SAND2015-8432R

Evaluating Moving Target Defense with PLADD

Stephen Jones, Alexander Outkin, Jared Gearhart,
Jacob Hobbs, John Siirola, Cindy Phillips, Stephen Verzi,
Daniel Tauritz, Samuel Mulder, and Asmeret Naugle

September 15, 2015

Abstract

This project evaluates the effectiveness of moving target defense (MTD) techniques using a new game we have designed, called PLADD, inspired by the game FlipIt [28]. PLADD extends FlipIt by incorporating what we believe are key MTD concepts. We have analyzed PLADD and proven the existence of a defender strategy that pushes a rational attacker out of the game, demonstrated how limited the strategies available to an attacker are in PLADD, and derived analytic expressions for the expected utility of the game's players in multiple game variants. We have created an algorithm for finding a defender's optimal PLADD strategy. We show that in the special case of achieving deterrence in PLADD, MTD is not always cost effective and that its optimal deployment may shift abruptly from not using MTD at all to using it as aggressively as possible. We believe our effort provides basic, fundamental insights into the use of MTD, but conclude that a truly practical analysis requires model selection and calibration based on real scenarios and empirical data. We propose several avenues for further inquiry, including (1) agents with adaptive capabilities more reflective of real world adversaries, (2) the presence of multiple, heterogeneous adversaries, (3) computational game theory-based approaches such as coevolution to allow scaling to the real world beyond the limitations of analytical analysis and classical game theory, (4) mapping the game to real-world scenarios, (5) taking player risk into account when designing a strategy (in addition to expected payoff), (6) improving our understanding of the dynamic nature of MTD-inspired games by using a martingale representation, defensive forecasting, and techniques from signal processing, and (7) using adversarial games to develop inherently resilient cyber systems.

Contents

1 Motivation

National security and our national economy are both highly dependent on the safe and secure operation of a huge range of networked computing and control systems.

Software developers are getting better at building commodity computing systems that have a smaller average defect rate [6]. The fact that we experience less frequent random faults and crashes when interacting with our personal computers, phones, and web services than we once did provides anecdotal evidence for this. Fewer flaws are good news for reliability. Today, low to medium consequence enterprises run their organizations on commodity hardware and software with high confidence.

However, the current safety and reliability state of the art is inadequate for systems that are subject to attack by an intelligent, adaptable, and well-resourced adversary. In this case, we cannot depend on faults being exercised with low probability based on their relatively low frequency. Attackers find and exploit low density flaws routinely. As long as we cannot remove all flaws and produce perfect systems and associated processes, we need additional security techniques that help us operate effectively in an imperfect world.

Moving target defenses (MTDs) are one category of mitigations that use diversity, randomization, and change to make exploiting vulnerabilities more costly for an adversary without having to find and remove all flaws from a system [10]. MTDs come in different flavors that typically transform some aspect of a system from static and known into dynamic and/or hidden. For example an MTD might identify information required by an adversary to mount a successful attack, then randomize and hide that information with the hope that the additional cost of discovering it will make the attack infeasible, or at least uneconomical. Another approach limits the scope of a successful attack, often without knowing that it occurred, by periodically performing a reset to a known good state. Still other techniques promote eventual recovery by holding some set of resources in reserve. Okhravi et al. have provided a thorough survey of contemporary MTDs [18].

Some MTDs, like address-space layout randomization (ASLR) [24] and managed entropy random number generators [14], have been widely deployed and have seen major success; others like network address randomization are often discussed, but have not been widely adopted. Many questions remain about which MTDs increase adversary costs, by how much, and how to use them effectively without breaking the defender's bank.

This paper explores the value of MTD. We have developed a strategic game called PLADD, which stands for **P**robabilistic **L**earning **A**ttacker, **D**ynamic **D**efender (PLADD), that captures what we believe are important, basic aspects of MTD. We take an abstract modeling and analysis approach by intentionally ignoring details that are potentially important for implementing realistic MTD, but whose complexity might detract from our understanding of MTD's fundamental properties and behavior. We first present a simple taxonomy of MTDs to frame our discussion. We then describe PLADD, followed by an analysis of the game and finally present some initial findings and design guidance based on results from experiments within our simplified context. The key questions we hope to begin to address are:

- What benefit does using MTD provide?

- If MTD is effective, when does one best use it, i.e., what are the best parameters and most suitable application scenarios?

We use a variety of techniques to explore the behavior and value of our basic formulation of MTD including analytic game theory, simulation, and stochastic programming. Each approach has its own advantages and disadvantages. For example, simulation is quick and easy to implement and can capture environmental details that are hard to represent analytically. Analytic approaches generalize more readily to whole classes of abstract scenarios. Stochastic programming combines the advantages of a bottom up implementation strategy while supporting the derivation of rigorous bounds on MTD performance.

1.1 Categories of Moving Target Defense

Based on a comprehensive survey of proposed MTD techniques [18], we have distilled MTDs into several general classes. We list them here in the order of their frequency of occurrence in the survey.

1. **Randomized secret**: Randomize some aspect of a system that an attacker needs to know about to mount a successful attack. Hide a key value that describes the random state from an attacker. For example, ASLR [24], randomized, keyed instruction sets [2, 13], randomized system call table [5], randomized IP or other network identification [1].
2. **N-variant systems with voting**: Deploy N versions of a system or component that are thought to fail independently. A proxy distributes user inputs to all N variants. Outputs are majority voted to select the ultimate response or to detect deviant behavior [7]. In the ideal case, an attacker must find a way to exploit a majority of the set of N systems *simultaneously* to be successful.
3. **Scheduled renewal**: Reset an environment to a desirable or known-good state according to some schedule or strategy. For example, password reset, Apache process pools, revert to virtual machine snapshot, or micro-reboots [4].
4. **Reserve for recovery**: Ensure that some fraction of a defender's resources are withheld from attacker access long enough that it is impractical for an attacker to compromise a complete system. Examples of this style of MTD include the Yarrow/Fortuna cryptographic random number generator [14], or the dynamic generation algorithms and the peer to peer controls used by botnets [22].
5. **Shuffle**: Deploy N versions of a system or component that are thought to fail independently. Each request is serviced by an unpredictable member of the pool. It is hoped that the uncertainty about which member of the pool will service a given request complicates exploitation.
6. **Chaff**: Pollute the space of systems and services with decoys that distract, delay and potentially reveal an attacker as they attempt to access and exploit the decoys [20]. Legitimate users have access to additional information, withheld from attackers, so they can avoid decoys.

1.2 The Essence of MTD

All of the categories of MTD described above have two important characteristics in common that we believe form the *essence* of moving target defense.

1. There is some information that, when held by the attacker, gives them a competitive advantage.
2. The defender has the ability to take that information away from the attacker (at least temporarily).

We believe PLADD incorporates these essential MTD features and may allow us to study the basic behaviors and effectiveness of MTD without becoming bogged down in implementation details.

2 Definition of the PLADD Game

The PLADD game is based on the FlipIt game [28]. FlipIt is intentionally simple, but exhibits interesting and complicated behaviors that provide insight into certain kinds of adversarial cyber scenarios. PLADD attempts to retain FlipIt's simplicity, but adds features that capture the key elements of MTD. It intentionally avoids including important, but complicating implementation details. Our goal is to achieve some general insights about MTDs using this incomplete, but illustrative model.

2.1 FlipIt Review

In FlipIt, two players, the defender and the attacker, vie for control of a single shared resource. Each player can make a move at any time by incurring a fixed cost. Defender and attacker costs are independent and the magnitude of the costs is a game parameter. When a player moves, they immediately gain control of the resource if it was controlled by their opponent or retain control of the resource if they already controlled it. Control is retained until the other player moves. The utility of each player is calculated as the player's payout, which is the amount of time they controlled the resource minus the sum of costs the player incurred for all moves. Note that costs and payouts are both specified in units of time to make payout and cost comparable. A strategy in FlipIt consists of the schedule of moves a player makes.

In FlipIt, the players cannot observe when their adversary makes a move (i.e. when they lose control of the resource). A player only knows that they control the resource immediately after their own moves, which earns FlipIt the sub-title "game of stealthy takeover". Figure 1 shows an example of a FlipIt game and how the resource changes hands between an attacker and a defender as the game progresses.

The original FlipIt paper analyzes this basic game and several extensions where the attacker is afforded progressively more information about the defender's strategy such as their average

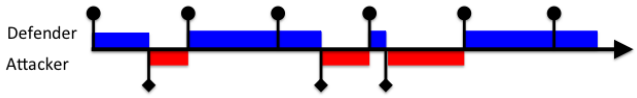

Figure 1: An illustrative example of the FlipIt game showing the resource changing hands between the attacker (red) and the defender (blue) as time progresses from left to right.

move rate. Subsequent papers by the original authors and others extend FlipIt to study multiple resources [15], multiple classes of attacker [16], and the ability of a defender to probe the state of the resource [19].

2.2 PLADD Extensions to the FlipIt Game

PLADD has several features that distinguish it from FlipIt. First, attackers do not take immediate control of the resource when they move, but are subject to a random time-to-success delay. Second, the probability distribution of the attacker's time-to-success changes according to the state of the game. This change represents the attacker's ability to learn. Third, defenders have two moves at their disposal. The first move, referred to as a "take", seizes control of the resource. The second move, referred to as a "morph", seizes control of the resource and takes information away from an attacker. Both types of defender moves succeed immediately. Fourth, the attacker is able to detect defender moves under certain conditions. Costs are incurred by the attacker and defender for each type of move. The following subsections provide detailed descriptions each of the features of the PLADD game.

2.2.1 Attacker Time-to-Success

In PLADD, an attacker does not seize control of the resource immediately when they move. Rather, the attacker will gain control at some random time after initiating an attack. For example, if an attacker starts an attack at time t it will succeed at some future time $t+s$ where s, the attacker's time-to-success, is a random variable drawn from a probability distribution $f(s)$. The random time-to-success represents the typically stochastic nature of cyber attacks.

Figure 2 illustrates how the time-to-success delay impacts the progression of the PLADD game. In this example, the attacker begins an attack immediately following a defender move, however the attack does not succeed until a future point in time. The attacker controls the resource from the time the attack succeeds until the defender's next move.

2.2.2 Attacker Learning

In PLADD, the distribution governing the attacker's time-to-success changes depending on the state of the game. These changes represent the attacker's ability to learn about the defender's system. At the beginning of the game, $f(s) = f_{base}(s)$, a distribution that represents

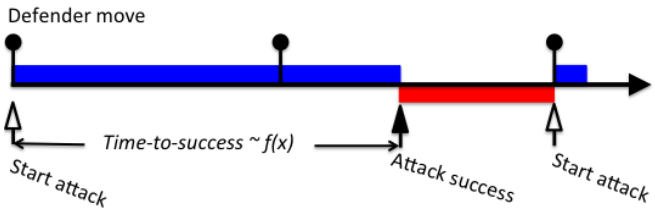

Figure 2: Illustration of the impact of the time-to-success distribution on the control of the resource.

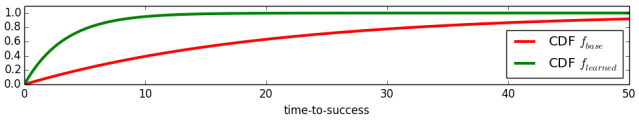

Figure 3: The attacker is subject to two different time-to-success distributions, indicated here with exponential CDFs.

a low state of knowledge for the attacker. We expect that this distribution will have relatively little density at small values of s which means that it is unlikely, but not impossible, for an attacker to succeed quickly against a new defense they are not familiar with. Once an attacker succeeds, $f(s)$ changes to $f_{learned}(s)$ to reflect the fact that the attacker has learned something new about the defender's system in the process of preparing for and executing a successful attack. Hence, $f_{learned}$ is likely to have greater density closer to $s = 0$, reflecting the attacker's ability to succeed earlier due to decreased uncertainty. Figure 3 shows a cumulative distribution function (CDF) for a notional $f_{base}(s)$ and $f_{learned}(s)$.

2.2.3 Defender Moves

The defender begins the game in control of the resource and has two move types available to them. A take immediately seizes control of the resource if it is controlled by the attacker, much like the standard FlipIt move. Take moves have no effect if the defender currently controls the resource. Ongoing attacks are not impacted by the take move.

The morph move, has multiple effects. Like the take move, it immediately takes control of the resource if the attacker has control. Unlike the take move, it cancels any attack that has been initiated, but hasn't yet succeeded and sets the attacker's time-to-success distribution to f_{base}. These two additional features make the morph move more powerful than the take move.

The ability of the morph to unconditionally set the attacker's time-to-success distribution to f_{base} represents the defender's moving target defense, i.e., it takes information away from the attacker.

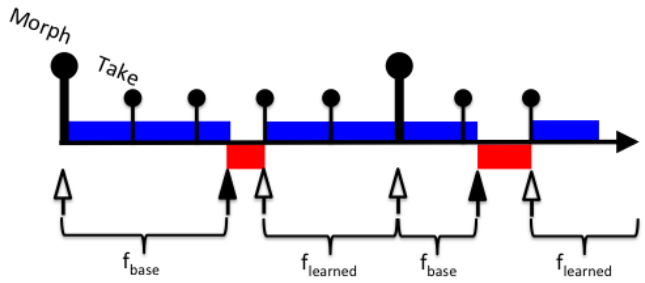

Figure 4: Illustration of the progress of the PLADD game for the attacker and defender.

2.2.4 Information Available to the Players

In PLADD, the information available to the defender is the same as in FlipIt. The defender cannot obtain information on the state of the resource unless they make a move, at which point they will know they control the resource.

Unlike FlipIt, the attacker is able to detect defender moves under certain conditions. First, an attacker can detect both types of defender moves when the attacker controls the resource. This implies that if the attacker has control of the resource, they can readily detect when they lose control. Second, an attacker can detect a morph move even if they do not control the resource since morph moves are assumed to cancel ongoing attacks. It is assumed that only a single attack can be in progress at any point in time.

Figure 4 shows a portion of the time line for a notional example of the PLADD game. This example begins when the defender makes a morph move. Since the attacker can detect this move, they start a new attack using the f_{base} distribution. While this attack is in progress, the defender makes two take moves, which have no impact since the defender controls the resource. Once the attack succeeds, the attacker controls the resource until the next take move. At this point the attacker detects that they have lost control of the resource and they begin an attack using the $f_{learned}$ distribution. This attack does not succeed before the defender's next morph move. When the morph move occurs, the attack is interrupted and the attacker starts a new attack using the f_{base} distribution.

2.2.5 Player Move Costs

A defender is not informed about any attacker moves and incurs a fixed cost of c_{take} or c_{morph} for the take and morph moves respectively. Because morph is a more capable move, we expect morphs to be more expensive than takes.

An attacker incurs two attack costs. As in FlipIt, they pay a fixed cost to initiate a new attack, which is denoted by α. They also incur a variable cost proportional to the duration of the attack, which we will call β.

2.2.6 Summary of PLADD Parameters and Notation

The list below summarizes the parameters used in the PLADD game:

- $f_{base}(s)$: The attacker's time-to-success distribution that represents a low state of knowledge for the attacker
- $f_{base}(s)$: The attacker's time-to-success distribution when the attacker has learned something new about the defender's system
- α: The attacker's fixed cost to initiate a new attack
- β: The attacker's variable cost related to the duration of an attack
- c_{take}: The defender's fixed cost for initiating a take move
- c_{morph}: The defender's fixed cost for initiating a morph move
- τ: The defender period or, when subscripted, the gaps between defender moves
- N: The number of take moves between morphs in a game that includes morphs

2.3 Strategies and Utilities

As in FlipIt, both the attacker and defender receive a payout in units of time proportional to the time they control the resource. They also incur costs, as described above, for making moves. The attacker's and defender's objective is to maximize their utility.

$$utility = \text{“}cumulative\ duration\ of\ control\text{”} - \text{“}cumulative\ costs\text{”}.$$

Strategies in the game consist of selecting if, when, and what type of moves should be made.

2.4 Game Types

In this paper, our ultimate goal is to examine the value of MTD. To that end, we distinguish between two types of PLADD games.

- A version of PLADD in which both take and morph moves are available. We call this version the "finite" game because the morph moves split games into finite chunks played between subsequent morph moves.
- A version of PLADD in which no morph move is available. We call this the "infinite" version of the game in contrast to the finite version described above because morph moves do not break the game into chunks.

2.5 Limited Attacker Strategy

An interesting, and unintentional, consequence of the game definition, is that the attacker appears to have limited freedom to specify a strategy. In PLADD, if the attacker's time to success distribution monotonically decreases, i.e., they are more likely to succeed sooner than later, it is always in the attacker's best interest to initiate an attack immediately if they do not control the resource. A proof appears in Section 3.1.2.

3 Analysis of PLADD

In this section we describe our analysis of PLADD. We begin by deriving analytic solutions for the attacker and defender utilities for the infinite and finite versions of the PLADD game given a variety of assumptions and restrictions. For the specific case of defenders using a periodic strategy and exponentially distributed attacker time-to-success, we perform an analysis of the value of MTD in PLADD. Finally, we derive general equations for defender utility with no restriction on defender strategy or time-to-success distribution and describe an algorithm for selecting the optimal defender deterrence strategy in PLADD.

3.1 Infinite Game

This subsection analyzes the infinite version of the PLADD game and derives expressions for calculating the attacker utility and controlling it by means of the defender strategy. We start with an equation for the utility of an attacker given a general defender strategy. We then consider the case where a defender is restricted to periodic strategies and show that such strategies are sufficient to push the attacker out of the game in the case of general time-to-success distributions. We then apply our results to an exponential distribution of the attacker time-to success, show that a periodic defender strategy that pushes the attacker out of the game is unique, and derive an explicit equation from which such a strategy can be calculated.

In general, we would like to characterize the dependence of the attacker's expected utility on the defender's strategy and its parameters, in expectation that such a dependence may be used to the defender's advantage.

We make the following assumptions, which ensure that strategies are realistic:

- Both the attacker and the defender have finite budgets with which to execute their strategies and therefore will do a finite number of moves in any finite time interval. We believe this assumption does not limit the applicability of our analysis or of the results.
- Aside from the attacker time-to-success switching from f_{base} to $f_{learned}$ when they first succeed, there is no other learning occurring during the game. We understand that implicit information about the defender's strategy and the attacker's time-to-success distribution is available in the game. For example, the attacker is notified when they

lose control of the resource and can use this fact to infer information about the defender's strategy. However, for simplicity, we have chosen not to take these sources of information into account. The more general question of adaptive strategies that respond to online learning is an interesting and promising area of future work.

The attacker's utility depends on the defender's strategy. In the case of the infinite PLADD game, the defender's strategy can be represented as $s = \{t_i,\, i = 1, ..., \infty\}$, which represents an infinite sequence of times where the defender will make moves. In the infinite game all moves are take moves, there are no morph moves. Given the defender strategy, we can calculate the attacker expected utility for the next move of the game, until they lose control of the resource.

As one innovative part of this analysis, we analyze the attacker expected utility at time t, when they lose control[1]. By definition, this time t coincides with a move by the defender. In this way we are partitioning the infinite game into manageable chunks defined by the points when the attacker loses control of the resource. Calculating the expected utility at the time of the attacker's loss of control allows us to roll in the attacker costs and benefits into the expected value of the attacker utility for the next period and treat the attacker cumulative net utility function as a martingale (see [25] or [29] for a more detailed description of such an approach).

In the infinite duration game, the attacker utility, u, for an attack duration x drawn from $f(x)$ and given a defender strategy s can be expressed as follows:

$$u = -\alpha - \beta x + \left(\min_{t_i \in s}(t_i : t_i \geq x) - x \right), \tag{1}$$

where $f(x)$ is a probability density function of attacker time-to-success.

The last term in this equation is simply the duration of attacker control of the resource and is visually represented in Figure 2 as the red line.

