A Game Theoretic Approach to Strategy Determination for Dynamic Platform Defenses[*]

Kevin M. Carter, James F. Riordan, and Hamed Okhravi
MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 0242
kevin.carter, james.riordan, hamed.okhravi}@ll.mit.edu

ABSTRACT

Moving target defenses based on dynamic platforms have been proposed as a way to make systems more resistant to attacks by changing the properties of the deployed platforms. Unfortunately, little work has been done on discerning effective strategies for the utilization of these systems, instead relying on two generally false premises: simple randomization leads to diversity and platforms are independent. In this paper, we study the strategic considerations of deploying a dynamic platform system by specifying a relevant threat model and applying game theory and statistical analysis to discover optimal usage strategies. We show that preferential selection of platforms based on optimizing platform diversity approaches the statistically optimal solution and significantly outperforms simple randomization strategies. Counter to popular belief, this deterministic strategy leverages *fewer* platforms than may be generally available, which *increases* system security.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

Keywords

Moving target; game theory; system diversity

1. INTRODUCTION

Developing secure systems is difficult and costly. The high cost of effectively mitigating all vulnerabilities and the far lesser cost of exploiting a single one creates an environment which advantages cyber attackers. New cyber defense paradigms have been proposed to re-balance the landscape and create uncertainty for the attackers [10]. One such paradigm is moving target (MT) defense based on dynamic platform techniques, which dynamically change the properties of a computing platform in order to complicate attacks [12]. These properties that are dynamically adjusted focus around: the instructions being executed [20, 19, 9, 7] and their architecture [4], system and application memory [17, 16, 6, 15], operating system (OS) distribution [11], and machine instance [14].

While there has been significant work in developing these techniques from a systems engineering perspective, little work has been done on defining effective strategies for an optimal system deployment. When utilizing a dynamic platform defense, the defender's strategic considerations are two-fold: the duration of time to keep a platform active before migrating to another, and the destination platform[1]. The duration of time on each platform before migration is often scenario dependent, which we will generally define as an input variable d. The question of *which* platforms to deploy, however, is of critical importance. In many scenarios, the defender utilizes a platform migration system specifically because she lacks knowledge of the vulnerability states of the available platforms. In order to gain security through diversity, it is important to strategically deploy a diverse pool of platforms.

In this paper, we present a game theoretic approach towards the determination of optimal strategies for using a dynamic platform defense. We present the interactions between attacker and defender as a two-player, incomplete, leader-follower game in which the defender has imperfect information while the attacker has perfect information [8]. We discuss two threat models for which a dynamic platform defense is appropriate, focused around preventing adversary persistence for some extended duration on a protected system. We also consider the cases of both static and adaptive adversaries, with respect to the development of new exploits against the observed platforms. We will show that a deterministic scheduling strategy that optimizes for selecting the most locally – in a temporal sense – diverse platform approaches the statistically optimal solution.

This paper proceeds as follows: In Section 2, we discuss some threat models that are relevant for a dynamic platform defense. We leverage these threat models to define statistically optimal strategies in Section 3, and present simulation results comparing several potential strategies in Section 4. We discuss the insights developed throughout this work in Section 5, and summarize in Section 6.

[*]This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

MTD'14, November 03 2014, Scottsdale, AZ, USA
Copyright 2014 ACM 978-1-4503-3150-0/14/11 ...$15.00.
http://dx.doi.org/10.1145/2663474.2663478.

[1]These considerations do not apply to all systems, as some run all platforms at all times.

1.1 Related Work

Colbaugh and Glass [3] use game theory to analyze strategies for deploying MT defenses against adaptive adversaries, concluding with the result that uniform randomization is optimal. Their threat model differed from ours in that it did not require adversary persistence and the defensive strategy was designed to minimize predictability. Most importantly, they operated under the assumption that defensive systems are independent, such that any developed countermeasure works against a single system. We argue that this is not the case in many scenarios, such as exploits against operating system vulnerabilities, and will demonstrate that uniform randomization is highly suboptimal in these cases.

Self-cleansing techniques, such as SCIT [2, 5, 1], offer security by continuously rotating across many virtual machines and re-imaging the inactive ones. This impacts attack persistence by bringing a single platform to its pristine state, but offers no protection against the same adversary re-infecting the system with the same exploits.

Talent [11] is a framework for live-migrating critical applications across diverse platforms that has several design goals: i) diversity at the instruction set architecture level, ii) diversity at the operating system level, iii) preservation of the state of the application, including the execution state, open files and sockets, and iv) working with a general-purpose, system language such as C. While the analysis we present in this paper is applicable across a broad range of technologies, we notionally use Talent as our target system as it provides both temporal changes (periodic migrations) and diversity (multiple operating systems).

Our recent work has explored Talent and other dynamic platform technologies to quantitatively evaluate their effectiveness [13] and explore attacker strategies against their utilization [21, 22]. The work presented here builds upon the insights developed under our prior work.

2. THREAT MODEL

In our model, the defender has a number of different platforms to run a critical application. The attacker has a set of exploits (attacks) that are applicable against some of these platforms, but not the others. We call the platforms for which the attacker has an exploit "vulnerable" and the others "invulnerable." In a strict systems security terminology, vulnerable does not imply exploitable; without loss of generality, we only consider exploitable vulnerabilities. An alternative interpretation of this threat model is that the vulnerabilities are exploitable on some platforms, but not on the other ones.

The defender does not know which platforms are vulnerable and which are invulnerable, nor does she have detection capabilities for the deployed exploits. This scenario, for example, describes the use of zero-day exploits by attackers, for which no detection mechanism exists by definition.

Since there is little attempt to isolate the inactive platforms in dynamic platform systems, we assume that all platforms are accessible by the attacker, and the attacker attempts to exploit each one. Moreover, we assume that the defender does not have a recovery or re-imaging capability to restore a compromised platform. The reason for this assumption is that typical recovery methods (e.g. clean installation of the operating system) can take a long time to complete and unless the defender has many spare platforms, it is hard to accomplish it effectively. A large number of spare platforms also implies large hardware cost for these systems (e.g. SCIT [2]).

The attacker's goal is what creates the variations in our threat model. For example, one success criteria may be for the adversary to compromise the system for a given period of time to cause irreversible damage (e.g. crash a satellite), while a different success criteria gives the attacker gradual gain the longer the system is compromised (e.g. exfiltration of information). These variations and their impact on the effectiveness of dynamic platforms are discussed in the subsequent sections.

