H}$) or sampled then
8:             $l$ += 1; update $f(l)$;
9:             if $f(l) \geq 1 - \vartheta$ then
10:                droppingOut = True;
11:                Output *"Adversary dropped out! Defender won!"*;
12:            end if
13:        end if
14:        if targetHost $\in \mathcal{K}$ and not filtered then
15:            $\mathcal{I} = \mathcal{I} \cup$ targetHost; $\mathcal{K} = \mathcal{K} \setminus$ targetHost;
16:            if $|\mathcal{I}| \geq k$ then
17:                winningState = True;
18:                Output *"Adversary won!"*;
19:            end if
20:        end if
21:    end while
22: end function
```

total number of so far captured attack traffic (i.e., malware samples) by the defense system. The attacker's view of the system state will change if an agent's effort in infecting a new node is successful (or if it possibly loses its control over an agent). The defender's view gets updated as a consequence of discovering an adversarial move (i.e., an attack/malware sample). Algorithm 3 shows the above description of the game in which the game simulator takes $\mathcal{N}, \mathcal{K}, \mathcal{H}, k$, and an adversarial target discovery strategy σ and an exploit χ, in addition to the learning rate $f(\,\cdot\,)$, the sampling rate γ, and an acceptable "threshold" $1-\vartheta$ as the input, and outputs who wins the game. The termination rule is whether the attacker compromises its desired number of hosts k (a specific size for its botnet) or if the attacker decides not to play anymore,[18] that is $f(.) \geq 1-\vartheta$. Note that the transmit method returns a tuple, i.e., if the transmitted packet by the attacker is filtered by IDPS with probability $f(.)$ or is sampled at the flow classifier level with probability γ.

Security Analysis of Case Study 3.5.1

In Section 3.4, we investigated the role of convergence rates of $f(\cdot)$ on the attacker's chances of winning the game. We know that for a stagnating learning rate, i.e., $f(l) = 1 - \tau$, for some constant $\tau > 0$, the adversary will always win the game. Therefore, f must not stagnate unless the attacker decides to drop out of the game. To show how effective a learning mechanism could be for containing the adversary's progress in the game, we consider an attacker who is using a polymorphic worm to infect susceptible nodes in the environment and therefore construct a zombie army

[18]For instance, the attacker concludes that not enough gain can be made in a reasonable time because of not being able to make progressive moves in the game due to defender's high detection rates that keep on improving.

of a specific size. For the purpose of illustration, we consider the address space to be $|\mathcal{N}| = 2^{32}$ (the entire *IPv4*), the number of hosts vulnerable to the malware attack as $|\mathcal{K}| = 350,000$ (i.e., for a memoryless uniform scan strategy $p \approx 8.15 \times 10^{-5}$), and the number of scans performed by an infected machine to be $10,188$ scans per hour, and an initially one infected node at time zero.[19]

Now consider a learning function of the form $f(l) = 1 - 1/(l/1000 + 1)$ for the defender. In addition, let us assume $\gamma = 0.05$ that is only a small fraction of adversarial moves (in this case worm traffic) will be observed by the defender on network levels. Using Algorithm 3, we simulate the malware propagation game for the mentioned parameters and game settings. Figure 3.6 shows the average number of infected nodes (based on 100 simulation runs) in the first 24 hours of the epidemic for two cases. In the first case, we give the attacker the advantage of $p = \sup_i p_i = |\mathcal{K}|/|\mathcal{N}|$, and for the second case, we assume a state-dependent adversarial move success probability i.e., $p_i = (|\mathcal{K}| - i)/|\mathcal{N}|$. As it can be seen from this figure, this is a reasonable assumption as the total number of infected nodes are almost the same for both cases.[20] Also, notice that based on the well-accepted simple deterministic epidemic model [103] (where there is no learning during the malware propagation), we know that the total number of infected machines reaches its maximum (i.e., 350,000) in almost 24 hours of the attack initialization, however, as Figure 3.6 depicts this number remains

[19]Notice that these are the actual infamous *CodeRed1v2* worm's attack settings and parameters [71]. Although the malware was not a polymorphic one, for illustration, we use these parameters since the codeRed epidemic is well studied in the literature and the parameters make sense from a practical perspective. The same analysis can be used for more recent and advanced malware such as Stuxnet and WannaCry ransomware crypto worm, only if the attack settings and parameters are known.

[20]This is because with a proper learning mechanism, the attacker's state in the game i almost always remains very small, and therefore $p_i = (|\mathcal{K}| - i)/|\mathcal{N}| \approx |\mathcal{K}|/|\mathcal{N}|$ meaning that giving the attacker the advantage of $p = \sup_i p_i$ in our analysis is a reasonable assumption as it does not boost its progress in the game significantly.

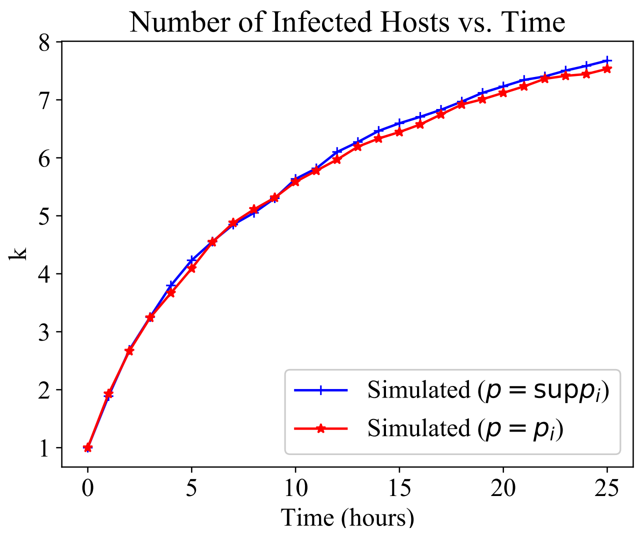


FIGURE 3.6: Simulation of malware propagation game using Algorithm 3

overwhelmingly small in k (around 8, which means that the attacker's progress is almost contained due to high-value detection rates of the defender). Notice that this plot shows simulations, and in this sense, it depicts the average case; no information regarding the worst-case scenarios can be inferred.

For the worst-case analysis, let us assume that the defender is interested in figuring out the highest number of infected nodes during a long time of window (e.g., a month) for a designated negligible probability of reaching the attack objective by the adversary (e.g., $2^{-128} \approx 10^{-38}$). Figure 3.7 depicts $\bar{w}(k, T_{bud})$ (as an upper bound of $w(k, T_{bud})$) when considering a fixed learning function of the form $f(l) = 1 - 1/(l/1000 + 1)$, while $p = 8.15 \times 10^{-5}$, and $\gamma = 0.05$ as the classifier's accuracy. Table 3.1 shows how a simple extrapolation of k can be done for a time window of a month (i.e., $T_{bud} \leq 31 \times 24 \times 10,188$ adversarial moves in our example), and setting $\bar{w}$ to 10^{-38}. By defining $k(t)$ as the 'fractional' k when 10^{-38} is reached for different t of the form $T_{bud} = 2^t$, it can be seen that the differences i.e., $k(t+1) - k(t)$

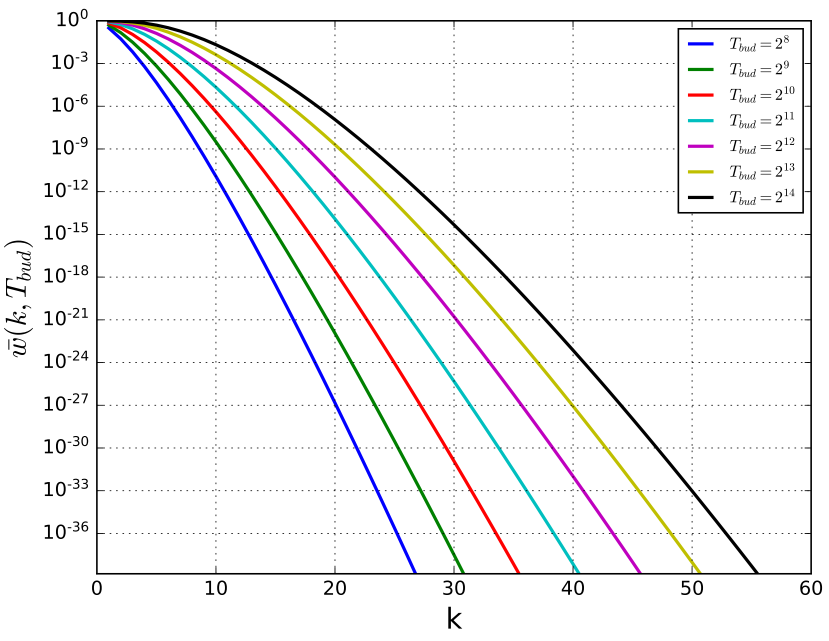


FIGURE 3.7: $\bar{w}(k, T_{bud})$ for different values of $T_{bud} = 2^t$ and a learning function of the form $f(l) = 1 - 1/(l/1000 + 1)$, $p = 8.15 \times 10^{-5}$, and $\gamma = 0.05$ as the classifier's accuracy

is decreasing for $t > 11$. We can conclude that by taking the last difference, say κ, and computing the $\log_2$ of the one month time budget t', the value of k is computed to be at most 130 infected nodes using $k = (t' - 14) \cdot \kappa + k(14)$. This means that in practice, with these attack-defense parameters, *the probability that the attacker can construct an army of* 130 *infected nodes within a one-month time window is at most* 2^{-128}, which depicts how important the learning function is when it comes to containing the adversary.

TABLE 3.1: Extrapolating k for a One-month Time Window and $\bar{w} \approx 10^{-38}$

t $(T_{bud} = 2^t)$	$k(t)$	$k(t+1) - k(t)$
8	25.61	4.01
9	29.62	4.64
10	34.26	5.03
11	39.29	5.13
12	44.42	5.02
13	49.44	4.78
14	54.22	-

3.5.2 A Moving Target Defense Game

As another example, we consider the "Multiple-Target Hiding" (MTH) game introduced in Chapter 2 with minor modifications to the game. In the studied MTH game, the adversary is interacting with a probabilistic defender taking advantage of a moving target defense strategy in which it reallocates/shuffles its resources at each time step of the game with some probability λ (for instance, consider an IP hopping strategy). The targets may represent valuable assets or vulnerable machines within the environment, as in early reconnaissance stages to the exfiltration of the data phase of an attack. Initially, the attacker has no knowledge of the targets and it has not yet found any of the targets $\mathcal{I} = \emptyset$. In order to win the game, the adversary needs to locate k out of $|\mathcal{K}|$ sensitive targets/resources distributed in the environment while there exist in total $|\mathcal{N}|$ "locations." The defender moves by reallocating targets causing the attacker to redo its search for the locations. The attacker moves by selecting one of the $\mathcal{N}$ possible locations and examining whether it corresponds to one of the targets or not. Notice that MTD strategies are usually costly for the defender as they have an immediate impact on the availability of the resources and system performance [19]. Therefore, instead of a randomized defense strategy (i.e., a probabilistic move at each time step), we consider that the defender issues a move if the number of observed adversarial actions reaches a threshold L^* based on which an attack detection signal is being generated.

Algorithm 4 MTD Game Simulator

```
function SIMULATOR($\mathcal{N}, \mathcal{K}, \mathcal{H}, k, \sigma, \gamma, L^*$)
    $\mathcal{I} = \emptyset$; $l = 0$;
    reallocateResources = False; winningState = False;
    while not reallocateResources and not winningState do
        observed,targetHost $\xleftarrow{(\gamma,h),p_i}$ $locate(\mathcal{N}, \sigma)$;
        if observed then
            $l$ += 1;
            if $l \geq L^*$ then
                reallocateResources = True; $\mathcal{I} = \emptyset$;
                Output "Defender won!";
            end if
        end if
        if targetHost $\in \mathcal{K}$ then
            $\mathcal{I} = \mathcal{I} \cup$ targetHost; $\mathcal{K} = \mathcal{K} \setminus$ targetHost;
            if $|\mathcal{I}| \geq k$ then
                winningState = True;
                Output "Adversary won!";
            end if
        end if
    end while
end function
```

The system state (i, l) represents the total number of located target machines by the attacker $\mathcal{I}$, and the total number of recognized adversarial moves so far by the defender. The attacker's view of the system state will change if it successfully finds a sensitive target based on its target discovery strategy σ, while the defender's view gets updated as a consequence of discovering an adversarial move (whether on network levels with probability γ or on host levels with probability h via a defense agent).

Algorithm 4 shows the above description of the game in which the game simulator takes $\mathcal{N}, \mathcal{K}, \mathcal{H}, k$, and an adversarial target discovery strategy σ, in addition to the sampling rate γ, and a "threshold" L^* as the input and outputs who wins the game. The termination rule is whether the attacker finds its desired number of sensitive targets k or if the defender reallocates its resources as it discovers enough amount of attack evidence in the environment which in turn pushesh the attacker back to the state 0 in the game.

Security Analysis of Case Study 3.5.2

In this case, the defender's defense strategy is to reallocate/shuffle its resources (or at least a fraction of sensitive ones) when a sufficient number of attack evidence is captured via the defense system (or in other words, the defender's objective is to generate a proper attack detection flag). Notice that the presented analysis in Section 3.4.6 can immediately be applied to this case study. As $f(l)$ remains 0 during this period, therefore, the delayed learning analysis (see Theorem 8) can be used to study how many targets will be discovered during the game by the adversary, before a threshold L^* is reached by the defender after which the defender shuffles its resources pushing the adversary to the beginning state of the game in the Markov model. The probability that more than k targets are discovered with an unlimited time budget is equal to $1 - u(k, L^*)$ which is exponentially small in k. A more precise upper bound for limited time budgets can be found by applying Theorem 6.

3.6 Concluding Remarks and Future Directions

In this Chapter, we presented a first attempt at constructing a theory of security and developing a general framework for modeling cyber attacks prevalent today, including opportunistic, targeted and multi-stage attacks while taking practical constraints and observations into consideration. To this end, we have modeled the interactions of an adversary (and possibly its agents) with a defensive system during the lifecycle of an attack as an incremental online learning game. The presented framework is comprehensive and general in the sense that it can be used to represent the dynamic interplay between the attacker and the defender in different game scenarios. In particular, instead of considering a "dummy defender" in our modeling, we allowed it to learn regarding the attack technology incrementally. As time elapses, and the defender captures more attack samples, it can reach better detection rates. As the defender's learning rate increases, the attacker's chances of making progressive moves in the game decrease, and therefore the likelihood of fulfilling its attack objective becomes smaller and smaller.

Moreover, by focusing on the most significant and tangible aspects of sophisticated cyber attacks i.e., (1) the amount of time it takes for the adversary to accomplish its mission and (2) the success probabilities of fulfilling the attack objectives, we were able to study under which circumstances the defense system can provide an effective response that makes the probability of reaching an attack objective k to be $negl(k)poly(T_{bud})$ in which T_{bud} is the attacker's (time) budget. This led us to the definition of a *security capacity region* as a metric for gauging a defensive system's efficiency from a security perspective. In particular, we show that a stagnating learning rate allows the attacker to win meaning that being able to reach its attack objective within a limited budget (e.g., time, US dollars), whether it is constructing a botnet of any specific size or locating information on a distributed number of nodes. The defender cannot wait for too long learning about a used attack vector/exploit, and once learning starts, it must continue learning with an associated detection probability converging fast enough to 1. Our security analysis gives precise recommendations for the defender, i.e., $1 - f(l) \leq O(l^{-a})$ for some $a \geq 2$ with a proof for $a = 2$ in our framework. The attacker needs to find just one attack vector/exploit for which the defender is too slow to react or too slow in learning.

An essential venue of future work is to estimate the learning rate $f(l)$ based on the number of observed malicious acts given a "worst-case adversary." Our framework lays the foundation for such work and allows to give a worst-case probabilistic bound on the maximal reached attack objective based on the estimated $f(l)$. This, in turn, provides guidance to the defender on how to allocate its resources.

A remaining challenge for future work is dealing with false positives as they can disrupt the defender's learning process, and therefore trouble the normal operation of the system. Providing bounds on the total number of false positives the defender can tolerate in its learning process is a next essential step in extending this presented framework. Also, the presented model can be fine-tuned to investigate various scenarios for a defender with a more extensive defensive action set, such as host-level cleaning and patching of network nodes. Although this can potentially lead to a more comprehensive model, we left the analysis for this case for future studies since assuming a universal host-level defense system installed on all the hosts is an unrealistic assumption in most practical scenarios.

The presented *worst-case* analysis in this chapter is beneficial when it comes to bootstrapping cryptographic protocols and having a proper reduction to common assumptions such as the learning rate of a defender. In Chapter 4, we switch gears and focus on the worm/bot-like malware propagation phenomenon. We investigate how the learning rate of a defender can lead to the containment of the malware by studying the effectiveness of content-based filtering strategies as one of the most effective and feasible solutions to tackle the spreading of such malware. From a practical perspective, we present an *average-case* analysis which is valuable for understanding how the adversary advances in its mission (without being able to bootstrap cryptographic protocols based on this analysis).

Chapter 4

A Learning-based Malware Propagation and Containment Modeling

4.1 Introduction

With the rise of advanced and novel malware and the expanding population of Internet residents, especially the explosion of connected Internet of Things (IoT) devices, new methods, and research for understanding malware behaviors and jousting with sophisticated botnets seem to be crucial. Of specific interest to us are two types of malware, worms [92], and autonomous bots [64], mainly due to their wide-ranging audience, and self-propagating characteristics. These types of malware execute a "seek-and-infect" mission strategy. For instance, a scanning worm conventionally infects a target by utilizing vulnerability exploitation, and it automatically probes the environment to *independently* propagate itself (or modified copies of itself) from the infected machine to other vulnerable hosts in a network. Consequently, a straightforward way of assembling an army of infected hosts is spreading the malicious code in the form of a scanning worm. This zombie army which is also known as a botnet is indeed a collection of compromised machines distributed over a network (usually the Internet) and controlled by its originator (known as a botmaster) through a Command and Control (C&C) structure [25]. Afterward, these infiltrated workstations (bots) can be managed for different malicious purposes including but not limited to ransom campaigns, massive spam-marketing, large-scale information harvesting and processing, click fraud, pay-per installation, and, Distributed Denial-of-Service (DDoS) attacks.

The never-ending battle between adversaries and system defenders drives both parties to improve their methods and technologies to enhance their chances of prosperity. On the one hand, attackers and network penetrators come up with new attack methods and evasion techniques to bypass (or even deny) defensive systems in any possible manner. Code obfuscation, session splicing, fragmentation attacks, string matching and DDoS attacks [51] are among the most common techniques used by the adversaries to decrease the attack detection probability or to increase the chances of attack prosperity. An armored malware (e.g., a metamorphic or polymorphic worm) buys itself more propagation time by being hard to identify and disassemble. On the other hand, a large number of studies on malware detection/prevention techniques, Intrusion Detection and Prevention Systems (IDPSes), and Honeypot (HP) technologies have been carried out to present more effective methods and designs to reach better accuracy and performance [9, 14, 31]. For instance, fast and automatic signature generation schemes are introduced to fight against even the mightiest type of malware [37, 39, 42, 55, 78, 82, 90]. Some claim that they can generate high-quality signatures (with negligible false alarm rates) based on only a few malware samples[1] usually captured via honeypots or a heuristic flow classifier.

Although there exist various number of research for worm-like malware propagation modeling, in addition to proposals for containment strategies such as content-based filtering based on automatic and distributed signature generation of the malware, the question unanswered here is *how and why does the malware spread, especially the kinds targeting a broad audience, if it is the case that we can generate high quality and reliable signatures, even for an armored malware?*

Our Approach. To answer such questions, in this chapter, we study the effectiveness of content-based filtering strategies[2] based on automatic signature generation schemes. Instead of presenting yet another malware detection/prevention methodology, we revisit the spread of malware phenomenon from a new perspective, especially the self-propagating ones such as worms and bots.

We introduce a new model for capturing the interactions of an adversary and its agents with the defensive system during the early phases of a zombie army construction. Unlike a malware propagator who usually follows a passive attack strategy for malware distribution to a broad audience, meaning that the attack technology usually remains the same after unleashing the first few samples of the malware, the defender can incrementally learn regarding the attack technology (specifically, from observed malware payloads). As time elapses, and the defender captures more attack data, it can reach better detection rates and accuracy. This learning rate indeed reflects into higher quality malware signatures which can be used for online malicious traffic filtering and hence malware containment purposes.