The expected utility from this attack can be calculated as

$$E[u] = -\alpha - \beta \int_0^\infty x f(x)\ \mathrm{d}x + \int_0^\infty \left(\min_{t_i \in s}(t_i : t_i \geq x) - x \right) f(x)\ \mathrm{d}x \tag{2}$$

3.1.1 Periodic Solution for the Infinite Game

In the infinite case we do not have an a priory reason to assume a non-periodic strategy for the defender. We therefore start with a periodic defender strategy, calculate the expected attacker

[1]The analytic approach we take is inspired by work from Glenn Shafer, Vladimir Vovk, and Akimichi Takemura on Game Theoretic Probability and Defensive Forecasting. Their approach to strategic interaction using martingales served as a starting point for the formulation of the problem, the attacker expected utility definition, and the choice of deterrence as a cornerstone of the analysis

utility given a periodic defender strategy and show that the periodic defender strategy is sufficient in the infinite case to drive the attacker out of the game. We specifically show that the defender has the ability to control the attacker expected utility and in particular drive it to zero. We interpret the ability to achieve an outcome where the attacker expected payoff is zero as the ability to drive a rational attacker out of the game. The defender also has an option of maximizing his own utility. In this latter case, there is generally no guarantee that the attacker would choose not to play, however there are game parameters where the attacker expected utility would be strictly negative in this case.

As a starting point, we derive the attacker expected utility and show that for a periodic defender strategy with period τ it is a function of τ. This explicit dependence of the attacker utility on the defender strategy gives the defender an ability to make attacks unprofitable and to partially control the attacker's utility or risk taken.

Our initial goal is to find out the conditions under which such a break-even value of τ exists and calculate its value. An explicit value for τ also allows calculating if a specific budget would allow a defender to force the attacker out of the game. For the purposes of this analysis, we assume that the attacker success distribution, $f(x)$, remains constant. We further assume that the defender moves at periodic time intervals τ and the defender strategy s can therefore be represented as follows: $s = \{t_i, : t_{i+1} - t_i = \tau, \, i = 1, ..., \infty, t_1 = 0\}$

The attacker incremental utility can now be expressed by:

$$u = -\alpha - \beta x + \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right), \tag{3}$$

where the last term is the duration of the attacker control, and $\lceil . \rceil$ is the ceiling function. The more general formulation in Equation 1 can be simplified because of the defender move periodicity.

The expected utility from this attack can therefore be calculated as:

$$E[u] = -\alpha - \beta \int_0^\infty x f(x) \, \mathrm{d}x + \int_0^\infty \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x) \, \mathrm{d}x \tag{4}$$

Given that the ceiling function is just a step-wise function that goes up by 1 at each integer increment of τ, the expression for the attacker utility can be simplified to:

$$E[u] = c_1 + \sum_{k=1}^{\infty} \int_{(k-1)\tau}^{k\tau} (k\tau - x) f(x) \, \mathrm{d}x \tag{5}$$

It is interesting to notice that the expression under the integral in Equation 5 can be thought of as a convolution of the defender strategy or its effect on the attacker payoff and the attacker success probability density function, thus bringing multiple connections to signal processing, reliability, stochastic control, and Fourier transformations in addition to the, already noted,

connections to martingales and game-theoretic probability. In this interpretation, the effective defender strategy is represented by a saw-tooth function that linearly decreases in the range of $(\tau, 0)$ on each interval (t_i, t_{i+1}) for any $i \in \{1, ..., \infty\}$. This also allows a possibility of using well-established deconvolution algorithms by both the attacker and defender to estimate each others' strategies, in addition to the possibility of applying learning approaches to the same problem.

Equation 5 can be further simplified to

$$E[u] = c_1 + \tau \sum_{k=1}^{\infty} k \int_{(k-1)\tau}^{k\tau} f(x)\ \mathrm{d}x - \sum_{k=1}^{\infty} \int_{(k-1)\tau}^{k\tau} x f(x)\ \mathrm{d}x, \tag{6}$$

where

$$c_1 = -\alpha - \beta \int_0^{\infty} x f(x)\ \mathrm{d}x \tag{7}$$

Existence of Optimal τ. In this section we present the analysis of the dependence of $E[u]$ on τ and show that there is an optimal value of τ which ensures that the attacker's expected payoff is zero. We investigate this dependence by deriving an explicit expression for $\frac{dE[u]}{d\,\tau}$. Starting with Equation 6 and differentiating it with respect to τ, we obtain:

$$\frac{dE[u]}{d\,\tau} = \sum_{k=1}^{\infty} k \int_{(k-1)\tau}^{k\tau} f(x)\ \mathrm{d}x + \tau \sum_{k=1}^{\infty} k \Big[k f(k\tau) - (k-1) f(k-1)\tau \Big] - \\ - \sum_{k=1}^{\infty} \Big[k\, k\, \tau f(k\tau) - (k-1)(k-1)\tau f((k-1)\tau) \Big] \tag{8}$$

Some of the terms above immediately cancel each other out. By simplifying the remaining terms, we further obtain:

$$\frac{dE[u]}{d\,\tau} = \sum_{k=1}^{\infty} k \left(\int_{(k-1)\tau}^{k\tau} f(x)\ \mathrm{d}x - \tau f(k\tau) \right) \tag{9}$$

The geometric interpretation of this equation is that the positive and negative component for each k represent the difference between the exact probability of attacker success between points $(k-1)\tau$ and $k\tau$ and its approximation obtained by multiplying the length of the interval (τ) by the value of the probability density function value at its right end $(f(k\tau))$. Therefore, it follows immediately that

$$\lim_{\tau \to 0} \frac{dE[u]}{d\,\tau} = 0 \tag{10}$$

We could also show that

$$\lim_{\tau \to \infty} \frac{dE[u]}{d\,\tau} = 1 \tag{11}$$

The latter equation simply means that if the defender never re-takes, then the attacker utility grows linearly with time.

We observe that: $\lim_{\tau \to 0} E[u] < 0$, because the attacker has a negative fixed cost, and that there exist a τ at which the attacker expected utility $E[u]$ is strictly positive, given Equation 11. Given that $E[u]$ is a continuous function of τ and it takes a negative value at 0 and a positive value at some $\tau > 0$, we conclude by the intermediate value theorem that there exists a τ_0, such that $E[u(\tau_0)] = 0$ for any $f(x)$ with a finite first moment.

We have therefore proven that there exists a value of τ below which the attacker expected utility is negative and at which the attacker expected utility is zero. The attacker cumulative utility is also a martingale at τ_0. Therefore, the expected value of the attacker total expected payoff as calculated along any path in the game is 0 as well.

Defender optimal strategy: τ_0. The optimal defender period τ for any continuous distribution $f(x)$ that has a valid first moment can be calculated by setting the left side of Equation 5 to zero and numerically solving for τ_0 that sets the right side of that equation to zero as well.

3.1.2 Value of delaying attacks

The PLADD game gives the attacker only two strategic choices when they do not control the resource. They can 1) delay their attack, and 2) restart an ongoing attack after some time. Restarting an attack ensures on average that the attacker will increase their cost and time-to-success and does not appear to be a viable option. Delaying an attack, however, may bring the attacker payoff and utility benefits, as well as reductions in the expected cost of the attack. Intuitively, the attacker may be able to shift the most likely part of their success distribution to the right in time, so that they are more likely to succeed right after a defender take move, and thus may have longer control of the resource before it is retaken by the defender.

We illustrate this intuition with a simple example: Assume that the defender plays with period $\tau > 0$ and that the attacker's time-to-success function is $\delta(x - \frac{3\tau}{4})$, where $\delta(.)$ is the Dirac delta function, and x is the time since beginning of the attack. By definition of the game, when played without delay, the beginning of the attack coincides with the defender take moves and results in the attacker controlling the resource for $\frac{\tau}{4}$ between the defender take moves. Therefore, the attacker would benefit from delaying the beginning of their attack by $\frac{\tau}{4} + \epsilon$ in order have control of the entire following period between defender moves (ϵ is an arbitrary small real number that is negligible as compared to τ. It is only needed here to ensure no ties associated with defender and attacker moving at the same time). By delaying the attack by $\epsilon + \frac{1}{4}$ the expected average attacker payoff can be made arbitrarily close to $\frac{\tau}{2}$. By design of the game the attacker would then have control in exactly half of the control

periods and therefore achieve a payoff of $\tau - \epsilon$ in every other period, thus bringing their average payoff arbitrary close to $\frac{\tau}{2}$. By the design of the game, when delaying the attack by $\frac{\tau}{4} + \epsilon$ the attacker would also be attacking in every other time period, rather than in every period in the game without delay, therefore ensuring 50% cost reduction in addition to the payoff benefits.

There are two caveats with this example: 1) $f(x) = \delta(.)$ is a special function and does not belong to the set of functions for which we've proven the existence of an optimal τ_0, and 2) periodic play may not be the optimal strategy in this case for the defender. Yet, even with those caveats, this example demonstrates that there are situations when delay is a viable strategic option for the attacker.

This example also demonstrates the richness of such an apparently simple game as PLADD: even in the situation where the attacker seemingly has no strategic choices, such actions as delaying the attack may still tilt the odds significantly in the attacker's favor. Combined with the attacker's ability to estimate the defender strategy by learning or by deconvolution, this may change the game outcomes significantly, and constitutes a promising venue for future research.

In the remainder of this section, we demonstrate that the delay brings no benefit to the attacker in the case of a monotonically declining time-to-success distribution $f(x)$ and provide an additional example when delay is beneficial to the attacker.

As before, we assume that the attacker time-to-success, x, is distributed according to $f(x)$, when starting at time 0. To evaluate the contribution of a delay to attacker utility, we assume the current time is 0, and the attacker will wait until time δ to start their attack. We call the time when the attacker succeeds x, as before, with the difference that x can only occur starting at δ and is therefore distributed by $f(x - \delta)$. The delay does not affect the fixed cost α, however it changes the variable cost to $\beta(x - \delta)$, because we assume that the variable cost will not be incurred until the attack started at time δ.

By not incurring any cost while waiting to start the attack, we are assigning the most optimistic value to delay, because in reality, waiting for a proper time to start an attack may incur costs as well. In reality, the attacker will incur at least opportunity costs, i.e., they may have fewer opportunities to learn about the defender without attacking, and the defender will have an opportunity to direct their efforts on strengthening their system during the lull in the attacks.

With the delay δ, the attacker utility can be expressed, similar to Equation 3, as follows:

$$u = -\alpha - \beta(x - \delta) + \left\lceil \frac{x}{\tau} \right\rceil \tau - x, \tag{12}$$

where x is drawn from $f(x - \delta)$. The expected utility, given a delay δ, can be expressed as follows:

$$E[u(\delta)] = -\alpha - \beta \int_0^\infty x f(x)\ \mathrm{d}x + \int_\delta^\infty \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x-\delta)\ \mathrm{d}x \tag{13}$$

We can verify that $E[u(\delta)]|_{\delta=0}$ from Equation 13 equals $E[u(0)]$ from Equation 4. The incremental utility $\Delta E[u(\delta)]$ from delaying the attack by δ can be calculated by subtracting the right hand side of Equation 4 from the right-hand side of Equation 13:

$$\Delta E[u(\delta)] = \int_\delta^\infty \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x-\delta)\ \mathrm{d}x - \int_0^\infty \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x)\ \mathrm{d}x \tag{14}$$

Given that we know that $\Delta E[u(0)] = 0$, we can evaluate the derivative of $\Delta E[u(\delta)]$ with respect to δ in the positive vicinity of 0. If the sign is positive or negative, then introducing small delays would result in increased or decreased attacker payoffs respectively. If the sign of the derivative remains the same on R^+ for a particular attacker distribution, then that would imply that delay always has either positive or negative effect on the attacker payoff. We can express the derivative of $\Delta E[u(\delta)]$ with respect to δ as follows

$$\frac{d\,\Delta E[u(\delta)]}{d\,\delta} = -\int_\delta^\infty \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) \left(\frac{d\,f(x-\delta)}{d\,x} \right)\ \mathrm{d}x - \left(\left\lceil \frac{\delta}{\tau} \right\rceil \tau - \delta \right) f(0) \tag{15}$$

We assume that $\delta < \tau$. This assumption is based on the intuition that delaying by exactly τ gives the exact same expectation for the attacker utility as if no delay occurred in an infinite game. Therefore, a larger than τ delay would bring no additional benefit as compared with a smaller than τ delay.

After integrating by parts, we can simplify Equation 15 as follows:

$$\frac{d\,\Delta E[u(\delta)]}{d\,\delta} = -\left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x-\delta) \Bigg|_\delta^\infty + \int_\delta^\infty \frac{d}{d\,x} \left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x \right) f(x-\delta)\,\mathrm{d}x - \left(\left\lceil \frac{\delta}{\tau} \right\rceil \tau - \delta \right) f(0) \tag{16}$$

The first and the last term in the above cancel each other, when we recognize that

$$\lim_{x\to\infty} -\left(\left\lceil \frac{x}{\tau} \right\rceil - x \right) f(x-\delta) = 0.$$

Now we are left with

$$\frac{d\,\Delta E[u(\delta)]}{d\,\delta} = \int_{\delta}^{\infty} \frac{d}{d\,x}\left(\left\lceil \frac{x}{\tau} \right\rceil \tau - x\right) f(x-\delta)\,\mathrm{d}x =$$
$$= \int_{\delta}^{\infty} \text{Ш}(\frac{x}{\tau}) f(x-\delta) - f(x-\delta)\,\mathrm{d}x =$$
$$= -\int_{0}^{\infty} f(y)\,\mathrm{d}y + \sum_{k=1}^{\infty} f(k\tau-\delta) = -1 + \sum_{k=1}^{\infty} f(k\tau-\delta) \quad (17)$$

where Ш is the Shah sampling function. Therefore, a delay may be worthwhile depending on the value of f at τ-length intervals.

We will now show that for any monotonically decreasing[2] f, the value of the delay is at best zero. Suppose, given the constraint that f must be monotonically decreasing, that the attacker had the power to shape f to try to make a delay worthwhile.

For a given δ and τ, the best the attacker can do is design f as a function which takes discrete downward steps at intervals of τ length, beginning at $\tau - \delta$. The reason for doing so is that the attacker wants $f(k\tau - \delta)$ to be as high as possible, and $f(k\tau - \delta + \epsilon)$ for $0 < \epsilon < \tau$ to be as low as possible, so as to not "waste" any of their distribution. Therefore, they will let $f(k\tau - \delta + \epsilon) = f((k+1)\tau - \delta)$. See Figure 5 for an idea of what this function would look like.

We may describe the optimal f, f^*, by a non-increasing sequence of points $a_1, a_2, a_3...$ such that

$$f^*(x) = \begin{array}{ll} 0 & \mid x < 0 \\ a_k & \mid (k-1)\tau - \delta < x \leq k\tau - \delta \end{array} \quad (18)$$

We then have

$$\frac{d\,\Delta E[u(\delta)]}{d\,\delta} = -1 + \sum_{k=1}^{\infty} f(k\tau-\delta) = -1 + \sum_{k=1}^{\infty} a_k \quad (19)$$

The requirement that f^* is a pdf constrains the possible values of a_k, since

$$\begin{array}{rcl} \int_0^{\infty} f^*(x)\,\mathrm{d}x & = & (\tau-\delta)a_1 + \sum_{k=2}^{\infty} \tau a_k \\ & = & -\delta a_1 + \tau \sum_{k=1}^{\infty} a_k \\ & = & 1 \end{array} \quad (20)$$

We now compare the expected value for the adversary using f^* with a delay, versus no delay. It is helpful to use Figure 5 as a guide for the derivation of these formulas.

[2]We include weak monotonicity in our definition of monotonically increasing or decreasing functions.

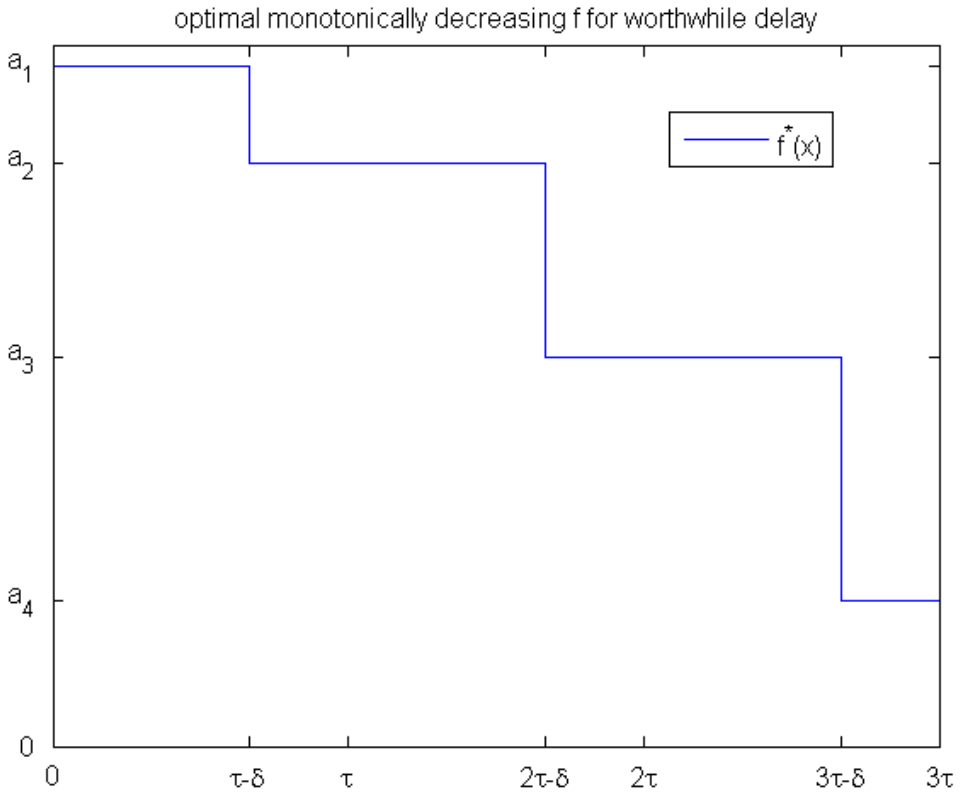

Figure 5: Notional picture of the candidate optimal monotonically decreasing function that makes delay worthwhile. The y axis is not to any scale, but preserves the decreasing sense of f^*.

$$\begin{aligned} E[u(\delta)|f^*] &= \tfrac{\tau}{2}\sum_{k=2}^{\infty} a_k + \tfrac{\tau-\delta}{2}a_1 \\ &= \tfrac{1}{2}\left(\tau\sum_{k=2}^{\infty} a_k + (\tau-\delta)\,a_1\right) \\ &= \tfrac{1}{2}(-\delta a_1 + \tau\sum_{k=1}^{\infty} a_k) \\ &= \tfrac{1}{2} \end{aligned} \tag{21}$$

Compare this to the expected value for the adversary given no delay.

$$\begin{aligned} E[u(0)|f^*] &= \sum_{k=1}^{\infty}(\tau - \tfrac{\tau-\delta}{2})a_k + \sum_{k=2}^{\infty}\tfrac{\delta}{2}a_k \\ &= -\tfrac{\delta}{2}a_1 + \tfrac{\tau}{2}\sum_{k=1}^{\infty} a_k + \delta\sum_{k=1}^{\infty} a_k \\ &= \tfrac{1}{2} + \delta\sum_{k=1}^{\infty} a_k \end{aligned} \tag{22}$$

Thus, the expected score given any delay is less than or equal to the expected score given no delay.

This completes the proof that delay brings no additional utility to the attacker when their time to success pdf is monotonically decreasing. In general, however, the question still remains if delay has value in finite duration games. Delay may also be a viable and necessary choice in multi-asset games, where the attacker has a finite amount of resources to direct at a set of targets.