Much of the analysis of one system using dynamic platforms as a security strategy applies to any such system. In this section we will specify an abstraction that generically describes dynamic platforms, which is useful as a security strategy in protecting against attacks that have a temporal requirement. We analyze a number of cases where this holds. To better convey the meanings, we use simple phrases which are examples of these cases to refer to them: crash the satellite and exfiltration over a difficult channel.

1. Crash the Satellite - once the attacker has controlled the system continuously for a specified time period, the game is over and the attacker has won.

2. Exfiltration Over a Difficult Channel - the attacker needs control for a specified time period for an attack to succeed. Once the attacker has control for the period, the attackers payoff function starts to increase from a positive value.

We will define the abstract model of a dynamic platform system $\mathcal{P}$ as a system that migrates through a finite fixed collection of platforms $\{p_i\}$ (all of the notations used in this paper are summarized in Table 1). Each platform either has or does not have a property exploitable by the attacker which we call vulnerable, specified by the Boolean function $v(p)$, which is generally unknown.

Given the presented threat scenarios, the goal of the defender is not to globally minimize system vulnerability, but to prevent persistent vulnerability for a duration of some specified length T. Specifically, an attack is not successful unless the attacker has maintained such persistence, and a proper defender strategy takes advantage of this. Without loss of generality, we discretize time such that an adversary needs to be present for $K = T/d$ intervals. For simplicity of analysis, we assume that the time required to migrate platforms is negligible. This time is indeed unimportant for our analysis, however it would be if one were to optimize d.

2.1 Defining the Game

One can view the interactions between the attacker and defender as an iterative *leader-follower* game, for which each turn the *leader* plays a move first, and the *follower* plays subsequently. In the case of the presented scenarios, the defender always acts as leader, where moves are defined as selecting an active platform during turn k, p^k. The attacker always acts as the follower, where moves are defined as playing an exploit iff the active platform is vulnerable, $v(p^k)$. We frame this game as an incomplete and imperfect information game, as the defender does not know the moves available to the attacker (e.g. which platforms are vulnerable), nor does she know the moves that have been played by the attacker

$p_i \in \mathcal{P}$	The space of all platforms available
p^k	Platform at migration step k
d	The duration of a platform
T	Attacker's goal for disrupting the mission
$v\left(p^k\right)$	Platform at migration step k is vulnerable
$\neg v\left(p^k\right)$	Platform at migration step k is not vulnerable
$Pr(x)$	Probability of x

Table 1: Notation describing dynamic platform system

(e.g. no sensing capability). Meanwhile, while the attacker has perfect information (e.g. observes defender moves), it is incomplete since she does not know the full slate of available moves and must observe them.

Given that we are considering only threat models that require adversary persistence, system compromise is only possible iff there has been a succession of vulnerable platforms played by the defender. That is, compromise at time k is defined as a Boolean:

$$C(k) \quad = \quad \begin{cases} 1 & v(p^k), v(p^{k-1}), \ldots, v(p^{k-K+1}) \\ 0 & otherwise \end{cases}.$$

In Table 2, we relate the different threat scenarios presented in Section 2 to different objective games, where $1\{\cdot\}$ is the standard indicator function. The objectives in each scenario are optimized by the same strategies due to the incomplete and imperfect nature of the game; maximizing the time of first compromise generally minimizes the collective sum of compromise and vice versa. We will demonstrate this empirically in later sections.

2.2 Simplifying Assumptions

While optimal implementation of a dynamic platform defense is specific to a given threat model and set of available platforms, the goal of this work is to demonstrate a framework and general insights for developing these strategies. As such, we have made simplifying assumptions to make our model broadly applicable.

In many real exfiltration scenarios, an adversary gains partial success even when disrupted, such as reaching an intermediate attack state. While we simplify our metrics such that partial success is not granted, this assumption can be adjusted as necessary by changing the cost function.

A key tenant to our model is defining the similarity between platforms. This can be defined in a variety of ways, such as code, application, or protocol diversity. Additionally, the similarity between platforms can be defined based on commonly vulnerable portions, or the entire system. For the purposes of this paper, we focus on code vulnerabilities and define platform similarity based on similarity across the entire codebase (see Section 4.1).

Finally, while we model the correlation of exploit success between platforms by code similarity, we model exploit development as independent processes across platforms. The development lifecycle is modeled with a fixed distribution and is not dependent on previous exploit development. Essentially, our model does not account for increased efficiency with experience.

3. STRATEGY DETERMINATION

Recall that the goal of the defender is not to globally minimize system vulnerability, but to prevent persistent vulnerability for a duration of length K. This is best accomplished by, at each migration time, assuming the most recent $K-1$ platforms were vulnerable and migrating to a platform that is most likely to break the chain of adversary persistence. Intuitively, if the defender knows an adversary has been able to exploit some unknown vulnerability on the recent platforms, the proper course of action is to migrate to the least similar platform[2].

We consider two different operating modes for adversary capabilities, and it is important to differentiate them. In the *static* mode, all potential adversary capabilities are available from the beginning of time. The adversary gains nothing by observing the system; either she can exploit it or she cannot. Hence, a defender need not be concerned with overexposing a system. In the *adaptive* mode, an adversary is able to develop system countermeasures as time progresses, and this development is wholly dependent on the number of observations of the system. Consider a spam filter as an example, for which an adversary learns something about the features with each email that is sent (e.g. whether or not it was filtered) and can craft their messages in response. In this scenario, it is important for the defender to understand the information being leaked to the adversary and adjust their own strategy accordingly.

3.1 Static Adversary

The static setup corresponds to a short-term game, where the development of additional countermeasures is not feasible during the game. For example, one could consider the scenario in which a critical system needs to be available for a 24-hour window. It is generally not feasible for an adversary to develop a zero-day exploit for this system in such a short duration, so any exploits must already be available. As such, a defender's sole goal is to minimize the probability that an adversary can effect compromise at each given time step, which we can define as

$$Pr(C(k)) = Pr(v(p^k), v(p^{k-1}), \ldots, v(p^{k-K+1})). \qquad (1)$$

Recall, however, that the defender cannot see the moves played by the adversary. Given this constraint, let us redefine Eq. (1) using conditional probability as

$$Pr(C(k)) = Pr\left(v(p^k)| \bigcap_{j=k-K+1}^{k-1} v(p^j)\right) Pr\left(\bigcap_{j=k-K+1}^{k-1} v(p^j)\right) \qquad (2)$$

where $\bigcap$ is the standard intersection operator. Since the defender's goal is to minimize Eq. (2), and $Pr\left(\bigcap_{j=k-K+1}^{k-1} v(p^j)\right)$ is a constant at time k, the optimal move by the defender is to select p^k by solving

$$p^k = \arg\min_i Pr\left(v(p^i)| \bigcap_{j=k-K+1}^{k-1} v(p^j)\right). \qquad (3)$$

[2]In some circumstances this may lead to a locally optimal decision that is globally suboptimal as time progresses. While these conditions are not well defined, our emperical experience suggests they are not common.