[1]E.g., see Hamsa [42] which only requires 100-500 malware samples in the suspicious flow pool to reach a %5 false negative rate, even for a previously unknown worm.

[2]As one of the most effective and feasible solutions to tackle the spreading of worm-like malware.

Contributions & Results. The main contributions of our work can be summarized as follows:

- Motivated by the advances in networking infrastructure and technologies such as the introduction of Software-Defined Networking (SDN), and also the widespread embrace of Automated Machine Learning (AutoML) techniques in different realms of networking and security fields, we introduce a novel malware propagation and containment modeling called the "learning-based model" suitable for today's technologies and infrastructure (e.g., Cloud environments) and more advanced malware.

- The previous outdated models are mainly focused on the attacker's strategies (specifically target discovery and scanning rates [16, 71, 75]), or in best cases, an ill-defined cleaning of the infection or patching processes [15, 21, 103]. However, we show how the increased knowledge of the defender can be taken into consideration by modeling the learning rate of the defense system as a function $f(l)$ (which is the probability of detecting and filtering a next malicious payload) where l represents the number of malicious payloads for a specific exploit so far collected.

- The model leads to the development of an automatic self-replicating malware containment strategy that prevents the spread of the malware beyond its early stages of propagation.

- We provide precise conditions on the convergence rates of a learning-based signature generation algorithm that determines whether the malware spread will ultimately settle or not.

- We derive tight upper and lower bounds on the total number of hosts the malware propagator can infect.

- The model is general enough and enables us to evaluate the effectiveness of learning-based signature generation schemes and containment strategies. It is especially suitable for "Cloud environments" where the defender (in this case the Cloud service provider) has more control, and better visibility over its network/infrastracture and the environment exposes more homogeneity by having high-level similarities in the different layers of the software stack (e.g., hypervisors, OS).

- We study learning functions of the form $1 - f(l) = (\frac{A}{A+l})^{\alpha}$ in which A is considered to slow down the learning process, and α is the amplification factor in the learning. The amplification and deceleration parameters enable the fine-tuning of the model to capture characteristics of both yesteryears malware (e.g., a monomorphic worm) and also today's more advanced ones such as polymorphic malware.

- Numerical analysis and simulation results show that regardless of an attacker's scanning rates, with a proper learning function that converges fast enough to 1, the propagation will be contained and only a negligible number of susceptible population will be infected.

We offer a fresh look at a 15-year old research area (worm epidemic modeling). Although the self-propagating class of malware may seem to be an *old*[3] type of a threat due to the decline in the "worm-like" Common Vulnerabilities Exposures (CVEs) [18],

[3]The Morris worm was the first computer worm released on the Internet on Nov. 1988, almost three decades ago.

the rise of self-propagating ransomware (e.g., the WannaCry[4] cryptoworm) and the highly sophisticated malware such as Stuxnet worm [35] attest that the network vector is one of the few attack vectors which is capable of passing the test of time. Taking advantage of the network vector and releasing the malware in a worm-like fashion, meaning that the malware is equipped with a self-propagating functionality, can provide the attackers a more potent malware which is capable of causing widespread damage. This means that the security community and professionals need to take the self-propagating class of malware more serious to first better understand such malware's life cycle and then to be able to come up with feasible solutions to tackle this problem.

We think the learning-based defense modeling is an interesting approach which deserves further attention especially by having immediate applications in the Cloud environments due to the specific characteristics of the Cloud, and the role service providers play in coordinating network defense across all their tenants.

Organization. The rest of this chapter is organized as follows. Section 4.2 briefly reviews existing worm-like malware's propagation models in addition to automatic signature generation schemes and content-based filtering of such malware. We present a new learning-based propagation model in Section 4.3 followed by the corresponding numerical analysis and simulation results of this model for various learning functions in Section 4.4, and 4.5. We conclude this chapter and discuss future directions in Section 4.6.

[4]The WannaCry ransomware incident was the first time that attackers combined the business model of ransomware along with the worm tactics, and this was the main reason the campaign was very flourishing and impactful. WannaCry was able to infect hundreds of thousands of machines suffered from unpatched MS-17-010 Windows OS vulnerabilities by the weaponization of the Eternal Blue exploit [52]. Taking advantage of this vulnerability, an attacker could launch a remote code execution by sending specially crafted messages to a Microsoft SMBv1 server [53].

4.2 Background and Related Work

The three potential solutions to mitigate a self-replicating code's rampancy are prevention, treatment, and containment strategies. The prevention and treatment schemes can be summarized as following secure software/application development procedures in hopes of having vulnerability-free software, or patching/updating vulnerable systems as soon as vulnerabilities are discovered. Unlike the containment techniques, such conducts are usually vital for pre/post incident time and not during the period of an incident. In this Section, we first review content-based filtering and automatic signature generation schemes as the most effective containment strategy [54] to hinder the spread of a malware. Moreover, we study some of the most relevant propagation models existing in the literature before presenting our model.

4.2.1 Content-based Filtering and Automatic Signature Generation Schemes

Human interventions and attack response time must be minimized in order to be able to prevent widespread infections. To this end, automatic, distributed and real-time detection and containment strategies, even for the mightiest type of worm, i.e., a polymorphic one, are introduced in the literature. Such signature-based detection and prevention techniques are of particular interest of this work as "an ounce of prevention is worth a pound of cure" meaning that such containment strategies can be very efficient for thwarting the spread of malware, especially during the early phases of a botnet construction and worm propagation, only if a high-quality signature can be generated (automatically). Such signatures can be created based on various attributes (e.g., length of the fields or invariant substrings of the byte sequences) of a class of malware, and later these signatures will be used for content-based malicious traffic filtering (even for extreme cases such as previously unknown malware or an armored one, e.g., a polymorphic worm). While designing a new automatic signature generation scheme is not the objective of this manuscript, here we review the most notable works in this area in order to better understand such schemes especially their architecture so that we can build a realistic propagation model based on a general learning algorithm (on which the signature generator is built upon) which we explain in later sections.

Autograph [37] is one of the first proposals for automatic (and optionally distributed) signature generation for a polymorphic worm, utilizing a naive portscan-based flow classifier (for TCP worms) to lessen the volume of traffic on which it performs a content-prevalence analysis. The flow classification enables Autograph to construct a suspicious flow pool on which it executes TCP flow reassembly for the payloads and outputs the most frequently occurring byte sequences across the flows as signatures. EarlyBird [78] uses a similar approach as Autograph in generating signatures by taking advantage of Rabin fingerprints.

On the other hand, Polygraph [55] forms signatures that consist of multiple disjoint content substrings to address the inefficiency of single, contiguous string-based signature generation techniques (such as Honeycomb [39], EarlyBird, and Autograph). Its underlying assumption for signature generation is that for a real-world exploit to function correctly, various invariant substrings must often be present in all alternatives of a malware payload; and these substrings typically correspond to return addresses, protocol framing, and in some cases, defectively obfuscated code. Hamsa [42] is another fast content-based signature generation method which generates multiset tokens as signatures. In comparison with Polygraph, it can provide better attack resilience and noise tolerance (accuracy). PolyTree [82] can show how the worm variants evolve and make the signature refinement task upon a new worm sample arrival quick using an incremental signature tree construction. This is based on the observation that worm signatures are related and a tree structure can properly reflect their familial resemblance which enables organizing the extracted signatures from worm samples into a tree structure. Unlike the aforementioned proposals which generate *exploit-specific* signatures, the paper in [90] presented an automatic *vulnerability-driven* network-based length-based signature generator called LESG for zero-day polymorphic worms exploiting buffer overflow vulnerabilities based on the fact that specific protocol fields in such attacks are usually longer than those in a conventional protocol usage.

4.2.2 Propagation Modeling and Estimating Size of a Botnet

In malware propagation models, possible states concerning vulnerable population during a malware attack are Susceptible (S), Infectious (I), and Recovered (R). According to a host's state at different times, and based on the transition between such states, malware propagation models can be categorized into three classes: if a host can only have one of the susceptible or infectious states and not being able to be recovered after an infection, the model is called Susceptible-Infectious (SI). If the model considers a permanent revival for an infected machine, meaning that the host remains in an immune state after the recovery process, the model is called Susceptible-Infectious-Recovered (SIR), and finally if there is the chance of being infected again after a recovery the model is called Susceptible-Infectious-Susceptible (SIS).

The classical simple epidemic model which is an SI modeling is the most commonly used model in the literature and can be simply described by the following differential equation in which k denotes the total number of vulnerable nodes to the epidemic (i.e., the susceptible population), and at each time step, the so far infectious hosts $i(t)$ propagate the epidemic with a constant rate β.

$$\frac{di(t)}{dt} = \beta i(t)[k - i(t)] \tag{4.1}$$

One of the most widely used SIR modelings in the literature is the Kermack-McKendrick model (also known as the classical general epidemic model [69]) which is an extension to the classical simple epidemic model. In this modeling, the effects of a removal process in the infectious hosts' population is taken into consideration. Consider $r(t)$ as the number of removed hosts from previously infected hosts, and ζ as the constant rate of removal, hence, the Kermack-McKendrick model can be expressed by:

$$\frac{di(t)}{dt} = \beta i(t)[k - i(t) - r(t)] - \frac{dr(t)}{dt} \tag{4.2}$$

where $\frac{dr(t)}{dt} = \zeta i(t)$.

Another proposal based on Kermack-McKendrick model is [21] in which it provides a diurnal model for different time zones for botnet propagation modeling by introducing a correction factor $\alpha(t)$ ("the diurnal shaping function") in the total number of online (available) hosts based on the time region. Therefore the diurnal worm propagation model can be represented by:

$$\frac{di(t)}{dt} = \beta\alpha^2(t)i(t)[k(t) - i(t) - r(t)] - \zeta\alpha(t)i(t) \tag{4.3}$$

The two-factor model [103] enhances the Kermack-McKendrick's model in two ways. First, it separates the removal process into two parts, one for the elimination of infectious hosts (due to cleaning) and the other for purging the susceptible population (due to patching and system updates). Second, it considers a time-varying infection rate to take worm traffic's impact on the network infrastructure into consideration (e.g., congestion in the network).

$$\frac{di(t)}{dt} = \beta(t)i(t)[k(t) - i(t) - (r(t) + q(t))] - \frac{dr(t)}{dt} \tag{4.4}$$

where $\frac{dr(t)}{dt} = \zeta i(t)$, and $q(t)$ denotes the number of removed hosts from the susceptible population.

Analytical Active Worm Propagation model (AAWP) which is a similar approach in the discrete time setting is presented in [15].

Stochastic modeling of active worms is another line of research in this area. Sellke *et al.* [75] model the propagation of the malware through a branching process, i.e., each infected machine in one generation will independently produce some random number of infected machines in the next-generation according to a fixed probability distribution. The model can determine the extinction condition of the malware and provide an upper bound on the total number of infected hosts. Rohloff *et al.* [71] used a simple stochastic density-dependent Markov jump process to model worm propagation. Each state in the chain represents the number of so far infected machines $i(t)$, in addition to the total number of remaining susceptible population $s(t)$. The model does not consider any removal/cleaning actions meaning that the total number of vulnerable hosts $k = i(t) + s(t)$ is always constant at each transition of the Markov chain. The time it takes for this chain to reach its absorbing state is calculated. For more information regarding malware propagation models we refer to the survey presented in [91].

The models mentioned above mainly focus on the treatment processes –such as patching a susceptible node or cleaning an infected machine– for vulnerable population and infection rate reduction purposes. In practice, these treatment strategies, however, are not suitable for the expeditious spread of the malware and therefore short-term reliefs for an outbreak. Moore *et al.* [54] showed that a content-based filtering strategy is the most vital containment strategy to limit the spread of the malware via isolating it from the susceptible population. Therefore, in this work, we mainly focus on the containment solutions made possible based on a defense system's learning engine which makes the propagation of the epidemic more difficult. We utilize the same notion of learning that we discussed in Chapter 3, i.e., the attacker's chances of making a progressive move in the game depend on its previous actions and the denfeder's learning process [86]. This notion of learning has been previously used in learning-based signature generation schemes for the polymorphic worm but never been applied to propagation models. The advances in network technologies (especially SDN) and AutoML provide more control and better visibility to the network defenders in addition to facilitating the automation processes. These advances motivate us to take advantage of such learning mechanisms and introduce a new *learning-based* propagation modeling suitable for today's and future's infrastructure and technology.

Moreover, the most critical shortcoming of the traditional models is that they suffer from not taking *both parties*' capabilities, actions and strategies into consideration for the model development. Therefore, these *static* models are incapable of representing the *dynamic* nature of today's network attack-defense scenarios. Our modeling differs from the previously mentioned models in the sense that as the time elapses, we can take enhanced attack/defense strategies into consideration by recognizing the players' interactions as a learning process, especially from the defense systems' perspective. This presented framework enables us to explicitly study how the learning rates of a defense system can affect future interplays of the players and their probability of success in the malware propagation game. Moreover, instead of considering an ill-defined removal/cleaning rate, we can describe where the containment process could come from by considering an incremental learning mechanism for the defender.

4.3 A Learning-based Malware Propagation and Containment Model

In this section, we study how a content-based filtering strategy–which utilizes a general learning-based signature generation scheme– can play a role in containing the propagation of a self-replicating code. More specifically, regardless of the limits on the *accuracy* of any learning-based automatic signature generation algorithm, we study the *efficacy* of such schemes in ceasing the propagation of the malware under the assumption of their constructability.

To understand the effect of a defense system's learning engine on the containment of the malware, we model the defender's learning engine (i.e., a signature generator algorithm) as a function f, which takes so far collected/observed suspicious traffic l (i.e., attack payloads) and outputs a signature for that class of malware. This generated signature is then used for online content-based traffic filtering in which each incoming malware traffic can be filtered with probability $f(l)$ (true positive) or with $1 - f(l)$ the system fails in detecting a malicious packet (or it falsely labels it as benign, i.e., false negative). Later, we briefly discuss the occurrence of false positives in the model. Moreover, we are interested in the infection modeling during the Window of Vulnerability (WoV) time as the number of vulnerable systems is not yet shrunken to insignificance and the attacker's exploit is useful in this period. This means that we do not take patching and cleaning of the susceptible and infected population into consideration.

4.3.1 Idealized Deployment, Network and Learning Model

We consider a logically centralized defense system (e.g., an IDPS) employed in the network, that is the containment system is universally deployed within the address space, and the learning engine works with all the observed/collected information regarding the worm infections (i.e., samples) and then distribute the generated signatures to defense system agents (e.g., edge routers, inline Network-based Intrusion Detection Systems (NIDS)). Similar to [89], we also consider that the learning algorithm will update its internal state after observing each new incoming batch of data, and these updates will be made over all of the so far accumulated samples. In this regard, with these updates, the learning engine might be able to find a high-quality signature over a more extended period, while without such updates, no learning would be possible. Moreover, we assume that the signature generation task on the accumulated samples is performed with no delay. Also, the signature distribution to the defense agents will be done immediately.

Automatic signature generation schemes (e.g., see [37, 42, 55, 82]) commonly take advantage of a heuristic flow classifier for constructing a suspicious flow pool in order to reduce the volume of traffic on which further analysis must be performed. Therefore, we also consider that a copy of traffic is given to a flow classifier for suspicious flow pool construction. For simplicity and generality, however, we model the defense system's classifier and traffic analysis task as a probabilistic sampling process [80] in which each malicious packet can be sampled with probability λ. Note that λ reflects the classifier's accuracy in which it is the probability that an incoming packet will be marked as suspicious traffic conditioned on the fact that it is indeed malicious. The captured payloads from the classifier are given to the signature generator engine for signature generation. The generated signatures then are used for the inline packet filtering by the defense system. This allows us not to be concerned about the transmission protocol (TCP/UDP), and in general more stealthier propagation schemes (e.g., second channel delivery) which are usually more robust against anomaly-based detection systems, as they will not trigger any events during the propagation.[5]

One very common problem among any IDPS is the inability to provide absolutely complete and accurate detection rates. This incompleteness and inaccuracy usually lead to the occurrence of a false detection of a malicious activity as benign or vice versa. Due to the existence of false alarms (especially false positives), it is a common practice not to utilize black/whitelisting prevention policies for the IDPS. This means that we do not care about *stealthy* attacks in which the attacker can conceal its real identity and disguise the source of attack traffic to decrease the chance of being located through common known practices such as IP address spoofing, use of proxies, etc. However, false positives (labeling a benign packet/activity as suspicious on the classifier level or malicious for inline filtering) play an essential role in the accuracy of the generated signatures and therefore the normal operations of the network in which the defense system is implemented. Note that in our model, false positives can

[5]Note that this is a practical assumption as the most common form of a self-replicating code found in the wild is a *self-carried* (malware payload is transferred in a packet by itself) malware which, regardless of the transmission scheme (TCP or UDP) utilizes a blind scan strategy as its target discovery (see Table 1 in [41]).

Table 4.1: Notations in this Chapter

Notation	Explanation
n	Total number of nodes in the address space
k	Total number of vulnerable nodes (i.e., susceptible population)
$i(t)$	Total number of infected nodes at time t
η	Average scanning rate of an infected machine
$f(l)$	Probability of filtering a malicious packet by the defender
λ	Probability of marking malicious traffic as suspicious by the classifier
$j(t)$	Total number of attack samples collected at time t
γ	Average rate of attack sample collection by the defense system
α	Amplification factor in learning
A	Deceleration factor in learning

occur on two levels. First, the classifier may mark a benign packet as suspicious, which eventually could lead to inaccurate signatures or taking longer times to generate a true signature. To address this problem, we consider a deceleration factor in our modeling to take the impact of such false positives in delaying the signature generation into consideration. Second, since the generated signatures are used for online filtering of incoming packets, a false positive at this level may drop a benign packet at the network level. However, notice that such incidents do not impact the accuracy of our model, i.e., the total number of infected nodes at any instance of time will not be affected.

4.3.2 The Learning-based Model

For completeness, one can apply the learning-based model (i.e., the impact of a defender's learning in filtering malicious traffic) to any of the studied models in Section 4.2.2. This allows one to include other factors such as human involvement, different time zones, cleaning the infection, and patching the susceptible population into the model. However, for brevity, simplicity, and generality, we develop the learning-based model based on the simple classical epidemic to purely study the impact of a defender's learning engine on the malware containment process.

Our goal is to show that a sufficiently increasing learning rate will stop malware from propagating and only a few number of susceptible nodes k will be infected. Consider the SI modeling represented in (4.1), as follows

$$\frac{di(t)}{dt} = \eta\frac{k - i(t)}{n}i(t),$$

where $i(t)$ represents the number of infected nodes at time t, n is the size of address space, and η is equal to the average scanning rate of an infected machine. Rather than modeling the above equation, let us assume that we already will be able to bound $i(t)$ to a number $\ll k$ so that

$$\frac{di(t)}{dt} = p \cdot i(t) \tag{4.5}$$

would be a good approximation for some constant

$$p = \eta k/n.$$

In fact this approximation gives the adversary an advantage since $\eta\frac{k-i(t)}{n}i(t) \leq pi(t)$ which makes the new differential equation (4.5) a best case scenario for the adversary.

We adapt (4.5) to include prevention of infection: If the defender gathers samples of malicious payloads at a certain rate λ, then the defender collects $\lambda{\cdot}\eta{\cdot}i(t)$ samples at time t. Notice that unlike the previous models in which the higher the scanning rate of the malware, the sooner the whole population become infected, in our modeling, however, a high scanning rate will also potentially provide more samples to the defense system which can lead to constructing a valid signature sooner and therefore holding the attack's progress. If we denote the number of samples collected up to time t by $j(t)$, and defining $\gamma = \lambda \cdot \eta$, then $j(t)$ satisfies

$$\frac{dj(t)}{dt} = \gamma \cdot i(t). \tag{4.6}$$

The defender's knowledge of how to recognize and prevent malicious payloads increases as a result of observing and collecting malicious traffic. This work does not propose a learning algorithm per se. However, we mainly provide a high-level overview of the learning process by abstracting it as a probabilistic function. We assume $f(l)$ captures this learning – with probability $f(l)$, where $l = j(t)$ is the number of samples collected so far, the adversarial payload is prevented from doing any harm. With probability $1 - f(l) = 1 - f(j(t))$, the payload may proceed and this leads to the following adjustment of (4.5):

$$\frac{di(t)}{dt} = p \cdot i(t) \cdot (1 - f(j(t))). \tag{4.7}$$

Equation (4.7) is the learning-based model and this is what we study; It shows that in addition to hitting a vulnerable target, an infected machine's endeavors to infecting a new device by submitting an ominous packet to the target must not get filtered/blocked by the defense system.