An example of a continuous distribution where delay is worthwhile

Consider the following (damped oscillator) function

$$f(x) = \frac{5\pi}{2\tau}\sin^2(\frac{\pi x}{\tau})e^{-\frac{\pi x}{\tau}}. \tag{23}$$

This is a non-negative function that integrates to 1 on the range $[0, \infty]$. We take advantage of the memorylessness of the exponential and the fact that for every k,

$$\int_{(k-1)\tau}^{k\tau}(\lceil\frac{x}{\tau}\rceil\tau - x)\sin^2(\frac{\pi x}{\tau})\,\mathrm{d}x = \frac{\tau^2}{4} \tag{24}$$

to come up with exact results for expected scores. Without any delay, the expected score for the attacker is

$$
\begin{aligned}
E[u(0)] &= \tfrac{5\pi}{2\tau}\int_0^{\infty}(\lceil\tfrac{x}{\tau}\rceil\tau - x)\sin^2(\tfrac{\pi x}{\tau})e^{-\frac{\pi x}{\tau}}\,\mathrm{d}x \\
&= \textstyle\sum_{k=1}^{\infty}\tfrac{5\pi}{2\tau}\int_{(k-1)\tau}^{k\tau}(\lceil\tfrac{x}{\tau}\rceil\tau - x)\sin^2(\tfrac{\pi x}{\tau})e^{-\frac{\pi x}{\tau}}\,\mathrm{d}x \\
&= \textstyle\sum_{k=1}^{\infty}\tfrac{\tau e^{-k\pi}(7+e^{\pi}(5\pi-7))}{5\pi} \\
&= \tfrac{\tau(7+e^{\pi}(5\pi-7))}{5\pi}\textstyle\sum_{k=1}^{\infty}e^{-k\pi} \\
&= \tfrac{\tau(7+e^{\pi}(5\pi-7))}{5\pi(e^{\pi}-1)} \\
&\approx 0.6\tau
\end{aligned}
\tag{25}
$$

Suppose we added a delay of $\frac{3\tau}{4}$. Then the expected score for the attacker is

$$
\begin{aligned}
E[u(\tfrac{\tau}{2})] &= \tfrac{5\pi}{2\tau}\int_{\frac{3\tau}{4}}^{\infty}(\lceil\tfrac{x}{\tau}\rceil\tau - x)\sin^2(\tfrac{\pi(x-\frac{3\tau}{4})}{\tau})e^{-\frac{\pi(x-\frac{3\tau}{4})}{\tau}}\,\mathrm{d}x \\
&= \textstyle\sum_{k=1}^{\infty}\tfrac{5\pi}{2\tau}\int_{(k-1)\tau}^{k\tau}(\lceil\tfrac{x}{\tau}\rceil\tau - x)\sin^2(\tfrac{\pi(x-\frac{3\tau}{4})}{\tau})e^{-\frac{\pi(x-\frac{3\tau}{4})}{\tau}}\,\mathrm{d}x \\
&\quad -\textstyle\int_0^{\frac{3\tau}{4}}(\lceil\tfrac{x}{\tau}\rceil\tau - x)\sin^2(\tfrac{\pi(x-\frac{3\tau}{4})}{\tau})e^{-\frac{\pi(x-\frac{3\tau}{4})}{\tau}}\,\mathrm{d}x \\
&= \textstyle\sum_{k=1}^{\infty}\tfrac{\tau e^{-k\pi+\frac{3\pi}{4}}(29+e^{\pi}(35\pi-29))}{20\pi} - \tau\tfrac{28-5*\pi+e^{\frac{3\pi}{4}}(35\pi-29)}{20\pi} \\
&= \tfrac{\tau e^{\frac{3\pi}{4}}(29+e^{\pi}(35\pi-29))}{20\pi}\textstyle\sum_{k=1}^{\infty}e^{-k\pi} - \tau\tfrac{28-5*\pi+e^{\frac{3\pi}{4}}(35\pi-29)}{20\pi} \\
&= \tfrac{\tau e^{\frac{3\pi}{4}}(29+e^{\pi}(35\pi-29))}{20\pi(e^{\pi}-1)} - \tau\tfrac{28-5*\pi+e^{\frac{3\pi}{4}}(35\pi-29)}{20\pi} \\
&\approx 0.64\tau \\
&> E[u(0)]
\end{aligned}
\tag{26}
$$

In this case, the attacker's time-to-success distribution is not monotonically decreasing and has local maxima for positive x. Unlike our earlier example of a situation where attacker delay has value which used the Dirac delta function, this distribution is continuous and therefore our earlier analysis shows it should be countered by a periodic defender strategy. This is another example of a time-to-success distribution where delaying an attack is worthwhile for the attacker. However, it remains to be shown whether such functions meaningfully represent any real-world time-to-success distributions.

3.1.3 Defender Strategy for Exponential Attacker Success Distribution

In this section we will analyze the use of the exponential distribution as the attacker's distribution for time-to-success. We will start with the attacker expected utility until the beginning of the next attack from Equation 6

$$
E[u] = -\alpha - \beta\int_0^{\infty} xf(x)\,\mathrm{d}x + \tau\sum_{k=1}^{\infty}k\int_{(k-1)\tau}^{k\tau}f(x)\,\mathrm{d}x - \sum_{k=1}^{\infty}\int_{(k-1)\tau}^{k\tau}xf(x)\,\mathrm{d}x,
$$

and replace the general distribution with the exponential distribution: $f(x) = \lambda e^{-\lambda x}$.

By substituting $f(x) = \lambda e^{-\lambda x}$ into the equation above, we obtain:

$$E[u] = -\alpha - \frac{1+\beta}{\lambda} + \tau \frac{e^{\lambda\tau}}{e^{\lambda\tau} - 1} \tag{27}$$

Therefore, the value of τ that pushes the attacker out of the game satisfies the following equation:

$$\tau = c(1 + e^{-\lambda\tau}), \tag{28}$$

where

$$c = \alpha + \frac{1+\beta}{\lambda} \tag{29}$$

By graphically analyzing both sides of the equation above, we can show that there is a unique value of τ that satisfies this equation as illustrated in the Figure 6.

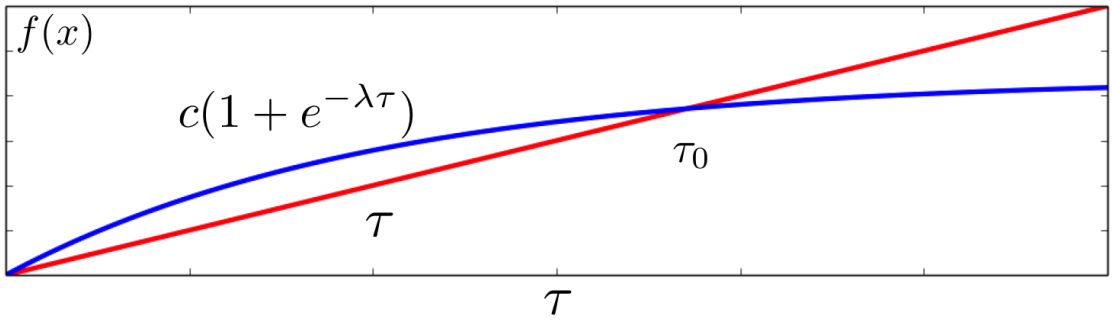

Figure 6: Both sides of Equation 28. Red - left side, blue - right side. This illustrates that there exists a single value of τ_0 that ensures expected utility of the attacker to equal to zero.

The Figure 6 shows that there are two points at which the left and the right side of Equation 28 intersect. The intersection at the origin is of no interest to us, and the intersection at positive value of τ represents the solution to the equation τ_0 where the expected attacker utility equals to zero.

Choice of the Exponential Distribution

In this section, we assume that the attacker's time-to-success distribution is limited to the exponential family of distributions. While this simplifies the derivation of attacker and defender utilities, we came to this choice based on another line of reasoning.

Conceptually, we model an attacker's activity as a series of attempts to compromise the contended resource. In this simple model, each attempt independently succeeds with a fixed

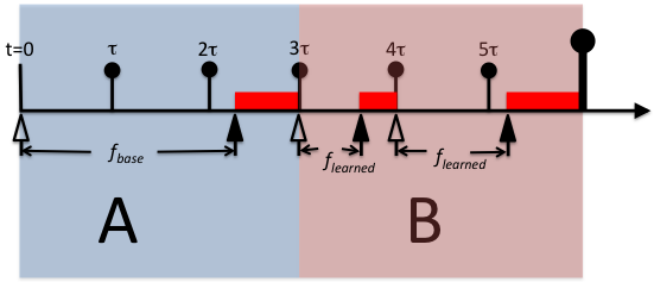

Figure 7: Illustration of the two regimes of the finite PLADD game between two morph moves. In regime A, the duration of the attack is governed by $f_{base}(s)$ and during regime B it is governed by $f_{learned}(s)$

probability p, hence, the attempts are Bernoulli trials. The number of Bernoulli trials until the first success is distributed according to the discrete geometric distribution with parameter p. The continuous analog of the geometric is the exponential. So, we choose to model the attacker's time-to-success as a continuous, exponential random variable which captures the notion of repeated, fixed probability-of-success attack attempts.

3.2 Finite Game

In the finite version of the game, the analysis must be modified to account for the morph move and the $f_{learned}(s)$ and $f_{base}(s)$ time-to-success distributions. Since the defender's morph move resets the attacker's time-to-success distribution back to $f_{base}(s)$ and no other learning occurs during the game, the finite version of the PLADD game can be analyzed by considering a single window between successive morph moves.

Next, the interval between morph moves can be analyzed by dividing it into two regimes, as illustrated in Figure 7. The first regime begins as soon as the defender makes a morph move. Assuming that the attacker does not delay the start of their next attack (see Section 3.1.2) they will start an attack at $t = 0$ using $f_{base}(s)$. This regime lasts until the attack succeeds and the defender makes their next move or until the next morph move if the attack has not succeeded by that point in time. Assuming that the attack succeeds and that the next defender move is not a morph move, there will be a regime with one or more intervals where the time-to-success distribution for attacks is $f_{learned}(s)$. In the figure, the first regime, labeled A, lasts for three intervals. In this example, the attack succeeds after the second take move and the attacker controls the resource until the next take move. The game then enters the second regime, labeled B, where $f_{learned}(s)$ applies, which lasts for three intervals. The attacker succeeds in two of these intervals before the game is reset by the morph move.

The following subsections describe the utility functions for the finite version of the PLADD game. The first subsection presents the general equations for computing the attacker and defender utility. The results in this section do not assume that periodic strategies are used by the defender nor do they make any assumptions about $f_{base}(s)$ or $f_{learned}(s)$. The second subsection describes the utility functions when the defender uses a periodic strategy, and $f_{base}(s)$ and $f_{learned}(s)$ are characterized by exponential distributions. The second subsection also compares the results for the infinite and finite cases when exponential distributions are

used.

3.2.1 General Expected Utility Equations

Here we wish to compute the expected attacker and defender utility for the general finite PLADD game. Let, $E_A[f_{base}, f_{learned}, \tau]$ and $E_D[f_{base}, f_{learned}, \tau]$ represent the expected utility for the attacker and defender between morph moves, respectively. The attacker's utility can also be represented by $E_A[f_{base}, f_{learned}, \tau; \alpha, \beta]$ to specify the attack costs. It is assumed that attacks always start as soon as the attacker loses control of the resource. Let τ be a sequence of $N+1$ times where the defender will make moves and $\tau_0 = 0$ corresponds to the start of the interval being analyzed. The value of τ_N corresponds to the morph move which ends the interval being analyzed. Let $E_L[f_{learned}, j, \tau]$ be the expected attacker utility following a successful attack in interval j. Following a successful attack only the $f_{learned}(s)$ distribution applies. Given this, the expected attacker utility is given by the equation below.

$$E_A[f_{base}, f_{learned}, \tau] = -\alpha - \beta\tau_N \int_{\tau_N}^{\infty} f_{base}(x)\,\mathrm{d}x + \sum_{j=1}^{N} \int_{\tau_{j-1}}^{\tau_j} f_{base}(x)(\tau_j - x - \beta x + E_L[f_{learned}, j, \tau])\,\mathrm{d}x \quad (30)$$

The first term in this equation represents the cost the attacker will pay for starting the attack at the beginning of the game. The second term represents the expected cost of the initial attacks that do not succeed before the morph move. In this case, there is no benefit for the attacker since they will not control the resource. The third term represents the utility to the attacker given that the initial attack succeeds before the next morph move. This term is similar to the equation used for the infinite version of the game with two exceptions. First, the summation in the finite game only goes to N versus ∞ in the infinite game since the the initial attack will be interrupted by the morph move (move N). Second, this term now includes $E_L[f_{learned}, j, \tau]$. This term states that the utility is equal to the utility of the initial attack plus the utility of all subsequent attacks given that the first attack succeeds in interval j.

The equation for calculating the utility for the second regime of the game can be derived by noting that given an attack is successful in interval j, the expected utility of that attack is independent of the history of the game. Given this, the equation below shows how $E_L[f_{learned}, j, \tau]$ can be calculated. This equation is essentially the same as the previous equation. There is a cost for starting the attack, a cost given that it does not succeed and an expected utility given that it does succeed in interval i. If the attack succeeds in interval i there will be a utility of $E_L[f_{learned}, i, \tau]$ for the subsequent attacks. This defines a recursive relationship for calculating the expected utility. This recursion ends when the end of the last interval (N) is reached. $E_L[f_{learned}, N, \tau]$ is defined to be zero, since the game is reset when the defender moves at interval N.

$$E_L[f_{learned}, j, \tau] = -\alpha - \beta(\tau_N - \tau_j) \int_{\tau_N}^{\infty} f_{learned}(x - \tau_j)\,\mathrm{d}x + \sum_{i=j+1}^{N} \int_{\tau_{i-1}}^{\tau_i} f_{learned}(x - \tau_j)(\tau_i - x - \beta(x - \tau_j) + E_L[f_{learned}, i, \tau])\,\mathrm{d}x \quad (31)$$

The equations for the defender's utility are related to the attacker's utility. Let $E_A[f_{base}, f_{learned}, \tau; \alpha, \beta]$ be the attacker's utility for a given α and β. If α and β are set to zero, $E_A[f_{base}, f_{learned}, \tau; 0, 0]$ is the expected total time that the attacker controls the resource between morph moves. Since the interval between morph moves is τ_N, the defender should expect to control the resource for $\tau_N - E_A[f_{base}, f_{learned}, \tau; 0, 0]$ units of time. During each interval the defender will have two types of costs. First, the defender will have to pay for a single morph move, which can be charged at the beginning or end of the interval. Second, the defender will have to pay for $N - 1$ take moves. Given this, the defender's utility is represented by the equation below.

$$E_D[f_{base}, f_{learned}, \tau] = \tau_N - E_A[f_{base}, f_{learned}, \tau; 0, 0] - c_{morph} - (N-1)c_{take} \quad (32)$$

3.2.2 Expected Utility Equations for Exponential Distribution

In this section, the expected utility equations for the attacker and defender are derived for the finite game when $f_{base}(s)$ and $f_{learned}(s)$ are assumed to be exponential distributions and a periodic strategy is used by the defender. These results are compared to the infinite version of the game using an exponential distribution and periodic defender strategy.

Utility of a Single Interval Using $f_{learned}(s)$

The probability density function of the exponential distribution is $f(x) = \lambda e^{-\lambda x}$. A key feature of this distribution is that it is memoryless. A random variable is memoryless if the probability that the variable exceeds a value s is equal to the probability that it exceeds a value of $s + t$ given that the value of the variable is greater than any t, as shown in the equation below. In the context of PLADD, this means that given an attack has not succeed at some point after it has begun, the distribution function of time until the attack succeeds is the same as when the attack was initially started.

$$P(x > s + t | x > t) = P(x > s), t > 0 \quad (33)$$

One challenging aspect of the general utility equations for the finite game is that they they are recursive. However, the memoryless nature of the exponential distribution can be used to avoid directly calculating this recursion. When the defender makes a take move, one of two events can occur. If the attacker controlled the resource, the attacker will start a new attack using $f_{learned}(s)$. If the attacker did not control the resource, their attack will continue.

However, the memoryless nature of the exponential distribution means that the distribution of the time remaining on the attack will be $f_{learned}(s)$, in either case. The distribution of the time-to-success following a take move is independent of the previous state of the resource. Given this, this version of the finite game can be analyzed by computing the expected utility of the interval between defender take moves and multiplying the results by the number of intervals between take moves.

Let $p_{learned}$ be a random variable that represents the utility for a single attack that uses $f_{learned}(s) = \lambda_\ell e^{-\lambda_\ell s}$. The expected value of $p_{learned}$ is given by the equations below. The expected utility can be calculated by dividing the expectation into two cases, one case where the attack is successful and one where it is not successful. It is successful if it succeeds before τ. The probability of this event can be found by integrating $f_{learned}(s)$ from zero to τ. If the attack succeeds the attacker will have to pay the ongoing attack costs until the attack succeeds. They will also have to pay to start a new attack after the defender moves next. The benefit they will gain will be the time they control the resource until the next move. The attack will fail when the duration of the attack is greater than τ. The probability of this event can be found by integrating $f_{learned}(s)$ from τ to infinity. In this case, there is no benefit to the attacker and they must pay the ongoing attack costs over the entire interval τ.

$$\begin{aligned} E[p_{learned}] &= P(Attack\ success)E[p_{learned}|Attack\ success]+ \\ &\qquad P(Attack\ failure)E[p_{learned}|Attack\ failure] \\ &= \left(\int_0^\tau \lambda_\ell e^{-\lambda_\ell x}\,\mathrm{d}x\right)\left(-\alpha + \int_0^\tau \frac{\lambda_\ell e^{-\lambda_\ell x}}{\int_0^\tau \lambda_\ell e^{-\lambda_\ell x'}\,\mathrm{d}x'}(\tau - x - \beta x)\,\mathrm{d}x\right) \\ &\qquad + \left(\int_\tau^\infty \lambda_\ell e^{-\lambda_\ell x}\,\mathrm{d}x\right)(-\beta\tau) \end{aligned} \tag{34}$$

The equation below shows the expected attacker utility when the integrals in the equation above are calculated and the result is simplified. This equation represents expected attacker utility between take moves when $f_{learned}(s)$ applies.

$$E[p_{learned}] = \tau - \left(\alpha + \frac{1+\beta}{\lambda_\ell}\right)(1 - e^{-\lambda_\ell \tau}) \tag{35}$$

Infinite Exponential Game Revisited

Before completing the analysis of the finite exponential it is useful to revisit the infinite game for the exponential case, using the results for $E[p_{learned}]$. Since the equation above gives the expected utility between moves when $f_{learned}(s)$ is in effect, it can be divided by τ to produce an equation for the utility rate of the attacker (U_{att}^{inf}). This will allow for comparisons of PLADD games that use different values for τ. In the case where there are no morph moves (i.e. the infinite version of the game), this equation describes the utility rate of the attacker. It is useful to compare this result to the equation derived for the infinite version of the game using the exponential distribution. Equation 27, in the infinite game section of this report, provides the expected utility per attacker move, whereas this equation describes the expected

utility between take moves. Despite this difference both equations should have the same root. Setting the equation below to zero and solving for τ results in the solution shown in Equation 28.

$$U_{att}^{inf} = 1 - \left(\alpha + \frac{1+\beta}{\lambda_\ell}\right)\left(\frac{1-e^{-\lambda_\ell \tau}}{\tau}\right) \tag{36}$$

The equation above can be used to determine the defender's utility rate for the infinite game (U_{def}^{inf}). Setting α and β to zero in the equation above yields the expected fraction of time that the attacker controls the resource. This result can be subtracted from 1.0 to determine the expected fraction of time that the defender controls the resource. Finally, the rate that the defender incurs costs must be subtracted. The defender incurs a cost of c_{take} every τ units of time. The defender's utility rate for the infinite game is given by the equation below.

$$U_{def}^{inf} = \frac{1-e^{-\lambda_\ell \tau}}{\lambda_\ell \tau} - \frac{c_{take}}{\tau} \tag{37}$$

Utility Equations for the Finite Exponential Game

Next we use the expected attacker utility between moves when $f_{learned}(s)$ applies to calculate the expected utility between morph moves for the finite game. For clarity, we divide this analysis into two sections. First, we consider the expected utility for the attacker when there are no fixed costs for moving ($\alpha = 0$). Then, we consider the expected fixed costs between morph moves. The second result can be subtracted from the first result to determine the total utility for the attacker when all costs are considered in the finite game.