Scenario	Defender	Attacker
Exfiltration Over Difficult Channel	$\min \sum_k C(k)$	$\max \sum_k C(k)$
Crash the Satellite	$\max\left[\arg\min_k 1\{C(k)=1\}\right]$	$\arg\min_k 1\{C(k)=1\}$

Table 2: Threat Scenarios Mapped to Game Objectives

As detailed in [13], this forms a Markov chain of order $K-1$. Effectively, the optimal strategy is to *assume* the adversary has been successful for the entirety of the recent history, such that a success at the current move leads to a reward for the adversary. One should select the platform statistically most diverse from the recent history, which can be defined in a variety of methods, such as common lines of code or shared modules. While defining this jointly across a collection of platforms is often infeasible, one may approximate by assuming conditional independence such that

$$Pr\left(v(p_i)|\mathcal{V}^K\right) \approx \prod_{j=1}^{K-1} Pr(v(p_i)|v(p^{k-j})) \tag{4}$$

$$= 1 - \prod_{p_j \in \mathcal{P}^K} \left[1 - Pr\left(v(p_i)|v(p_j)\right)\right], \tag{5}$$

where $\mathcal{P}^K = \{p^{k-1}, \ldots, p^{k-K+1}\}$ is the set of $K-1$ previous platforms. For ease of future notation, let us now define $v(\mathcal{P}^K) = \bigcap_{j=k-K+1}^{k-1} v(p^j)$, meaning that each of the prior $K-1$ platforms was vulnerable.

3.2 Adaptive Adversary

We now explore the case for which adversary adaptation is feasible. In longer-term games, an adversary is able to observe the strategy of the defender and develop targeted countermeasures. In these scenarios, systems that are played more frequently are more likely to become vulnerable. As such, the defender has two considerations: prevent the adversary from compromising the system and minimize the probability that the adversary will develop a new exploit for the presented platform, $Pr(E(k))$.

The new consideration provides interesting aspects to game strategies, as it often makes sense for the defender to present a known vulnerable system in order to prevent the adversary from developing new capabilities. This is due to the persistence requirement on the adversary; the defender can maintain the upper hand in the next iteration by invoking the diversity strategy. Hence, we now define the optimal strategy as one that solves the following:

$$p^k = \arg\min_{p_j} \lambda Pr(C(k)) + (1-\lambda) Pr(E(k)), \tag{6}$$

where λ is a parameter which trades off the strategic emphasis between current ($\lambda \to 1$) and future ($\lambda \to 0$) success.

Adaptation Model

Let us also define the probability that an exploit has been developed for p, based on $t(p)$ observations, as

$$F_p(t(p)) = \int_0^{t(p)} f_p(\tau)\, d\tau, \tag{7}$$

the cumulative distribution function evaluated at $t(p)$, where f_p is the probability density function governing exploit development. Without loss of generality, we assume that the adversary begins with no available countermeasures and the development of a new countermeasure is only possible during the turn in which it is presented by the defender. This assumption can be viewed in the sense that each time a move is played, additional information is leaked to the adversary which gives additional opportunity for development of an effective countermeasure. This assumption was relaxed in Winterrose et. al. [22] when exploring evolving attackers against static defenders.

Defining $Pr(C(k))$

$Pr(C(k))$ is defined similarly to how it was in Eq. (2), although now we must incorporate the probability that an exploit has already been developed. We are able to define the probability that a platform p^k is vulnerable as

$$Pr(v(p^k)) = F_{p_i}(t(p^k)) + (1 - F_{p^k}(t(p^k))) \times$$
$$\left(1 - \prod_{p_j \in \mathcal{P}\setminus p^k} \left[1 - F_{p_j}(t(p_j)) Pr(v(p^k)|v(p_j))\right]\right)$$
$$= 1 - \prod_{p_j \in P} \left[1 - F_{p_j}(t(p_j)) Pr(v(p^k)|v(p_j))\right]. \tag{8}$$

Intuitively, this can be viewed as the probability that a countermeasure has been developed specifically for p^k added to the probability that a countermeasure has been developed for a separate platform p_j and works on p^k (e.g. a cross-platform exploit), conditioned on a p^k exploit not being developed. These are the two ways that an exploit for p^k could come into existence[3].

Defining $Pr(C(k))$ in the same way as in Eq. (2), and therefore conditioning (8), we yield

$$Pr\left(v(p_i)|v(\mathcal{P}^K)\right) = 1 - \prod_{p_j \in P} \left[1 - F_{p_j}\left(t(p_j)|v(\mathcal{P}^K)\right) \times Pr\left(v(p_i)|v(p_j), v(\mathcal{P}^K)\right)\right] \tag{9}$$

$$\approx 1 - \prod_{p_j \in \mathcal{P}^K} \left[1 - Pr(v(p_i)|v(p_j))\right] \times \prod_{p_j \notin \mathcal{P}^K} \left[1 - F_{p_j}(t(p_j)) Pr(v(p_i)|v(p_j))\right], \tag{10}$$

where the (10) follows (9) from partitioning $\mathcal{P}$ into two sets, noting that $F_p(t(p)|v(p)) = 1$, and approximating the conditional independence in the same manner as (4). One should note the similarity to (5), where (10) differs by incorporating exploits developed for platforms not in the recent history that may still work on p_i.

Defining $Pr(E(k))$

Of critical importance to an adaptive adversary is considering not only the short term gains, but also the long term impact of moves played. Each move leaks information that

[3]Note that the adversary will not develop an exploit for a platform she already has available to her.

the opponent may use to develop countermeasures, and this must be factored into the defender's decision making process. Recalling that an adversary can only access a platform for which she has developed an exploit, one would like to minimize the probability of developing a *new* exploit during move k, $Pr(E(k))$. Note that this is strictly with new exploits; any information leaked about a platform for which the adversary already has exploited provides no benefit.