Although one can study any general form of a learning function f by plugging it in (4.7), we will analyze monotonically increasing learning rates of the form

$$1 - f(l) = \left(\frac{A}{l + A}\right)^{\alpha}$$

for some exponent α and constant A. The exponent α reflects how collected samples amplify the learning rate while constant A is included to slow down the learning process, i.e., delayed learning. The motivation behind choosing this specific learning function is twofold. First, such family of functions can potentially satisfy the convergence rate criterion (converging fast enough to 1), which guarantees the possible containment of the malware as we proved in the previous chapter. Second, this specific learning function can be adjusted to take both yesteryears' worms and today's more advanced malware's characteristics into consideration in the model. For instance, for a naive, monomorphic malware, which the signature generation is more straightforward and can be done based on only a few malware samples, a small slow-down factor A and a large enough amplification factor α can be used in the model to capture the swiftness of signature generation task. On the other hand, for a more potent malware, e.g., polymorphic/metamorphic worms, a large slow-down factor and small amplification factor can depict the hardness of signature generation task when dealing with such malware.

Substituting $1 - f(l)$ into (4.7) yields

$$\frac{di(t)}{dt} = p \cdot i(t) \cdot \left(\frac{A}{j(t) + A}\right)^{\alpha}. \tag{4.8}$$

Now notice that (4.6) expresses $i(t)$ as $i(t) = j'(t)/\gamma$ where $j'(t) = \frac{dj(t)}{dt}$ is the first derivative of $j(t)$. By denoting $j''(t)$ the second derivative of $j(t)$ and substituting these into (4.8) gives the final differential equation which we wish to solve:

$$j''(t)/\gamma = p \cdot j'(t)/\gamma \cdot \left(\frac{A}{j(t)+A}\right)^{\alpha}. \tag{4.9}$$

We are interested in bounding $j'(t)/\gamma = i(t)$. Initially,

$$\begin{aligned} j(0) &= 0 \text{ and} \\ j'(0) &= \gamma \cdot i(0) = \gamma, \end{aligned}$$

meaning that there is a single infected node within the network at time zero (patient zero) and the defender's knowledge of the attack is zero at the beginning (zero-day vulnerability/exploit).

We now present the following theorem (for the analysis and the proof see A.3) which provides lower and upper bounds on the total number of nodes the attacker can infect.

Theorem 10 (Containment Criterion). *For $\alpha < 1$, $i(t) = \Omega(t^{1-\alpha})$. For $\alpha = 1$, $i(t) = \Omega(\ln t)$. For $\alpha > 1$,*

$$i(t) \le 1 + \frac{Ap}{(\alpha-1)\gamma}.$$

Corollary 5. *In order to contain the malware from propagation, the convergence rate of a learning-based signature generation scheme must have an amplification factor $\alpha > 1$.*

Corollary 6. *For $\alpha > 1$, we have:*

- *the number of nodes that will be infected is proportional to the deceleration factor A.*

- *substituting $p = \eta k/n$, and $\gamma = \lambda\eta$, gives $i(t) \leq 1 + \frac{A}{(\alpha-1)\lambda}(\frac{k}{n})$ meaning that $i(t)$ is independent from the malware's scanning rate η.*

For the defense system, the smaller the deceleration factor and the more significant the amplification factor, the minimal will be the impact of an attack (i.e., the total number of infected machines). The environment must expose a minimum vulnerability density k/n as well.

4.3.3 Model Extensions and Limitations

In this section, we briefly discuss some possible scenarios which require further investigations or can be modeled through the same methodology presented in this work. Where possible, we provide general guidance for the curious reader on how such cases can be analyzed with a few adjustments to the model.

An adaptive adversary with multiple exploits: After observing no progress in the malware propagation mission (i.e., being contained with some limited number of agents), an adaptive adversary may decide to improve and update its attack technology. For instance, a bot herder can update the malware binary on its bots for different purposes. This includes enhancing and extending attack vectors and technologies by adding new functionalities or evasion techniques, migrating to different C&C servers, and considering recent and revised exploits. Our framework, can also capture such strong and possibly well-funded adversary. From the attacker's perspective, each new exploit χ_j (or in other words an enhanced attack technology) should be associated with some probability p_j (as each exploit may correspond to a different set of vulnerable nodes within the network). From the defender's perspective, the signature generator, if it cannot generate a single universal signature which matches all the malware's instances, must be able to generate multiple signatures each of which matches some subset of flows in the suspicious flow pool. Therefore, multiple detection rate functions and learning rates f_j should be considered for each class of malware.

A deceptive attacker capable of misleading the learning engine: There exist various attacks against signature generation schemes especially those with learning based on pattern extraction (see [27, 56, 65, 88]). Noise injection attacks in which the attacker can systematically inject noise (or deliberately crafted attack samples) in the training pool to mislead the defender's learning engine are amongst the most notable challenges for signature generation schemes in adversarial settings. In developing the learning-based model, we decided not to consider a *deceptive* adversary since in practice, a malware propagator usually follows a passive attack strategy for the propagation of the malware with a broad audience (e.g., worms/bots), meaning that the attack technology usually remains the same after unleashing the first few samples of the malware. That is the malware propagation game is not a dynamic interaction between the attacker and the defender (especially from the attacker's perspective) and although such attacks are theoretically possible, they have not yet been observed in the wild. In general, when dealing with a delusive adversary, the learning rate of the defense system may not always be positive, and the learning engine's false positive rates should be taken into consideration in the model. We leave this problem for future studies and refer the reader to [89] which studies the signature generation in adversarial settings and proves lower bounds on the number of mistakes any pattern extraction learning algorithm can take under common assumptions.

Host-level detection and prevention: When the malware resides on a machine, it surely exhibits abnormal and erratic behavior (both externals such as unusual port usage [32] and internals such as irregular system call sequences [38]) on that machine. Therefore, one possibility is taking the knowledge gained at the host-level into consideration for the purpose of attack/malware signature generation by means of monitoring events on endpoint devices through a Host-based Intrusion Detection System (HIDS). Such signatures can be generated based on the malware residual activity on the target machine or the specific exploit/vulnerability used by the adversary during the infection. Hence, a cumulative function f_h can be considered for the host-level signatures which depends on the total number of so far infected machines. Therefore, one can easily fine-tune the model to capture such defensive scenarios by multiplying the adversary's probability of success at each time step p_i with $1 - f_h(i)$ meaning that not only it must hit the right target (i.e., a vulnerable host), the malware should not be neutralized too at the host level. The f_h function gets updated once a new node becomes infected. There exist proposals for combined detection methods, i.e., taking both host-level and network-level information into consideration to fight against a worm/bot malware. For example, [95] introduced a C&C protocol-independent detection framework based on the combination of information gained from both host and network level bot activities. We decided not to count such strategies, since considering a universal host-level defense system installed on all the network devices is an unrealistic assumption in almost any practical situation. In brief, the model per se is an excellent fit for the environments where the rapid patching of tenants' applications/OS is not possible for the service provider (e.g., in the Infrastructure as a Service (IaaS) Cloud service model), i.e., the defense is purely based on network filtering. However, with only a few modifications, the model can also capture other scenarios such as host-level detections, patching, and cleaning.

Scalability and heterogeneity aspects of the problem: Implementing the learning-based solution on very large scales (e.g., the entire IP address space) is undoubtedly an arduous task or even maybe impossible. However, we think the model is a perfect fit for smaller scales such as a class B enterprise/organization network. Such environments relax the assumptions used in the model in two aspects. First, the negligible delay assumption in the signature generation and distribution process becomes more realistic when the size of the network under the watch is relatively smaller. Furthermore, the central focus of this article is on the defense aspects of the problem, and the main novelty lies in the dynamic learning of the defender. From the attack perspective, though, the model can be strengthened by weakening the homogeneity assumption and investigating the spread of the malware in more heterogeneous/arbitrary networks. However, for smaller-scale networks, especially when a single entity is in charge of providing and maintaining the infrastructure and the entire software stack on top of it (for instance, Cloud environments), the homogeneous susceptible-infection assumption is more valid in that the chances that the environment exhibits a constant vulnerability density is very high. Moreover, the logically centralized defense assumption makes more sense for signature generation and distribution due to more visibility and control over the network.

4.4 Numerical Analysis of the Model

The learning-based model is a general malware propagation model with several parameters. Notice that when the learning function is set to zero, i.e., $f(.) = 0$, and $p = \eta\frac{k-i(t)}{n}$, we have exactly the classical epidemic model. Although the effects of cleaning the infection and patching the susceptible population can be easily included in the model, we decided to ignore such conducts since they have been well studied in the literature. Also, we are interested in evaluating the impacts of learning even in the worst case scenarios for the defender (i.e., giving a leg up to the attacker by considering $p = \sup_i p_i$).

For the purpose of illustration, we consider a hypothetical polymorphic worm, propagating in an address space of size $n = 2^{32}$ (the entire IPv4 address space), while the size of susceptible hosts' population is $k = 350,000$. We assume a blind scan strategy for which each infected host performs a uniform scanning in the address space with an average rate of $10,188$ scans per hour.[6] Moreover, we consider $i(0) = 1$, and $j(0) = 0$, meaning that initially there exists one infected machine at time zero, and the defender has no knowledge regarding the attack (signature).

Although we proved upper and lower bounds on the total number of infected nodes $i(t)$ (see Theorem 10), for the general learning-based model, we cannot get closed-form solutions. Instead, we present numerical solutions of the differential equation by using Python *scipy.integrate* package.

Figure 4.1 depicts the solution of the learning-based model when the learning parameter is set to zero. As mentioned earlier, without a learning process the model is equivalent to the classical epidemic model, and the saturation in the curve depicts how the whole susceptible population will be infected as time elapses.

We now study the impact of different parameters on the total number of infected nodes in the learning-based model. Figure 4.2 depicts how the amplification factor α plays a role in the modeling. The $\Omega(t^{0.5})$ and $\Omega(\ln t)$ growth in the population of infected hosts, for $\alpha = 0.5$, and $\alpha = 1$ respectively, can be seen from this plot which

[6]These are the actual infamous codeRed1v2's epidemic parameters [71]. We use these values only for the purpose of illustrations since the malware is well studied and the parameters are already known.

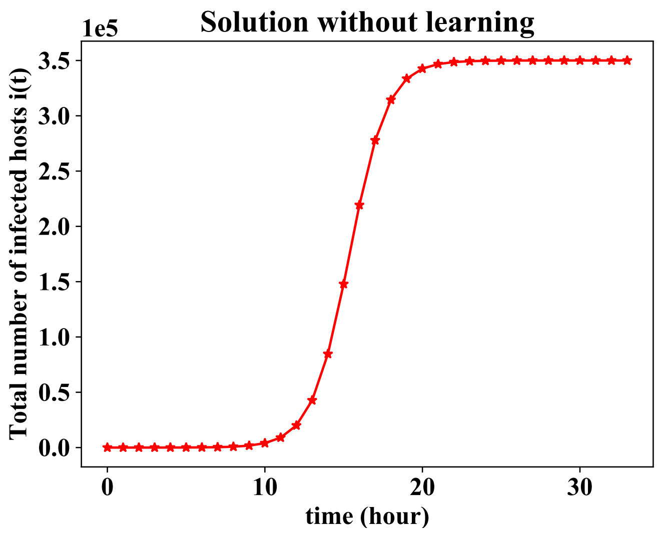


FIGURE 4.1: No learning, classical epidemic model

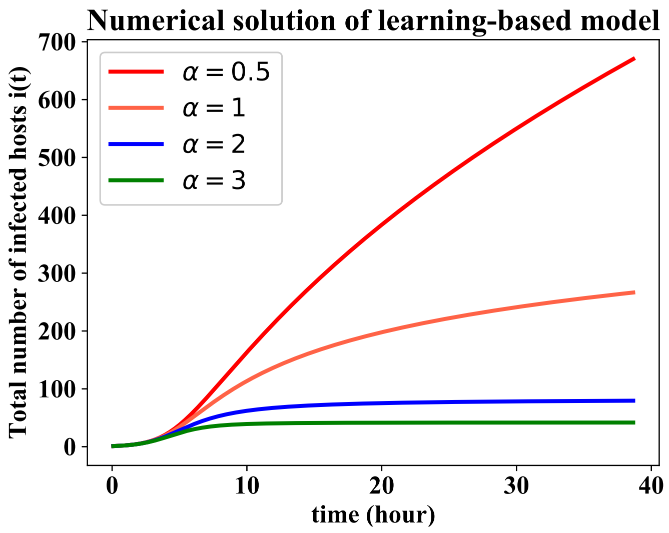


FIGURE 4.2: Numerical solution of learning-based model for $1 - f(l) = (1000/(l + 1000))^{\alpha}$ while $\lambda = 0.001$

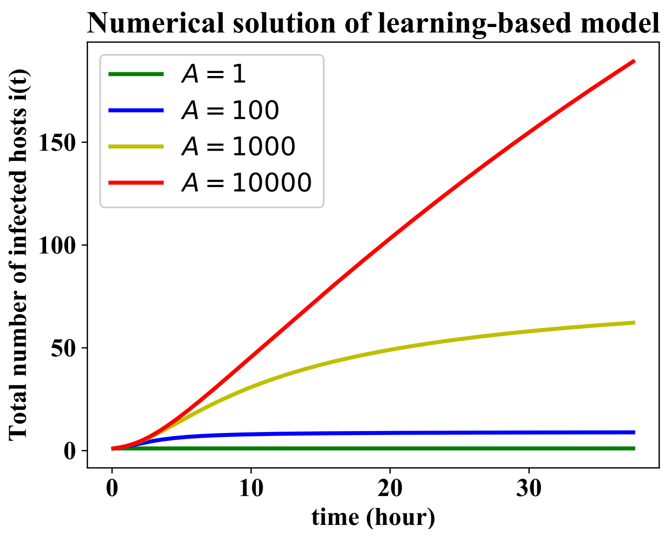


FIGURE 4.3: Numerical solution of learning-based model for $1 - f(l) = (A/(l + A))^2$ while $\lambda = 0.001$

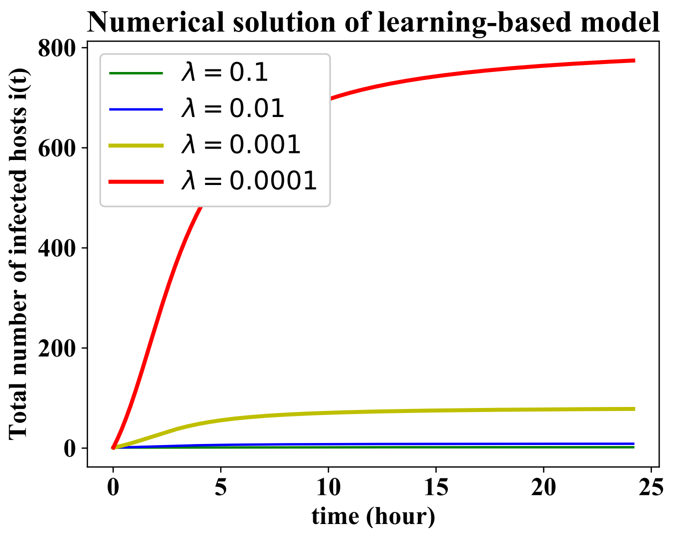


FIGURE 4.4: Numerical solution of learning-based model for $1 - f(l) = (1000/(l + 1000))^2$ and different values of λ

is consistent with Theorem 10. In addition, notice that based on the theorem, for $\alpha > 1$, the upper bound on the total number of infected nodes is $i(t) \leq 1+82/(\alpha-1)$, which gives 83, and 42 for $\alpha = 2$ and 3 respectively. The value of $i(t)$ based on the numerical solution of the differential equation for these two cases tends to 79.99 and 41.77 which is very close to the upper bound.

In order to study the impact of other parameters (i.e., A, and λ) in the learning-based model, we numerically solved the differential equation for a constant $\alpha = 2$ to see how the other parameters play a role while the amplification factor is set to an acceptable value. Figure 4.3 depicts how the deceleration factor A impacts $i(t)$. The deceleration factor in learning must remain small enough for the containment purposes meaning that the defender cannot wait for too long learning about a used exploit, and once the learning starts it must continue with an associated detection probability converging fast enough to 1. In addition, the impact of λ on the malware containment is shown in Figure 4.4. Note that λ is the probability that incoming malicious traffic is marked as suspicious by the classifier. As it can be seen from this plot, the larger the λ, the sooner the attacker's progress is contained in the environment and that is because more samples are being provided to the signature generator engine.

4.5 Simulation Results

In this section, we want to see how reasonable is the assumption of giving the adversary the advantage of $p = \sup_i p_i$ in our analysis by simulating the malware propagation process. Algorithm 5 provides a high-level overview of malware's self-propagation activity. The simulator's inputs are $\mathcal{N}, \mathcal{K}$ (set of all nodes and susceptible nodes within the address space respectively), and an adversarial exploit χ, in addition to the filtering probability $f(l)$ based on the so far collected samples l (initially set to 0 to depict a zero-day vulnerabilty/exploit situation) and the corresponding generated signatures, and the defense system's sampling rate λ. At each time step, an infected machine transmits an ominous packet which includes the attack sample to a (random) target. If the defense system does not filter the submitted packet, and the destination is indeed vulnerable, the number of infected nodes is increased by one and the target will be added to the set of so far infected machines $\mathcal{I}$. Excluding the "patient zero," initially, we assume that $|\mathcal{I}|$ is zero. If an attack sample is captured by the system, the number of samples will be increased by one consequently. Note that the transmit method returns a tuple, i.e., if the transmitted packet by an infected machine is filtered by the defense system or is sampled at the flow classifier level. The termination rule is whether the attacker compromises all the susceptible population or the defender finds a true signature, that is $f(l) = 1$.

We simulate two different scenarios. In the first scenario, we give the attacker a leg-up by considering a constant p (i.e., $p = \sup_i p_i = \eta k/n$; this requires removing line 18 in Algorithm 5, and updating the termination rule in line 19 to $|\mathcal{I}| == |\mathcal{K}|$). For the second scenario, we consider p to be a function of so far infected nodes (i.e., $p = \eta(k - i(t))/n$). For both cases, we consider $\lambda = 0.001$ and a learning function of the form $1 - f(l) = (\frac{1000}{l+1000})^2$ as it satisfies the containment requirement (i.e., $\alpha > 1$). We again use the same propagation parameters we used in the numerical analysis section only for the purpose of illustration.