When the fixed attack costs are ignored, the equation below can be used to calculate the utility for the regime of the game where $f_{learned}(s)$ applies. j is the interval where the attack using $f_{base}(s)$ succeeds. If the first attack succeeds in interval j there will be $N - j$ intervals where the attacker attacks using the learned distribution. Therefore the utility in this interval can be calculated by multiplying the per-period utility by the number of periods.

$$E_L[f_{learned}, j, \tau] = \left(\tau - \frac{1+\beta}{\lambda_\ell}(1-e^{-\lambda_\ell \tau})\right)(N-j) \tag{38}$$

This result can be substituted into the general equation for the attacker utility (Equation 30), as show in the equation below. In this equation, α has been set to zero, the τ values have been updated to reflect the defender's periodic strategy, and $f_{base}(s)$ and $f_{learned}(s)$ have been replaced with the exponential distribution.

$$\begin{aligned} E_A[f_{base}, f_{learned}, \tau] = -\beta\tau N \int_{\tau N}^{\infty} \lambda_b e^{-\lambda_b x}\,\mathrm{d}x \\ + \sum_{j=1}^{N} \int_{\tau(j-1)}^{\tau j} \lambda_b e^{-\lambda_b x}(\tau j - x - \beta x + E_L[f_{learned}, j, \tau])\,\mathrm{d}x \end{aligned} \tag{39}$$

The equation above can be simplified by calculating the integrals and summations. This results in the equation below.

$$E_A[f_{base}, f_{learned}, \tau] = \tau N - \frac{(1+\beta)N}{\lambda_\ell}(1 - e^{-\lambda_\ell \tau}) + \frac{(1+\beta)}{\lambda_b}(1 - e^{-\lambda_b \tau N})\left(\frac{\lambda_b(1 - e^{-\lambda_\ell \tau})}{\lambda_\ell(1 - e^{-\lambda_b \tau})} - 1\right) \quad (40)$$

The expected fixed costs for attacks can be calculated using the equation below. The first term represents the cost that will always be paid by the attacker following a morph move. The second term represents the subsequent attack costs that will be paid once the game enters the the $f_{learned}(s)$ regime. The integral is the likelihood that the initial attack succeeds in interval j. If the initial attack using $f_{base}(s)$ succeeds in interval j, a new attack will start in interval $j+1$ and an additional fixed cost will be incurred. After interval $j+1$, there will be $N-j-1$ intervals where additional fixed costs could be incurred. Since the exponential distribution is memoryless, the probability of incurring a fixed cost in each of these remaining intervals (i.e. the probability of attack success in the previous interval) is independently and identically distributed. This implies that the fixed attack costs in this interval follow a binomial distribution, where the probability of success is $1 - e^{-\lambda_\ell \tau}$. This probably is calculated by integrating $f_{learned}(s)$ between zero and τ. Multiplying this probability by $N-j-1$ results in the expected number of fixed attack costs between intervals $j+2$ and N. Note that if an attack succeeds in the last interval (N) an additional fixed attack cost will not be generated. This is due to the fact that the next move will be a morph move that will reset the game and that the cost of the attack that starts immediately after the morph move is already accounted for.

$$E[\textit{Fixed attack costs}] = \alpha + \sum_{j=1}^{N-1}\left(\int_{\tau(j-1)}^{\tau j} \lambda_b e^{-\lambda_b x}\,\mathrm{d}x\right)\left(\alpha + \alpha(N-j-1)(1 - e^{-\lambda_\ell \tau})\right) \quad (41)$$

The equation above can be simplified by calculating the integrals and summations. This results in the equation below.

$$E[\textit{Fixed attack costs}] = \alpha\left(1 + (N-1)(1 - e^{-\lambda_\ell \tau}) + (1 - e^{-\lambda_b \tau(N-1)})\left(1 - \frac{1 - e^{-\lambda_\ell \tau}}{1 - e^{-\lambda_b \tau}}\right)\right) \quad (42)$$

Subtracting this result from the finite case where α was set to zero yields the expected attacker utility, given by the equation below.

$$E_A[f_{base}, f_{learned}, \tau] = \tau N - \frac{(1+\beta)N}{\lambda_\ell}(1 - e^{-\lambda_\ell \tau}) + \frac{(1+\beta)}{\lambda_b}(1 - e^{-\lambda_b \tau N})\left(\frac{\lambda_b(1 - e^{-\lambda_\ell \tau})}{\lambda_\ell(1 - e^{-\lambda_b \tau})} - 1\right) - \alpha\left(1 + (N-1)(1 - e^{-\lambda_\ell \tau}) + (1 - e^{-\lambda_b \tau(N-1)})\left(1 - \frac{1 - e^{-\lambda_\ell \tau}}{1 - e^{-\lambda_b \tau}}\right)\right) \quad (43)$$

The defender's utility function can be calculated from the attacker's utility function. The attacker costs, α and β, can be set to zero to calculate the expected time the attacker controls the resource. This result can be subtracted from τN to determine the expected time the defender controls the resource between morphs. Finally, the morph and take costs can be subtracted to determine the defender's utility function.

$$E_D[f_{base}, f_{learned}, \tau] = \frac{N(1 - e^{-\lambda_\ell \tau})}{\lambda_\ell} - \frac{(1 - e^{-\lambda_b \tau N})}{\lambda_b}\left(\frac{\lambda_b(1 - e^{-\lambda_\ell \tau})}{\lambda_\ell(1 - e^{-\lambda_b \tau})} - 1\right) - c_{morph} - (N-1)c_{take} \quad (44)$$

Comparison of Infinite and Finite PLADD Game Equations

The utility equations for the finite and infinite versions of the game can be compared to understand the impact of the morph move. In the case of the infinite game, the utility equations were divided by τ to determine the cost rate. Since the utility equations for the finite version of the game correspond to the utility between morph moves, these equations can be divided by $N\tau$ to obtain the utility rate for the finite game (U_{att}^{fin}). This allows these two versions of the game to be compared.

The equations below show the attacker's utility rate equations for the infinite and finite game. The terms in the equations have been arranged so that more direct comparisons can be made. The equation for the finite version of the game contains the equation for the infinite version of the game and two additional terms. The term that is subtracted from the infinite utility rate represents the impact of the morph move on the fraction of time the attacker controls the resource and spends attacking. If it is assumed that $\lambda_b/\lambda_\ell \leq 1$, this term will always decrease the attacker's utility. This is reasonable since the morph move should increase the duration of attacks. This term implies that the addition of a morph move will increase the time the attacker spends attacking and reduce the time they control the resource. The term that is added represents the change in fixed attack costs when the morph move is added. Under the previous assumption, this term will always increase the utility rate. When the morph move is added, the attacker will succeed less often, therefore they will start fewer attacks (they will also control the resource less and spend more time attacking). Whether or not these additional terms produce a net increase or decrease in the utility function compared to the infinite game depends on the values of the parameters.

$$U_{att}^{inf} = 1 - \left(\alpha + \frac{1+\beta}{\lambda_\ell}\right)\left(\frac{1-e^{-\lambda_\ell \tau}}{\tau}\right) \tag{45}$$

$$\begin{aligned} U_{att}^{fin} = 1 &- \left(\alpha + \frac{1+\beta}{\lambda_\ell}\right)\left(\frac{1-e^{-\lambda_\ell \tau}}{\tau}\right) - \\ &(1+\beta)\frac{1-e^{-\lambda_b N\tau}}{\lambda_b N\tau}\left(1 - \frac{\lambda_b(1-e^{-\lambda_\ell \tau})}{\lambda_\ell(1-e^{-\lambda_b \tau})}\right) + \\ &\frac{\alpha}{N\tau}\left(e^{-\lambda_\ell \tau} + (1-e^{-\lambda_b(N-1)\tau})\left(\frac{1-e^{-\lambda_\ell \tau}}{1-e^{-\lambda_b \tau}} - 1\right)\right) \end{aligned} \tag{46}$$

The next set of equations shows the defender's utility rate equations for the infinite and finite game. Again, the equation for the finite game contains two additional terms. The term that is added corresponds to the additional time that the defender controls the resource when the morph move is added. This term is always positive given the previous assumption about λ_b and λ_ℓ. The term that is subtracted represents the change in utility related to the cost of the morph move. If it is assumed that $c_{morph} \geq c_{take}$, this term will always decrease the defender's utility. This assumption is reasonable since the morph move will generally cost more than the take move. If the morph move was more powerful than the take move and cost less, the defender would never choose to make a take move. Again, the net change in the utility rate for the finite game compared to the infinite game depends on the values of the parameters.

$$U_{def}^{inf} = \frac{1-e^{-\lambda_\ell \tau}}{\lambda_\ell \tau} - \frac{c_{take}}{\tau} \tag{47}$$

$$\begin{aligned} U_{def}^{fin} = &\frac{1-e^{-\lambda_\ell \tau}}{\lambda_\ell \tau} - \frac{c_{take}}{\tau} \\ &+ \frac{(1-e^{-\lambda_b N\tau})}{\lambda_b N\tau}\left(1 - \frac{\lambda_b(1-e^{-\lambda_\ell \tau})}{\lambda_\ell(1-e^{-\lambda_b \tau})}\right) - \\ &\frac{c_{morph} - c_{take}}{N\tau} \end{aligned} \tag{48}$$

3.2.3 Verification via Simulation

To help verify our analytic solutions for attacker and defender utility in PLADD with exponential attacker time-to-success, we developed a simple discrete event simulation of the game. In the simulation, the defender uses a periodic strategy whose rate is given as a parameter. The attacker and defender are subject to move costs and time-to-success distributions, also given as parameters. The simulation yields attacker and defender utility rates. Figures 8 and 9 show two example plots of the attacker and defender utility rates in the infinite (no morph) and finite (morph available) games for a given set of parameters. We compared our analytic results to the simulation for many values of the game parameters with essentially

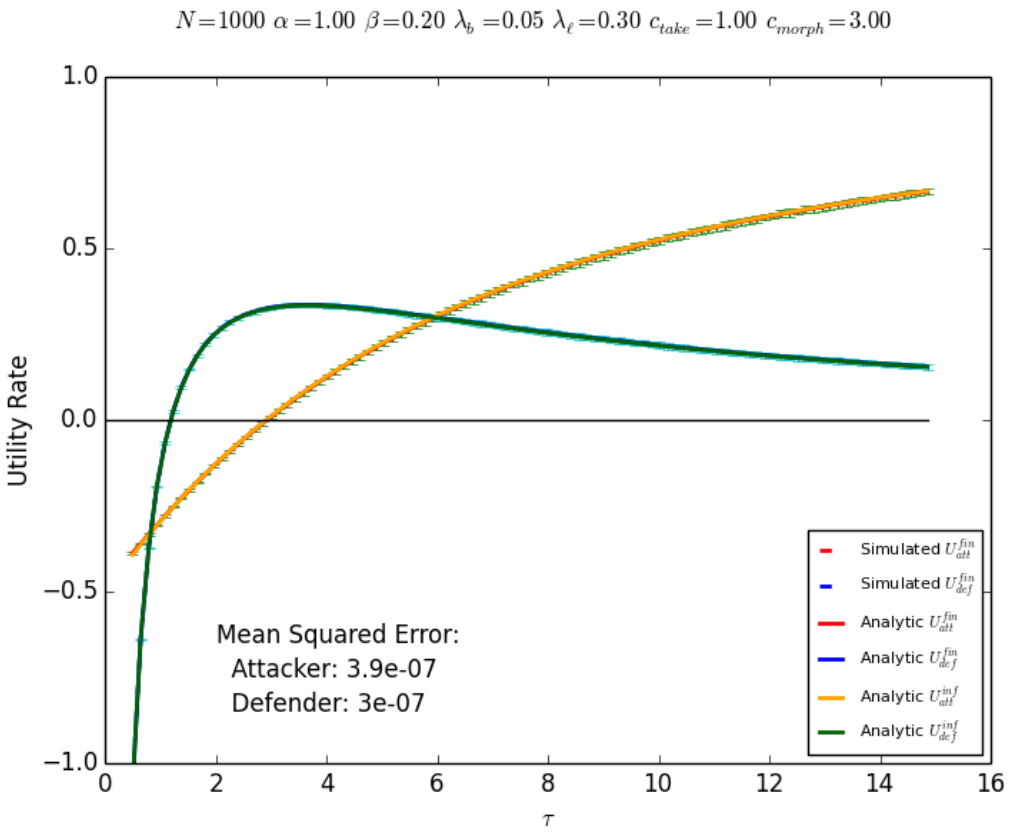

Figure 8: Example simulation-based verification plot for the *infinite* game, i.e., without morphs.

identical results. The values produced by our analytic solutions and those resulting from simulation match extremely closely for games with relatively long duration. The figures show games of length 1000 time units. For games with shorter durations, the match is still good statistically, but the variance between runs is higher. Comparing the results from our equations and the simulator helped us identify both derivation mistakes and simulator bugs. The results ultimately increased our confidence in the correctness of the analytic game solutions.

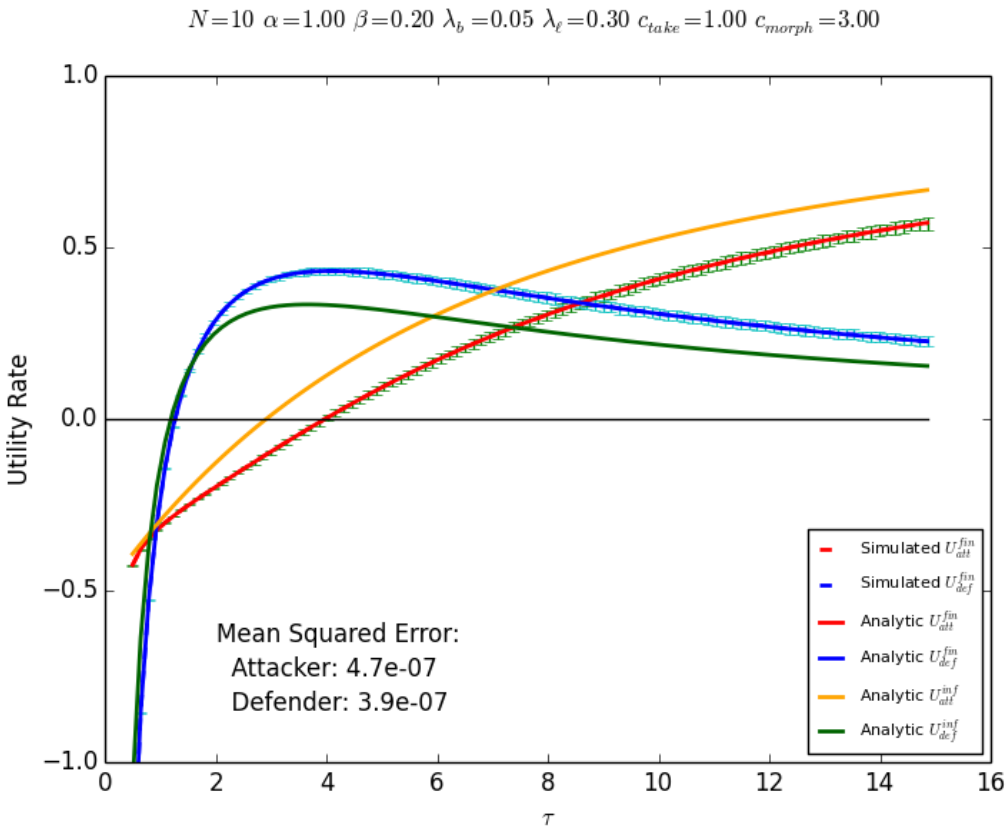

Figure 9: Example simulation-based verification plot for the *finite* game, i.e., including morph moves.

3.3 Value of MTD in PLADD

We are now in a position to attempt to answer our original question, namely, "how much value does MTD provide to a defender?". We cannot yet answer that question in general (See Section 3.4 for the initial derivation of more general defender utility equations). We can, however, try to answer it for a specific definition of value within the version of PLADD where defenders use a periodic strategy and attackers are subject to an exponentially distributed time-to-success. To proceed, we will select a realistic defender goal and compare the cost of achieving that goal in the infinite PLADD game, where no morph move and, hence, no MTD is available, vs. the cost required to achieve the same goal in the finite game where morph moves are available. We believe that this unconventional solution concept tells us more about our game and the value of MTD within it than more traditional solution concepts like the locations of Nash or other equilibria.

3.3.1 Defender Goal and Metric

The defender goal we have chosen to use in our MTD analysis is pushing a rational attacker out of the game. This is typically the ultimate goal of any defender. Other equilibrium-based metrics often leave defenders in a strange loss equilibrium state that no serious system owner would willingly accept. A defender pushes an attacker out of the game by making it more expensive than it is worth to continue. We have a proof, described in Section 3.1.1, that demonstrates it will always be possible for the defender to play fast enough to force the attacker out of a PLADD game. We name the defender strategy that results in the attacker dropping out of the game "deterrence". The metric by which we will judge MTD within PLADD is the degree to which having MTD available in an otherwise identical game makes achieving deterrence less expensive for the defender.

Our evaluation methodology is as follows:

1. Using equation 45, find the optimal (least cost) periodic deterrence strategy (P_{inf}^{opt}) for the infinite game given the remainder of the parameters
2. Determine defender cost C_{inf}^{det} for the strategy P_{inf}^{opt}
3. Using equation 46, find the optimal (least cost) periodic deterrence strategy (P_{fin}^{opt}) for the finite game, which may include morph moves, given the same parameters
4. Determine defender cost C_{fin}^{det} for the strategy P_{fin}^{opt}
5. Determine the value of MTD for that parameter combination ($V = C_{inf}^{det} - C_{fin}^{det}$)
6. Plot V against a low-dimensional subset of the PLADD parameters, choosing parameters that have the largest impact on the outcome of V

Our utility equations cannot be solved directly for P_*^{opt}, the defender period that makes the attacker's utility zero, so we solve implicitly for P_*^{opt} numerically in both the infinite and finite game cases.

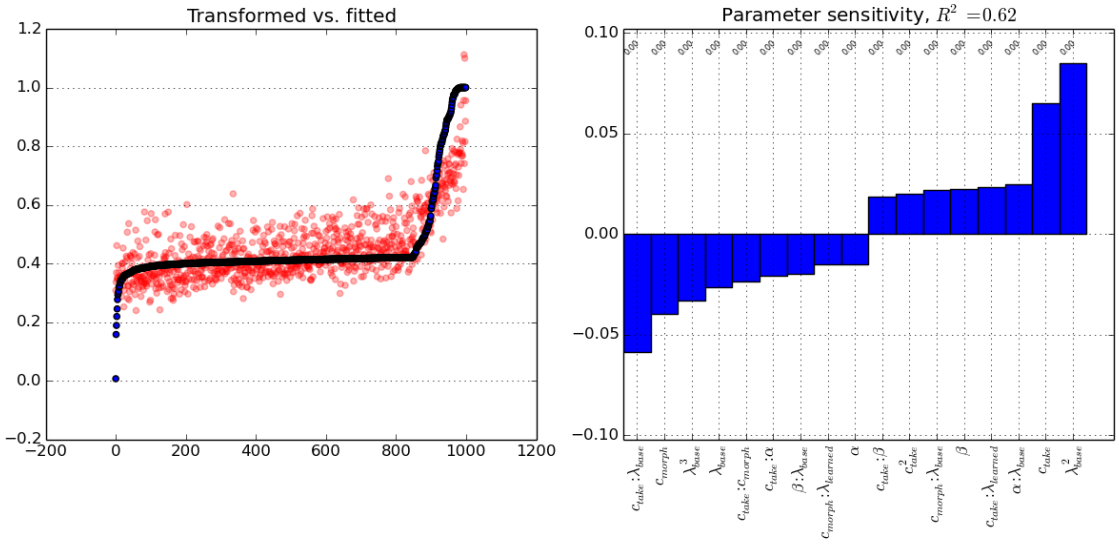

Figure 10: Sensitivity analysis of PLADD model parameters vs. V, the value of MTD.

Parameter name	Minimum value	Maximum value
τ	0.1	100
N	1	100
c_{take}	0.1	30.0
c_{morph}	0.1	30.0
α	0.1	30.0
β	0.1	30.0
λ_{base}	0.01	10.0
$\lambda_{learned}$	0.01	10.0

Table 1: Parameter ranges used in sensitivity analysis.