We define $Pr(E(k))$ as the probability that an exploit will be developed during move k, multiplied by the probability that the platform is not already vulnerable:

$$Pr(E(k)) = f_{p^k}(t(p^k))(1 - Pr(v(p^k))), \quad (11)$$

where $Pr(v(p^k))$ is defined as in (8).

Defining λ

When selecting the next move, it is important for the defender to appropriately weight the objectives with respect to the threat model. Consider, for example, the 'crash the satellite' scenario; the defender's first priority is to prevent the crash, then only after that is assured should she consider future impact. Hence, the defender truly wants to minimize $Pr\left(E(k), \neg v(\mathcal{P}^K)\right)$, which is the probability that a new exploit will be developed *and* the entirety of the recent history was not vulnerable – that is, the system cannot be compromised on this turn. This results in the following approximation:

$$Pr\left(E(k), \neg v(\mathcal{P}^K)\right) \approx f_{p^k}(t(p^k))\left(1 - Pr\left(v(p^k)\right)\right) Pr\left(\neg v(\mathcal{P}^K)\right). \quad (12)$$

Given that the defender's goal is to jointly minimize $Pr(C(k))$ and $Pr(E(k))$, both appropriately conditioned as discussed, we can minimize the sum of (10) and (12), choosing the optimal move as

$$p^k = \arg\min_k \lambda Pr(C(k)) + (1 - \lambda) Pr(E(k)), \quad (13)$$

where $\lambda = Pr\left(v(\mathcal{P}^K)\right)$. Note that this equation is the original goal expressed in Eq. (6) with an optimized value for λ. This value could easily be adjusted depending on the goals of the defender, although gains in short term success ($\lambda \to 1$) will lead to increased failure later in the game, while increased success late ($\lambda \to 0$) will lead to earlier short term failures. Adaptively defining λ each turn will optimally balance both these criteria.

The derivations for both Eq. (12) and (13) can be found in Appendix A.

3.3 Deterministic Strategies

For the static adversary, Eq. (3) results in a periodic scheduling strategy, which oscillates between the K most diverse platforms. For the adaptive adversary, there is not necessarily any periodicity derived from Eq. (6), but the complete sequence of moves can be computed given the initial move. We note that this determinism is acceptable under the adversary model, as the adversary acts as a follower and hence observes the defender's move each turn.

This relation is important to note, as the typical thought process has been that platform diversity is achieved through randomization. In fact, the opposite is true; randomization and diversity are two different criterion that are often orthogonal to one another. A strategy that optimizes for randomization would uniformly select from the available platforms, and would almost surely – as $k \to \infty$ – select consecutive platforms that are highly similar and vulnerable to the same exploits. Contrarily, a strategy that optimizes for diversity returns a deterministic schedule which minimizes the probability of consecutive platforms with similar vulnerabilities, according to some measure of similarity.

	CentOS	Fedora	Debian	Gentoo	FreeBSD
CentOS	1.0	0.6645	0.8067	0.6973	0.0368
Fedora	0.6645	1.0	0.5928	0.8658	0.0324
Debian	0.8067	0.5928	1.0	0.6202	0.0385
Gentoo	0.6973	0.8658	0.6202	1.0	0.0330
FreeBSD	0.0368	0.0324	0.0385	0.0330	1.0

Table 3: Code Similarity Scores, $S(i, j)$, including device drivers

4. SIMULATIONS

4.1 Set-up

We now demonstrate the optimality of our presented control strategies through a Monte Carlo simulation, leveraging a pool of five different platforms: CentOS, Fedora, Debian, Gentoo, and FreeBSD. We use the Measures of Software Similarity (MOSS)[4] [18] tool to compute a similarity score between each pair of operating systems. The results are presented in Table 3, where each similarity is on a scale of (0-1) where 1.0 implies identical code and 0.0 implies entirely distinct code. The input for each operating system was the kernel code and a set of standard device drivers. As discussed earlier, one should notice that FreeBSD is highly dissimilar to the 4 Linux distributions presented.

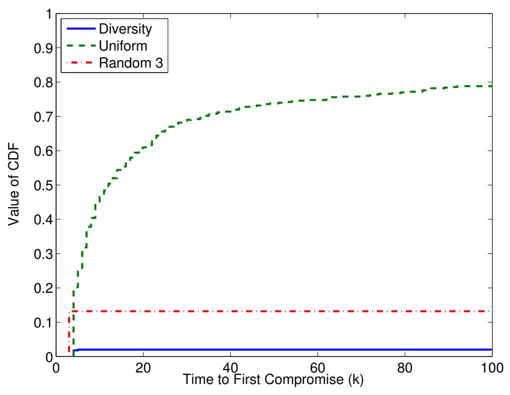
During the following simulations, we set $K = 3$, such that the adversary must be present for 3 consecutive moves. This could be viewed, for example, as $T = 90$ seconds with $d = 30$ seconds. During each of 500 Monte Carlo (MC) trials lasting $M = 100$ moves each, we evaluate five different game strategies for the defender selection process:

- *Uniform*: Uniform random selection at each interval from the fully available set without immediate repeat
- *Random 3*: Uniform random selection of $K = 3$ platforms prior to the trial, then periodic rotation between them
- *Diversity*: Optimizing diversity through Eq. (3)
- *Evolution:* Optimize against the development of new capabilities with Eq. (11) (only for adaptive adversaries)
- *Optimal*: Optimizing for both diversity and adaptation through Eq. (6) (only for adaptive adversaries)

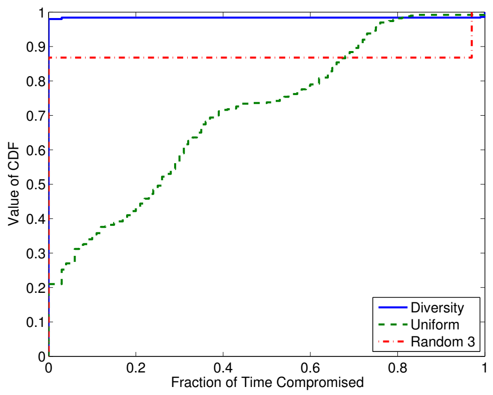
Throughout the simulations, we compute Eqs. (3) and (6) by setting $Pr(v(p_i)|v(p_j)) = S(i, j)$, where $S(i, j)$ is the similarity score between p_i and p_j given in Table 3.