Figure 4.5 shows the expected number of infected nodes among 100 simulation runs of the learning-based model. As it can be seen from this figure, the infected

Algorithm 5 Worm Propagation Simulator

```
function SIMULATOR($\mathcal{N}, \mathcal{K}, \chi, \lambda, f(.)$)
    $\mathcal{I} = \emptyset$; $l = 0$;
    trueSignature = False;
    thoroughInfection = False;
    while not trueSignature and not thoroughInfection do
        targetHost $\overset{\$}{\leftarrow}$ $probe(\mathcal{N})$;
        filtered,sampled $\xleftarrow{f(l),\lambda}$ $transmit(\chi$,targetHost);
        if sampled then
            $l$ += 1;
            update $f(l)$;
            if $f(l) == 1$ then
                trueSignature = True;
                Output "Attacker cannot make any progress!";
            end if
        end if
        if targetHost $\in \mathcal{K}$ and not filtered then
            $\mathcal{I} = \mathcal{I} \cup$ targetHost;
            $\mathcal{K} = \mathcal{K} \setminus$ targetHost;
            if $|\mathcal{K}| == 0$ then
                thoroughInfection = True;
                Output "All vulnerable targets are infected!";
            end if
        end if
    end while
end function
```

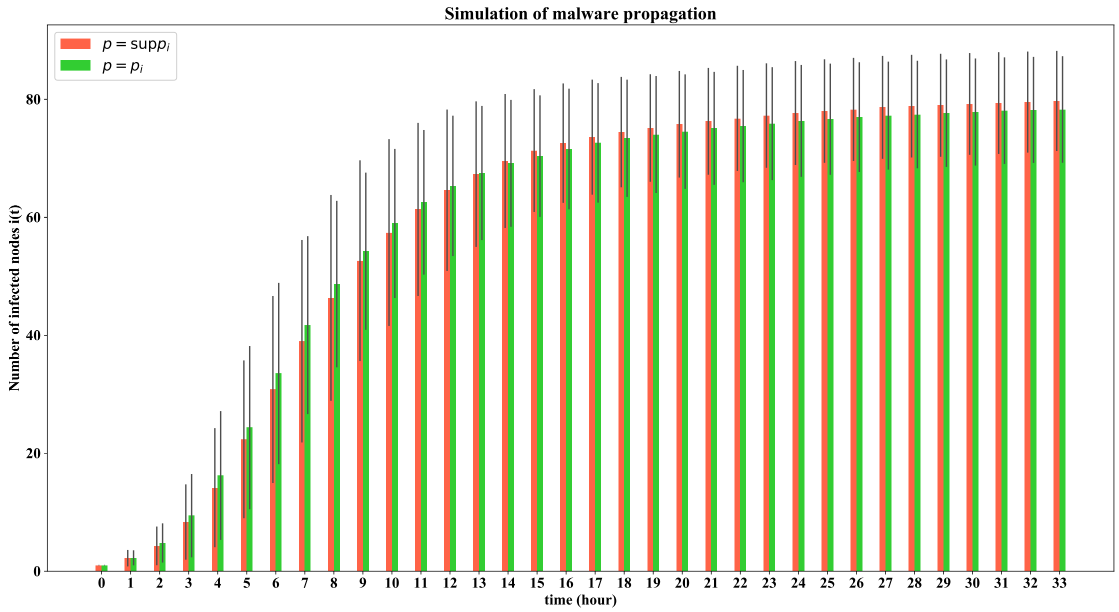


FIGURE 4.5: Simulation of malware propagation using Algorithm 5 for two cases: $p = \sup_i p_i$, and $p = p(i)$

population are almost the same for both cases of $p = \sup_i p_i$, and $p = p(i)$ meaning that $\eta\frac{k-i(t)}{n} \approx \eta k/n = p$ is indeed a good approximation, and considering a constant p in the model is a reasonable assumption. Notice that this behavior was expected since in the learning-based model as time elapses and $f(l) \to 1$, the chances of making progress for the malware propagator get smaller and smaller (causes the saturation observed in the plots). This means that the value of $i(t)$ remains relatively small in comparison to k and therefore $i(t)$ can be bounded to a number $\ll k$. Figure 4.6 on the other hand, compares the result of simulations with the numerical solution of the learning-based model. For both scenarios, the upper bound on the size of the infected population can be used as a tight estimate.

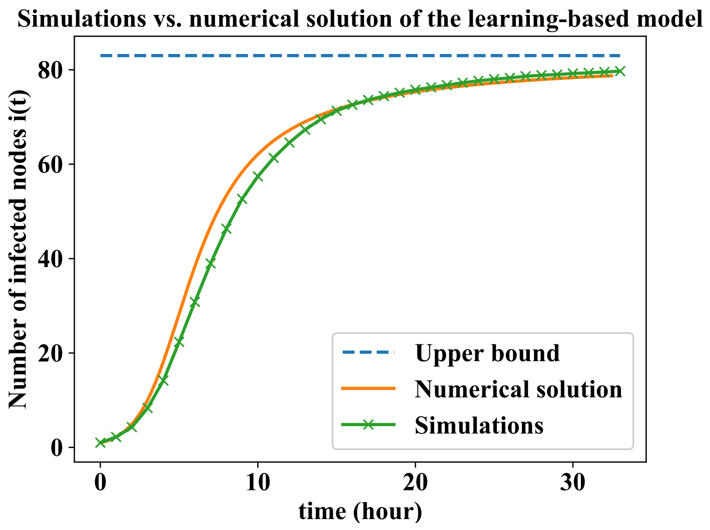


FIGURE 4.6: Comparing the total number of infected nodes based on simulations (using Algorithm 5 with $p = \sup_i p_i$) and numerical solution of the learning-based model for $1 - f(l) = (1000/(l + 1000))^2$ while $\lambda = 0.001$

4.6 Concluding Remarks and Future Directions

We revisited the spread of malware phenomenon, especially worm and bot type malware, from a new perspective. We have modeled the interactions of an adversary and its agents with a defensive system during the construction of a botnet as an incremental online learning process. By focusing on the learning rate of a defense system's learning engine and signature generation algorithm, we presented a novel and general propagation model called the *learning-based model* suitable for today's technology and infrastructure.

Unlike the existing outdated *static* modelings in this area, the learning-based model is a *dynamic* one which can capture the increased knowledge of the defender regarding the attack technology into consideration for bringing next adversarial actions into a halt.

We studied monotonically increasing learning functions for which we showed how different system parameters play a role in the malware containment process. In particular, we showed that a learning function with amplification factor $\alpha \leq 1$ allows the attacker to succeed in its zombie army construction mission. The deceleration factor in learning must remain small enough for the containment purposes meaning that the defender cannot wait for too long learning about a used exploit, and once the learning starts it must continue with an associated detection probability converging fast enough to 1. Using the learning-based model, we provided a precise bound on the convergence of a learning-based scheme that ensures that the worm propagation reaches minimal saturation in the number of infected hosts during the worm's life cycle.

The presented analysis and simulation results back up the validity of the learning-based approach and suggest that the idea worth to be investigated further and to be implemented to collect experimental results. We have shown that learning-based signature generation schemes can be very effective for malware propagation containment purposes only if their convergence rates satisfies our presented criteria. The presented security analysis recommends $1 - f(l) \leq O(l^{-\alpha})$ with a proof for $\alpha > 1$ in our framework, which were consistent with the numerical analysis and simulation results. The attacker needs to find just one exploit for which the defender is too slow to react or too slow in learning. Instead of focusing on higher scan rates, the attacker must invest in more stealthier and robust target discovery schemes and malware delivery techniques, which would not raise suspicions, leading to fewer attack samples provided to the defense system.

A possible direction for future work is to estimate the learning-based model's parameters, especially the learning rate $f(l)$, based on the size of suspicious traffic pool (i.e., number of observed malicious payloads) given a "worst-case adversary." Our framework lays the foundation for such work and allows to provide a worst-case probabilistic bound on the number of compromised nodes based on the estimated $f(l)$ which in turn, offers guidance to the system defender in how to earmark its resources and use the model for damage assessment and prediction purposes.

We believe our study can provide network security researchers and architects valuable lessons and more robust understandings of both attack and defense technologies concerning malware with a broad audience. Besides, with the advent of software-defined networking in which the entire network infrastructure can be controlled from a centralized software controller, and the embrace of AutoML in defense technologies, new generations of traditional cyber defense technologies (e.g., HPs, IDPSes, firewalls, etc.), which are more capable, scalable and secure, are already being introduced. This could be a new opportunity and a fresh start to fight against the botnet phenomenon or the spreading of the malware in general. For future work, we would like to evaluate the performance of common automatic signature generation schemes using real data from enterprise networks by porting the learning-based containment mechanism to edge and local routers, especially in a software-defined network environment.

Chapter 5

Conclusion

In this dissertation, we showed how mathematical modeling could be used to represent real-world cyber-attack situations. We investigated how such models could play a beneficial role when it comes to the design and evaluation of systems/infrastructures, especially from a *security* perspective. We presented a mathematical framework for modeling various cyber-attack and defense scenarios. We first defined a formal language for Moving Target Defense (MTD) analysis, as the state of the art defense strategy. Using this language, we were able to evaluate the effectiveness of different MTD schemes from a security standpoint by relating the time/cost spent by an adversary to the chances of winning the introduced MTD games. Motivated by the IP address hopping scheme as a well-known MTD technique, we introduced and analyzed two different MTD game scenarios called Single-Target Hiding and Multiple-Target Hiding to explain how our presented framework can be used to model real-world cyber scenarios. More importantly, we investigated how the composition of MTD schemes could be beneficial in that we proved that layered MTD games could provide larger security capacity and, therefore, more robust security guarantees for a system.

We extended our Markov-based framework to other cyber scenarios and further investigated a general game of consequences in multi-stage and APT-like attacks, in which the defender could learn from the observed adversarial actions and therefore bring next attacker moves in the game to a standstill with a higher probability. We defined the notion of security capacity region for cybersecurity games in which we related the most important aspects of cyberattacks, i.e., success probabilities and the time it takes to reach the attack objective(s) in the game. The presented *worst-case* analysis is beneficial when it comes to bootstrapping cryptographic protocols and having a proper reduction to common assumptions such as the learning rate of a defender.

Lastly, we introduced a new malware propagation modeling and containment strategy called the "learning-based model" in which we showed how the spread of the malware could be settled based on the learning rate of the defender and various attack-defense parameters. From a practical perspective, the presented *average-case* analysis is valuable for understanding how the adversary advances in its mission (without being able to bootstrap cryptographic protocols based on this analysis).

All in all, we showed how mathematical approaches and modeling could be helpful in studying the essential aspects of today's cybersecurity problems. Utilizing mathematical modeling, we can have a better understanding of system vulnerabilities, attack vectors, points of failure, and in general, the life cycle of a cyber attack. Besides, one could analyze various defense mechanisms/strategies, provide rigorous security guarantees for a system, perform a cost-benefit analysis by finding the best attack-response strategies in different scenarios, and also define what it means for a system/infrastructure/strategy to be *secure*.

Appendix A

Proofs and Math Derivations

A.1 Proofs of Theorems in Chapter 2

A.1.1 Adversary's Winning Probability of an M-MTD Game – Proof of Theorem 1

In order to prove Theorem 1, we give the adversary the advantage to initially start in a state drawn from the stationary distribution π rather than starting in the state 0 which represents the worst starting state for the adversary. For π we prove the following lemma:

Lemma 2. *If the MTD game starts in a state drawn from the stationary distribution π, then E, the expected number of times the attacker enters state k within the first T time steps, is equal to $E = T \cdot \pi(k)$.*

Proof. Let Z be the random variable representing the sequence of T states entered in the first T time steps, i.e., if $Z = (z_1, \ldots, z_T)$ then in the i-th time slot the game entered state z_i. Let X_i be the indicator random variable with $X_i = 1$ if $z_i = k$ and $X_i = 0$ if $z_i \neq k$. By the definition of expectation

$$\begin{aligned}
E &= \sum_{(z_1,\dots z_T)} Prob[Z = (z_1, \dots z_T)] \cdot |\{i : z_i = k\}| \\
&= \sum_{(z_1,\dots z_T)} Prob[Z = (z_1, \dots z_T)] \cdot \sum_i X_i \\
&= \sum_i \sum_{(z_1,\dots z_T)} Prob[Z = (z_1, \dots z_T)] \cdot X_i \\
&= \sum_i Exp[X_i].
\end{aligned}$$

Notice that $Exp[X_i]$ is equal to the probability that the game enters state k during the i-th time slot. Since π is the stationary distribution and the game is assumed to start from a state drawn from π, this probability is equal to the k-th entry of $\pi M^i = \pi$, i.e., $\pi(k)$. This proves $E = \sum_i Exp[X_i] = T \cdot \pi(k)$.

Lemma 3. *Let E be the expected number of times the attacker enters state k within the first T time steps. Then, the probability that the attacker wins in the first T time steps is $\leq E$.*

Proof. Let $p(j)$ be the probability the attacker enters state k exactly j times in the first t time steps. By the definition of expectation, $E = \sum_j p(j) \cdot j$. Now notice that the probability that the attacker wins in the first T time steps is equal to $\sum_{j=1}^{T} p(j) \leq \sum_{j=0}^{T} p(j) \cdot j = E$.

The two lemmas together prove[1] Theorem 1.

[1]We notice that the upper bound is close to the actual probability for $T \cdot \pi(k) \ll 1$ since (1) the assumed stationary distribution has its probability mass at and around the initial state 0 for well designed MTD strategies and (2) $\sum_{j=1}^{T} p(j) \approx \sum_{j=0}^{T} p(j) \cdot j$ as $p(j) \ll p(1)$ for $j > 1$.

A.1.2 Cost Analysis – Proof of Theorem 2

We prove Theorem 2: Let $A_{i,j}$ represent the probability the adversary moves in state i and reaches state j as a consequence of the move. Let the real number $c_{i,j} \geq 0$ indicate the adversarial cost associated to $A_{i,j}$. Then

$$\alpha = \sum_{i,j} \pi_i A_{i,j} c_{i,j} \tag{A.1}$$

is a convex combination of costs $c_{i,j}$ (since $\sum_{i,j} \pi_i A_{i,j} \leq \sum_{i,j} \pi_i M_{i,j} = \sum_j \pi_j = 1$). We need to prove that the probability the attacker wins the M-MTD game by paying at most C is

$$\leq \frac{C \cdot \pi_k}{\sum_{i,j} \pi_i A_{i,j} c_{i,j}}.$$

We first introduce a mapping of matrix M to a new matrix M^* which we can analyze using our theorem but has all the necessary cost information embedded: First, we split each transition from state i to state j with probability $M_{i,j}$ into two transitions. One from i to j with probability $A_{i,j}$ and the other from i to j with probability $D_{i,j} = M_{i,j} - A_{i,j}$: $A_{i,j}$ represents the probability that the adversary moves in state i and reaches state j as a consequence of that move, while $D_{i,j}$ represents the probability of a defender's move or no move (if $i = j$) from state i to state j. Second, let $c_{i,j} \geq 0$ be a real number representing the cost for the adversary if he moves from i to j with probability $A_{i,j}$. Let $\epsilon \geq 0$ (later in the proof we will take the limit $\epsilon \to 0$) and define

$$C_{i,j} = \lceil \frac{c_{i,j}}{\epsilon} \rceil \in \mathbb{N}$$

We transform each edge from $i \to j$ with probability $A_{i,j}$ into a path $i \to (i,j;1) \to (i,j;2) \to \ldots \to (i,j;C_{i,j}) \to j$ with probabilities as depicted in Figure A.1. The

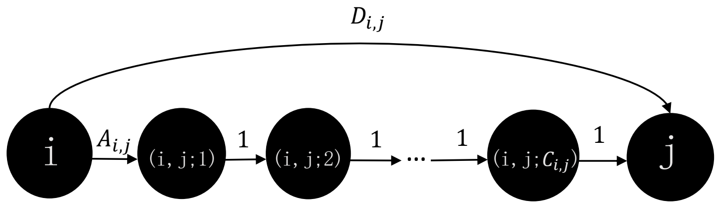


FIGURE A.1: Each transition $M_{i,j}$ from $i \to j$ will be replaced with above transition from state i to j

transition $i \to (i,j;1)$ has probability $A_{i,j}$. If $C_{i,j} = 0$, then i is directly connected to j without intermediary nodes and $i \to j$ is transitioned with probability $A_{i,j}$ by the adversary. We denote the transformed matrix by M^*.

Lemma 4. *If M has an stationary probability distribution π, then M^* has an stationary distribution π^* defined by:*

$$\begin{aligned} \pi_i^* &= \frac{\pi_i}{1+\alpha} \textit{ and} \\ \pi_{i,j;h}^* &= \frac{\pi_i \cdot A_{i,j}}{1+\alpha} \textit{ for } 1 \le h \le C_{i,j}. \end{aligned}$$

Proof. From $\pi^* = \pi^* M^*$ and the definition of M^* we infer the following equations:

$$\begin{aligned} \pi_{i,j;1}^* &= \pi_i^* \cdot A_{i,j}, && \text{if } C_{i,j} \ge 1, \\ \pi_{i,j;h+1}^* &= \pi_{i,j;h}^*, && \text{for } 1 \le h \le C_{i,j}, \\ \pi_j^* &= \sum_i \pi_i^* D_{i,j} + \sum_{\substack{i \\ C_{i,j} \ne 0}} \pi_{i,j;C_{i,j}}^* + \sum_{\substack{i \\ C_{i,j}=0}} \pi_i^* \cdot A_{i,j}. \end{aligned}$$

By combining the above equations we have,

$$\begin{aligned}\pi_j^* &= \sum_i \pi_i^* D_{i,j} + \sum_{\substack{i \\ C_{i,j}\neq 0}} \pi_{i,j;1}^* + \sum_{\substack{i \\ C_{i,j}=0}} \pi_i^* \cdot A_{i,j} \\ &= \sum_i \pi_i^* D_{i,j} + \sum_i \pi_i^* \cdot A_{i,j} = \sum_i \pi_i^* M_{i,j}\end{aligned}$$

We conclude that the vector $(\pi_1^*, \pi_2^*, ...)$ is proportional to vector π and this proves $\pi_i^* = \pi_i/\rho$ for some ρ. As a result

$$\pi_{i,j;h}^* = \pi_{i,j;1}^* = \pi_i^* \cdot A_{i,j} = \frac{\pi_i \cdot A_{i,j}}{\rho} \text{ for } 1 \leq h \leq C_{i,j}.$$

The proportionality factor ρ satisfies

$$\begin{aligned}1 &= \sum_i \pi_i^* + \sum_{\substack{i,j,h \\ C_{i,j}\neq 0 \\ 1\leq h\leq C_{i,j}}} \pi_{i,j;h}^* \\ &= \sum_i \frac{\pi_i}{\rho} + \sum_{\substack{i,j,h \\ C_{i,j}\neq 0 \\ 1\leq h\leq C_{i,j}}} \frac{\pi_i \cdot A_{i,j}}{\rho} \\ &= \sum_i \frac{\pi_i}{\rho} + \sum_{i,j} \frac{\pi_i \cdot A_{i,j} \cdot C_{i,j}}{\rho}\end{aligned}$$

Together with $\sum_i \pi_i = 1$ and (A.1), this proves $\rho = 1 + \alpha$ and the lemma follows.

In order to prove the theorem we introduce new random variables: Let T_M be the random variable indicating the number of transitions $i \xrightarrow{D_{i,j}} j$, $i \xrightarrow{A_{i,j}} (i, j; 1)$ (if $C_{i,j} \geq 1$) and $i \xrightarrow{A_{i,j}} j$ (if $C_{i;j} = 0$), made before the adversary wins the MTD-game corresponding to M^*.

Let $T_C = \sum_{i,j} N_{i,j} \cdot C_{i,j}$ be the random variable indicating the number of all other transitions where $N_{i,j}$ represents the random variable counting the number of times the path$(i,j;1) \xrightarrow{1} (i,j;2) \xrightarrow{1} \ldots \xrightarrow{1} (i,j;C_{i,j})$ is traversed. Notice that $T = T_M + \sum_{i,j} N_{i,j}$ is the random variable representing the number of transitions in the M-MTD game before the attacker wins the game.

Let the random variable C_R represent the real cost of the adversary before he wins the MTD-game corresponding to M. Notice that, for random variables $N_{i,j}$ in the MTD-game corresponding to M^*,

$$C_R = \sum_{i,j} N_{i,j} \cdot c_{i,j}.$$

This shows how the MTD games based on M and M^* are related.

We derive

$$\begin{aligned} T_C &= \sum_{i,j} N_{i,j} \cdot C_{i,j} = \sum_{i,j} N_{i,j} \cdot \lceil \frac{c_{i,j}}{\epsilon} \rceil \\ &= \frac{1}{\epsilon} \sum_{i,j} N_{i,j} \cdot (\lceil \frac{c_{i,j}}{\epsilon} \rceil \cdot \epsilon) \\ &= \frac{1}{\epsilon} \sum_{i,j} N_{i,j} \cdot c_{i,j} + \frac{1}{\epsilon} \sum_{i,j} N_{i,j} \cdot (\lceil \frac{c_{i,j}}{\epsilon} \rceil \cdot \epsilon - c_{i,j}) \\ &= \frac{C_R}{\epsilon} + E \text{ where } 0 \le E \le \sum_{i,j} N_{i,j} \end{aligned} \tag{A.2}$$

Since $T = T_M + \sum_{i,j} N_{i,j}$ represents the number of transitions in the M-MTD game before the attacker wins the game, we can use Theorem 1 in combination with

(A.2) and derive for all t,

$$\begin{aligned}
\frac{C\cdot\pi_k^*}{\epsilon} &\geq Prob(T_M+T_C\leq\frac{C}{\epsilon})=Prob(T_M+\frac{C_R}{\epsilon}+E\leq\frac{C}{\epsilon}) && \text{(A.3)}\\
&= Prob(C_R\leq C-\epsilon(T_M+E))\\
&\geq Prob(C_R\leq C-\epsilon t|T_M+E\leq t)Prob(T_M+E\leq t)\\
&= Prob(C_R\leq C-\epsilon t)-\\
&\quad Prob(C_R\leq C-\epsilon t|T_M+E\geq t)Prob(T_M+E\geq t)\\
&\geq Prob(C_R\leq C-\epsilon t)-Prob(T_M+E\geq t)\\
&\geq Prob(C_R\leq C-\epsilon t)-Prob(T_M+\sum_{i,j}N_{i,j}\geq t)\\
&= Prob(C_R\leq C-\epsilon t)-Prob(T\geq t). && \text{(A.4)}
\end{aligned}$$

If we choose $\epsilon=\frac{1}{t^2}$ and let $t\to\infty$ then (A.4) tends to $Prob(C_R\leq C)$.

