A Queueing-Theoretic Framework for Dynamic Attack Surfaces: Data-Integrated Risk Analysis and Adaptive Defense

ANONYMOUS AUTHORS

We develop a queueing–theoretic framework to model the temporal evolution of cyber-attack surfaces, where the number of active vulnerabilities is represented as the backlog of a queue. Vulnerabilities arrive as they are discovered or created, and leave the system when they are patched or successfully exploited. Building on this model, we study how automation affects attack and defense dynamics by introducing an AI amplification factor that scales arrival, exploit, and patching rates. Our analysis shows that even symmetric automation can increase the rate of successful exploits. We validate the model using vulnerability data collected from an open source software supply chain, and show that it closely matches real-world attack-surface dynamics. Empirical results reveal heavy-tailed patching times, which we prove that they induce long-range dependence in vulnerability backlog and help explain persistent cyber risk. Utilizing our queueing abstraction for the attack surface, next we build a systematic approach for cyber risk mitigation. Toward that end, we formulate the dynamic defense problem as a constrained Markov decision process with resource-budget switching-cost constraints, and develop a reinforcement-learning (RL) algorithm that achieves provably near-optimal regret. Numerical experiments validate the approach and demonstrate that our adaptive RL-based defense policies significantly reduce successful exploits and mitigate heavy-tail queue events. Using trace-driven experiments on the ARVO dataset, we show that the proposed RL-based defense policy reduces the average number of active vulnerabilities in a software supply chain by over 90% compared to existing defense practices, without increasing the overall maintenance budget. Our results allows defenders to fundamentally quantify the cumulative exposure risk under long-range dependent attack dynamics and to design adaptive defense strategies with provable efficiency.

CCS Concepts: • **Security and privacy** → **Vulnerability management**; • **Computing methodologies** → **Markov decision processes**; **Model verification and validation**; **Reinforcement learning**.

Additional Key Words and Phrases: computer security, vulnerability dynamics, queueing theory, reinforcement learning, long-range dependence

ACM Reference Format:
Anonymous Authors . 2026. A Queueing-Theoretic Framework for Dynamic Attack Surfaces: Data-Integrated Risk Analysis and Adaptive Defense. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Cyber risk exhibits temporal dependence and cannot be adequately described by static or stationary reliability models. Much of the existing approaches in cybersecurity focus on isolated attack models or mitigation mechanisms, offering limited understanding of the holistic and time-varying nature of vulnerabilities that define an organization's *attack surface*. Modern infrastructures, spanning

Author's Contact Information: Anonymous Authors.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

cloud services, software-defined networks, and distributed APIs, further amplify these dynamics, producing attack surfaces whose scale and evolution are often unknown even to their operators.

To address this gap, we develop a *dynamic stochastic model* for the evolution of the attack surface. The model generalizes from an individual software component to an entire organization, and ultimately to large-scale ecosystems such as industry sectors or nation-state infrastructures. We formalize the instantaneous size of the attack surface as the number of active vulnerabilities, represented by the queue length of a stochastic service process. Arrivals to the queue correspond to the discovery or creation of new vulnerabilities, while departures represent either (a) successful exploitation or (b) successful patching. This queueing abstraction makes explicit the role of limited defense capacity, allowing attack-surface management to be studied as a resource-allocation and backlog-control problem.

Building on this foundation, we extend the model to capture the growing influence of automation and AI in both offensive and defensive operations. We introduce an *AI amplification factor* that scales vulnerability arrival, exploit, and patching rates. This abstraction is not intended to model AI systems in detail, but to examine how this factor reshapes backlog dynamics. Our analysis shows that even when attack and defense capabilities scale symmetrically, the rate of successful exploits can still increase superlinearly.