Finally, let us define our metrics, relating back to Table 2. The first, related to the 'crash the satellite' scenario, is the *Time to First Compromise*, and is computed as $\arg\min_k 1\{C(k) = 1\}$. This is the time it takes before the system is fully compromised, that is the minimum value

[4]http://theory.stanford.edu/~aiken/moss/



(a) Time to first compromise



(b) Fraction of time compromised

Figure 1: Evaluation metrics on different migration strategies against a static adversary

of k for which $C(k) = 1$. The other metric, *Fraction of Time Compromised*, is related to the 'exfiltration over a difficult channel' scenario, and is computed as $\frac{1}{M}\sum_k C(k)$.

4.2 Static Adversary Results

We begin our analysis with the static adversary; one which has all available moves defined at the start of the game. Rather than require an extensive set of vulnerability exploits, we define the moves (e.g. exploits) available to the adversary by randomly selecting one of the five available platforms as vulnerable, $v(p')$. Next, we determine the vulnerability status of each other platform i by performing a Bernoulli trial with a probability of success equal to the similarity score between p' and p_i. The intuition behind this setup is that the greater the similarity between platform code, the more likely they share a vulnerability. While this is not a direct mapping, the intuition enables a robust analysis, and different measures of similarity could be used.

In Fig. 1a, we plot the cumulative distribution function (CDF) for the time it takes before the system is fully compromised, (e.g. 'Crash the Satellite' scenario). Optimizing for diversity results in system compromise only 2% of the time, while uniform selection eventually results in system compromise in 79% of the trials. We note that because the diversity optimization strategy results in periodic scheduling, if system compromise is going to happen, it does so nearly instantaneously – either at $k = K$ or shortly thereafter (once the period solution is reached). This is due to the fact that the system can only be compromised if all of the platforms in the resultant solution are vulnerable. The same can be said about the Random 3 strategy, but we note that performance is significantly reduced due to the increased probability of drawing 3 platforms that are highly similar. Contrarily, the Uniform strategy will almost surely result in system compromise as $k \to \infty$, if there exists K vulnerable platforms in the fully available set.

To study the 'Exfiltration over a Difficult Channel' threat scenario, we plot the CDF for the fraction of time that the system is in a compromised state in Fig. 1b. Once again, optimizing diversity yields far superior performance to other methods. In 98% of the trials, the system was never compromised (note this matches Fig. 1a), while in that other

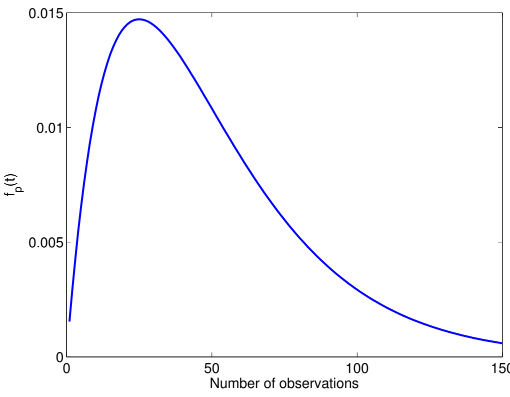


Figure 2: Adversary evolution model

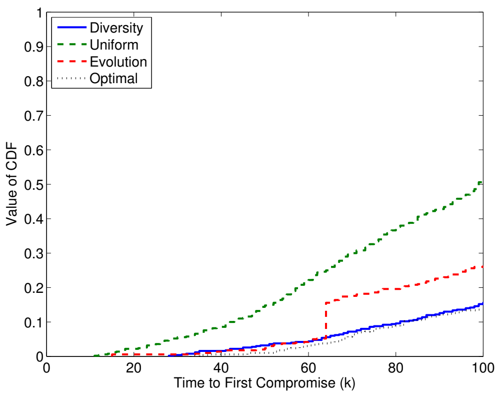
2% it was compromised for the entire duration. Meanwhile, uniform migration results in system compromise on average 26% of the time, and only in less than 2% of the trials did it result in less compromise than diversity optimization (13% of trials when comparing to Random 3).

The significant difference in performance emphasizes the need to develop a deployment strategy that is optimal towards the specific threat model of interest. Given the threat model requiring adversary persistence, deterministically optimizing for diversity shows superior performance to attempting to achieve security through randomization.

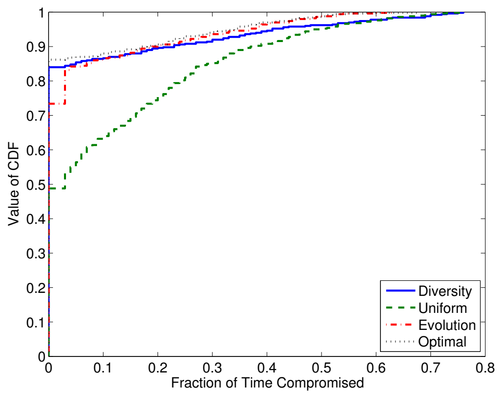
4.3 Adaptive Adversary Results

Prior work [3] has stated that uniform strategies are optimal because any defensive strategy that is biased towards certain moves leaks information to the adversary. More specifically, an adversary can expend resources developing countermeasures toward that work best against the defender's strategy. Let us now compare different defender strategies against an adaptive adversary.

Recall from our adaptation model that we assume that the adversary begins with no available countermeasures and the development of a new countermeasure is only possible



(a) Time to first compromise



(b) Fraction of time compromised

Figure 3: Evaluation metrics on different migration strategies against an adaptive adversary

during the turn in which it is presented by the defender. For these simulations, we leverage a Gamma distribution as our adaptation model, such that

$$f_p(t) = \frac{1}{\Gamma(\phi)\theta^{\phi}} t^{\phi-1} e^{-\frac{t}{\theta}},$$

where $\theta > 0$ is the scale parameter, $\phi > 0$ is the shape parameter, and $\Gamma(\cdot)$ is the gamma function. The Gamma distribution can be parametrized to equivalency with many distributions, including the Gaussian distribution, but we set the parameters $\theta = 25, \phi = 2$ for our simulations. This distribution is illustrated in Fig. 2, where intuitively the attacker quickly gains knowledge from observation, but eventually the information gained diminishes after a number of observations. We make no claim that this is the proper distribution for countermeasure development, but note that the equations computed in Section 3.2 are independent of the chosen distribution. Hence, any PDF could be swapped in, and we will analyze this further in Section 4.3.2. In our current parametrization, $E[f_p(t)] = 50$, meaning that on average, only 1 usable exploit will be developed[5], but there will be many runs where several exploits will be developed. We note, once again, that each exploit has a probability of working on other platforms drawn from a binomial distribution parametrized by platform similarity; this draw occurs on the turn the exploit is created and remains constant for the duration of the MC trial.