3.3.2 Choosing Important Parameters

To keep the number of dimensions reasonable for plotting, we have chosen a small subset of the model parameters against which to plot V using a simple linear regression-based sensitivity analysis. In the analysis, we:

1. Define numeric ranges for all PLADD parameters (ranges shown in Table 1)
2. Sample 300 points randomly across those ranges
3. For each sample, use the solution methodology described above to compute V
4. Center the parameter values so that regression coefficients can be compared
5. Fit a linear model of the parameters, two-way parameter interactions, and quadratic parameters to $Probit^{-1}(V)$
6. Identify the independent variables with the largest influence on V by comparing model coefficients β_i

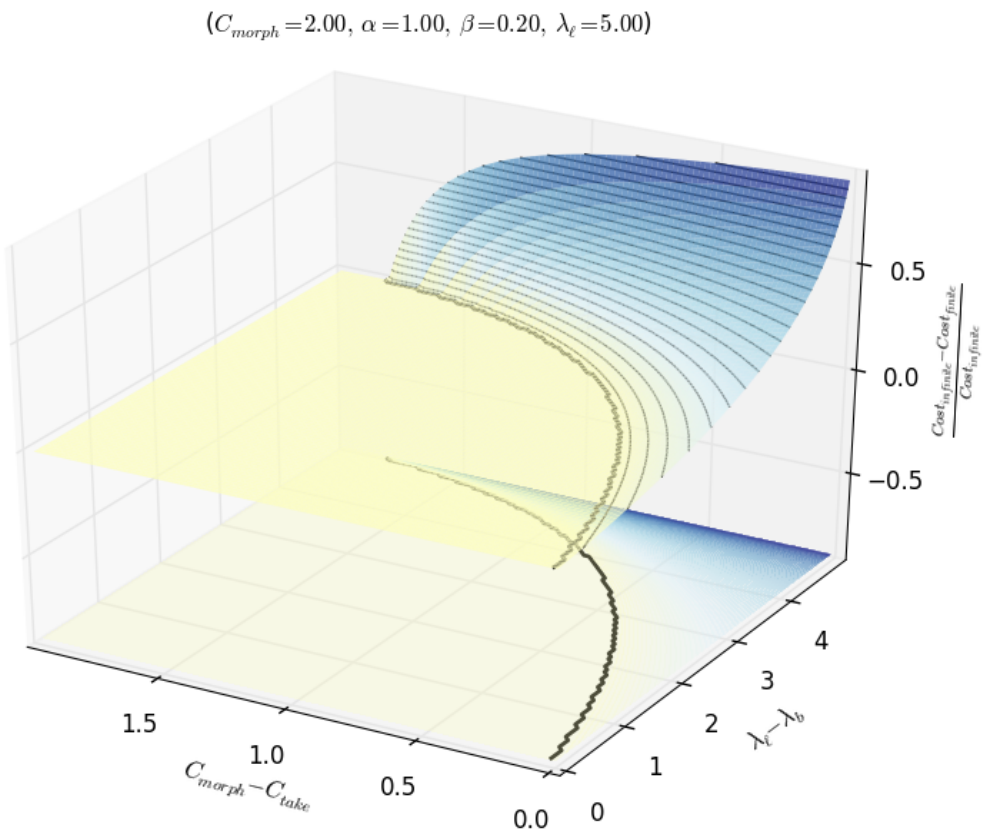

Figure 11: Cost benefit of MTD vs. relative defender cost and relative attacker time-to-success

Figure 10 shows the results of the analysis of the model parameters. The exercise indicates that the parameters which have the largest impact on V are the defender's costs (C_{morph} and C_{take}) and the rate of the attacker's time-to-success distributions (λ_{base} and $\lambda_{learned}$). The fact that those parameters have a larger influence on the value of MTD to the defender matches our intuition.

3.3.3 Results

Figure 11 shows how the value of having MTD available to a defender changes as the relative defender cost on the X axis ($C_{morph} - C_{take}$) and the relative attacker time-to-success rate on the Y axis ($\lambda_{learned} - \lambda_{base}$) change for fixed remaining parameters as shown in the graph title.

The large area of the graph where value of MTD is at or near zero shows immediately that, at least in PLADD, there are many configurations where MTD provides no additional value and, hence, shouldn't be used by a defender even if available. The areas where MTD demonstrates good value are intuitive, namely when MTD is very cheap (i.e., $C_{morph} - C_{take}$ is small) and when the MTD move has a large impact (i.e., $\lambda_{learned} - \lambda_{base}$ is large). Once some combination of the chosen parameters passes a threshold (the bold black line in the

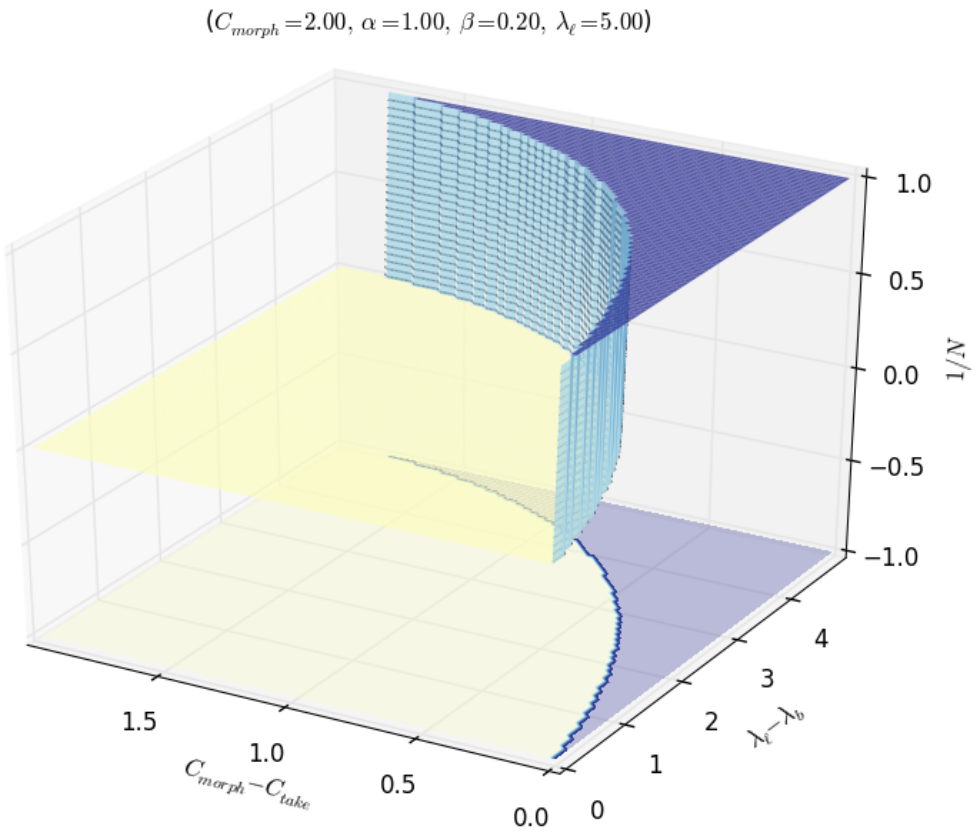

Figure 12: Optimal use of the MTD (morph) move by a defender.

graph), the value of MTD grows rapidly. We do not yet know the exact combination of parameters that determines the location of the transition.

Another interesting feature of MTD in PLADD is that, for many combinations of parameters, the transition from a regime of not using MTD at all (because it provides no value) to a regime where the defender uses MTD as aggressively as possible is quite abrupt with little or no intermediate stage where MTD moves are mixed with non-MTD moves more evenly. Figure 12 shows the sharp divide between the region where MTD is not used at all ($1/N$ is near zero) to the region where MTD is used as aggressively as possible ($1/N$ is near one) for the same parameter combination as shown in Figure 11. This feature of the graph is dependent on the attacker's costs. When attacks are dramatically cheaper than defender moves, the shape of the surface changes and the sharp distinction between regions softens as shown in Figure 13.

3.4 Exact Solutions for the Finite PLADD Game

In this section we present an initial derivation of a general solution for the defender utility in the finite PLADD game. In this scenario, the defender is not constrained to periodic moves and the attacker's time-to-success is governed by an unspecific probability distribution. In this case, the defender places $n-1$ take moves between morphs. The target scenario we

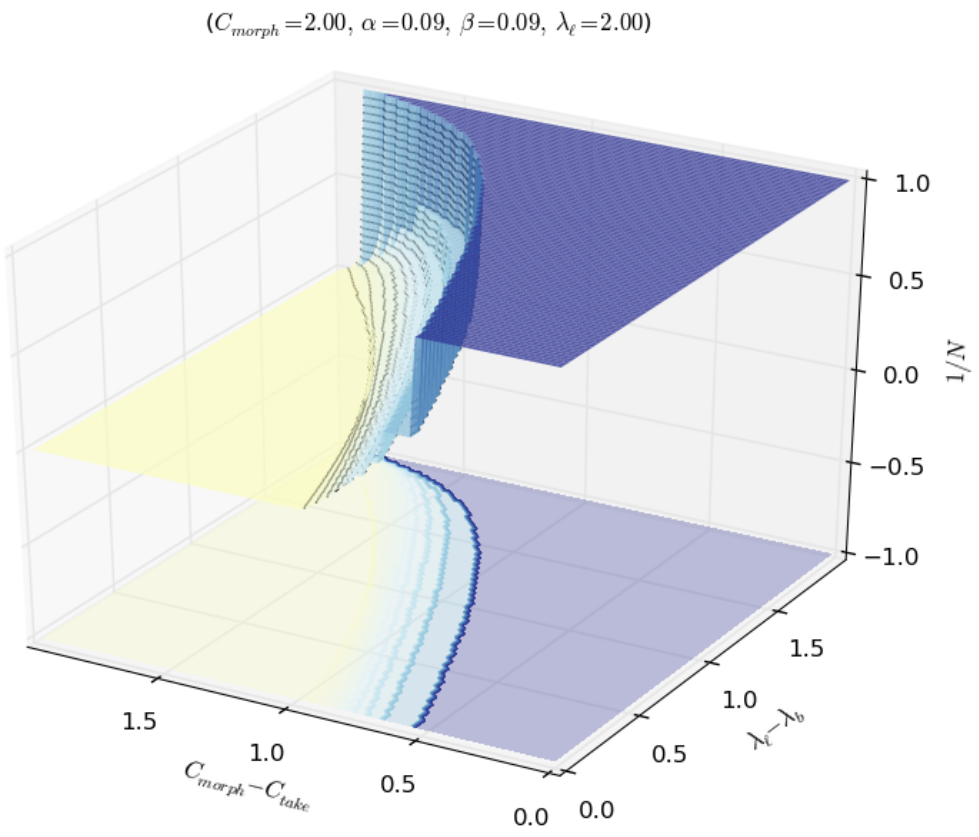

Figure 13: More gradual shift between "no use" and "use aggressively" regions when attacks are cheap.

solve for is when the defender acts to drive an attacker out of the game; a point we have called deterrence. Subsequently, we use the general equations to derive and briefly explore a solution for the cases when the attacker's time-to-success is governed by the exponential and the uniform distributions. We were unable to completely explore the implications of these new equations prior to project completion, so that remains future work.

We will start with a general solution for the attacker's utility. Since we are interested in the deterrence game setting, we can take the attacker's utility equation and set it to 0 and use this result to solve for the optimal set of defender take moves, $\{\tau_k\}$ where $k \in [1, n-1]$. Let us start by returning to the recursive definition of the attacker's utility. In the set of equations below, the first one defines the recursive step for all intervals after the attacker's first successful attack, which occurs before τ_j. The second equation defines the end of the recursion at interval τ_n, the defender's next morph, and the third equation, handles the intervals up to and including the attacker's first successful attack after the defender's last morph at τ_0. Note, as before, we segment the time between defender morph moves, which occur at τ_0 and τ_n, into intervals of defender take moves, which occur starting at τ_1 and continuing through τ_{n-1}.

$$
\begin{aligned}
E_L[f,j,\tau_j] &= \max\Bigg(0, -\alpha - \beta(L-\tau_j)\textstyle\int_{\tau_n}^{\infty} f(x-\tau_j)\,\mathrm{d}x \\
&\quad + \textstyle\sum_{i=j+1}^{n}\int_{\tau_{i-1}}^{\tau_i} f(x-\tau_j)(\tau_i - x - \beta(x-\tau_j) \\
&\quad + E_L[f,i,\tau_i])\,\mathrm{d}x\Bigg) \\
E_L[f,n,\tau_n] &= 0 \\
E_A[f_{base}, f_{learned}, \tau_0] &= -\alpha - \beta L\textstyle\int_{\tau_n}^{\infty} f_{base}(x)\,\mathrm{d}x \\
&\quad + \textstyle\sum_{k=1}^{n}\int_{\tau_{k-1}}^{\tau_k} f_{base}(x)(\tau_k - x - \beta x \\
&\quad + E_L[f_{learned}, k, \tau_k])\,\mathrm{d}x
\end{aligned}
$$

,

In the deterrence game context, the defender will manipulate the frequency of his periodic game play in order to assure that the attacker's utility will be zero. One way that the defender can accomplish this goal is to choose each τ_j such that $E_L[f_{learned}, j, \tau_j] = 0$ for all j. This choice eliminates the recursion, which simplifies the analysis. These are also points for which a rational attacker would choose not to continue to play the game.

$$
\begin{aligned}
E_L[f,j,\tau_j] = -\alpha - \beta(\tau_n - \tau_j)\int_{\tau_n}^{\infty} f(x-\tau_j)\,\mathrm{d}x \\
+ \sum_{i=j+1}^{n}\int_{\tau_{i-1}}^{\tau_i} f(x-\tau_j)(\tau_i - x - \beta(x-\tau_j))\,\mathrm{d}x = 0
\end{aligned}
$$

At this point we will reorganize the analysis by reversing our perspective and proceeding from the end of the last interval at τ_n and move backward toward τ_1. We will also introduce a more compact notation for the expected utility in each interval as

$$E_k = E_L[f_{learned}, n-k, \tau_{n-k}].$$

Here $k \in [1, n-1]$. We will further streamline our notation by introducing $\Delta_k = \tau_n - \tau_{n-k}$ and $\Delta_{i,k} = \tau_{n-i} - \tau_{n-k} = \Delta_k - \Delta_i$, which gives us the following equation for E_k

$$
\begin{aligned}
E_k &= -\alpha - \beta\Delta_k\textstyle\int_{\Delta_k}^{\infty} f(x)\,\mathrm{d}x + \sum_{i=0}^{k-1}\int_{\Delta_{i+1,k}}^{\Delta_{i,k}} f(x)(\Delta_{i,k} - x(1+\beta))\,\mathrm{d}x \\
&= -\alpha - \beta\Delta_k(1 - \textstyle\int_0^{\Delta_k} f(x)\,\mathrm{d}x) + \sum_{i=0}^{k-1}\Bigg(\Delta_{i,k}\textstyle\int_{\Delta_{i+1,k}}^{\Delta_{i,k}} f(x)\,\mathrm{d}x \\
&\quad -(1+\beta)\textstyle\int_{\Delta_{i+1,k}}^{\Delta_{i,k}} x f(x)\,\mathrm{d}x\Bigg) \\
&= -\alpha - \beta\Delta_k(1 - \textstyle\int_0^{\Delta_k} f(x)\,\mathrm{d}x) - (1+\beta)\textstyle\int_0^{\Delta_k} x f(x)\,\mathrm{d}x \\
&\quad + \textstyle\sum_{i=0}^{k-1}\Delta_{i,k}\int_{\Delta_{i+1,k}}^{\Delta_{i,k}} f(x)\,\mathrm{d}x
\end{aligned}
$$

Note that $\int_{\Delta_{i+1,k}}^{\Delta_{i,k}}$ represents an integral of the span of the i^{th} take interval, counting up from the last interval before the next morph, shifted so that τ_{n-k} is the 0 point.

In the deterrence game setting, the defender wants to ensure that $E_k = 0$ for each k, which allows us to rearrange the equation above as follows.

$$\begin{aligned} \alpha + \beta\Delta_k + (1+\beta)\int_0^{\Delta_k} x f(x)\,\mathrm{d}x &= (1+\beta)\Delta_k \int_0^{\Delta_k} f(x)\,\mathrm{d}x \\ &\quad - \sum_{i=1}^{k-1} \Delta_i \int_{\Delta_{i+1,k}}^{\Delta_{i,k}} f(x)\,\mathrm{d}x \end{aligned} \tag{49}$$

The equation above holds for all $k \in [1, n-1]$ and can be used with a given attacker time-to-success distribution (f) to determine the optimal set of $\{\tau_k\}$ for the defender. Note that the sum on the right hand side has an interesting interpretation. Each term in the sum can be rewritten as

$$(\tau_n - \tau_{n-i}) \int_{\tau_{n-(i+1)}-\tau_{n-k}}^{\tau_{n-i}-\tau_{n-k}} f(x)\,\mathrm{d}x$$

This can be interpreted as the distance between the end of a "take" interval at τ_{n-i} and the end of the morph cycle at τ_n multiplied by the fraction of the probability of the attacker's time-to-success within the "take" interval.

When the system of equations (Equation 49 above) is solved for a given k, one more step is required to figure out the exact numerical value of τ_1. (The above system will allow Δ_k for $k \in [1, n-1]$ to be determined). In order to solve for τ_1, the formula for E_A must be solved and set to 0. That is,

$$\alpha + \beta\tau_n \int_{\tau_n}^{\infty} f_{base}(x)\,\mathrm{d}x + (1+\beta)\int_0^{\tau_n} x f_{base}(x)\,\mathrm{d}x = \sum_{k=1}^{n} \tau_k \int_{\tau_{k-1}}^{\tau_k} f_{base}(x)\,\mathrm{d}x$$

Here we will rearrange this equation to better match equations presented previously (see Equation 49),

$$\alpha + \beta\tau_n + (1+\beta)\int_0^{\tau_n} x f_{base}(x)\,\mathrm{d}x = \beta\tau_n \int_0^{\tau_n} f_{base}(x)\,\mathrm{d}x + \sum_{k=1}^{n} \tau_k \int_{\tau_{k-1}}^{\tau_k} f_{base}(x)\,\mathrm{d}x$$

We will now use the Δ notation and simplification as before to make these equations even more similar.

$$\begin{aligned} \alpha + \beta\Delta_n + (1+\beta)\int_0^{\Delta_n} x f_{base}(x)\,\mathrm{d}x &= (1+\beta)\Delta_n \int_0^{\Delta_n} f_{base}(x)\,\mathrm{d}x \\ &\quad - \sum_{k=1}^{n-1} \Delta_{n-k} \int_{\tau_{k-1}}^{\tau_k} f_{base}(x)\,\mathrm{d}x \\ &= (1+\beta)\Delta_n \int_0^{\Delta_n} f_{base}(x)\,\mathrm{d}x \\ &\quad - \sum_{i=1}^{n-1} \Delta_i \int_{\tau_{n-(i+1)}-\tau_0}^{\tau_{n-i}-\tau_0} f_{base}(x)\,\mathrm{d}x \\ &= (1+\beta)\Delta_n \int_0^{\Delta_n} f_{base}(x)\,\mathrm{d}x \\ &\quad - \sum_{i=1}^{n-1} \Delta_i \int_{\Delta_{i+1,n}}^{\Delta_{i,n}} f_{base}(x)\,\mathrm{d}x \end{aligned} \tag{50}$$

We now present Equation 49 with $f_{learned}$ replacing f.

$$\begin{aligned}\alpha + \beta\Delta_k + (1+\beta)\int_0^{\Delta_k} x f_{learned}(x)\,\mathrm{d}x &= (1+\beta)\Delta_k \int_0^{\Delta_k} f_{learned}(x)\,\mathrm{d}x \\ &\quad - \sum_{i=1}^{k-1}\Delta_i \int_{\Delta_{i+1,k}}^{\Delta_{i,k}} f_{learned}(x)\,\mathrm{d}x\end{aligned}$$

The form of these two sets of equations (Equation 49 and Equation 50) are exactly the same, except for the attacker's time-to-success distribution function (f_{base} versus $f_{learned}$) we are optimizing with and n versus k in the Δ notation. These can be solved to determine the exact sequence of $\{\tau_k\}$, given $n-1$ take moves before a morph. Furthermore, we can let the expected score for the defender given these equations be E_n^*, after which we can compute the optimal n to maximize the defender expected score, given the defender is using the deterrence strategy to drive the attacker out of the game.

We solve for the optimal expected score E_n^*, and thus the optimal n, using Equation 49 and Equation 50 by successively incrementing from the value $n = 1$. At some point extra moves will provide diminishing returns. At $n = 1$, we only use Equation 50 since $k < n$. For all $n > 1$, we use Equation 49 for $k \in [1, n-1]$ and Equation 50 for n.

In the next subsection we will use Equation 49 and Equation 50 to determine the optimal defender set of take times ($\{\tau_k\}$) for the exponential attacker time-to-success distributions.

3.4.1 Exponential Solution

In this section we will estimate optimal move times for the defender given an exponential time-to-success distribution for the attacker.