We now analyze upper bound (A.3) by using Lemma 4:

$$\frac{C\cdot\pi_k^*}{\epsilon}=\frac{C\cdot\pi_k}{\epsilon(1+\alpha)},\ \text{where}$$

$$\begin{aligned}
\epsilon(1+\alpha) &= \epsilon+\sum_{i,j}\pi_iA_{i,j}C_{i,j}\cdot\epsilon=\epsilon+\sum_{i,j}\pi_iA_{i,j}\lceil\frac{c_{i,j}}{\epsilon}\rceil\epsilon\\
&= \epsilon+\sum_{i,j}\pi_iA_{i,j}c_{i,j}+\sum_{i,j}\pi_iA_{i,j}(\lceil\frac{c_{i,j}}{\epsilon}\rceil\epsilon-c_{i,j})
\end{aligned}$$

Hence,

$$\epsilon(1+\alpha) - \sum_{i,j} \pi_i A_{i,j} c_{i,j} \leq \epsilon + \sum_{i,j} \pi_i A_{i,j} (\lceil \frac{c_{i,j}}{\epsilon} \rceil \epsilon - c_{i,j})$$
$$\leq \epsilon + \sum_{i,j} \pi_i M_{i,j} \epsilon = \epsilon + \sum_j \pi_j \epsilon = 2\epsilon.$$

This proves that, for $\epsilon = \frac{1}{t^2}$ and $t \to \infty$, $\epsilon(1+\alpha) \to \sum_{i,j} \pi_i A_{i,j} c_{i,j} = \alpha$ and the theorem follows:

$$Prob(C_R \leq C) \leq C \cdot \pi_k / \alpha.$$

A.1.3 Upper Bound on the Winning Probability of a Layered MTD Game – Proof of Theorem 3

Proof. To analyze this composition game, we slightly modify the composition game by giving the adversary two extra advantages. First, the attacker is also able to play in M_j if it is currently already in a winning state in M_j (as we will explain this is not really an advantage that helps the adversary). Second, the defender can only play game M_j if the attacker is also allowed to play that game. This restriction of the defender implies that in the modified composition game only if the attacker is in a winning state in M_j or in M_{j+1}, then the defender and attacker are able to play in M_{j+1}; if the attacker is not in a winning state in M_j or M_{j+1}, then M_{j+1} is put "on hold" as both the defender and attacker do not issue moves in M_{j+1}. The modified composition game is formally defined as follows:

Definition 7. *Let M_j with winning (for the adversary) states k_j, $0 \leq j \leq f$, represent a sequence of MTD games. We define the $(M_0, \ldots, M_f)$-MTD game as the following composition of the individual M_j-MTD games:*

1. *The M_0-MTD game is played in every time slot.*

2. *For $0 \leq j < f$, the M_{j+1}-MTD game is only played in a time slot if*

    - *the M_j-MTD game is in its winning state k_j at the start of the time slot or*

    - *the M_{j+1}-MTD game is in its winning state k_{j+1} at the start of the time slot.*

3. *The composed game is won by the adversary as soon as it enters the winning state k_f of the M_f-MTD game.*

We say that the M_j-MTD game is played at level j in the composition of the M_j-MTD games.

In order to get a feeling for how a $(M_0, \ldots, M_f)$-MTD game behaves we consider the toy example of Figure A.2 which depicts the Markov chains of a sequence of (in our example equivalent) games M_0, M_1, and M_2 (with similar time scales) at *levels* 0, 1, and 2. A sample run of a simulation of the (M_0, M_1, M_2)-MTD game is illustrated in Figure A.3 in which after 439 time steps the attacker wins. Notice that the defender and attacker can play moves in games at different levels at the same time. E.g., if at the start of a time slot the attacker is not in the winning position in the game at level 0 but is in the winning position in the game at level 1, then during the time slot it can play moves in both M_0 and M_2, but not in M_1.

From the attacker's point of view (in Definition 7), if it is in the winning state of the game at level $j+1$, then playing the game at level $j+1$ means not doing anything at all as it does not want to lose its winning position. So, equivalently, the adversary only plays the game at level $j+1$ if it is in the winning state of the game at level

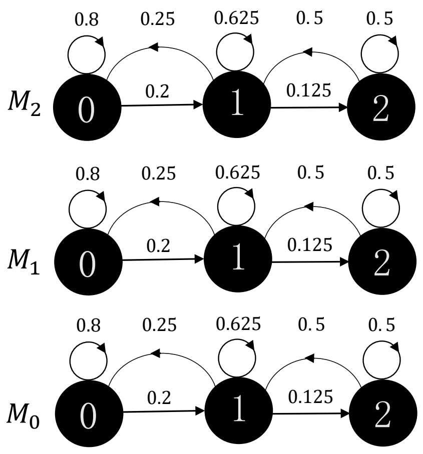


FIGURE A.2: An example of a composed game made of 3 similar levels

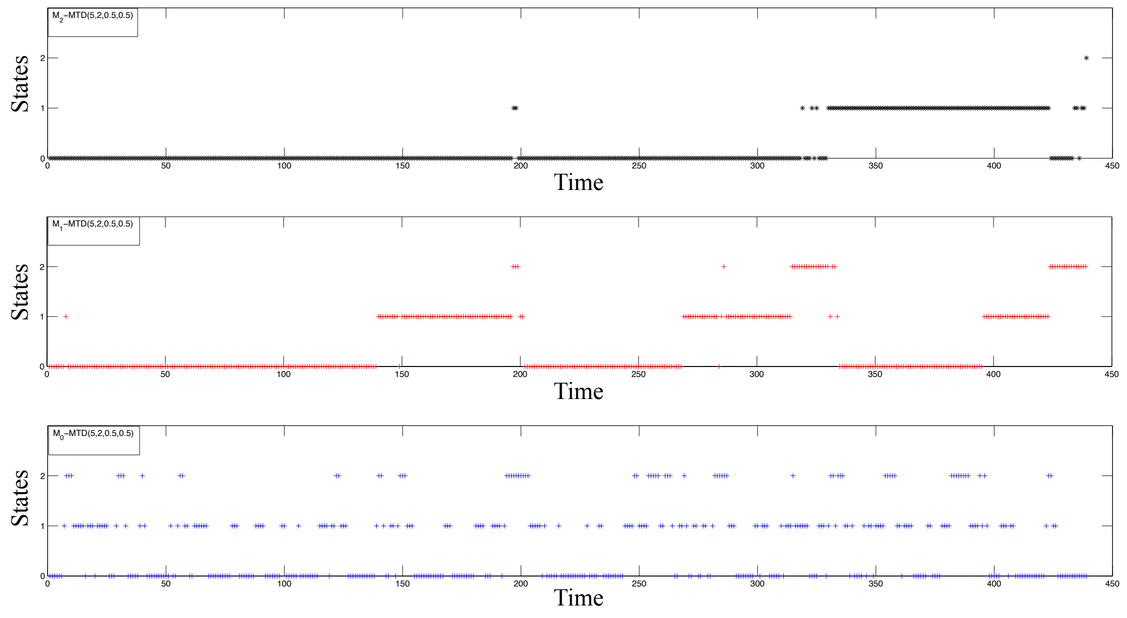


FIGURE A.3: Simulated modified composed game example (sample run) where for each level $N = 5, K = k = 2$, and $\mu = \lambda = 0.5$

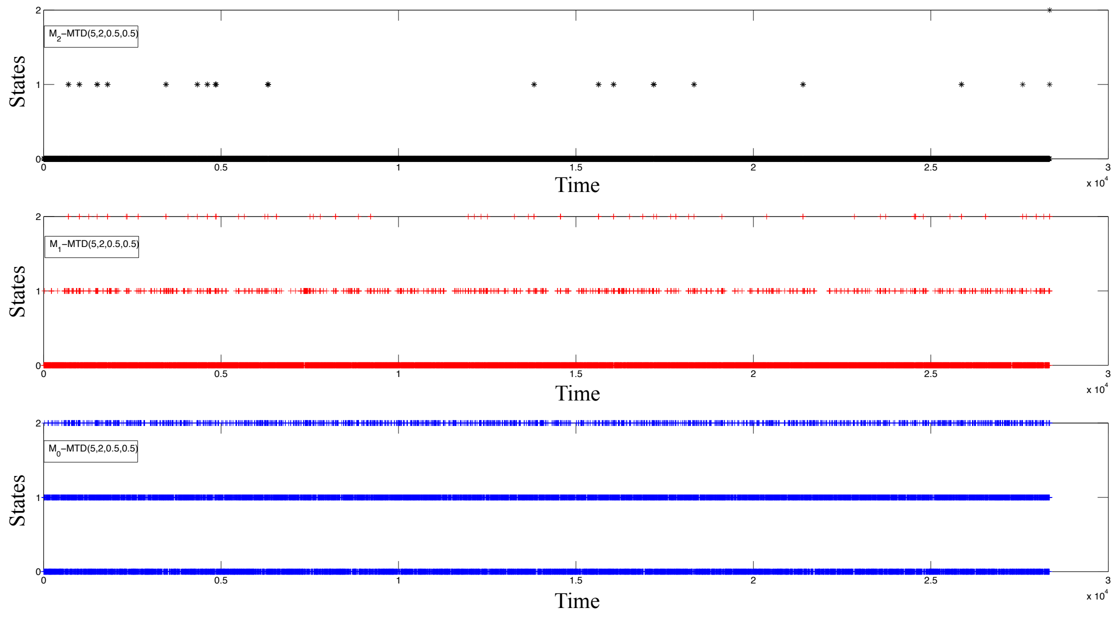


FIGURE A.4: Simulated composed game example (sample run) where for each level $N = 5, K = k = 2$, and $\mu = \lambda = 0.5$

j. On the other hand, from the defender's point of view, the definition unnecessarily restricts it: The defender stops playing the game at level $j+1$ if the game at level j is not in state k_j and at the same time the game at level $j+1$ is not in state k_{j+1}. This is an advantage for the adversary which does not happen in practice i.e., the defender does not know the states of the various games/levels, and it will continuously play its MTD strategy at all times.

The $(M_0, \ldots, M_f)$-MTD game is only introduced for proving Theorem 3: We now show that the theorem holds for $(M_0, \ldots, M_f)$-MTD games which implies the theorem must hold for layered MTD games as these represent adversaries without additional advantages.

Let E_j be the expected number of times the attacker enters state k_j in the game at level j within the first T time steps. Let $p_j(t)$ be the probability that the attacker enters state k_j exactly t times in the game at level j in the first T time steps. Let $q_j(w)$ be the probability that the game at level j is played during exactly w time slots in the first T time steps. We define $p_j(t|w)$ as the probability that the attacker enters state k_j exactly t times in the game at level j in the first T time steps conditioned on the assumption that the game at level j is played during exactly w time slots in the first T time steps. Notice that $p_j(t) = \sum_{w=t}^{T} q_j(w)p_j(t|w)$.

We derive

$$\begin{aligned} E_{j+1} &= \sum_{t=0}^{T} p_{j+1}(t) \cdot t \\ &= \sum_{w=0}^{T} q_{j+1}(w) \sum_{t=0}^{w} p_{j+1}(t|w) \cdot t. \end{aligned}$$

Notice that Lemma 2 applies directly to the expectation $\sum_{t=0}^{w} p_{j+1}(t|w) \cdot t$: By Lemma 2,

$$\sum_{t=0}^{w} p_{j+1}(t|w) \cdot t = w \cdot \pi_{j+1}(k_{j+1}).$$

Hence,

$$E_{j+1} = \sum_{w=0}^{T} q_{j+1}(w) \cdot w \cdot \pi_{j+1}(k_{j+1}).$$

By the definition of composition, the game at level $j+1$ is only played at time slot t when the game at level j is in k_j at the start of time slot t or when the game at level $j+1$ is in k_{j+1} at the start of time slot t. Let t_0 be a time slot during which the game at level $j+1$ transitions from k_{j+1} to a state $\neq k_{j+1}$. Let t_1 be the most recent time slot before t_0 during which the game at level $j+1$ transitioned from a state $\neq k_{j+1}$ to state k_{j+1}. This means that (a) the game at level $j+1$ enters (or remains in) state k_{j+1} at time slots $t_1, t_1+1, \ldots, t_0-1$, and (b) the game at level

$j+1$ was played in time slot t_1 when at the start of time slot t_1 the game at level $j+1$ was not in k_{j+1}.

By the definition of composition and choice of t_0, (a) implies that the game at level $j+1$ is played during time slots $t_1+1,\ldots,t_0$; at the start of each of these time slots the game at level $j+1$ is in k_{j+1}; during time slot t_0 the game at level $j+1$ exits state k_{j+1}. This process is described by a Poisson process with parameter λ_{j+1}, i.e., the probability that the game at level $j+1$ behaves this way is equal to $(1-\lambda_{j+1})^{t_0-1-t_1}\lambda_{j+1}$. Therefore the expectation of t_0-t_1 is equal to $1/\lambda_{j+1}$.

By the definition of composition, (b) implies that the game at level j must have been in k_j at the start of time slot t_1. It may happen that during the Poisson process as described above the game at level j enters state k_j once more in which case the current Poisson process proceeds and a new Poisson process is not spun off. In the worst case analysis for the defender, however, each time the game at level j is in k_j at the start of a time slot a Poisson process (as described above) is spun off (which adds more time slots during which the game at level $j+1$ is being played). We conclude that the expected number of times the game at level $j+1$ is played is at most the expected number of times the game at level j is played times $1/\lambda_{j+1}$. That is,

$$\sum_{w=0}^{T} q_{j+1}(w)\cdot w \le E_j/\lambda_{j+1}.$$

This leads to the recurrence

$$E_{j+1} = E_j \cdot \pi_{j+1}(k_{j+1})/\lambda_{j+1}$$

with $E_0 = T \cdot \pi_0(k_0)$ by Lemma 2. Its solution is

$$E_f = T \cdot \pi_0(k_0) \cdot \prod_{j=1}^{f} \pi_j(k_j)/\lambda_{j+1}.$$

Application of Lemma 3 proves the theorem.

Since an $(M_0, \ldots, M_f)$-MTD game assumes a much weaker defender than the corresponding layered MTD game, the layered MTD game may have a security capacity which is significantly larger than (2.7). In the example of Figure A.3 the average (simulated) time to win the (M_0, M_1, M_2)-MTD game is 471.8 time steps while the average time to win the corresponding layered MTD game is $14,257$ time steps.[2] Theorem 3 is a first analysis of layered MTD games and future research is needed to understand the exact effect of such composition. Is it possible to derive a much stronger bound on the security capacity for $(M_0, \ldots, M_f)$-MTD games if the defender has stronger capabilities? E.g., if the defender is able to detect whether the adversary enters a state k_j, then we may give the defender the ability to immediately kick the adversary out of state k_j after one time slot. In the proof of Theorem 3 this means that we do not need to consider the arguments related to the Poisson process; the upper bound can be improved by discarding the factors λ_j. However, this does not change the asymptotic behavior of the upper bound and therefore extra investment in such advanced detection will likely not pay off (we expect this intuition to also hold for layered MTD games).

[2]The security capacity of an individual game M_i is at least $-\log \pi_i(2) = -\log 0.1 = 3.32$ and (2.7) gives at least a security capacity of $3 \cdot 3.32 + \log 0.25 = 7.97$ for the (M_0, M_1, M_2)-MTD game. The simulation results fit up to a constant factor the theoretical prediction as the average time to win an individual game M_i is 24.3 and is approximately twice $2^{3.32}$ and the average time to win the (M_0, M_1, M_2)-MTD game is 471.8 and is approximately twice $2^{7.97}$.

A.2 Proofs of Theorems in Chapter 3

A.2.1 Closed Form Winning Probability – Proof of Theorem 4

Before proving any bounds on $w(k, T_{bud})$ we first provide a closed form for $w(k, T_{bud})$ itself. To this purpose we introduce $p(k-1, L, T_{bud})$ defined as *the probability that the adversary reaches state* $(k-1, L)$ *and is ready to leave state* $(k-1, L)$ *(after zero or more self-loops at state* $(k-1, L)$*) within time budget* T_{bud}. Any possible path from state $(0,0)$ to state $(k-1, L)$ in the Markov model with self-loops along the way in each of the visited states (including $(k-1, L)$) contributes to this probability. In our notation we use $k-1$ rather than k because after a path to $(k-1, L)$ for some $L \geq 0$ only a single horizontal or diagonal move is needed to reach (k, L) for the first time, and this probability is what we will need for our characterization of $w(k, T_{bud})$:

The formula for $w(k, T_{bud})$ considers all the possible paths towards a state $(k-1, L)$ within $T_{bud} - 1$ transitions and a single final horizontal or diagonal transition (via $m_2(L)$ or $m_3(L)$) towards the winning state (k, L). We are interested in the probability of entering state (k, L) for some $L \geq 0$ and the time it takes to reach there *for the first time*. Appropriately summing over probabilities $p(k-1, L, T_{bud} - 1)$ gives

$$w(k, T_{bud}) = \sum_{L \geq 0} p(k-1, L, T_{bud} - 1) \cdot \left(\frac{m_2(L)}{1 - m_1(L)} + \frac{m_3(L)}{1 - m_1(L)} \right), \quad \text{(A.5)}$$

where each path represented by $p(k-1, L, T_{bud})$ is ready to leave $(k-1, L)$ implying that the horizontal and diagonal transition probabilities are conditioned on "not having a self-loop" and this explain the division by $1 - m_1(L)$.

A path to $(k-1, L)$ till the moment it is ready to leave $(k-1, L)$ is uniquely represented by

- g_l; the number of states on the path that have the same level l,
- b_l; if level l is entered via a vertical transition, then $b_l = 0$; if level l is entered via a diagonal transition, then $b_l = 1$,
- $t(i, l)$ counts the number of self-loops on the path in state (i, l).

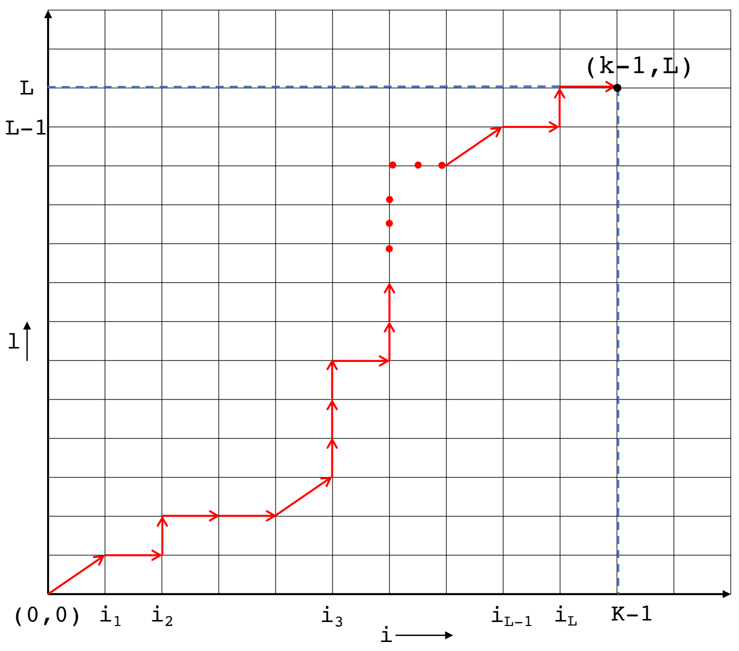
Notice that b_0 is undefined and $g_l \geq 1$ for all $0 \leq l \leq L$. Fig. A.5 depicts a typical path from $(0, 0)$ to $(k-1, L)$ without showing any self-loops, where

- $i_{l+1} = i_l + b_{l+1} + g_{l+1}$ and $i_0 = 0$; when the path enters level l for the first time, $k = i_l$.