We now demonstrate the efficacy of different strategies against an adaptive adversary. In Fig. 3a, we illustrate the 'Crash the Satellite' scenario, and see that the statistically optimal strategy is indeed optimal empirically. One should note as well, that the diversity strategy is near optimal. The strategy which optimizes against adversary evolution is near optimal for some fraction of MC trials, but diverges in 36% of trials. This is due to the circumstances for which the adversary was able to develop multiple exploits within the trial. Meanwhile, the uniform strategy performs the worst by far. This is once again due to the threat model, as the strategy does not account for adversary persistence. Although there will not be an immediate repeat in moves played, the uniform strategy often results in move sequences such as (p_i, p_j, p_i), which reduces the attack surface by a third.

[5]The 2nd would be developed on the last turn

Diversity	Uniform	Evolution	Optimal
0.213	0.180	0.161	0.165

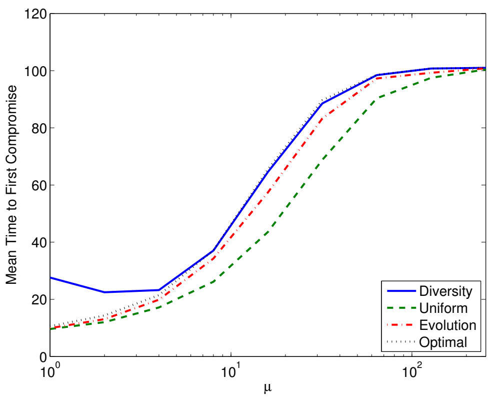
Table 4: Mean vulnerability rate against adaptive adversary not requiring persistence

We present the results for the 'Exfiltration over a Difficult Channel' in Fig. 3b, where similar results are observed. The diversity strategy is near optimal, as is the evolution strategy except for 3% of the cases. Once again, the uniform strategy is highly sub-optimal due to the threat model not being considered. In addition to not optimizing against persistence, the uniform strategy does not consider cross-platform exploits and will play randomly selected sequences of similar platforms, leading to increased adversary gain.

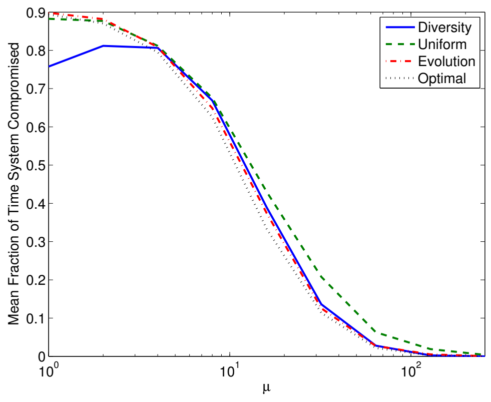
We note that in 94% of the trials, the diversity strategy saw no compromise at all, while the same can be said for only 49% of the trials with a uniform strategy, and 73% for the evolution strategy. Contrast this with mean vulnerability rate, $\frac{1}{M}\sum_k v(p^k)$, presented in Table 4, which is relevant only for threat scenarios that do not require adversary persistence, such as exfiltration over an easy channel. That the diversity strategy is by far the worst stresses that different strategies are required under different threat models.

4.3.1 *Performance vs. Adaptability*

Let us now briefly study the impact on adversary adaptability to the strategic performance of the defender. Specifically, we look to explore the impact of different strategies as a function of how quickly the adversary adapts; this is governed by $\mu = E[f_p(t(p))] = \theta/\phi$. In Fig. 4, we present the results over varying values of μ, each plot presenting the mean value of our defined metrics on a log-scale. While the diversity strategy is near optimal in most cases, it significantly outperforms the statistically optimal strategy for the rapidly evolving adversary. While the verbiage here seems counter-intuitive, recall that the optimal strategy considers the prevention of exploit development; when $\mu = 1$ it results in a strategy that actually prefers playing a previously used

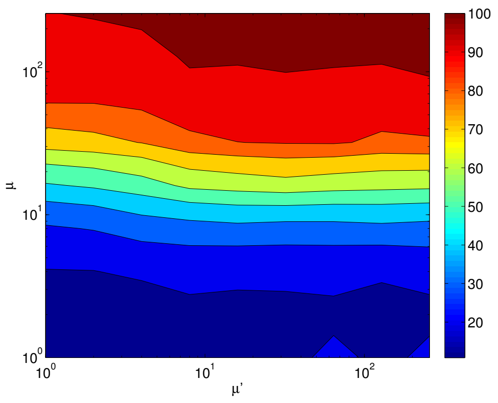


(a) Mean time to first compromise

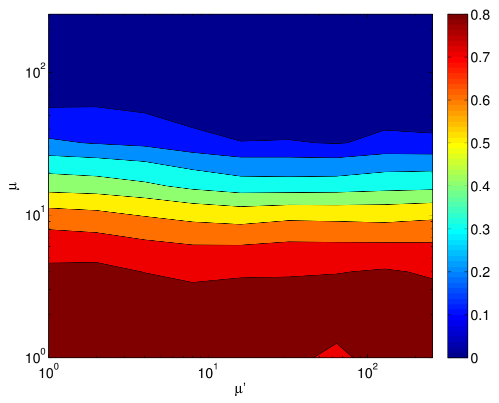


(b) Mean fraction of time compromised

Figure 4: Evaluation metrics vs. Adversary adaptability (μ)



(a) Mean time to first compromise



(b) Mean fraction of time compromised

Figure 5: Evaluation metrics as a function of model uncertainty

platform over an unused one, as $f_p(2) < f_p(1)$. This leads to more certain exploit development (and cross-platform exploits), while the diversity strategy minimizes the impact of cross-platform development. It should be noted, however, that an adversary that if an adversary can adapt that quickly, the defender is almost certain to lose regardless of the strategy she deploys.