Since the attacker's time-to-success distribution is exponential, we will assume that $f_{base}(x) = \lambda_b e^{-\lambda_b x}$ and that $f_{learned}(x) = \lambda_l e^{-\lambda_l x}$. We can compute the optimal set $\{\tau_k\}$ as follows.

$$\begin{aligned}\alpha + \beta\Delta_k + (1+\beta)\int_0^{\Delta_k} x\lambda_l e^{-\lambda_l x}\,\mathrm{d}x = (1+\beta)\Delta_k \int_0^{\Delta_k} \lambda_l e^{-\lambda_l x}\,\mathrm{d}x \\ - \sum_{i=1}^{k-1}\Delta_i \int_{\Delta_{i+1,k}}^{\Delta_{i,k}} \lambda_l e^{-\lambda_l x}\,\mathrm{d}x\end{aligned}$$

After handling the integrals we get

$$\begin{aligned}\alpha + \beta\Delta_k + (1+\beta)\frac{1 - e^{-\Delta_k\lambda_l}(1+\Delta_k\lambda_l)}{\lambda_l} = (1+\beta)\Delta_k(1 - e^{-\Delta_k\lambda_l}) \\ - \sum_{i=1}^{k-1}\Delta_i(e^{-\Delta_{i+1,k}\lambda_l} - e^{-\Delta_{i,k}\lambda_l})\end{aligned}$$

Here we simply replaced $f_{learned}$ with $\lambda_l e^{-\lambda_l x}$ in Equation 49. Given this equation we will solve for Δ_1 (i.e., for $k = 1$),

$$\alpha + \beta\Delta_1 + (1+\beta)\frac{1 - e^{-\Delta_1\lambda_l}(1 + \Delta_1\lambda_l)}{\lambda_l} = (1+\beta)\Delta_1(1 - e^{-\Delta_1\lambda_l})$$

$$\Delta_1 = (1+\beta)\frac{1 - e^{-\Delta_1\lambda_l}}{\lambda_l} + \alpha$$

$$\Delta_1\lambda_l = (1+\beta)(1 - e^{-\Delta_1\lambda_l}) + \alpha\lambda_l$$

$$\Delta_1 = \frac{W(-(1+\beta)e^{-(1+\beta+\alpha\lambda_l)}) + 1 + \beta + \alpha\lambda_l}{\lambda_l}$$

In the last step, W is positive branch of the Lambert W function, or product log function. That is, $W(x)$ makes the following true: $x = W(x)e^{W(x)}$. Now we will return the general case to solve for Δ_k.

$$\alpha + \beta\Delta_k + (1+\beta)\frac{1 - e^{-\Delta_k\lambda_l}(1 + \Delta_k\lambda_l)}{\lambda_l} = (1+\beta)\Delta_k(1 - e^{-\Delta_k\lambda_l}) - \frac{\sum_{i=1}^{k-1}\Delta_i(e^{\Delta_{i+1}\lambda_l} - e^{\Delta_i\lambda_l})}{e^{\Delta_k\lambda_l}}$$

which can be re-arranged and simplified as

$$\alpha + \beta\Delta_k + (1+\beta)\frac{1 - e^{-\Delta_k\lambda_l} - \lambda_l\Delta_k}{\lambda_l} + \frac{\sum_{i=1}^{k-1}\Delta_i(e^{\Delta_{i+1}\lambda_l} - e^{\Delta_i\lambda_l})}{e^{\Delta_k\lambda_l}} = 0$$

We can separate the first element from the rest of the summation and re-arrange as

$$\alpha + \beta\Delta_k + \Delta_{k-1} + (1+\beta)\frac{1 - e^{-\Delta_k\lambda_l} - \lambda_l\Delta_k}{\lambda_l} + \frac{-\Delta_1 e^{\Delta_1\lambda_l} + \sum_{i=2}^{k-1}(\Delta_{i-1} - \Delta_i)e^{\Delta_i\lambda_l}}{e^{\Delta_k\lambda_l}} = 0$$

Since we have previously defined that $\Delta_0 = 0$ we get

$$\alpha + \beta\Delta_k + \Delta_{k-1} + (1+\beta)\frac{1 - e^{-\Delta_k\lambda_l} - \lambda_l\Delta_k}{\lambda_l} + \frac{\sum_{i=1}^{k-1}(\Delta_{i-1} - \Delta_i)e^{\Delta_i\lambda_l}}{e^{\Delta_k\lambda_l}} = 0$$

Now let

$$y(\lambda, j) = -(1 + \beta + \lambda(-\sum_{i=1}^{j-1}(\Delta_{i-1} - \Delta_i)e^{\Delta_i\lambda}))e^{-(1+\beta+\lambda(\alpha+\Delta_{j-1}))}$$

Then we can apply the Lambert W function as before to solve for Δ_k

$$\Delta_k = \Delta_{k-1} + \alpha + \frac{W(y(\lambda_l, k)) + 1 + \beta}{\lambda_l}$$

Note that

$$\Delta_n = \Delta_{n-1} + \alpha + \frac{W(y(\lambda_b, n)) + 1 + \beta}{\lambda_b}$$

and so for $n = 1$,

$$\Delta_n = \frac{W(-(1+\beta)e^{-(1+\beta+\alpha\lambda_b)}) + 1 + \beta + \alpha\lambda_b}{\lambda_b}$$

Since the cost for a take move is c_{take} and the cost for a morph move is c_{morph}, then

$$E_n^* = 1 - \frac{c_{morph} + (n-1) * c_{take}}{\Delta_n}$$

The best n and therefore Δ_n depend on the parameters of the problem.

Suppose we have the following:

$$\begin{aligned} c_t &= 1.0 \\ c_m &= 3.0 \\ \alpha &= 1.0 \\ \beta &= 0.2 \\ \lambda_b &= 0.01 \\ \lambda_l &= 0.1 \end{aligned}$$

In this case, it is more worthwhile for the defender to only have morph moves, spaced every $\Delta_1 \approx 42.7201$ apart, for an expected score of

$$E_1^* \approx 0.92978$$

Suppose instead we have the following:

$$\begin{aligned} c_t &= 1.0 \\ c_m &= 3.0 \\ \alpha &= 1.0 \\ \beta &= 0.2 \\ \lambda_b &= 0.05 \\ \lambda_l &= 0.1 \end{aligned}$$

In this case, it is more worthwhile for the defender to always take and never morph, with moves spaced approximately 5.49861 apart, for an expected score of

$$\lim_{k\to\infty} E_k^* \approx 0.81814$$

In performing a parameter search, it appears that there is no parameterization in which mixing take and morph moves outperforms either only taking or only morphing. The difference between this result and our earlier simulation results appears to be due to our allowing take moves to be spaced non-periodically. However, where the expected score for only taking is very close to the expected score for only morphing, it may be worthwhile to mix moves such that the defender expected score is roughly the same, but the attacker can't reliably predict the upcoming move sequence of the defender.

3.4.2 Uniform Solution

In this section we will compute the optimal defender period in the case where the attacker's time-to-success distribution is uniform. Here we assume that f_{base} is a uniform distribution with probability p_{base} and that $f_{learned}$ is a uniform distribution with probability $p_{learned}$, and furthermore that $p_{base} \leq \frac{1}{\tau_n}$ and $p_{learned} \leq \frac{1}{\tau_n - \tau_1}$, then we can compute the optimal τ as follows.

$$\begin{aligned} \alpha + \beta\Delta_k + (1+\beta)p_{learned}\textstyle\int_0^{\Delta_k} x \,\mathrm{d}x &= (1+\beta)p_{learned}\Delta_k \textstyle\int_0^{\Delta_k} 1 \,\mathrm{d}x \\ &\quad - p_{learned}\textstyle\sum_{i=1}^{k-1}\Delta_i \int_{\Delta_{i+1,k}}^{\Delta_{i,k}} 1 \,\mathrm{d}x \end{aligned}$$

Thus,

$$\alpha + \beta\Delta_k = (1+\beta)p_{learned}\frac{\Delta_k^2}{2} - p_{learned}\sum_{i=1}^{k-1}\Delta_i(\Delta_{i+1} - \Delta_i)$$

This is merely a quadratic equation in Δ_k, given the Δ_i values for $i < k$.

$$\frac{(1+\beta)p_{learned}}{2}\Delta_k^2 + (-\beta - p_{learned}\Delta_{k-1})\Delta_k + (-\alpha + p_{learned}(\sum_{i=1}^{k-1}\Delta_i^2 - \sum_{i=1}^{k-2}\Delta_i\Delta_{i+1})) = 0$$

Thus,

$$\Delta_k = \frac{(\beta+p_{learned}\Delta_{k-1})}{(1+\beta)p_{learned}} + \frac{\sqrt{(\beta+p_{learned}\Delta_{k-1})^2+2(\alpha-p_{learned}(\sum_{i=1}^{k-1}\Delta_i^2-\sum_{i=1}^{k-2}(\Delta_{i+1}\Delta_i)))(1+\beta)p_{learned}}}{(1+\beta)p_{learned}}$$

This is the solution for Δ_k where $k < n$. Δ_n is computed by using the Δ_k solutions for all but $k = n$, and for Δ_n solve the above equation with $p_{learned}$ replaced by p_{base}.

$$\Delta_n = \frac{(\beta+p_{base}\Delta_{n-1})}{(1+\beta)p_{base}} + \frac{\sqrt{(\beta+p_{base}\Delta_{n-1})^2+2(\alpha-p_{base}(\sum_{i=1}^{n-1}\Delta_i^2-\sum_{i=1}^{n-2}(\Delta_{i+1}\Delta_i)))(1+\beta)p_{base}}}{(1+\beta)p_{base}}$$

If the cost for a take move is c_{take} and the cost for a morph move is c_{morph}, then

$$E_n^* = 1 - \frac{c_{morph} + (n-1) * c_{take}}{\Delta_n}$$

The best n and therefore Δ_n depend on the parameters of the problem.

We can make a few additional statements about the nature of Δ_k as k grows. Notice that

$$\lim_{\Delta_{k-1}\to\frac{1}{p_{learned}}} \Delta_k = \frac{1}{p_{learned}} + O(\frac{1}{(1+\beta)\sqrt{p_{learned}}})$$

Here we are doing some hand waving; the α determines some of the behavior and the O bound is very approximate. However, empirically it appears that the equations are forcing all $\Delta_k \leq \frac{1}{p_{learned}}$, which means our initial constraint on $p_{learned}$ will always be met. Likewise for p_{base}. Our unconstrained uniform solution (see next subsection) provides some math to show this happening when $p_{learned}\Delta_k \leq 1$.

Suppose we have the following:

$$\begin{aligned} c_{take} &= 1.0 \\ c_{morph} &= 3.0 \\ \alpha &= 1.0 \\ \beta &= 0.2 \\ p_{base} &= 0.01 \\ p_{learned} &= 0.1 \end{aligned}$$

In this case, the relative cost of a take to a morph is small compared to the relative likelihood of attacker time-to-success in base versus learned, so the best expected score comes where $n = 1$, where there are no take moves. In this case,

$$\Delta_1 = \frac{0.2 + \sqrt{0.04 + 0.024}}{0.012} \approx 37.7485$$

yielding

$$E_1^* \approx 0.9205$$

To get more interesting behavior, we can make the base time-to-success likelihood closer to the learned time-to-success likelihood. Consider the following:

$$\begin{aligned} c_{take} &= 1.0 \\ c_{morph} &= 3.0 \\ \alpha &= 1.0 \\ \beta &= 0.2 \\ p_{base} &= 0.05 \\ p_{learned} &= 0.1 \end{aligned}$$

This time, $n = 2$ yields the highest benefit, and

$$\Delta_1 = \frac{0.2 + \sqrt{0.04 + 0.24}}{0.12} \approx 6.0763$$

$$\Delta_2 = \frac{0.2+0.05*\Delta_1+\sqrt{(0.2+0.05*\Delta_1)^2+2*(1-0.05*\Delta_1^2)*1.2*0.05}}{1.2*0.05} \approx 14.9012$$

yielding

$$E_2^* \approx 0.7315$$

We speculate that more interesting behaviors and better defender solutions will be found if we resolve these equations removing the constraints on how high p_{base} and $p_{learned}$ can be.

Unconstrained Uniform Solution

The defender will want to space their moves as far apart as possible, so they must consider moves that guarantee attacker success before a morph. Unfortunately, the more take moves a defender considers, the more the equations determining their decisions may differ, depending on the magnitude of p_l.

E_1 and Δ_1

Let $f_{learned} = f$ and $\int_0^{\frac{1}{p}} f(x)\,\mathrm{d}x = 1$. Then when $p\Delta_1 \geq 1$,

$$\begin{aligned} E_1 &= -\alpha + \textstyle\int_0^{1/p} f(x)(\Delta_1 - x(1+\beta))\,\mathrm{d}x \\ &= -\alpha + \Delta_1 - (1+\beta)\textstyle\int_0^{1/p} xf(x)\,\mathrm{d}x \\ &= 0 \end{aligned}$$

Thus,

$$\Delta_1 = \alpha + (1+\beta)\int_0^{1/p} xf(x)\,\mathrm{d}x$$

When $f(x) = p_l$, then

$$\Delta_1 = \alpha + \frac{(1+\beta)}{2p_l}$$

$p\Delta_1 \geq 1$ implies the constraint

$$\alpha p_l + \frac{(1+\beta)}{2} \geq 1$$

$$p_l \geq \frac{1-\beta}{2\alpha}$$

The constraints for our two versions align perfectly. Thus, we have

$$\begin{aligned} \Delta_1 &= \tfrac{\beta+\sqrt{\beta^2+2\alpha(1+\beta)p_l}}{(1+\beta)p_l} \quad ; \quad p_l \leq \tfrac{1-\beta}{2\alpha} \\ \Delta_1 &= \alpha + \tfrac{(1+\beta)}{2p_l} \quad ; \quad p_l \geq \tfrac{1-\beta}{2\alpha} \end{aligned}$$

E_2 and Δ_2

We can extend the last section's result to a second "take" move. The new cases to consider are where $\Delta_2 - \Delta_1 \leq \frac{1}{p} \leq \Delta_2$ and where $\frac{1}{p} \leq \Delta_2 - \Delta_1$.

Case $\frac{1}{p} \leq \Delta_2 - \Delta_1$

This is where the attacker is guaranteed success sometime after the penultimate take move and before the final take move.

$$E_2 = -\alpha + \int_0^{\frac{1}{p}} f(x)(\Delta_2 - \Delta_1 - x(1+\beta))\,\mathrm{d}x = 0$$

This simplifies to

$$\Delta_2 = \alpha + \Delta_1 + (1+\beta)\int_0^{\frac{1}{p}} xf(x)\,\mathrm{d}x$$

From here, if $p\Delta_1 \geq 1$, we get

$$\Delta_2 = 2\Delta_1$$

We can here generalize.

$$\Delta_k = k\left(\alpha + (1+\beta)\int_0^{1/p} xf\left(x\right)\,\mathrm{d}x\right) \quad ; \quad p\Delta_1 \geq 1$$

For $f(x) = p_l$,

$$\Delta_k = k(\alpha + \tfrac{(1+\beta)}{2p_l}) \quad ; \quad p_l \geq \tfrac{1-\beta}{2\alpha}$$

On the other hand, if $p\Delta_1 \leq 1$, we again will go straight to the case $f(x) = p_l$, so that $\Delta_1 = \frac{\beta+\sqrt{\beta^2+2\alpha(1+\beta)p_l}}{(1+\beta)p_l}$ and $p_l \leq \frac{1-\beta}{2\alpha}$. We have

$$\Delta_2 = \alpha + \Delta_1 + \frac{1+\beta}{2p_l}$$

Consider when this formula is valid.

$$\frac{1}{p_l} \leq \Delta_2 - \Delta_1 = \alpha + \frac{1+\beta}{2p_l} \rightarrow 2 \leq 2p_l\alpha + 1 + \beta \rightarrow p_l \geq \frac{1-\beta}{2\alpha}$$

In other words, this is a case that won't happen. The implication here is that if we can space two take moves apart far enough to guarantee success for an attacker yet also an expected utility of 0 for the attacker, we will space all take moves that same distance apart.

Case $\Delta_2 - \Delta_1 \leq \frac{1}{p} \leq \Delta_2$

This is where, beginning at the penultimate take move, the attacker is guaranteed success before the next morph move, but not necessarily before the next take move. It implies $\Delta_2 - \Delta_1 \leq \Delta_1$.

$$\begin{aligned} E_2 &= -\alpha + \int_0^{\Delta_2-\Delta_1} f(x)(\Delta_2 - \Delta_1 - x(1+\beta))\,\mathrm{d}x \\ &\quad + \int_{\Delta_2-\Delta_1}^{\frac{1}{p}} f(x)(\Delta_2 - x(1+\beta))\,\mathrm{d}x \\ &= 0 \end{aligned}$$

This simplifies to

$$\Delta_2 = \alpha + \Delta_1 \int_0^{\Delta_2-\Delta_1} f(x)\,\mathrm{d}x + (1+\beta)\int_0^{\frac{1}{p}} xf(x)\,\mathrm{d}x$$

From here, if $p\Delta_1 \geq 1$ (meaning attack success was guaranteed within the final take period), we get

$$\Delta_2 = \Delta_1 \left(1 + \int_0^{\Delta_2-\Delta_1} f\left(x\right)\,\mathrm{d}x\right)$$

This implies $\Delta_2 \leq 2\Delta_1$. The Δ_2 value derives no advantage from the constraint $\Delta_2 - \Delta_1 \leq \Delta_1$, and rather we should let $\Delta_2 = 2\Delta_1$ as outlined in the previous case section.

So we conclude that $p\Delta_1 \leq 1$. We have no simple formula for Δ_1; rather, we have the following.

$$\alpha + \beta\Delta_1 + (1+\beta)\int_0^{\Delta_1} xf(x)\,\mathrm{d}x = (1+\beta)\Delta_1 \int_0^{\Delta_1} f(x)\,\mathrm{d}x$$

We move ahead to the case where $f(x) = p_l$, so that $\Delta_1 = \frac{\beta+\sqrt{\beta^2+2\alpha(1+\beta)p_l}}{(1+\beta)p_l}$ and $p_l \leq \frac{1-\beta}{2\alpha}$.

$$\Delta_2 = \alpha + p_l\Delta_1(\Delta_2 - \Delta_1) + \frac{(1+\beta)}{2p_l}$$

$$\Delta_2 = \frac{2p_l(\alpha - p_l\Delta_1^2) + (1+\beta)}{2p_l(1-p_l\Delta_1)}$$

$$\Delta_2 = \frac{(1-\beta)\sqrt{\beta^2+2\alpha(1+\beta)p_l} + 1 + 3\beta}{2(1+\beta)p_l}$$

The constraint $p_l(\Delta_2 - \Delta_1) \leq 1$ implies our old constraint, $p_l \leq \frac{1-\beta}{2\alpha}$. The constraint $p_l\Delta_2 \geq 1$ lets us derive a new constraint.

$$(1-\beta)\sqrt{\beta^2+2\alpha(1+\beta)p_l} + 1 + 3\beta \geq 2(1+\beta)$$

$$(1-\beta)(\sqrt{\beta^2+2\alpha(1+\beta)p_l}-1)\geq 0$$

$$p_l \geq \frac{1-\beta}{2\alpha}$$

We are forced to conclude, therefore, that the circumstance $\Delta_2 - \Delta_1 \leq \frac{1}{p_l} \leq \Delta_2$ doesn't occur for uniformly distributed attackers, and therefore that when $p\Delta_1 \leq 1$, it follows that $p\Delta_2 \leq 1$. We believe that it also follows that $p\Delta_k \leq 1$ for all $k < n$.

Summary

We have a general solution for when the opponent costs are high enough that we can allow success between every take and still give the attacker an expected score of 0.

$$\Delta_k = k\left(\alpha + (1+\beta)\int_0^{1/p} x f(x)\,\mathrm{d}x\right) \quad ; \quad p\Delta_1 \geq 1$$

We have solution methods and partial solutions for when relative costs versus success likelihood are lower. The following is for uniform probability attack distributions $f(x) = p_l$

$$\Delta_k = \frac{(\beta+p_l\Delta_{k-1})+\sqrt{(\beta+p_l\Delta_{k-1})^2+2(\alpha-p_l(\sum_{i=1}^{k-1}\Delta_i^2-\sum_{i=1}^{k-2}(\Delta_{i+1}\Delta_i)))(1+\beta)p_l}}{(1+\beta)p_l};$$

$$\Delta_k \leq 1$$

A surprising consequence here is that either defender take moves will not extend past $\Delta_k < \frac{1}{p_l}$, or else they will be spaced evenly apart and $\Delta_k \geq \frac{1}{p_l}$, whenever the defender wishes to drive out the attacker with an expected score of 0. There is no space of interesting behaviors in between.