The probability of having exactly $t(i, l)$ self-loops in state (i, l) after which the path exits (i, l) is equal to

$$Prob(t(i, l) = u) = m_1(l)^u (1 - m_1(l))$$

and describes a Poisson process. The probability that the path exits (i, l) along a



b_l	g_l
0	2
1	2
.	.
.	.
.	.
0	1
0	1
0	2
0	1
0	1
1	1
0	3
1	2
-	1

FIGURE A.5: A sample path toward the winning state without depicting the self-loops at each state (i, l) in the path

horizontal, diagonal or vertical transition given no more self-loops in (i,l) is equal to

$$\begin{aligned}
Prob(\text{horizontal transition}) &= Prob((i,l) \to (i+1,l)|\text{no self-loop}) \\
&= \frac{m_2(l)}{1-m_1(l)}, \\
Prob(\text{diagonal transition}) &= Prob((i,l) \to (i+1,l+1)|\text{no self-loop}) \\
&= \frac{m_3(l)}{1-m_1(l)}, \text{ and} \\
Prob(\text{vertical transition}) &= Prob((i,l) \to (i,l+1)|\text{no self-loop}) \\
&= \frac{m_4(l)}{1-m_1(l)}.
\end{aligned}$$

Combination of the probabilities describing diagonal and vertical transitions shows that

$$Prob((i_l+b_{l+1}+g_{l+1},l) \to (i_{l+1},l+1)|\text{no self-loop}) = b_{l+1}\frac{m_3(l)}{1-m_1(l)}+(1-b_{l+1})\frac{m_4(l)}{1-m_1(l)}.$$

The above analysis proves that the probability of having the Markov model transition along a path which is represented by $g_l, b_l, t(i,l)$, and i_l is equal to

$$\prod_{l=0}^{L-1}\left(b_{l+1}\frac{m_3(l)}{1-m_1(l)} + (1-b_{l+1})\frac{m_4(l)}{1-m_1(l)}\right) \tag{A.6a}$$

$$\cdot\prod_{l=0}^{L}\left(\frac{m_2(l)}{1-m_1(l)}\right)^{g_l-1} \tag{A.6b}$$

$$\cdot\prod_{l=0}^{L}\prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)), \tag{A.6c}$$

where (A.6a) corresponds to diagonal and vertical transitions, (A.6b) corresponds to horizontal transitions, and (A.6c) corresponds to self-loops.

The length of the path, i.e., total number of transitions without self-loops, is equal to

$$(k-1)+L-B, \text{ where } B=\sum_{l=1}^{L} b_l$$

is the total number of diagonal transitions. The total number of vertical transitions is equal to $L-B$ and the number of horizontal transitions is equal to $\sum_{l=0}^{L}(g_l-1) = (k-1)-B$.

Summing the above probability over all possible combinations of g_l, b_l, and $t(.,.)$ for paths that reach state $(k-1, L)$ and are ready to leave $(k-1, L)$ such that the number of self-loops is equal to T and the total number of transitions including self-loops is equal to $(k-1)+L-B+T \leq T_{bud}$ leads to an expression for $p(k-1, L, T_{bud})$:

$$p(k-1, L, T_{bud}) = \sum_{\substack{B,T s.t. \\ (k-1)+L-B+T\leq T_{bud}}}$$

$$\sum_{\substack{b_1,\ldots,b_L\in\{0,1\} s.t. \\ \sum_{l=1}^{L} b_l=B}} (A.6a) \tag{A.7a}$$

$$\sum_{\substack{g_0\geq 1,\ldots,g_L\geq 1 s.t. \\ \sum_{l=0}^{L}(g_l-1)=(k-1)-B}} (A.6b) \tag{A.7b}$$

$$\sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l=T}} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} (A.6c) \tag{A.7c}$$

Notice that $(A.6a) = 0$ if $B > L$ and $(A.6b) = 0$ if $B > k-1$. So, the main sum only needs to consider $B \leq \min\{L,(k-1)\}$. Plugging the above formula into (A.5) gives

$$w(k, T_{bud}) = \sum_{L=0}^{T_{bud}-1} \sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L) + m_3(L)}{1 - m_1(L)} \cdot (A.7a) \cdot (A.7b) \qquad \text{(A.8a)}$$

$$\cdot \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]} (A.7c). \qquad \text{(A.8b)}$$

This closed form together with

$$(A.7c) = \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1 - m_1(l))$$

$$= \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} \prod_{l=0}^{L} m_1(l)^{t_l}(1 - m_1(l))^{g_l}$$

$$= \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \prod_{l=0}^{L} \sum_{\substack{t(i_l,l),\ldots,t(i_l+g_l-1,l) \\ s.t. \sum_{i=0}^{g_l-1} t(i_l+i,l)=t_l}} m_1(l)^{t_l}(1 - m_1(l))^{g_l}$$

$$= \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \prod_{l=0}^{L} \binom{t_l + g_l - 1}{t_l} m_1(l)^{t_l}(1 - m_1(l))^{g_l}$$

proves Theorem 4.

A.2.2 Stagnating Learning Rate – Proof of Theorem 5

In order to prove a lower bound on $w(k, T_{bud})$ we give a benefit to the defender and assume the learning rate $f(l)$ is as large as possible given $1 - f(l) \geq \tau$, i.e., $1 - f(l) = \tau$

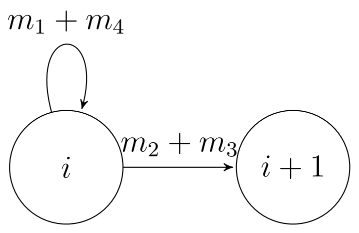


FIGURE A.6: Reduced Markov model of the game for a constant $1 - f(l) = \tau > 0$

with equality.

If $1 - f(l) = \tau$, then m_1, m_2, m_3 and m_4 are independent of l and the Markov model reduces to Fig. A.6 where each state i represents the *collection* of states (i, l) for $l \geq 0$ in the original Markov model.

Within time budget T_{bud} we reach $i = k$ with probability

$$w(k, T_{bud}) = \sum_{T=0}^{T_{bud}-k} \sum_{\substack{f_0,\ldots,f_{k-1} s.t. \\ \sum f_j = T}} \prod_j (m_1 + m_4)^{f_j} (m_2 + m_3)^k,$$

where the time budget is distributed over k steps from each state to the next until state k is reached and $T \leq T_{bud} - k$ transitions consisting of f_i self-loops in state i for

$0 \le i \le k-1$. This probability is equal to

$$\begin{aligned}
& \sum_{\substack{f_0,\ldots,f_{k-1} s.t.\\ \sum f_j \le T_{bud}-k}} \prod_j (m_1+m_4)^{f_j}(m_2+m_3)^k \\
\ge\ & \sum_{f_0=0}^{(T_{bud}-k)/k} \cdots \sum_{f_{k-1}=0}^{(T_{bud}-k)/k} \prod_j (m_1+m_4)^{f_j}(m_2+m_3)^k \\
=\ & \prod_{j=0}^{k-1} \sum_{f_j=0}^{(T_{bud}-k)/k} (m_1+m_4)^{f_j}(m_2+m_3)^k \\
=\ & \prod_{j=0}^{k-1} \frac{1-(m_1+m_4)^{T_{bud}/k}}{1-(m_1+m_4)}(m_2+m_3)^k \\
=\ & [1-(m_1+m_4)^{T_{bud}/k}]^k \ge 1-k(m_1+m_4)^{T_{bud}/k},
\end{aligned}$$

where the last equality follows from $1-(m_1+m_4) = m_2+m_3$. Now we substitute $m_1+m_4 = 1-\tau p$ which yields Theorem 5.

A.2.3 Upper and Lower Bounds – Proof of Theorem 6

By noticing that (A.8b) for $T_{bud} = \infty$ is equal to 1, a straightforward upper bound on $w(k, T_{bud})$ is given by just formula (A.8a): We have $w(k, T_{bud})$ is at most equal to $\bar{w}(k, T_{bud})$.

In order to prove a lower bound we first analyze

$$
\begin{aligned}
(A.8b) &= \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]} (A.7c) \\
&= \sum_{T=0}^{(T_{bud}-k-L+B)} \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l = T}} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} \\
&\qquad \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)) \\
&= \sum_{\substack{t_0,\ldots,t_L s.t. \\ \sum_{l=0}^{L} t_l \le (T_{bud}-k-L+B)}} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} \\
&\qquad \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)).
\end{aligned}
$$

Notice that (A.8b) is used within another sum with $B \le \min\{L, (k-1)\}$. This implies that $B \le (k-1)L$ or equivalently $-(L+1)k \le -k-L+B$. Let $v \ge 1$. For now we only consider $L \le (T_{bud}/v) - 1$, or equivalently $v \le T_{bud}/(L+1)$. Then $(L+1)(v-k) \le T_{bud} - (L+1)k \le T_{bud} - k - L + B$, hence,

$$
v - k \le \frac{T_{bud} - k - L + B}{L+1}.
$$

This allows us to lower bound the above sums and obtain

$$\begin{aligned}
(A.8b) \;\geq\; & \sum_{t_0=0}^{v-k} \cdots \sum_{t_L=0}^{v-k} \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)=t_0}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)=t_L}} \\
& \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)) \\
\geq\; & \sum_{\substack{t(i_0,0),\ldots,t(i_0+g_0-1,0) \\ s.t. \sum_{i=0}^{g_0-1} t(i_0+i,0)\leq v-k}} \cdots \sum_{\substack{t(i_L,L),\ldots,t(i_L+g_L-1,L) \\ s.t. \sum_{i=0}^{g_L-1} t(i_L+i,L)\leq v-k}} \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)) \\
\geq\; & \sum_{t(i_0,0)\leq t_0/g_0} \cdots \sum_{t(i_0+g_0-1,0)\leq (v-k)/g_0} \cdots \sum_{t(i_L,L)\leq t_L/g_L} \cdots \sum_{t(i_L+g_L-1,L)\leq (v-k)/g_L} \\
& \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} m_1(l)^{t(i_l+i,l)}(1-m_1(l)) \\
=\; & \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} \sum_{t(i_l+i,l)\leq (v-k)/g_l} m_1(l)^{t(i_l+i,l)}(1-m_1(l)) \\
=\; & \prod_{l=0}^{L} \prod_{i=0}^{g_l-1} \frac{1-m_1(l)^{\frac{v-k}{g_l}+1}}{1-m_1(l)}(1-m_1(l)) \\
=\; & \prod_{l=0}^{L} (1-m_1(l)^{\frac{v-k}{g_l}+1})^{g_l} \geq \prod_{l=0}^{L} (1-g_l m_1(l)^{\frac{v-k}{g_l}+1}).
\end{aligned}$$

We observe that $g_l \leq k$ within the larger sum that contains (A.8b). If $v \geq k$, then this implies $v/k \leq 1 + (v-k)/g_l$ proving

$$(A.8b) \;\geq\; \prod_{l=0}^{L} (1 - g_l m_1(l)^{v/k})$$

Since $m_1(l) = [(1-\gamma)(1-(p+h)(1-f(l)))] \leq 1-\gamma$, we can further lower bound this to

$$\begin{aligned}(A.8b) &\geq \prod_{l=0}^{L}(1 - g_l(1-\gamma)^{v/k}) \\ &\geq 1 - (\sum_{l=0}^{L} g_l)(1-\gamma)^{v/k}.\end{aligned}$$

Within the larger sum that contains (A.8b), we have $\sum_{l=0}^{L} g_l = (L+1)+\sum_{l=0}^{L}(g_l-1) = (L+1)+(k-1-B) = k+L-B \leq k+T_{bud}/v$. This yields

$$(A.8b) \geq 1-(k+\frac{T_{bud}}{v})(1-\gamma)^{v/k}.$$

The obtained lower bound on $(A.8b) = \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]}(A.7c)$ is independent of any of the other summing variables used in (A.8a) but requires $L+1 \leq T_{bud}/v$. By restricting L to be $\leq (T_{bud}/v)-1$ in (A.8a), i.e., we substitute T_{bud} by T_{bud}/v, we obtain a lower bound on $w(k, T_{bud})$:

$$\begin{aligned}w(k,T_{bud}) &= \sum_{L=0}^{T_{bud}-1}\sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L)+m_3(L)}{1-m_1(L)} \cdot (A.7a)\cdot(A.7b)\cdot \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]} (A.7c) \\ &\geq \sum_{L=0}^{(T_{bud}/v)-1}\sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L)+m_3(L)}{1-m_1(L)} \cdot (A.7a)\cdot(A.7b)\cdot \sum_{T=0}^{(T_{bud}-1)-[(k-1)+L-B]} (A.7c) \\ &\geq \sum_{L=0}^{T_{bud}/v-1}\sum_{B=0}^{\min\{L,(k-1)\}} \frac{m_2(L)+m_3(L)}{1-m_1(L)} \cdot (A.7a)\cdot(A.7b)(1-(k+\frac{T_{bud}}{v})(1-\gamma)^{v/k}) \\ &= \bar{w}(k, T_{bud}/v)\cdot(1-(k+\frac{T_{bud}}{v})(1-\gamma)^{v/k}),\end{aligned}$$

where the last equality follows from the fact that (A.7a) and (A.7b) do not depend on T_{bud} (they depend on L). This completes the proof.

A.2.4 Analyzing the Learning Rate – Proof of Theorem 7

Theorem 7 has a couple of statements and we start by proving the first most general claim that upper bounds $\bar{w}(k, T_{bud})$ in terms of the general parameters of the Markov model.

We define and assume
$$\beta(l) = \sqrt{\frac{1-f(l)}{d}} \leq 1.$$

This allows us to bound

$$\begin{aligned} \alpha(l) &= \frac{p(1-f(l))}{\gamma + (1-\gamma)(p+h)(1-f(l))} \leq \frac{p(1-f(l))}{\gamma} = \frac{pd\beta(l)^2}{\gamma}, \\ \alpha(l) &= \frac{p(1-f(l))}{\gamma + (1-\gamma)(p+h)(1-f(l))} \geq p(1-f(l)) = pd\beta(l)^2. \end{aligned}$$

Since $\beta(l) \leq 1$, $\beta(l)^2 \leq \beta(l)$. Let θ be such that $(1-\gamma)/\gamma \leq \theta$. Then the above inequalities allow us to bound

$$\begin{aligned} \frac{m_2(l)}{1-m_1(l)} &= (1-\gamma)\alpha(l) \leq \frac{(1-\gamma)pd}{\gamma}\beta(l)^2 \leq \theta pd\beta(l)^2 \\ &\leq \theta pd\beta(l), \\ \frac{m_3(l)}{1-m_1(l)} &= \gamma\alpha(l) \leq pd\beta(l)^2, \text{ and} \\ \frac{m_4(l)}{1-m_1(l)} &= 1-\alpha(l) \leq 1-pd\beta(l)^2. \end{aligned}$$

Plugging the above bounds in expression (3.7) for $\bar{w}(k, T_{bud})$ with $\frac{m_2(L)}{1-m_1(L)} \leq$

$\theta p d\beta(L)^2$ and $\frac{m_2(l)}{1-m_1(l)} \leq \theta p d\beta(l)$ yields

$$\bar{w}(k, T_{bud}) \leq \sum_{L=0}^{T_{bud}-1} (1+\theta) p d\beta(L)^2$$

$$\cdot \sum_{B=0}^{L} \sum_{\substack{b_1,\dots,b_L \in \{0,1\} \\ s.t. \sum_{l=1}^{L} b_l = B}} \prod_{l=0}^{L-1} \begin{matrix} b_{l+1} p d\beta(l)^2 \\ +(1-b_{l+1})(1-pd\beta(l)^2) \end{matrix} \tag{A.9a}$$

$$\cdot \sum_{\substack{g_0,\dots,g_L \geq 1 s.t. \\ \sum_{l=0}^{L}(g_l-1)=(k-1)-B}} \prod_{l=0}^{L} \Big(\theta p d\beta(l)\Big)^{g_l-1} \tag{A.9b}$$

Let $b_{L+1} = 0$ and define

$$Z = \prod_{l=0}^{L} \Big(\theta p d\beta(l)\Big)^{b_{l+1}}.$$

We will multiply (A.9b) with Z and show an upper bound of the product which is independent of b_l and B. We will multiply (A.9a) with Z^{-1} and show that it behaves like a product of $1 + \frac{\beta(l)}{\theta}$, which (as we will see) remains 'small enough':

We choose d such that $\theta \leq (pd)^{-1}$ (if d does not satisfy this inequality, then the to be derived upper bound will be larger than 1 and will therefore trivially hold for

$\bar{w}(k, T_{bud}))$. Then, since $B = \sum_{l=0}^{L} b_{l+1}$,

$$(A.9b) \cdot Z = \sum_{\substack{g_0,\ldots,g_L \geq 1 s.t. \\ \sum_{l=0}^{L}(g_l-1)+b_{l+1}=(k-1)}} \prod_{l=0}^{L} \Big(\theta p d \beta(l)\Big)^{(g_l-1)+b_{l+1}}$$

$$\leq \sum_{\substack{g'_0,\ldots,g'_L \geq 0 s.t. \\ \sum_{l=0}^{L} g'_l=(k-1)}} \prod_{l=0}^{L} \Big(\theta p d \beta(l)\Big)^{g'_l}$$

$$\leq (\theta p d)^{k-1} \sum_{\substack{g'_0,\ldots,g'_L \geq 0 s.t. \\ \sum_{l=0}^{L} g'_l=(k-1)}} \prod_{l=0}^{L} \beta(l)^{g'_l}$$

$$\leq (\theta p d)^{k-1} \sum_{g'_0 \geq 0} \cdots \sum_{g'_L \geq 0} \prod_{l=0}^{L} \beta(l)^{g'_l}$$

$$= (\theta p d)^{k-1} \prod_{l=0}^{L} \sum_{g'_l \geq 0} \beta(l)^{g'_l} = (\theta p d)^{k-1} \prod_{l=0}^{L} \frac{1}{1-\beta(l)}.$$

Now notice that each b_{l+1} is either equal to 0 or 1 and therefore

$$(b_{l+1} p d \beta(l)^2 + (1-b_{l+1})(1-pd\beta(l)^2)) \cdot (\theta p d \beta(l))^{-b_{l+1}} = b_{l+1}\frac{\beta(l)}{\theta} + (1-b_{l+1})(1-pd\beta(l)^2).$$

We derive

$$(A.9a)\cdot Z^{-1} = \sum_{B=0}^{L} \sum_{\substack{b_1,\ldots,b_L\in\{0,1\} \\ s.t. \sum_{l=1}^{L} b_l = B}} \prod_{l=0}^{L-1} \begin{array}{l} b_{l+1}\frac{\beta(l)}{\theta} \\ +(1-b_{l+1})(1-pd\beta(l)^2) \end{array}$$

$$= \sum_{b_1,\ldots,b_L\in\{0,1\}} \prod_{l=0}^{L-1} \begin{array}{l} b_{l+1}\frac{\beta(l)}{\theta} \\ +(1-b_{l+1})(1-pd\beta(l)^2) \end{array}$$

$$= \prod_{l=0}^{L-1} \sum_{b_{l+1}\in\{0,1\}} \begin{array}{l} b_{l+1}\frac{\beta(l)}{\theta} \\ +(1-b_{l+1})(1-pd\beta(l)^2) \end{array}$$

$$= \prod_{l=0}^{L-1} \left(\frac{\beta(l)}{\theta} + (1-pd\beta(l)^2)\right) \le \prod_{l=0}^{L-1} \left(\frac{\beta(l)}{\theta} + 1\right).$$

Combination of the above results proves

$$\begin{aligned} \bar{w}(k, T_{bud}) &\le \sum_{L=0}^{T_{bud}-1} [\theta pd]^k (1+\frac{1}{\theta})\frac{\beta(L)^2}{1-\beta(L)} \prod_{l=0}^{L-1} \frac{1+\beta(l)/\theta}{1-\beta(l)} \\ &\le \sum_{L=0}^{T_{bud}-1} [\theta pd]^k (1+\frac{1}{\theta}) \prod_{l=0}^{L} \frac{1+\beta(l)/\theta}{1-\beta(l)}, \end{aligned}$$

where the last inequality uses $\beta(L)^2 \le 1 \le 1+\beta(L)/\theta$. The argument in the resulting product is equal to $(1+\beta(l)/\theta)/(1-\beta(l)) = 1+(1+\theta^{-1})\beta(l)/(1-\beta(l))$. This completes the proof of the first statement.