We also see, once again, that the uniform strategy is highly sub-optimal, in both threat scenarios. As the adversary evolution becomes slower, the strategy approaches the optimal solution, but one should note that this regime effectively leaves the adversary with no moves, such that any strategy played by the defender is a winning strategy.

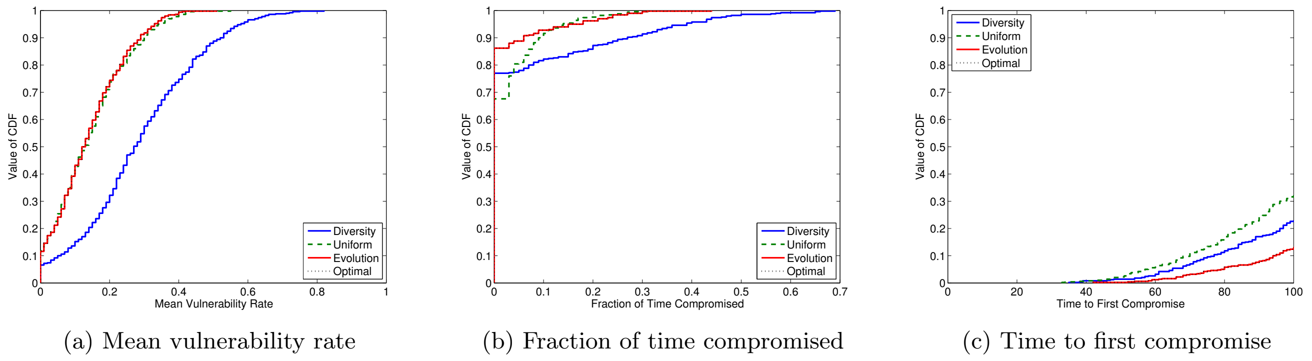
4.3.2 *Adversary Uncertainty*

Throughout the simulations, we have computed the statistically optimal solution under the assumption that the defender has perfect knowledge of the adversary adaptation model. Note that the defender never knows the realizations of the distribution, but did know the parameterization. We now analyze the performance against an unknown adversary. Specifically, we parametrize the adversary with μ, yet compute the statistically optimal strategy with μ'. In Fig. 5, we plot the contour map of our evaluation metrics over varying values of (μ, μ'), noting once again the log-log scaling.

The results show that the defender suffers little from misparamterizing the opponent. In fact, the defender suffers most when she assumes the adversary is rapidly evolving, *even if that is true*. As discussed earlier, this leads to an ineffective sequence of moves that adversely effects the success of the adversary. Contrarily, if the defender assumes a very slowly evolving adversary (e.g. $\mu' = 256$) at all times, the performance is roughly equivalent to setting $\mu' = \mu$. Given that a slowly evolving adversary is essentially static, the statistically optimal solution is indeed converging to the diversity strategy, regardless of the opponent being faced.

5. DISCUSSION

It is important to understand that the analyses presented in this paper are with respect to a very specific threat model: adversary requires persistence, adversary can observe the defender's moves and a single adversary move may be viable against multiple defender moves. We have demon-



(a) Mean vulnerability rate (b) Fraction of time compromised (c) Time to first compromise

Figure 6: Uniform strategies perform better in the absence of cross-platform exploits

strated that under these circumstances, a defender strategy that optimizes for diversity approaches the statistically optimal solution given perfect knowledge of the adversary's parametrization. These results proved consistent for both static and evolving adversaries.

As discussed in the introduction, prior studies [3] have demonstrated that a uniform deployment strategy is near optimal for the defender. These results may appear contradictory to our own, but they arise due to a different threat model. We have demonstrated that if the adversary *does not* require persistence – exfiltration over an easy channel – then a diversity strategy is suboptimal and these scenarios the strategy of broadening your attack surface is ideal. If the attacker cannot allocate resources towards adaptation in any biased and strategic fashion, her gains will be minimized. Additionally, and more critically, the scenarios for which uniform defender strategies are optimal do not account for cross-platform exploits. We illustrate this is Fig. 6, where we run the simulations in the absence of cross-platform exploits, such that each attacker move is only viable against one defender move[6]. One can see that the uniform strategies lead to much lower rates of vulnerability and compromise than the diversity strategy, and approach the optimal solution, which is equivalent to the evolution strategy. On the other hand, the uniform solution still results in the fastest compromise, and this is due to the preponderance of (p_i, p_j, p_i) sequences as discussed earlier.

While uniform strategies are near optimal against evolving adversaries on independent systems, we believe that these scenarios are very rare in reality. While not all adversaries attacks require persistence, exploits *do* work against multiple operating systems, applications, and hardware configurations. Defensive systems are not independent from one another, and this needs to be accounted for when developing defender strategies.

Predictability

One area for which a uniform strategy shows improvement is when the attacker and the defender play simultaneously. In this circumstance, the attacker is unable to first observe the defender's move, and must make his choice predicated on what she *thinks* the defender will play. For example, when dynamically allocating spam filters, the defender picks a filter to use against an unseen email, and the attacker sends an spam message against an unknown filter [3]. In these game scenarios, being predictable is to the detriment of the defender, so a deterministic strategy becomes suboptimal. However, in many scenarios, such as those presented in this paper, the attacker is able to wholly observe the defender's move before determining his own, making a stochastic strategy on the part of the defender yield no benefit.

[6]In these simulations we used $\mu = 30$ as to ensure enough exploits are created.

6. CONCLUSION

In this paper, we have proposed a game theoretic framework for studying moving target strategies using dynamic platforms. We were able to evaluate different threat scenarios and deployment strategies for both a static and adaptive adversary, discovering that a preferential platform selection strategy that optimizes the diversity of moves played converges to the statistically optimal solution. We also demonstrated that uniformly random selection strategies are highly suboptimal under the considered threat scenarios.

The key factor in the analysis provided in this paper is that optimal moving target strategies are highly dependent on the threat model under consideration. Employing defensive strategies that have not considered the threat model may result in the illusion of security, when one may actually be increasing their attack surface. While coupling a platform diversity system with a control mechanism may significantly improve its effectiveness, we have demonstrated that this is unnecessary under some circumstances, as the diversity solution is convergent with the statistically optimal one.

In future work we aim to study the efficacy of strategies under different conditions. Specifically, we are interested in changing the game from a leader-follower game to one in which both players play simultaneously, and studying the trade-offs between randomization and diversity.