4 Stochastic Programming Approach

Optimization arises frequently when analyzing game theoretic models. First, the players are frequently modeled as rational actors who seek to maximize their benefit subject to their knowledge of the opponent and system and their own particular objective function. Second, we can use optimization approaches to directly identify the best outcome a player can achieve subject to the game and the behavior of their opponent. It is this second class of problems that we will investigate here. In general, numerical optimization approaches attempt analysis "between" that of (exhaustive) simulation and closed form analytic models. In simulation, we select specific player parameterizations and then simulate the game (typically several times) to approximate the expected outcome (benefit) for each player. Determining the best outcome for a player requires enumerating over the range of possible combinations of player

parameterizations. In contrast, analytic approaches seek to develop closed-form expressions that explicitly capture a feature of interest like a player's best outcome or the point where a player will drop out of the game. Closed-form analytic approaches can provide insights and prescriptive solutions, but usually under very restrictive assumptions and at the cost of significant research. In this effort, our goal was to explore techniques based on stochastic programming as an alternative paradigm that fills the gap between exhaustive simulation and closed-form analytic approaches.

4.1 A Stochastic Programming Model for the FlipIt Game

The key feature of the FlipIt game is the stealthy aspect of both players' moves. As neither player can directly or indirectly observe the actions of the other, the game is separable: we can treat the attacker as exogenous by sampling from their space of allowable moves. We will then optimize the defender's move times in response to a finite set of attacker moves. In the context of stochastic programming, each time series of attacker moves forms an *attack scenario*. We want to identify the defender's optimal set of *non-anticipative* moves that best guard against the entire set of attack scenarios. We generate non-anticipative solutions by forcing the defender to select a single set of moves for all attack scenarios. This corresponds to a two-stage stochastic problem where the defender's moves are the (non-anticipative) first stage variables. Then, in the second stage, the opponent's moves are revealed and the scenario-specific utility is calculated. For the trivial case of a single scenario, the player will respond perfectly to the opponent's moves. However, as we add scenarios, the player is forced to *hedge* against a larger set of possible outcomes. This has the effect of enforcing the inability of the real defender to observe or respond to the attacker's moves.

An important feature of this model is that the optimization algorithm will always place defender moves at the same time as an attacker move in one of the scenarios. This can be shown by considering a defender move placed between two attack time points (not necessarily in the same scenario): the defender can always unilaterally improve their utility by shifting that move up in time to the first attack time in any scenario. This allows us to form a discrete time model where the defender can only move in a finite set of discrete time points (the union of all attack times in all scenarios). This model exactly reproduces the optimal solution for a model where the defender is allowed to move at any time (i.e., no discretization error), and significantly reduces the computational complexity of the model.

Mathematically, the FlipIt model is given by the following stochastic Generalized Disjunctive Program (GDP):

$$\max \quad |T|^{-1}|S|^{-1} \sum_{s\in S, t\in T} \rho_{s,t} - c_{take} \sum_{t\in T} d_t \tag{51}$$

$$s.t. \quad \begin{bmatrix} Y_{1,s,t} \\ a_{s,t} = 0 \\ d_t = 0 \\ \rho_{s,t} = \rho_{s,t-1} \end{bmatrix} \bigvee \begin{bmatrix} Y_{2,s,t} \\ a_{s,t} = 1 \\ d_t = 0 \\ \rho_{s,t} = 0 \end{bmatrix} \bigvee \begin{bmatrix} Y_{3,s,t} \\ \\ d_t = 1 \\ \rho_{s,t} = 1 \end{bmatrix}$$

$$\forall s \in S, \{t | t \in T, t > 0\} \tag{52}$$

$$Y_{1,s,t} + Y_{2,s,t} + Y_{3,s,t} = 1 \qquad \forall s \in S, \{t | t \in T, t > 0\} \tag{53}$$

$$\rho_{s,0} = 1 \qquad \forall s \in S \tag{54}$$

$$d_t \in \{0, 1\} \qquad \forall t \in T \tag{55}$$

$$\rho_{s,t} \in \{0, 1\} \qquad \forall s \in S, t \in T \tag{56}$$

$$Y_{i,s,t} \in \{0, 1\} \qquad \forall i \in \{1, 2, 3\}, s \in S, t \in T \tag{57}$$

Here, T is the set of discrete time points where either an attacker move or defender move is allowed to occur, S is the set of attack scenarios, $a_{s,t}$ is a binary parameter that is 1 if and only if the attacker moves at time t in scenario s, d_t is a binary variable that is one if and only if the defender moves (in all scenarios) at time t, and $\rho_{s,t}$ is a binary variable that is 1 if and only if the defender controls the resource at time t in scenario s. $Y_{i,s,t}$ is a Boolean (binary) variable that indicates the disjunct i in the disjunction at time t on scenario s is enforced. The resource state, ρ, is set by the disjunction (52), and the constraint (53) forces exactly one of the three transitions stated in (52) to be active at any time in any scenario. As written, the objective (51) assumes uniformly spaced discrete time points, which is the case we investigate here. This assumption can be relaxed by scaling the individual $\rho_{s,t}$ terms by the time between time point t and the next time point and removing the $|T|^{-1}$ prefactor.

An important aspect of this model is that we are not forcing the defender to move with any particular strategy: the strategy can be inferred from the final selected move times. Similarly, the attacker is not forced into a single strategy: we are free to use any scheme to generate the attack scenarios. It was very encouraging that, given a series of attacker scenarios sampled from an attacker moving periodically with a fixed rate, the optimal defender moves were indeed periodic, as predicted by the analytic results in [28].

It is also important to note that this model approximated the original FlipIt game in two ways: first is the approximation due to a finite number of attacker scenarios. The second is that the model solves for a fixed, finite time horizon, H. Obviously, as H and the number of scenarios go to infinity the model will recover the exact results of the original FlipIt game. We investigated this by varying the number of scenarios and the game horizon for a specific parameterization of the player costs. Figure 14 shows optimal defender move rates over a range of attacker move rates for differing horizons and number of scenarios. This shows that even at very low numbers of scenarios and short horizons we can get solutions very close to expected value for the infinite-time game.

We can now look at the effect the ratio of the defender's move cost to the benefit obtained for possessing control of the resource has. Figure 15 shows that as the defender's move cost decreases relative to the benefit of control, the optimal defender move rate increases for

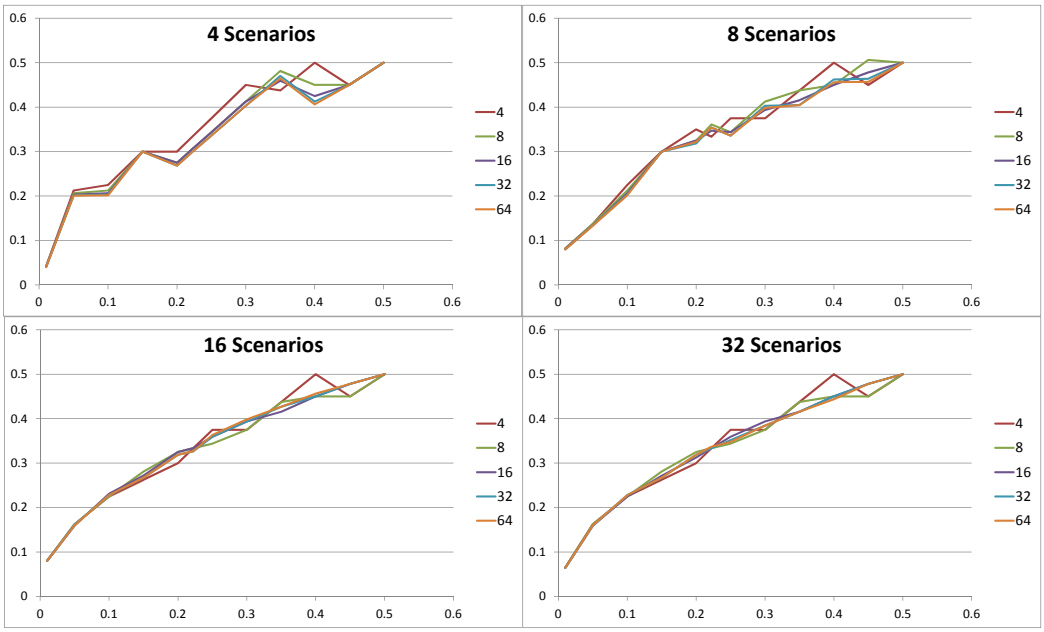

Figure 14: Effect of finite horizon and limited attack scenarios on the optimal defender move rate. Each trace shows the optimal defender move rate (Y-axis) for a given attacker move rate (X-axis) for a different horizon (4, 8, 16, 32, and 64 attacker moves).

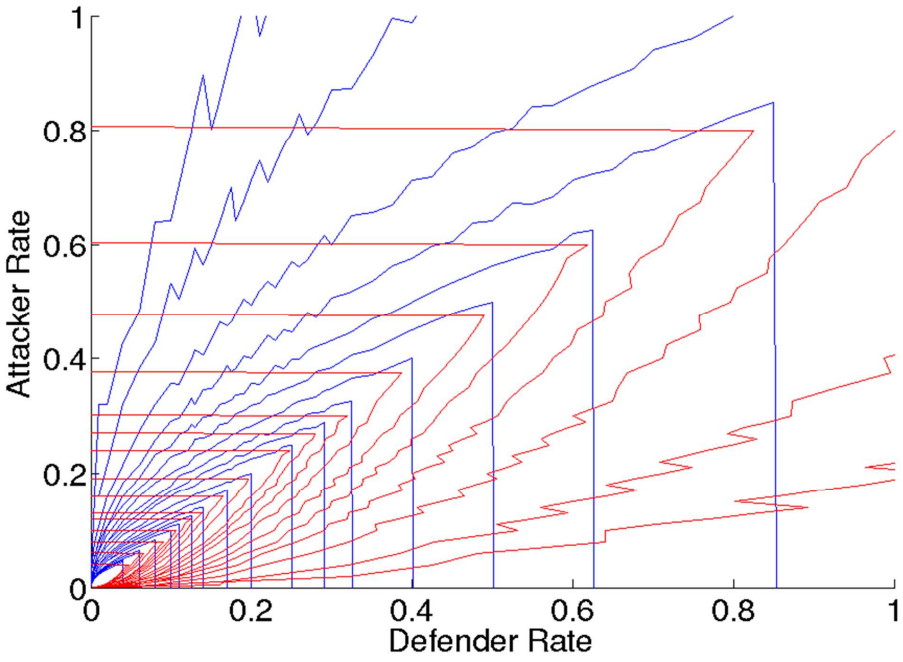

Figure 15: Optimal attacker (blue) and defender (red) move rates for various move costs. The noise is due primarily to limited samples at high move rates. The slight asymmetry in the response curves reflects underlying asymmetries in the FlipIt game.

any expected attacker move rate. The noise in the optimal response curves is due to error introduced by the finite model horizon and under-sampling of the attacker scenario space. We also note that as the game is separable and symmetric, we can switch the roles of the attacker and defender in our stochastic programming model and calculate optimal *attacker* profiles as well. It is significant to note that the results are not exactly symmetric: this is due to two asymmetries in the model: the resource is defined to be under control of the defender at $t = 0$, and in the event that both players move at the same time, the defender always gains control of the resource.

4.2 A Stochastic Programming Model for the PLADD Game

On the surface, the PLADD model is fundamentally different from FlipIt: whereas FlipIt was separable – allowing the attacker and defender to be analyzed independently – the presence of the morph move and the effect it has on the subsequent attacker time to success represents endogenous uncertainty that couples the two players together. However, because the morph move functions as a complete reset on the attacker's probability of success (f is completely

reset from $f_{learned}$ back to f_{base}), no information can leak across the morph boundary. This allows us to exactly analyze the infinite-time PLADD game as a single finite-time game where the period of the morph moves sets the time horizon, H. We will further simplify our model by considering the case of a fixed horizon (morph move interval) and a fixed number of defender moves. This changes our optimization from maximizing utility to minimizing the expected time that the attacker controls the resource.

The remaining challenge is capturing the additional information that is provided to the PLADD attacker, namely that the attacker is informed when they lose control of the resource. The net effect of this information is that the attacker will delay the start of their next attack until the first defender *take* move after their previously successful attack. As a result, we can no longer deterministically sample the time points that the attacker gains control of the resource as that time depends on when it lost control of the resource. However, the time until success (ϕ) for any individual attack can be sampled from a known distribution (f_{base} or $f_{learned}$). This allows us to create attack scenarios that consist of a vector of ϕ values, Φ, with Φ_0 sampled from f_{base} and $\Phi_{1..m_s}$ from $f_{learned}$). Each scenario will capture a different number of attacks (m_s), such that

$$\sum_{i \in \{0..(m_s-1)\}} \Phi_i < H \leq \sum_{i \in \{0..m_s\}} \Phi_i \tag{58}$$

The stochastic program then attempts to determine the amount of time the attacker controls the resource for each attack in each scenario ($\chi_{s,a}$). As the optimization algorithm will attempt to minimize the expectation of the total time that the attacker controls the resource, the key choice is to determine into which interval between defender moves each attack falls. The optimal defender moves will still occur at the same time as an attacker move in one of the attack scenarios. However, unlike FlipIt where the number of attack times grows as $|S| * |A|$, because the attack times are dependent on the defender move times, the finite set of attacker move times grows as $|S|^{|D|}$. This is large enough to prevent exact discretization of time. Instead, we pose the following continuous-time model:

$$\min \quad \sum_{s\in S, a\in m_s} \chi_{s,a} \tag{59}$$

$$s.t. \quad \begin{bmatrix} Y_{s,a,d} \\ t^A_{s,a} \le t^D_d \end{bmatrix} \bigvee \begin{bmatrix} \neg Y_{s,a,d} \\ \chi_{s,a} \ge \begin{cases} t^D_{d+1} - t^A_{s,a} & \text{if } d < \max(D); \\ H - t^A_{s,a} & \text{if } d = \max(D); \end{cases} \end{bmatrix}$$

$$\forall s \in S, a \in \{0..m_s\}, d \in D \tag{60}$$

$$t^A_{s,a} = t^A_{a-1} + \chi_{s,a-1} + \Phi_{s,a} \qquad \forall s \in S, a \in \{1..m_s\} \tag{61}$$

$$t^A_{s,0} = \Phi_{s,0} \qquad \forall s \in S \tag{62}$$

$$t^D_d \ge t^D_{d-1} \qquad \forall \{d | d \in D, d > 0\} \tag{63}$$

$$\psi_{s,a} = \sum_{i\in\{o..a\}} \Phi_{s,i} \qquad \forall s \in S, a \in \{0..m_s\} \tag{64}$$

$$\psi_{s,a} \le t^A_{s,a} \le H + \psi_{s,a} \qquad \forall s \in S, a \in \{0..m_s\} \tag{65}$$

$$0 \le t^D_d \le H \qquad \forall d \in D \tag{66}$$

$$0 \le \chi_{s,a} \le H \qquad \forall s \in S, a \in \{0..m_s\} \tag{67}$$

$$Y_{s,a,d} \in \{0,1\} \qquad \forall s \in S, a \in \{0..m_s\}, d \in D \tag{68}$$

Here $t^A_{s,a}$ is a continuous variable giving the success time of attack a in scenario s. Similarly, t^D_d is a continuous variable giving the time of the dth defender take move. $\psi_{s,a}$ a connstant reflecting the earliest possible success time for attack a in scenario s. Continuous variable $\chi_{s,a}$ is the amount of time the attacker controls the resource after the completion of attack a in scenario s.

Binary variables $Y_{s,a,d}$ control the disjunction (60), which either allows a particular scenario attack to succeed before the start of a specific defender interval (before the previous defender *take* time, t^D_{d-1}), or the time that the attacker controls the resource due to that attack is lower-bounded by the end of the interval (the next defender take time, t^D_d) minus the attack success time, $t^A_{s,a}$. Consider the ath attack success time and the dth take time in a scenario s. This is a relevant pair if the attacker control acquired in attack a is stopped by take d. The first disjunction handles the case where the attacker control gained by attack a is halted by an earlier defender take move ($d-1$ or earlier). In this case, this pair doesn't matter for setting $\chi_{s,a}$. If the attack a is stopped by a take later than d, then $t^D_d - t^A_{s,a}$ will be negative (the success time for attack a is after take d). Then the constraint in the second part of the disjunction is trivial. Only when the right-hand side of that constraint is positive will the constraint have an impact. This sets $\chi_{s,a}$ to the time between the ath attack success and the retake on the dth take. The objective function pressure forbids $\chi_{s,a}$ being any larger than the largest right-hand side from these disjunctions.

Constraints (62) enforce that the first attack begins at time 0 and succeeds after $\Phi_{s,0}$ (drawn from the distribution f_{base}). Constraints (61) enforce that the attack success times after the first attack are equal to the previous attack success time, plus the time the attacker holds the resource (that is, the time to the next take) plus the time to run the next attack. This assumes the attacker starts a new attack immediately after the defender takes control of the

resource. Constraints (63) enforce ordering for the defender take moves (that is, take $d+1$ must occur after take d). Constraints (64) define the constants $\psi_{s,a}$. The remainder of the model (65-68) specifies upper and lower bounds on all model variables.

It is also useful to note that this model reduces to a continuous time model of the FlipIt game by removing the $\chi_{s,a}$ term from (61):

$$t^A_{s,a} = t^A_{a-1} + \Phi_{s,a} \qquad \forall s \in S, a \in \{1..m_s\} \tag{69}$$

However, unlike the FlipIt model, which solves even relatively large models to optimality in a few minutes, neither the continuous time PLADD model nor the continuous time FlipIt model is computationally tractable. A single modest PLADD instance optimizing the time of 12 defender take moves over 30 scenarios had only closed the optimality gap to 31.9% after 10 days on a 32-core workstation. Scenario-based decomposition approaches were similarly unable to close the gap in a reasonable time. This will become even more problematic for realistic problem sizes: to reasonably capture the exponential distributions for f, we would need to solve problems with 100-200 scenarios. There are numerous opportunities for improving the performance of this approach, through innovations in both modeling and solution algorithms (see Section 6.7).

4.3 Alternative Combinatorial Scheduling Formulation

Expressing the stochastic programming version of PLADD in the language of combinatorial scheduling is another conceptual approach. Combinatorial scheduling is a popular area in theoretical computer science and operations research. This equivalent version of the problem may be easier to understand.

Suppose we have a set of m scenarios. Each scenario corresponds to one draw from the f_{base} distribution followed by up to t draws from the $f_{learned}$ distribution, where t is the number of takes before a morph. Let T be the number of ticks in the time horizon. In PLADD, T is the time between two morphs, which need not be consistent. We wish to schedule the set of t takes within this T-tick time horizon.

We can think of each scenario as a machine. Each machine has a list of jobs, each with a deterministic run time. The machine need not process all its jobs, but what it does process must be in the order from the list.

At the start of time (tick 0), each machine starts to process its first job. Whenever there is a start time, which corresponds to a take in PLADD, each machine that has finished its current job can start the next job on its list. Machines still working on a job must continue working on the current job. The goal is to schedule t global start times (take times) to minimize the total idle time over all machines. The idle time on a machine corresponds directly to the amount of time the attacker is in charge of a resource in the given scenario.

Without loss of generality, we can drop jobs and/or truncate the last job so that the sum of the processing times for the jobs for a given machine is at most T. If there is only one machine, then the optimal schedule is to start a new job at the point the previous job finishes. If a machine's list has $t+1$ jobs whose total length is $L < T$, then that machine must incur

$T - L$ ticks of idle time in any solution. These machines have more flexibility. That is, the start times can move away from job end times, with as much as $T-L$ total space between jobs while still being optimal. Of course an optimal global solution will in general be sub-optimal for many or all individual machines.