To demonstrate how accurately our proposed framework captures real-world vulnerability dynamics, we apply it to the problem of strengthening open-source software supply chains. Using the ARVO (Atlas of Reproducible Vulnerabilities for Open Source Software) dataset [19], which contains over 4,000 reproducible vulnerabilities from Google's OSS-Fuzz platform, we characterize the real-world dynamics of vulnerability discovery and patching across thousands of open-source projects. Event-level analysis reveals that vulnerability arrivals and lifetimes are bursty, heavy-tailed, and non-stationary, and that segmented queueing models accurately reproduce the temporal evolution of the attack surface size across development cycles. This temporal structure further exhibits *long-range dependence (LRD)*, indicating that correlations in exposure decay polynomially rather than exponentially. In practical terms, the effects of individual vulnerabilities persist far beyond their initial disclosure, highlighting systemic bottlenecks in patch deployment and motivating the need for continuous, adaptive, and resource-aware defense strategies to ensure supply-chain resilience.

Motivated by persistent patching delays and the imbalance between vulnerability arrivals and limited defense capacity observed in both data and AI-driven analysis, we develop a reinforcement learning (RL) approach for adaptive defense under resource-budget constraints. The defense resource, represented by the patching rate, directly influences the service process in our queueing model. Our dynamic framework allows defense rates to vary over time, while explicitly incorporating such switching costs into performance evaluation. Although the resulting control problem is analytically intractable in closed form, we design a low-complexity RL algorithm for adaptive defense allocation under uncertainty. In addition, we rigorously establish a near-optimal regret bound relative to an oracle defender and introduce new switching-reduction techniques that extend the theory of constrained Markov decision processes (CMDPs).

Finally, to illustrate the practical implications of our theoretical and empirical findings, we conduct numerical experiments to evaluate the proposed RL-based defense policy. The results show that adaptive resource allocation guided by our RL algorithm can substantially mitigate exploit success rates, achieving reductions of up to 55% compared to static defense strategies, while maintaining stable performance under both stochastic and adversarial vulnerability arrivals. In trace-driven experiments using the ARVO dataset, our adaptive defense policy reduces the average number of active vulnerabilities in a software supply chain by more than 90% compared to existing defense practices, while operating under the same overall maintenance budget. These

findings underscore that dynamic, learning-based defense policies not only outperform static benchmarks, but also yield smoother and more predictable system behavior. This demonstrates how our analytical framework can directly supports real-world cyber-defense decision making.

Contributions. This work establishes a foundational framework for analyzing and controlling the dynamics of organizational attack surfaces through a stochastic and queueing-theoretic lens. The key contributions are as follows:

- **Dynamic Queueing Model of the Attack Surface.** We develop a queueing-theoretic model that jointly captures the temporal and spatial evolution of active vulnerabilities, providing a unified representation of vulnerability discovery, exploitation, and patching across organizational or ecosystem scales.
- **AI-Amplified Threat Dynamics.** We extend the model with an *AI-amplification factor* that quantifies how automation accelerates vulnerability creation and exploitation. Analytical results show that the breach rate can grow superlinearly with automation, even when AI is deployed defensively.
- **Empirical Validation and LRD of Vulnerability Dynamics.** Using the ARVO dataset, we empirically validate the proposed queueing-theoretic framework on a real-world open-source repository. By fitting segmented queueing models to vulnerability discovery and patching events, we show that the model accurately captures the non-stationary and heavy-tailed evolution of the attack surface. We further prove that such heavy-tailed service distributions lead to LRD in attack surface size, explaining the persistent exposure patterns observed in practice and highlighting the structural limits of static defense strategies.
- **Near-Optimal Adaptive Defense via RL.** We formulate adaptive patching as a CMDP with resource-budget and switching-cost constraints, and develop a near-optimal RL algorithm for adaptive defense. The algorithm achieves a sublinear regret relative to an oracle defender and produces smoother, more stable defense actions under varying attack intensities.
- **Theoretical Advances in Learning-Based Defense.** Our analysis introduces switching-reduction techniques and, to our knowledge, provides the first sublinear regret guarantees for RL under the joint coexistence of resource-budget and switching-cost constraints. These results advance the theoretical foundation of learning-based cyber defense.
- **Defense Switching Cost.** To our knowledge, this work is the first to model and analyze the amount of defense change as a measurable switching cost in RL. Specifically, the switching cost in Eq. (6) quantifies the *magnitude* of consecutive policy adjustments, in contrast to previous approaches [2, 10, 26] that penalize only the *frequency* of policy changes.