In order to prove the second bound in Theorem 7 we define

$$B'(l) = \frac{\beta(l)}{1-\beta(l)} \text{ and } y = 1+\theta^{-1}$$

and apply the next lemma. Notice that $B'(l) \ge 0$ and, since $f'(l) \ge 0$ (because learning only increases),

$$B''(l) = \beta'(l)\frac{2-\beta(l)}{(1-\beta(l))^2} = \frac{-f'(l)}{2\beta(l)}\frac{2-\beta(l)}{(1-\beta(l))^2} \leq 0.$$

Lemma 5. *For differentiable continuous functions $B(.)$, if $B'(l) \geq 0$ and $B''(l) \leq 0$ for $l \geq 0$, then, for $y \geq 0$,*

$$\prod_{l=0}^{L}(1 + yB'(l)) \leq e^{y(B'(0)-B(0)+B(L))}.$$

Proof. We find an upperbound of $\ln \prod_{l=0}^{L}(1 + yB'(l)) = \sum_{l=0}^{L} \ln(1 + yB'(l))$. Differentiating w.r.t. y gives

$$\begin{aligned}
\sum_{l=0}^{L} \frac{1}{1+yB'(l)} \cdot B'(l) &\leq \sum_{l=0}^{L} B'(l) = B'(0) + \sum_{l=1}^{L} B'(l) \\
&\leq B'(0) + \int_{l=0}^{L} B'(l)dl \\
&= B'(0) + B(L) - B(0)
\end{aligned}$$

We also have

$$\sum_{l=0}^{L} \ln(1 + yB'(l))|_{y=0} = 0 = y \cdot [B'(0) - B(0) + B(L)]|_{y=0}.$$

We conclude that $\sum_{l=0}^{L} \ln(1 + yB'(l)) \leq y \cdot [B'(0) - B(0) + B(L)]$ for $y \geq 0$. $\square$

For our $B'(l)$ and y, application of the lemma to our upper bound yields

$$\bar{w}(k, T_{bud}) \leq (1+\theta^{-1}) \sum_{L=0}^{T_{bud}-1} [\theta pd]^k e^{(1+\theta^{-1})(B'(0)-B(0)+B(L))}$$
$$\leq T_{bud} \cdot (1+\theta^{-1})[\theta pd]^k e^{(1+\theta^{-1})(B'(0)-B(0)+B(T_{bud}-1))}.$$

Minimizing $\theta^k e^{(1+\theta^{-1})c}$ for

$$c = B(T_{bud}-1) - B(0) + B'(0)$$

gives $\theta = c/k$ if $c/k \geq (1-\gamma)/\gamma$ and gives $\theta = (1-\gamma)/\gamma$ if $c/k \leq (1-\gamma)/\gamma$. Substituting this in our upper bound (e denotes the natural number) yields

$$\bar{w}(k, T_{bud}) \leq \begin{cases} T_{bud}(1+\frac{k}{c})[\frac{c}{k}pd]^k e^{k+c} = T_{bud}(1+\frac{k}{c})[epd\frac{c}{k}e^{c/k}]^k, & \text{if } c/k \geq (1-\gamma)/\gamma \\ T_{bud}(1-\gamma)^{-1}[\frac{1-\gamma}{\gamma}pd]^k e^{(1-\gamma)^{-1}c}, & \text{if } c/k \leq (1-\gamma)/\gamma. \end{cases} \tag{A.10}$$

The first case of the upper bound is less interesting as $\frac{c}{k}e^{c/k}$ may be large yielding a bad upper bound. The second case of the upper bound gives the most insight and proves the second statement of Theorem 7.

As an example, suppose that

$$1 - f(l) \leq \frac{d}{(l+2)^2}.$$

To get an upper bound, we give the adversary the advantage of having the defender play with the smallest possible learning rates. i.e., $1 - f(l) = \frac{d}{(l+2)^2}$. This gives $B'(l) = 1/(l+1)$, hence, $B(T_{bud}-1) = \ln(T_{bud})$, $B(0) = 0$, and $B'(0) = 1$. This implies $c = 1 + \ln(T_{bud})$. The second case of the upper bound translates into: If

$$T_{bud} \leq e^{-1+k(1-\gamma)/\gamma}, \tag{A.11}$$

then the probability of winning for the adversary is at most

$$w(k, T_{bud}) \leq \bar{w}(k, T_{bud}) \leq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot T_{bud}^{1+1/(1-\gamma)} \cdot [\frac{1-\gamma}{\gamma} pd]^k.$$

This proves the last statement of the theorem.

A.2.5 Delayed Learning – Proof of Theorem 8

In practice, we may not immediately start learning at a rate $1-f(l) \approx d/(l+2)^2$. This will only happen after reaching for example L^* samples. During such a first phase the defender is not yet able to increase $f(l)$ and $f(l)$ remains 0, i.e., $1 - f(l) = 1$. The adversary tries to compromise as many, say k^*, nodes as possible before reaching L^* levels.

For $1 - f(l) = 1$, we have:

$$\begin{aligned}
m_1(l) &= (1-\gamma)(1-(p+h)),\\
m_2(l) &= (1-\gamma)p,\\
m_3(l) &= \gamma p, \text{ and}\\
m_4(l) &= [h - \gamma(h+p)] + \gamma = (1-\gamma)h + \gamma(1-p),
\end{aligned}$$

which are all independent of l. For this reason we use m_1, m_2, m_3, and m_4 where suited in our derivations below.

We are interested in k^* as a function of ϵ for which

$$\begin{aligned} u(k^*, L^*) &= \sum_{k=0}^{k^*-1} p(k, L^*-1, T_{bud}-1)(\frac{m_4(L^*-1)}{1-m_1(L^*-1)} + \frac{m_3(L^*-1)}{1-m_1(L^*-1)}) + \\ & \qquad p(k^*, L^*-1, T_{bud}-1)\frac{m_4(L^*-1)}{1-m_1(L^*-1)} \geq 1-\epsilon. \end{aligned} \tag{A.12}$$

Here $u(k^*, L^*)$ is equal to the probability that $\leq k^*$ nodes will be compromised before level $l = L^*$ is reached for the first time: The sum in (A.12) considers all paths reaching a state (k, L^*-1) for some $k < k^*$ after which a single vertical or diagonal transition reaches level L^* for the first time. The additional term considers paths that reach a state (k^*, L^*-1) after which only a vertical transition reaches level L^* without increasing the number of compromised nodes beyond k^*. If (A.12) holds, then the probability of winning is

$$\leq \epsilon + (1-\epsilon)w(k - k^*(\epsilon), T_{bud}), \tag{A.13}$$

where $w(k - k^*(\epsilon), T_{bud})$ is defined for learning rate $f(l + L^*)$ as a function of l. In other words, during the first phase a node with $i < k^*(\epsilon)$ is reached with probability $\geq 1-\epsilon$ after which the learning rate increases according to $f(l + L^*)$ leading to a winning state if the Markov model increments i at least another $k - k^*(\epsilon)$ steps.

In order to find a lower bound on $u(k^*, L^*)$ we give the adversary the benefit of an unlimited time budget (implying $(A.7c) = 1$) giving

$$\begin{aligned} u(k^*, L^*) &\geq \sum_{k=0}^{k^*-1} p(k, L^*-1, \infty)(\frac{m_4(L^*-1)}{1-m_1(L^*-1)} + \frac{m_3(L^*-1)}{1-m_1(L^*-1)}) && \text{(A.14)} \\ &= \sum_{B=0}^{\min\{L^*-1,(k^*-2)\}} \sum_{k=B+1}^{k^*-1} (A.7a)\cdot(A.7b)\cdot\frac{m_3(L^*-1)+m_4(L^*-1)}{1-m_1(L^*-1)}. && \text{(A.15)} \end{aligned}$$

We simplify this lower bound by noticing that (A.7a) does not depend on k and

$$\begin{aligned}
\sum_{k=B+1}^{k^*-1} (A.7b) &= \sum_{k=B+1}^{k^*-1} \sum_{\substack{g_0\geq 1,\ldots,g_{L^*-1}\geq 1 s.t. \\ \sum_{l=0}^{L^*-1}(g_l-1)=k-1-B}} \prod_{l=0}^{L^*-1} \left(\frac{m_2(l)}{1-m_1(l)}\right)^{g_l-1} \\
&= \sum_{\substack{g'_0,\ldots,g'_{L^*-1}\geq 0 s.t. \\ \sum_{l=0}^{L^*-1} g'_l \leq k^*-2-B}} \prod_{l=0}^{L^*-1} \left(\frac{m_2(l)}{1-m_1(l)}\right)^{g'_l} \\
&\geq \sum_{g'_0=0}^{(k^*-2-B)/L^*} \cdots \sum_{g'_{L^*}=0}^{(k^*-2-B)/L^*} \prod_{l=0}^{L^*-1} \left(\frac{m_2(l)}{1-m_1(l)}\right)^{g'_l} \\
&= \prod_{l=0}^{L^*-1} \sum_{g'_l=0}^{(k^*-2-B)/L^*} \left(\frac{m_2(l)}{1-m_1(l)}\right)^{g'_l} \\
&= \prod_{l=0}^{L^*-1} \frac{1}{1-\frac{m_2(l)}{1-m_1(l)}} \big(1-(\frac{m_2(l)}{1-m_1(l)})^{\frac{k^*-2-B}{L^*}+1}\big) \\
&= \prod_{l=0}^{L^*-1} \frac{1-m_1(l)}{m_3(l)+m_4(l)} \left(1-(\frac{m_2(l)}{1-m_1(l)})^{\frac{k^*-2-B}{L^*}+1}\right) \\
&= \left(\frac{1-m_1}{m_3+m_4}\right)^{L^*} \left(1-(\frac{m_2}{1-m_1})^{\frac{k^*-2-B}{L^*}+1}\right)^{L^*}.
\end{aligned}$$

Since $B \leq L^*-1$, this can be further lower bounded as

$$\sum_{k=B+1}^{k^*-1} (A.7b) \geq \left(\frac{1-m_1}{m_3+m_4}\right)^{L^*} \left(1-(\frac{m_2}{1-m_1})^{\frac{k^*-1}{L^*}}\right)^{L^*}.$$

Notice that this lower bound is independent from B, hence,

$$\begin{aligned}
u(k^*,L^*) &\geq \sum_{B=0}^{\min\{L^*-1,(k^*-2)\}} (A.7b)\cdot \frac{m_3(L^*-1)+m_4(L^*-1)}{1-m_1(L^*-1)} \\
&\quad \cdot\left(\frac{1-m_1}{m_3+m_4}\right)^{L^*}\left(1-(\frac{m_2}{1-m_1})^{\frac{k^*-1}{L^*}}\right)^{L^*} \\
&= \sum_{B=0}^{\min\{L^*-1,(k^*-2)\}} (A.7b)\cdot \\
&\quad \cdot\left(\frac{1-m_1}{m_3+m_4}\right)^{L^*-1}\left(1-(\frac{m_2}{1-m_1})^{\frac{k^*-1}{L^*}}\right)^{L^*}.
\end{aligned}$$

We assume $k^* \geq L^*+1$ and derive

$$\begin{aligned}
\sum_{B=0}^{\min\{L^*-1,(k^*-2)\}} (A.7b) &= \sum_{B=0}^{L^*-1} \sum_{\substack{b_1,\dots,b_{L^*-1}\in\{0,1\} s.t. \\ \sum_{l=1}^{L^*-1} b_l = B}} \prod_{l=0}^{L^*-2}\left(b_{l+1}\frac{m_3(l)}{1-m_1(l)}+(1-b_{l+1})\frac{m_4(l)}{1-m_1(l)}\right) \\
&= \sum_{b_1,\dots,b_{L^*-1}\in\{0,1\}} \prod_{l=0}^{L^*-2}\left(b_{l+1}\frac{m_3(l)}{1-m_1(l)}+(1-b_{l+1})\frac{m_4(l)}{1-m_1(l)}\right) \\
&= \prod_{l=0}^{L^*-2}(\frac{m_3(l)}{1-m_1(l)}+\frac{m_4(l)}{1-m_1(l)}) = \left(\frac{m_3+m_4}{1-m_1}\right)^{L^*-1}.
\end{aligned}$$

Substituting this in the lower bound for $u(k^*,L^*)$ yields

$$\begin{aligned}
u(k^*,L^*) &\geq \left(1-(\frac{m_2}{1-m_1})^{\frac{k^*-1}{L^*}}\right)^{L^*} \geq 1-L^*(\frac{m_2}{1-m_1})^{\frac{k^*-1}{L^*}} \\
&= 1-L^*[\frac{(1-\gamma)p}{1-(1-\gamma)(1-(p+h))}]^{\frac{k^*-1}{L^*}}.
\end{aligned}$$

A.2.6 Capacity Region – Proofs of Corollaries 3 and 4

Proof of Corollary 3. As an example of computing a capacity region, we translate Theorem 7 for $1-f(l)\leq \frac{d}{(l+2)^2}$ in terms of a capacity region: We have the security guarantee

$$T_{bud} \leq e^{-1+k(1-\gamma)/\gamma} \quad \Rightarrow \quad \bar{w}(k, T_{bud}) \leq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot T_{bud}^{1+1/(1-\gamma)} \cdot [\frac{1-\gamma}{\gamma} pd]^k. \tag{A.16}$$

If

$$\begin{aligned} 2^t &\leq e^{-1+k(1-\gamma)/\gamma}, \text{ and} \\ 2^{-s} &\geq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot (2^t)^{1+1/(1-\gamma)} \cdot [\frac{1-\gamma}{\gamma} pd]^k, \end{aligned} \tag{A.17}$$

then $T_{bud} \leq 2^t$ implies the condition on the left hand side of the implication in (A.16), hence,

$$\begin{aligned} \bar{w}(k, T_{bud}) &\leq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot T_{bud}^{1+1/(1-\gamma)} \cdot [\frac{1-\gamma}{\gamma} pd]^k \\ &\leq \frac{e^{1/(1-\gamma)}}{1-\gamma} \cdot (2^t)^{1+1/(1-\gamma)} \cdot [\frac{1-\gamma}{\gamma} pd]^k \\ &\leq 2^{-s}. \end{aligned}$$

This shows that the capacity region is characterized by (A.17), or equivalently after taking logarithms, assuming $\frac{pd(1-\gamma}{\gamma} < 1$, and reordering terms, $\boldsymbol{\delta}$, $\boldsymbol{\mu}$, and $\boldsymbol{\xi}$ are given by 2-dimensional vectors

$$\begin{aligned} \boldsymbol{\delta} &= (0, q \ln 2), \text{ where } q = \left[\ln(\frac{\gamma}{pd(1-\gamma)}) \right]^{-1}, \\ \boldsymbol{\mu} &= (\frac{\gamma \ln 2}{1-\gamma}, q(\ln 2)(1 + \frac{1}{1-\gamma})), \\ \boldsymbol{\xi} &= (-\frac{\gamma}{1-\gamma}, q(-\frac{1}{1-\gamma} + \ln \frac{1}{1-\gamma}). \end{aligned}$$

Proof of Corollary 4. Delayed learning shows that for all $k^* \geq L^* + 1$,

$$\begin{aligned} w(k, T_{bud}) &\leq (1 - u(k^*, L^*)) + w'(k - k^*, T_{bud}) \\ &\leq L^*[\frac{(1-\gamma)p}{1-(1-\gamma)(1-(p+h))}]^{\frac{k^*-1}{L^*}} + w'(k - k^*, T_{bud}). \end{aligned}$$

If

$$\begin{aligned} L^*[\frac{(1-\gamma)p}{1-(1-\gamma)(1-(p+h))}]^{\frac{k^*-1}{L^*}} &\leq 2^{-(s+1)} \text{ and} \qquad (A.18) \\ w'(k - k^*, T_{bud}) &\leq 2^{-(s+1)}, \qquad (A.19) \end{aligned}$$

then $w(k, T_{bud}) \leq 2^{-s}$.

We assume $\frac{(1-\gamma)p}{1-(1-\gamma)(1-(p+h))} < 1$. Then, after taking logarithms, substituting $k^* = \hat{k} + L^* + 1$ with $\hat{k} \geq 0$, and reordering terms, condition (A.18) is equivalent to $(s+1)\delta'' \leq \hat{k} + \xi''$, where $\delta'' = L^*(\ln 2)z$, and $\xi'' = L^*[1 - (\ln L^*)z]$ for $z = \left[\ln(\frac{1-(1-\gamma)(1-(p+h))}{(1-\gamma)p})\right]^{-1}$.

Suppose that $w'(\cdot, \cdot)$ corresponds to capacity region $(\boldsymbol{\delta'}, \boldsymbol{\mu'}, \boldsymbol{\xi'})$. Then, condition (A.19) is implied by $T_{bud} \leq 2^t$ and

$$(s+1)\boldsymbol{\delta'} + t\boldsymbol{\mu'} \leq (k - (\hat{k} + L^* + 1))\mathbf{1} + \boldsymbol{\xi'}.$$

If we assume learning is delayed sufficiently long such that

$$(s+1)\delta'' \geq \xi''$$

(e.g., $L^* \geq z^{-1}/2$ for $s = 0$), then we may choose

$$\hat{k} = (s+1)\delta'' - \xi'' \geq 0.$$

This satisfies condition (A.18). Condition (A.19) (after substituting $\hat{k}$) is now equivalent to $T_{bud} \leq 2^t$ and

$$(s+1)(\boldsymbol{\delta}' + \delta''\mathbf{1}) + t\boldsymbol{\mu}' \leq k\mathbf{1} + \boldsymbol{\xi} + (\xi'' - (L^*+1))\mathbf{1}.$$

In other words,

$$(\boldsymbol{\delta}' + \delta''\mathbf{1}, \boldsymbol{\mu}', \boldsymbol{\xi}' - \boldsymbol{\delta}' + (\xi'' - \delta'' - (L^*+1))\mathbf{1})$$

is a capacity region.

If we assume learning is moderately delayed such that

$$(s+1)\delta'' \leq \xi'',$$

then we may choose $\hat{k} = 0$ and

$$(\boldsymbol{\delta}', \boldsymbol{\mu}', \boldsymbol{\xi}' - \boldsymbol{\delta}' - (L^*+1)\mathbf{1}).$$

is a capacity region.

Substituting

$$\begin{aligned} s \geq \xi''/\delta'' - 1 = (L^*[1 - (\ln L^*)z])/L^*(\ln 2)z - 1 &= (L^*[1 - (\ln 2L^*)z])/L^*(\ln 2)z \\ &= 1/z\ln 2 - \ln(2L^*)\ln 2 \end{aligned}$$

and

$$\xi'' - \delta'' - (L^* + 1) = L^*[1 - (\ln L^*)z] - L^*(\ln 2)z - (L^* + 1) = L^*(\ln 2L^*)z - 1$$

proves the corollary.