7. REFERENCES

[1] D. Arsenault, A. Sood, and Y. Huang. Secure, resilient computing clusters: Self-cleansing intrusion tolerance with hardware enforced security (scit/hes). In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, ARES '07, pages 343–350, Washington, DC, USA, 2007. IEEE Computer Society.

[2] A. Bangalore and A. Sood. Securing web servers using self cleansing intrusion tolerance (scit). In

Dependability, 2009. DEPEND '09. Second International Conference on, pages 60 –65, june 2009.

[3] R. Colbaugh and K. Glass. Predictability-oriented defense against adaptive adversaries. In *Proceedings of the IEEE Intl. Conference on Systems, Man, and Cybernetics*, COEX, pages 2721–2727, 2012.

[4] D. A. Holland, A. T. Lim, and M. I. Seltzer. An architecture a day keeps the hacker away. *SIGARCH Comput. Archit. News*, 33(1):34–41, Mar. 2005.

[5] Y. Huang, D. Arsenault, and A. Sood. Incorruptible system self-cleansing for intrusion tolerance. In *Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International*, pages 4 pp. –496, april 2006.

[6] T. Jackson, B. Salamat, G. Wagner, C. Wimmer, and M. Franz. On the effectiveness of multi-variant program execution for vulnerability detection and prevention. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, MetriSec '10, pages 7:1–7:8, New York, NY, USA, 2010. ACM.

[7] P. Larsen, S. Brunthaler, and M. Franz. Security through diversity: Are we there yet? In *Proceedings of IEEE Security & Privacy*, Oct. 2013.

[8] R. B. Myerson. *Game Theory: Analysis of Conflict.* Harvard University Press, 1997.

[9] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. In *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '07, pages 89–100, New York, NY, USA, 2007.

[10] F. Networking, I. T. Research, and D. (NITRD). Federal Cybersecurity Game-change R&D Themes, 2012. http://cybersecurity.nitrd.gov/page/federal-cybersecurity-1.

[11] H. Okhravi, A. Comella, E. Robinson, and J. Haines. Creating a cyber moving target for critical infrastructure applications using platform diversity. *International Journal of Critical Infrastructure Protection*, 5(1):30 – 39, 2012.

[12] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. Finding focus in the blur of moving-target techniques. *IEEE Security & Privacy*, 12(2):16–26, Mar 2014.

[13] H. Okhravi, J. Riordan, and K. Carter. Quantitative evaluation of dynamic platform techniques as a defensive mechanism. In *Proceedings of 17th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Sept. 2014.

[14] A. Saidane, V. Nicomette, and Y. Deswarte. The design of a generic intrusion-tolerant architecture for web servers. *Dependable and Secure Computing, IEEE Transactions on*, 6(1):45 –58, jan.-march 2009.

[15] B. Salamat, A. Gal, and M. Franz. Reverse stack execution in a multi-variant execution environment. In *Workshop on Compiler and Architectural Techniques for Application Reliability and Security*, 2008.

[16] B. Salamat, A. Gal, T. Jackson, K. Manivannan, G. Wagner, and M. Franz. Multi-variant program execution: Using multi-core systems to defuse buffer-overflow vulnerabilities. In *Complex, Intelligent and Software Intensive Systems, 2008. International Conference on*, pages 843 –848, march 2008.

[17] B. Salamat, T. Jackson, G. Wagner, C. Wimmer, and M. Franz. Runtime defense against code injection attacks using replicated execution. *Dependable and Secure Computing, IEEE Transactions on*, 8(4):588 –601, july-aug. 2011.

[18] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 76–85, New York, NY, USA, 2003. ACM.

[19] K. Scott and J. Davidson. Strata: A Software Dynamic Translation Infrastructure. Technical Report CS-2001-17, 2001.

[20] D. Williams, W. Hu, J. W. Davidson, J. D. Hiser, J. C. Knight, and A. Nguyen-Tuong. Security through diversity: Leveraging virtual machine technology. *IEEE Security and Privacy*, 7(1):26–33, Jan. 2009.

[21] M. Winterrose and K. Carter. Strategic evolution of adversaries against temporal platform diversity active cyber defenses. In *Proceedings of Agent-Directed Simulation Symposium*, pages 68–76, April 2014.

[22] M. Winterrose, K. Carter, N. Wagner, and W. Streilein. Adaptive attacker strategy development against moving target cyber defenses. In *Proceedings of MODSIM World 2014*, April 2014.

APPENDIX

A. DERIVING λ

In this appendix we show the derivations of Eqs. (12) and (13), which are used to approximate λ, the optimal weight between objectives in defending the threat model. Recall the defender aims to minimize $Pr\left(E(k), \neg v(\mathcal{P}^K)\right)$, which is the probability that a new exploit will be developed *and* the entirety of the recent history was not vulnerable. Using conditional probability, we may condition this as follows:

$$\begin{aligned}
& Pr\left(E(k), \neg v(\mathcal{P}^K)\right) \\
= \; & Pr\left(E(k) | \neg v(\mathcal{P}^K)\right) Pr\left(\neg v(\mathcal{P}^K)\right) \\
= \; & f_{p^k}(t(p^k)) \left(1 - Pr\left(v(p^k) | \neg v(\mathcal{P}^K)\right)\right) Pr\left(\neg v(\mathcal{P}^K)\right) \\
\approx \; & f_{p^k}(t(p^k)) \left(1 - Pr\left(v(p^k)\right)\right) Pr\left(\neg v(\mathcal{P}^K)\right),
\end{aligned}$$

where the final approximation is derived from the lack of knowledge of which (or how many) of the recent history was not vulnerable.

To jointly minimize $Pr(C(k))$ and $Pr(E(k))$, we choose the optimal move as

$$\begin{aligned}
p^k \; = \; & \arg\min_k Pr\left(v(p_i), v(\mathcal{P}^K)\right) \\
+ \; & Pr\left(E(k), \neg v(\mathcal{P}^K)\right) \\
\approx \; & \arg\min_k Pr\left(v(p_i) | v(\mathcal{P}^K)\right) Pr\left(v(\mathcal{P}^K)\right) + \\
& Pr\left(E(k) | \neg v(\mathcal{P}^K)\right) Pr\left(\neg v(\mathcal{P}^K)\right) \\
= \; & \arg\min_k \lambda Pr(C(k)) + (1 - \lambda) Pr(E(k)),
\end{aligned}$$

where $\lambda = Pr\left(v(\mathcal{P}^K)\right)$.