There exists an optimal schedule where each start time corresponds to the time a job ends on some machine. To see this, consider a schedule where there are t' ticks where all machines are idle before a new start. Moving the start ahead t' ticks and keeping all the other start times at the same relative positions opens up t' ticks at the end of the schedule without removing any productive time from any machine. If a machine was previously running a job that would have run past time T, that machine replaces at least some idle time with time running that job. Therefore the idle time cannot be increased and could be decreased.

This is an unusual scheduling model because there is forced idle time on a machine. A job is available, the the machine cannot run it because it is awaiting the next start time. There are scheduling models where such unforced idle time is optimal, but for the cases we are aware of, such structure is beneficial for the objective function.

Suppose we are given t_1 the value of the first non-trivial start time (time 0 is a trivial start time). Then the placement of the remaining $t-1$ start times is a new problem with the same overall structure as the first. Machines that were idle at time t_1 (that is, the length of their first job is at most t_1) have job list equal to the original with the first job gone. Machines that are still busy at time t_1 (the length of their first job is $\ell > t_1$) have the previous job list with the first job's length reduced to $\ell - t_1$. If there were initially t jobs in the list, the last job is dropped since there are now only $t - 1$ job starts remaining.

Each choice of first start time t_1 gives a different new problem. So, using only the property that each start time (take) occurs at the time a job finishes on some machine, there are m^t possible schedules. This is generally far too many to enumerate. If job run times are general, we do not expect sums of job lengths to be coincidentally equal. For example, if ℓ_{ij} is the length of the jth job on the ith machine, we do not expect $\ell_{11} + \ell_{22} = \ell_{21} + \ell_{12}$. Thus we do not expect dynamic programming to yield a polynomial-time algorithm.

5 Related Work

The principles of MTD have been known and used by militaries for a very long time. The Art of War advises "Keep your army continually on the move, and devise unfathomable plans." [27]. Use in the computing realm is obviously more recent [18, 10, 9], but techniques like periodic password changes and crypto-key rotation are as old as the discipline.

5.1 Effectiveness of MTD

More recently, researchers have begun examining the effectiveness of MTD techniques. Shacham et al. studied the practical effectiveness of address space layout randomization (ASLR) in 2004 and found that, because of implementation limitations like incomplete randomization

and small address spaces, ASLR was less effective against a determined adversary than one might have expected [24]. Even without those shortcomings, they showed that dynamically updating ASLR's memory object mappings will provide at most a factor of two increase in the effort required by an attacker over single shot address space randomization.

Evans et al. considered the general effectiveness of MTD [8] against a variety of attacker strategies like circumvention, deputy, probing, and incremental. They confirmed Shacham's results for basic ASLR and proposed other defense scenarios where MTD may have more value. They show that scenarios where multiple MTD techniques are composed or when a defender can force an adversary into an incremental attack approach that MTD can have a larger effect.

Our work on PLADD complements this earlier work by making the technical impact of MTD a parameter and by explicitly incorporating attacker and defender costs. We show that, for our formulation of the game, that there are cost configurations where MTD provides little or no cost benefit in acheiving a defensive goal like driving an attacker out of the game. We also show that the transition between the regime where a rational defender will choose never to use an available MTD move, because it does not add value, and where they will use the MTD move almost exclusively is typically quite abrupt.

5.2 Game Theoretic Analysis

van Dijk et al. recently introduced the FlipIt game [12, 28, 3]. A simple game theoretic framework for examining attacker and defender strategies for competitive games where little is known about the moves of an adversary. While simple, the basic FlipIt game exhibits interesting and complicated behaviors. Subsequently, many researchers have extended FlipIt to include additional features like multiple contested resources [15], defender probes [19], multiple attacker classes [16] and MTD [31].

Our PLADD game follows this pattern by introducing new features to FlipIt that implement one view of MTD. Our model of MTD was developed independently of Wellman et al. [31], but has similar features. While their solution uses empirical game theoretic analysis [30] where simulations are used to fill out a large game matrix and then solved using an automated solver like Gambit [17], we provide analytic solutions for attacker and defender utilities and costs.

6 Future Work

PLADD has demonstrated its value as a tool for exploring the dynamics of MTD for a variety of simple scenarios. The project, however, has generated many more questions about the practice of modeling cyber conflict than it has answered. In this section, we highlight a few interesting research directions.

6.1 Adaptive Player Agents

This project introduced and studied PLADD, an extension of the FlipIt model to evaluate MTD. While supporting far richer dynamics than FlipIt, the PLADD model shares some common limitations with it, such as the lack of adaptability of the player agents and the lack of multiple, heterogeneous adversaries. In order to capture additional important elements of real-world adversarial environments, the PLADD model will need to be extended to be able to support learning by the player agents, as well as the presence of multiple adversaries, each with a different objective and cost function, such as the differences between cyber-criminals and nation state adversaries either trying to steal secrets or cripple capabilities. Where periodic strategies are currently optimal, once adversaries can learn, they will be able to exploit the predictability of periodicity, making less predictable strategies almost certainly superior. The presence of multiple, heterogeneous adversaries will make it difficult to choose a single defensive strategy, causing a critical bottleneck in cyber security decision-making.

6.2 Coevolution

Game theory allows for the mathematical analysis of adversarial interactions, but the characterizations are generally limited to very simple models. Computational game theory is focused on scaling classical game theory to large, complex systems modeling more real-world environments. One promising approach is coevolution, where each player's strategy and fitness is dependent on, and evolves to reflect, their complex environment and potentially heterogeneous adversaries [21]. The proposed extensions to PLADD will quickly make it intractable for analytical analysis and classic game theory. However, coevolution, a powerful heuristic approach which trades guarantees of optimality for vastly superior scalability, can be employed to evolve adaptive strategies for the defender and all adversaries, leveraging the analytical insight from simpler models to provide initial strategy seeding. It has previously successfully been applied in critical infrastructure protection [23]. This approach is particularly well suited for identifying superior, less predictable strategies for scenarios with multiple, heterogeneous adversaries.

6.3 Mapping to Real-World Scenarios

Another important step in extending the current work is to calibrate it to real-world scenarios through systematically tuning the model parameters in a principled manner to each scenario, and then to study its predictive capabilities for those scenarios. By doing so for both the current form of PLADD and the proposed extensions, the required model fidelity can be determined for obtaining satisfactory modeling capabilities. Previous work has provided solutions for Stackelberg games with a worst-case guarantee with respect to the solution quality without making any assumption about the attackers' behavior model, as well as more robust solutions with guaranteed good defender utility against all reasonable quantal response attackers [11]. It may be worthwhile to adapt that work to PLADD to form a starting point for the proposed computational game theory approaches.

6.4 Using Risks and Expected Payoffs for Strategy Design and MTD Implementation

The work presented in this report focuses on understanding the expected utility of attackers and defenders and assumes that their decisions will be made based on those values. This ignores the importance of risks associated with specific strategies and objectives. The ability to take defender and the attacker risks into account when designing their respective strategies may substantially impact strategy selection as well as in-game and real-world outcomes. For example, in addition to pushing the attacker out of the game, the defender may want to design a strategy that maximizes or otherwise controls the risk taken by the attacker as well as controls risks taken by the defender. It is also necessary to understand how the deployment of MTD affects the risk profile of attacker and defender. One possible approach for including the results of such analyses in the decision making process is to represent the risk-reward trade-offs as an efficient frontier where each payoff is associated with the minimum risk at which it can be achieved.

6.5 Apply Martingale Representation and Game-Theoretic Probability to Information Acquisition and Strategy Development

The approach this paper takes to defender strategy analysis has drawn inspiration from Glenn Shafer's, Vladimir Vovk's, and Akimichi Takemura's work on Game Theoretic Probability and Defensive Forecasting. Their approach to strategic interaction using martingales served as a starting point to the formulation of the problem, the attacker expected utility definition, and to the choice of the deterrence solution as a cornerstone of the analysis of PLADD.

However, this analysis can be expanded significantly by understanding 1) when (in general) the attacker payoff is a martingale for MTD?; 2) What generic strategies the defender can employ to enforce the martingale property of the attacker strategy?; 3) How the attacker's expected payoff can be conditioned on all the information acquired by the defender during the game? Understanding the last item would allow fine-grained real-time control of attacker outcomes by the defender.

We believe connections to the signal processing literature should be explored in conjunction with the martingale representation as well. It is interesting to notice that the expression under the integral in Equation 5 in Section 3 can be thought of as a convolution of the defender strategy or its effect on the attacker payoff and the attacker success probability density function, thus bringing multiple connections to signal processing, reliability, stochastic control, and Fourier transformations. In this interpretation, the effective defender strategy is represented by a saw-tooth function that linearly decreases in the range of $(\tau, 0)$ on each interval (t_i, t_{i+1}) for any $i \in \{1, ..., \infty\}$. This also allows a possibility of using well-established deconvolution algorithms by both the attacker and defender to estimate each others' strategies, in addition to the possibility of applying learning approaches to the same problem.

6.6 Using Adversarial Games for Creating Resilient Cyber Systems

Complex cyber systems are normally designed or evolve to optimize costs with resilience as a secondary consideration. Random or adversarial shocks are seen as something to be avoided. An alternative is to design systems that learn and benefit from shocks. We believe this is an achievable goal and draw inspiration from four areas: our current work described in this report in creating mathematical and game-theoretic models for MTD; game-theoretic probability and on-line conformal prediction pioneered by Shafer and Vovk [25, 29]; resilience science currently developed at Sandia; and our preliminary efforts on understanding the mechanisms for anti-fragility [26].

The distinctive feature of cyber attacks is the inability, at least so far, to create a model that represents the domain with sufficient completeness that attacker and defender strategies can be confidently developed and tested within such models. New attack and defense methods are discovered routinely. One such example is MTD. Therefore, the defender is generally playing an incomplete information game against novel and only partially observable strategies. This makes online learning from shocks and surprises necessary and unavoidable.

Our goal is to develop principles for designing cyber systems that benefit from shocks and to develop strategies that enable the necessary information processing and learning, while controlling for risks. We explicitly recognize that information derived from a system's response to shocks needs to be incorporated into system operator strategies.

The game-theoretic model described in this report could be used as an inspiration for this approach. PLADD treats an adversarial game between an attacker and a defender as a stochastic process. We have shown that, in certain games, the defender has the ability to control the attacker payoff process and, for example, force the attacker payoff to become a martingale with zero expected payoff. We also have an indication that certain defender strategies and system design choices allow the defender to introduce intentional shocks in order to delay attacker learning or allow the system to experience external or adversarial shocks without significant performance degradation.

In PLADD, implicit information is generated, but, at present, it is not taken into account by either defender or attacker. This information can be used to estimate attacker or defender strategies. Strategies learned on one asset can be deployed on a different asset. Actions, such as introducing intentional shocks to the system, for example by making potentially suboptimal moves, can be taken to speed up learning about an adversary, as well as to disrupt learning by the adversary.

The generally non-stationary character of repeated adversarial games makes much of existing machine learning, which is based on learning stationary distributions, only partially applicable. We propose using on-line conformal learning and game-theoretic probability as a starting point. This approach is based on martingales and is suitable for learning over time in systems with memory and path-dependence. Conformal predictors can be used with many existing machine-learning algorithms and allow hedging predictions in an on-line setting. Game-theoretic probability allows designing strategies against both probabilistic and non-probabilistic adversaries.

6.7 Stochastic Programming Approaches

The Stochastic Programming (SP) approach to analyzing models like PLADD is promising both due to its flexibility and expressivity. However, to deploy these approaches in practice will require significant development of both modeling tools and algorithmic kernels. We see significant opportunities in three key areas:

Dynamic discretization of continuous-time SP. Continuous-time representations are more compact and avoid discretization errors that impact adversarial systems model fidelity and corresponding solutions in practice, but they are computationally challenging. Extending ideas from Dantzig-Wolfe decomposition (column generation) to balance the accuracy of continuous-time models with the computational tractability of discrete-time models to preserve fidelity, while managing problem growth is a promising area of research.

Adaptive SP through lazy scenario generation. In many cases a handful of the scenarios in a SP problem have the bulk of the impact on the algorithm convergence and accuracy. It is possible that new SP algorithms could "lazily" incorporate scenarios only when necessary to improve SP convergence or accuracy. SP run-times are more tractable with fewer scenarios. A major impediment to scalable SP algorithms for adversarial systems analysis is the number of scenarios required to accurately represent the space of attack scenarios. A "lazy" approach to SP mitigates this complexity by initially considering a small number of representative scenarios. As convergence of SP decomposition algorithms proceeds, solutions can be assessed in terms of sensitivity to new candidate solutions. Those new attack scenarios that significantly impact SP solutions can then be incorporated incrementally, and in a targeted manner. The overall result will be a significant reduction in both the overall number of iterations required for convergence, and the total amount of computation performed across the iterations.

Improved SP decomposition solvers. There are numerous opportunities to accelerate scenario-based SP decomposition algorithms (Progressive Hedging) using approaches like cross-scenario cutting planes, upper bounding techniques, dynamic scenario rebundling, and dynamic algorithmic tuning. Scenario-based SP decomposition is a scalable solution approach for deploying parallel compute resources to reduce run times. However, novel extensions are required to address SPs in adversarial systems contexts. In particular, cycling and related behavior is a major issue observed when using scenario-based SP for solving adversarial systems models. The proposed techniques are critical for solution of such models, which we have demonstrated to be intractable even given weeks of compute time with current algorithmic approaches.

7 Conclusion

The goal of this project was to evaluate the general effectiveness of MTD using simple, abstract models. Our game theoretic model, PLADD, intentionally excludes many details that would be important when implementing meaningful MTD in a real system, but which would complicate analysis. PLADD does, however, include features that we believe represent the essence of MTD, namely information that is available to an attacker which, if known,

makes attacks easier, and the ability of a defender to take that information away, at least temporarily.

We prove that, in PLADD, it is always possible for the defender to play fast enough to drive the attacker out of the game. We also prove that, in PLADD, an attacker has very limited strategies. For the important class of monotonically decreasing time-to-success distributions, the rational attacker will always begin attacking immediately after losing the resource and, hence, cannot choose the time of their attacks.

We have analytic solutions for defender and attacker utility in PLADD for general distributions of attacker time-to-success and generic defender strategies. For a sub-case of PLADD where defenders are limited to periodic strategies and attacker time-to-success follows an exponential distribution, we have performed a detailed analysis of a new, non-equilibrium metric that measures the difference in cost for a defender to drive a rational attacker out of the game when MTD is available vs. when MTD is not available. That analysis shows:

1. For large fractions of this version of PLADD's parameter space, MTD is not cost effective.
2. The fraction of the parameter space where MTD is effective shrinks as attacker costs go down.
3. The defender (intuitively) receives the largest benefit from MTD when MTD moves are powerful and cheap.
4. In many configurations, the defender transitions abruptly from never using MTD, because it is not cost effective, to using MTD as aggressively as possible.

Analyzing the abstract PLADD game was a challenging exercise that resulted in insights into the dynamics of MTD and raised many interesting questions about modeling strategic, adversarial scenarios. We believe, however, that to be truly useful, future models should be based on real-world scenarios and be calibrated using real-world data.

References

[1] Spyros Antonatos, Periklis Akritidis, Evangelos P Markatos, and Kostas G Anagnostakis. Defending against hitlist worms using network address space randomization. *Computer Networks*, 51(12):3471–3490, 2007.

[2] Elena Gabriela Barrantes, David H Ackley, Stephanie Forrest, and Darko Stefanović. Randomized instruction set emulation. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):3–40, 2005.

[3] Kevin D Bowers, Marten Van Dijk, Robert Griffin, Ari Juels, Alina Oprea, Ronald L Rivest, and Nikos Triandopoulos. Defending Against the Unknown Enemy: Applying FlipIt to System Security. In *Decision and Game Theory for Security*, pages 248–263. Springer, 2012.

[4] George Candea, Shinichi Kawamoto, Yuichi Fujiki, Greg Friedman, and Armando Fox. Microreboot-A Technique for Cheap Recovery. In *OSDI*, volume 4, pages 31–44, 2004.

[5] Monica Chew and Dawn Song. Mitigating buffer overflows by operating system randomization. Technical report, Carnegie Mellon University, December 2002.

[6] Coverity. Coverity Scan: Open Source Report. Technical report, Coverity, 2013.

[7] Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser. N-variant systems: a secretless framework for security through diversity. In *Usenix Security*, volume 6, pages 105–120, 2006.

[8] David Evans. Effectiveness of Moving Target Defenses. In *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, pages 29–48. Springer, 2011.

[9] Sushil Jajodia, Anup K. Ghosh, V.S. Subrahmanian, Vipin Swarup, Cliff Wang, and X. Sean Wang, editors. *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*. Springer, 2013.

[10] Sushil Jajodia, Anup K. Ghosh, Vipin Swarup, Cliff Wang, and X. Sean Wang, editors. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer, 2011.

[11] Albert Xin Jiang, Thanh H Nguyen, Milind Tambe, and Ariel D Procaccia. Monotonic Maximin: A Robust Stackelberg Solution Against Boundedly Rational Followers. In *Decision and Game Theory for Security*, pages 119–139. Springer, 2013.

[12] Ari Juels. FlipIt: A Game-Theory Handle on Password Reset and Other Renewal Defenses. In *RSA Conference*, 2012.

[13] Gaurav S Kc, Angelos D Keromytis, and Vassilis Prevelakis. Countering code-injection attacks with instruction-set randomization. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 272–280. ACM, 2003.

[14] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In *Selected Areas in Cryptography: Proceedings of the 6th Annual International Workshop (SAC'99)*, volume 1758 of *Lecture Notes in Computer Science*, pages 13–33. Springer, August 2000.

[15] Aron Laszka, Gabor Horvath, Mark Felegyhazi, and Levente Buttyan. FlipThem: Modeling Targeted Attacks with FlipIt for Multiple Resources. Technical report, Budapest University of Technology and Economics, 2013.

[16] Aron Laszka, Benjamin Johnson, and Jens Grossklags. Mitigating covert compromises. In *Web and Internet Economics*, pages 319–332. Springer, 2013.

[17] Richard D. McKelvey, Andrew M. McLennan, and Theodore L. Turocy. Gambit: Software Tools for Game Theory, 2014.

[18] H. Okhravi, M.A. Rabe, T.J. Mayberry, W.G. Leonard, T.R. Hobson, D. Bigelow, and W.W. Streilein. Survey of Cyber Moving Targets. Technical Report ESC-EN-HA-TR-2012-109, MIT Lincoln Laboratory, 2012.

[19] Viet Pham and Carlos Cid. Are We Compromised? Modelling Security Assessment Games. In *Decision and Game Theory for Security*, pages 234–247. Springer, 2012.

[20] Ronald L Rivest et al. Chaffing and winnowing: Confidentiality without encryption. *CryptoBytes (RSA laboratories)*, 4(1):12–17, 1998.

[21] Christopher D. Rosin. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego, 1997.

[22] Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian Dietrich, and Herbert Bos. SoK: P2PWNED – Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *IEEE Symposium on Security and Privacy*, pages 97–111, 2013.

[23] Travis C. Service and Daniel R. Tauritz. Increasing Infrastructure Resilience through Competitive Coevolution. *New Mathematics and Natural Computation*, 5(2):441–457, July 2009.

[24] Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the Effectiveness of Address-Space Randomization. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 298–307. ACM, 2004.

[25] Glenn Shafer. Game-theoretic probability and its uses, especially defensive forecasting. Working Paper 22, The Game-Theoretic Probability and Finance Project, 2007.

[26] N.N. Taleb. *Antifragile: Things That Gain from Disorder*. Incerto. Random House Publishing Group, 2012.

[27] S. Tzu and L. Giles. *The Art of War by Sun Tzu - Special Edition*. The Art of War - Special Edition. El Paso Norte Press.

[28] Marten van Dijk, Ari Juels, and Alina Oprea. FlipIt: The Game of "Stealthy Takeover". *Journal of Cryptology*, 26(4):655–713, 2013.

[29] V. Vovk, A. Takemura, and G. Shafer. Defensive forecasting. *eprint arXiv:cs/0505083*, May 2005.

[30] Michael P Wellman. Methods for empirical game-theoretic analysis. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 1552. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[31] Michael P Wellman and Achintya Prakash. Empirical game-theoretic analysis of an adaptive cyber-defense scenario (preliminary report). In *Decision and Game Theory for Security*, pages 43–58. Springer, 2014.