A.3 Proofs of Theorems in Chapter 4

A.3.1 Containment Criterion – Proof of Theorem 10

In order to make the next derivations readable we introduce

$$\begin{aligned} y &= j(t), \\ v &= y' = j'(t) = \gamma \cdot i(t). \end{aligned}$$

Differential equation (4.9) in terms of y and v after reordering terms reads

$$y'' = A^\alpha p \cdot v \cdot (y + A)^{-\alpha}. \tag{A.20}$$

We derive

$$y'' = v' = \frac{dv}{dt} = \frac{dv}{dy}\frac{dy}{dt} = y'\frac{dv}{dy} = v\frac{dv}{dy}.$$

Substituting this expression into (A.20), dividing by v, and multiplying with dy yields

$$dv = A^\alpha p \cdot (y + A)^{-\alpha} \cdot dy.$$

Assuming $\alpha \neq 1$, then taking integrals on the left and right side gives the equation

$$v = \frac{A^\alpha p}{1-\alpha} \cdot (y+A)^{1-\alpha} + c_1 \tag{A.21}$$

for some constant c_1. For $t = 0$, we have $y(0) = j(0) = 0$ and $v(0) = y'(0) = j'(0) = \gamma \cdot i(0) = \gamma$. Substituting this into the above equation solves

$$c_1 = \gamma + \frac{Ap}{\alpha - 1}. \tag{A.22}$$

Now we substitute $v = \frac{dy}{dt}$ into (A.21) and multiply both sides with dt:

$$dy = \left\{ \frac{A^\alpha p}{1-\alpha} \cdot (y+A)^{1-\alpha} + c_1 \right\} \cdot dt. \tag{A.23}$$

Equivalently, we have

$$dt = \left\{ \frac{A^\alpha p}{1-\alpha} \cdot (y+A)^{1-\alpha} + c_1 \right\}^{-1} \cdot dy.$$

Taking integrals on the left and right side gives the expression

$$t = c_2 + \int_{y=y(0)}^{y(t)} \left\{ \frac{A^\alpha p}{1-\alpha} \cdot (y+A)^{1-\alpha} + c_1 \right\}^{-1} dy$$

for some constant c_2. Substituting $t = 0$ gives $c_2 = 0$ and $y(0) = j(0) = 0$. Plugging in (A.22) proves

$$\begin{aligned} t &= \int_{y=0}^{y(t)} \left\{ \frac{A^\alpha p}{1-\alpha} \cdot (y+A)^{1-\alpha} + \gamma + \frac{Ap}{\alpha-1} \right\}^{-1} dy \\ &= \frac{\alpha-1}{Ap} \cdot \int_{y=0}^{y(t)} \left\{ 1 - (y/A+1)^{1-\alpha} + \frac{\gamma(\alpha-1)}{Ap} \right\}^{-1} dy. \end{aligned}$$

As an immediate consequence we can take the derivative (using the chain rule) with respect to t on both sides. This shows that (this can also directly be concluded from (A.21))

$$1 = \frac{\alpha - 1}{Ap} \cdot \left\{ 1 - (y(t)/A + 1)^{1-\alpha} + \frac{\gamma(\alpha - 1)}{Ap} \right\}^{-1} \cdot y'(t). \qquad \text{(A.24)}$$

Notice that $y(t) = j(t)$ is increasing (since $j'(t) = \gamma \cdot i(t) \geq 0$) with $y(t) \geq y(0) = 0$. Therefore, if $\alpha > 1$, then $0 \leq (y(t)/A + 1)^{1-\alpha} \leq 1$ and

$$\left\{ 1 - (y(t)/A + 1)^{1-\alpha} + \frac{\gamma(\alpha - 1)}{Ap} \right\}^{-1} \geq \left\{ 1 + \frac{\gamma(\alpha - 1)}{Ap} \right\}^{-1}.$$

Case I ($\alpha > 1$): Substituting this into (A.24) yields for $\alpha > 1$,

$$1 \geq \frac{\alpha - 1}{Ap} \cdot \left\{ 1 + \frac{\gamma(\alpha - 1)}{Ap} \right\}^{-1} \cdot y'(t)$$

proving

$$i(t) = y'(t)/\gamma \leq \frac{Ap}{\alpha - 1} \cdot \left\{ 1 + \frac{\gamma(\alpha - 1)}{Ap} \right\} /\gamma = 1 + \frac{Ap}{(\alpha - 1)\gamma}.$$

Case II ($\alpha < 1$): As a second consequence of (A.24), we derive for $\alpha < 1$,

$$y'(t) = \frac{Ap}{\alpha - 1} \left\{ 1 - (y(t)/A + 1)^{1-\alpha} \right\} + \gamma \geq \gamma = y'(0). \qquad \text{(A.25)}$$

Hence,

$$y(t) \geq \gamma \cdot t.$$

Substituting this back into (A.25), and noticing that $i(t) = y'(t)/\gamma$ gives

$$i(t) \geq \frac{Ap}{\alpha - 1} \left\{ 1 - (\gamma \cdot t/A + 1)^{1-\alpha} \right\} / \gamma + 1 = \Omega(t^{1-\alpha}).$$

This proves that only a learning rate with amplification $\alpha > 1$ will prevent the malware from propagating to all k vulnerable nodes.

Case III ($\alpha = 1$): For completeness, if $\alpha = 1$, then taking integrals that led to (A.21) for $\alpha \neq 1$, now lead to

$$\frac{dy}{dt} = v = Ap \ln(y + A) + c_1 \tag{A.26}$$

with

$$c_1 = \gamma - Ap \ln A.$$

Notice that $y'(t) = Ap \ln(y/A + 1) + \gamma \geq \gamma$ and we use the same argument as we did above for $\alpha < 1$. We again have $y(t) \geq \gamma \cdot t$ and substituting this back gives

$$i(t) \geq Ap/\gamma \ln(\gamma \cdot t/A + 1) + 1 = \Omega(\ln t).$$

These lower bounds can be improved by recursively substituting lower bounds back into (A.25) and (A.26), respectively.

Bibliography

[1] N. Adams and N. Heard, *Dynamic Networks and Cyber-security.* World Scientific, 2016, vol. 1.

[2] E. Al-Shaer and M. A. Rahman, *Security and Resiliency Analytics for Smart Grids: Static and Dynamic Approaches.* Springer, 2016, vol. 67.

[3] T. Alpcan and T. Basar, "An intrusion detection game with limited observations," in *12th Int. Symp. on Dynamic Games and Applications, Sophia Antipolis, France*, vol. 26, 2006.

[4] P. E. Ammann and J. C. Knight, "Data diversity: An approach to software fault tolerance," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 418–425, 1988.

[5] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.

[6] Avast blog, "Seven new Mirai variants and the aspiring cybercriminal behind them," 2018, [Accessed Nov., 2018]. [Online]. Available: https://bit.ly/2GVoY2c

[7] S. Banerjee and P. Lofgren, "Fast bidirectional probability estimation in markov models," in *Advances in Neural Information Processing Systems*, 2015, pp. 1423–1431.

[8] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[10] C. Cadar, P. Akritidis, M. Costa, J.-P. Martin, and M. Castro, "Data randomization," Technical Report Microsoft Research TR-2008-120, Tech. Rep., 2008.

[11] T. E. Carroll and D. Grosu, "A game theoretic investigation of deception in network security," *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.

[12] K. M. Carter, J. F. Riordan, and H. Okhravi, "A game theoretic approach to strategy determination for dynamic platform defenses," in *Proceedings of the First ACM Workshop on Moving Target Defense*, 2014, pp. 21–30.

[13] L. Chen and J. Leneutre, "A game theoretical framework on intrusion detection in heterogeneous networks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 2, pp. 165–178, 2009.

[14] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection system," in *Information security and cryptology.* Springer, 2006, pp. 153–167.

[15] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003, pp. 1890–1900.

[16] Z. Chen and C. Ji, "An information-theoretic view of network-aware malware attacks," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 530–541, 2009.

[17] M. Christodorescu, M. Fredrikson, S. Jha, and J. Giffin, "End-to-end software diversification of internet services," in *Moving Target Defense.* Springer, 2011, pp. 117–130.

[18] Cisco, "Annual Cybersecurity Report," 2018, [Accessed Dec., 2018]. [Online]. Available: https://bit.ly/2ul3dOM

[19] W. Connell, D. A. Menasce, and M. Albanese, "Performance modeling of moving target defenses with reconfiguration limits," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[20] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-variant systems: a secretless framework for security through diversity," in *Usenix Security*, vol. 6, 2006, pp. 105–120.

[21] D. Dagon, C. C. Zou, and W. Lee, "Modeling botnet propagation using time zones." in *NDSS*, vol. 6, 2006, pp. 2–13.

[22] H. Debar, "An introduction to intrusion-detection systems," *Proceedings of Connect*, vol. 2000, 2000.

[23] C. C. Elisan, *Malware, Rootkits & Botnets A Beginner's Guide.* McGraw Hill Professional, 2012.

[24] D. Evans, A. Nguyen-Tuong, and J. Knight, "Effectiveness of moving target defenses," in *Moving Target Defense.* Springer, 2011, pp. 29–48.

[25] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on.* IEEE, 2009, pp. 268–273.

[26] FireEye, "M-Trends 2019, FireEye Mandiant Services — Special Report," 2019, [Accessed Nov., 2019]. [Online]. Available: https://bit.ly/2Oo9sMS

[27] P. Fogla, M. I. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee, "Polymorphic blending attacks." in *USENIX Security Symposium*, 2006, pp. 241–256.

[28] Y. Han, W. Lu, and S. Xu, "Characterizing the power of moving target defense via cyber epidemic dynamics," in *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, ser. HotSoS '14, 2014, pp. 10:1–10:12.

[29] D. A. Holland, A. T. Lim, and M. I. Seltzer, "An architecture a day keeps the hacker away," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 1, pp. 34–41, 2005.

[30] T. Huang, B. Satchidanandan, P. Kumar, and L. Xie, “An online detection framework for cyber attacks on automatic generation control,” *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6816–6827, 2018.

[31] N. Idika and A. P. Mathur, “A survey of malware detection techniques,” *Purdue University*, vol. 48, 2007.

[32] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, and K. Nakao, “Automated malware analysis system and its sandbox for revealing malware’s internal and external activities,” *IEICE transactions on information and systems*, vol. 92, no. 5, pp. 945–954, 2009.

[33] T. Jackson, B. Salamat, G. Wagner, C. Wimmer, and M. Franz, “On the effectiveness of multi-variant program execution for vulnerability detection and prevention,” in *Proceedings of the 6th International ACM Workshop on Security Measurements and Metrics*, 2010, p. 7.

[34] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “Openflow random host mutation: transparent moving target defense using software defined networking,” in *Proceedings of the first ACM Workshop on Hot topics in Software Defined Networks*, 2012, pp. 127–132.

[35] C. Jin, S. Valizadeh, and M. van Dijk, “Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems,” in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 824–829.

[36] C. Kil, J. Jun, C. Bookholt, J. Xu, and P. Ning, “Address space layout permutation (aslp): Towards fine-grained randomization of commodity software,” in

IEEE Proceedings of the 22nd Annual Computer Security Applications Conference, 2006, pp. 339–348.

[37] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection." in *USENIX security symposium*, vol. 286. San Diego, CA, 2004.

[38] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence.* Springer, 2016, pp. 137–149.

[39] C. Kreibich and J. Crowcroft, "Honeycomb: creating intrusion detection signatures using honeypots," *ACM SIGCOMM computer communication review*, vol. 34, no. 1, pp. 51–56, 2004.

[40] C. E. Lee, A. Ozdaglar, and D. Shah, "Computing the stationary distribution locally," in *Advances in Neural Information Processing Systems*, 2013, pp. 1376–1384.

[41] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, 2008.

[42] Z. Li, M. Sanghi, Y. Chen, M.-Y. Kao, and B. Chavez, "Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience," in *Security and Privacy, 2006 IEEE Symposium on.* IEEE, 2006, pp. 15–pp.

[43] X. Liang and Y. Xiao, "Game theory for network security," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 472–486, 2013.

[44] Z. Liang and R. Sekar, "Fast and automated generation of attack signatures: A basis for building self-protecting servers," in *Proceedings of the 12th ACM conference on Computer and communications security.* ACM, 2005, pp. 213–222.

[45] Y. Lindell and J. Katz, *Introduction to modern cryptography.* Chapman and Hall/CRC, 2014.

[46] K.-w. Lye and J. M. Wing, "Game strategies in network security," *International Journal of Information Security*, vol. 4, no. 1-2, pp. 71–86, 2005.

[47] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. van Dijk, "Markov modeling of moving target defense games," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense.* ACM, 2016, pp. 81–92.

[48] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," in *Moving Target Defense II.* Springer, 2013, pp. 1–13.

[49] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux, "Game theory meets network security and privacy," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 25, 2013.

[50] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection," *Computer Networks*, vol. 109, pp. 127–141, 2016.

[51] J. A. Marpaung, M. Sain, and H.-J. Lee, "Survey on malware evasion techniques: State of the art and challenges," in *Advanced Communication Tech-*

nology (ICACT), 2012 14th International Conference on. IEEE, 2012, pp. 744–749.

[52] McAfee Executive Perspectives, "Wannacry: The Old Worms and the New," 2017, [Accessed Aug., 2019]. [Online]. Available: https://bit.ly/2KMlkro

[53] Microsoft Security Bulletin, "Security update for microsoft windows smb server (4013389)," 2017, [Accessed Aug., 2019]. [Online]. Available: https://bit.ly/2jfsWVc

[54] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003, pp. 1901–1910.

[55] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Security and privacy, 2005 IEEE symposium on.* IEEE, 2005, pp. 226–241.

[56] ——, "Paragraph: Thwarting signature learning by training maliciously," in *International Workshop on Recent Advances in Intrusion Detection.* Springer, 2006, pp. 81–105.

[57] A. Nguyen-Tuong, D. Evans, J. C. Knight, B. Cox, and J. W. Davidson, "Security through redundant data diversity," in *IEEE International Conference on Dependable Systems and Networks*, 2008, pp. 187–196.

[58] A. Nochenson and C. L. Heimann, “Simulation and game-theoretic analysis of an attacker-defender game,” in *Decision and Game Theory for Security*. Springer, 2012, pp. 138–151.

[59] A. J. O’Donnell and H. Sethu, “On achieving software diversity for improved network security using distributed coloring algorithms,” in *Proceedings of the 11th ACM conference on Computer and Communications Security*, 2004, pp. 121–131.

[60] Official website of the Department of Homeland Security, “Moving target defense,” [Accessed Nov., 2019]. [Online]. Available: https://bit.ly/2RaI4VM

[61] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein, “Survey of cyber moving targets,” Report Massachusetts Inst of Technology Lexington Lincoln Lab MIT/LL-TR-1166, 2013.

[62] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein, “Finding focus in the blur of moving-target techniques,” *IEEE Security & Privacy*, vol. 12, no. 2, pp. 16–26, 2014.

[63] K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti, and E. Kirda, “G-free: defeating return-oriented programming through gadget-less binaries,” in *IEEE Proceedings of the 26th Annual Computer Security Applications Conference*, 2010, pp. 49–58.

[64] Panda Security, “What is a botnet?” 2017, [Accessed May, 2018]. [Online]. Available: https://bit.ly/2Sb2WJA

[65] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif, "Misleading worm signature generators using deliberate noise injection," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*. IEEE, 2006, pp. 15–pp.

[66] M. Petkac, L. Badger, and W. Morrison, "Security agility for dynamic execution environments," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 1. IEEE, 2000, pp. 377–390.

[67] N. Provos *et al.*, "A virtual honeypot framework." in *USENIX Security Symposium*, vol. 173, 2004, pp. 1–14.

[68] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[69] M. Roberts and J. Heesterbeek, "Mathematical models in epidemiology," 2004.

[70] T. Roeder and F. B. Schneider, "Proactive obfuscation," *ACM Transactions on Computer Systems (TOCS)*, vol. 28, no. 2, p. 4, 2010.

[71] K. R. Rohloff and T. Basar, "Stochastic behavior of random constant scanning worms," in *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*. IEEE, 2005, pp. 339–344.

[72] B. Salamat, A. Gal, and M. Franz, "Reverse stack execution in a multi-variant execution environment," in *Workshop on Compiler and Architectural Techniques for Application Reliability and Security*, 2008, pp. 1–7.

[73] B. Salamat, T. Jackson, G. Wagner, C. Wimmer, and M. Franz, "Runtime defense against code injection attacks using replicated execution," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 588–601, 2011.

[74] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999.

[75] S. H. Sellke, N. B. Shroff, and S. Bagchi, "Modeling and automated containment of worms," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 2, pp. 71–86, 2008.

[76] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "Moving target defense for the placement of intrusion detection systems in the cloud," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 326–345.

[77] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 298–307.

[78] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting." in *OSDI*, vol. 4, 2004, pp. 4–4.

[79] A. N. Sovarel, D. Evans, and N. Paul, "Where's the feeb? the effectiveness of instruction set randomization." in *Usenix Security*, 2005, p. 10.

[80] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection." *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[81] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 580–591, 2014.

[82] Y. Tang, B. Xiao, and X. Lu, "Signature tree generation for polymorphic worms," *IEEE transactions on computers*, vol. 60, no. 4, pp. 565–579, 2011.

[83] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, and K. Franke, "Combining expert knowledge with automatic feature extraction for reliable web attack detection," *Security and Communication Networks*, vol. 8, no. 16, pp. 2750–2767, 2015.

[84] S. Valizadeh and M. van Dijk, "Malpro: A learning-based malware propagation and containment modeling," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop.* ACM, 2019, pp. 45–56.

[85] ——, "On the convergence rates of learning-based signature generation schemes to contain self-propagating malware," *arXiv preprint arXiv:1905.00154*, 2019.

[86] ——, "Toward a theory of cyber attacks," *arXiv preprint arXiv:1901.01598*, 2019.

[87] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "Flipit: The game of "stealthy takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013.

[88] M. Van Gundy, D. Balzarotti, and G. Vigna, "Catch me, if you can: Evading network signatures with web-based polymorphic worms." in *WOOT*, 2007.

[89] S. Venkataraman, A. Blum, and D. Song, "Limits of learning-based signature generation with adversaries," 2008.

[90] L. Wang, Z. Li, Y. Chen, Z. Fu, and X. Li, “Thwarting zero-day polymorphic worms with network-level length-based signature generation,” *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 1, pp. 53–66, 2010.

[91] Y. Wang, S. Wen, Y. Xiang, and W. Zhou, “Modeling the propagation of worms in networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 942–960, 2014.

[92] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, “A taxonomy of computer worms,” in *Proceedings of the 2003 ACM workshop on Rapid malcode.* ACM, 2003, pp. 11–18.

[93] T.-F. Yen and M. K. Reiter, “Traffic aggregation for malware detection,” *Lecture Notes in Computer Science*, vol. 5137, pp. 207–227, 2008.

[94] K. Zaffarano, J. Taylor, and S. Hamilton, “A quantitative framework for moving target defense effectiveness evaluation,” in *Proceedings of the Second ACM Workshop on Moving Target Defense*, 2015, pp. 3–10.

[95] Y. Zeng, X. Hu, and K. G. Shin, “Detection of botnets using combined host-and network-level information,” in *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on.* IEEE, 2010, pp. 291–300.

[96] Q. Zhu and T. Başar, “Game-theoretic approach to feedback-driven multi-stage moving target defense,” in *Decision and Game Theory for Security.* Springer, 2013, pp. 246–263.

[97] R. Zhuang, “A theory for understanding and quantifying moving target defense,” Ph.D. dissertation, Kansas State University, 2015.

[98] R. Zhuang, A. G. Bardas, S. A. DeLoach, and X. Ou, “A theory of cyber attacks: A step towards analyzing mtd systems,” in *Proceedings of the Second ACM Workshop on Moving Target Defense*, 2015, pp. 11–20.

[99] R. Zhuang, S. A. DeLoach, and X. Ou, “A model for analyzing the effect of moving target defenses on enterprise networks,” in *ACM Proceedings of the 9th Annual Cyber and Information Security Research Conference*, 2014, pp. 73–76.

[100] ——, “Towards a theory of moving target defense,” in *Proceedings of the First ACM Workshop on Moving Target Defense.* ACM, 2014, pp. 31–40.

[101] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, “Simulation-based approaches to studying effectiveness of moving-target network defense,” in *National symposium on moving target research*, 2012.

[102] C. Zimmerman, “Ten strategies of a world-class cybersecurity operations center,” MITRE Corporation report release, 2014.

[103] C. C. Zou, W. Gong, and D. Towsley, “Code red worm propagation modeling and analysis,” in *Proceedings of the 9th ACM conference on Computer and communications security.* ACM, 2002, pp. 138–147.