Together, these contributions establish a quantitative and theoretically grounded foundation for modeling, analyzing, and dynamically defending evolving attack surfaces.

2 Related Work

Our work is related to probabilistic approaches to cyber risk analysis. The industry standard, Factor Analysis of Information Risk (FAIR) framework [12] formalizes cyber risk quantification through probabilistic factors such as threat events, vulnerabilities, and loss magnitude, providing a common language for risk assessment. Broader treatments of probabilistic cyber insurance and risk evaluation can be found in [16]. While these approaches are influential, they largely assume static system conditions and do not capture the evolving temporal dependencies characteristic of modern attack surfaces.

The concept of the attack surface was formalized by Manadhata and Wing [18], and a systematic review [29] revealed fragmented definitions across hundreds of studies. Recent large-scale analyses, such as [9], quantified attack surfaces across government infrastructures, highlighting their scale

and complexity. These works provide valuable measurement perspectives, but they typically treat the attack surface as a static quantity and do not model how it evolves over time or responds to defense actions.

A related line of work models interdependent vulnerabilities through probabilistic attack graphs [30] and their AI-based extensions [11]. Bayesian-network models [13, 23, 24] have been proposed to estimate compromise probabilities, but these frameworks describe the system at a single snapshot in time. We refer to such methods as *snapshot models of risk*, as they capture system state at a fixed point in time and do not represent the sequential or long-range evoluation of vulnerabilities.

Efforts to incorporate temporal evolution have used Bayesian networks for industrial and cloud systems [25, 32] and Markovian models for sequential attacks [14, 17]. These studies focus on specific environments rather than the evolution of the attack surface as a whole. Haldar and Mishra [8] and Feutrill et al. [5] observed that vulnerability disclosures exhibit burstiness and long-range dependence, suggesting queueing systems as a natural abstraction. However, existing studies do not combine such models with large-scale empirical validation or address the joint temporal and spatial dynamics of vulnerabilitiy backlogs.

A key enabler for such modeling is the availability of event-level vulnerability data. The recently released ARVO dataset [19] provides detailed timestamps of vulnerability discovery and patching. Our work is the first to leverage ARVO to calibrate and validate a queueing-theoretic model of attack surface evolution, bridging theoretical abstractions with empirical vulnerability dynamics.

The rapid integration of AI into both software development and exploitation further complicates this landscape. While large language models (LLMs) can assist in code repair [3, 28], they also accelerate exploit generation [4, 7, 31]. Reports by practitioners and agencies [20, 22] highlight this dual role of AI as both attacker and defender. Yet existing models do not provide a quantitative framework for studying how automation simultaneously affects vulnerability discovery, exploitation, and patching dynamics. Our use of an *AI amplification factor* is intended to capture these rate-level effects in a tractable way.

Finally, constrained and safe RL has been studied under budget [1, 21, 27] and policy-adaptation [2, 10, 26] constraints. Existing studies on policy-adaptation primarily penalize the number of policy changes, i.e., the frequency of updates. Without the magnitude of change, it is not completely possibly to quantify the operational cost of change actions in practical defense settings. In contrast, our formulation models the amount of change in the executed defense action and quantifies the magnitude of consecutive policy adjustments. This distinction allows us to model reconfiguration overhead in a more realistic way. Moreover, previous work does not consider the joint effect of resource-budget and switching-cost constraints on adaptive defense policies. Our formulation unifies these elements and provides the first theoretical regret guarantees for RL in this setting.

Overall, our study introduces a queueing-theoretic perspective that explicitly models the time-varying and heavy-tailed nature of vulnerability backlogs. By validating the model on real data and integrating it with adaptive control, we provide a quantitative framework for analyzing dynamic attack surfaces and defense resource allocation. Using the framework, we provide an RL-based systematic approach to allocating constrained defensive resources to achieve a significantly improved attack surface dynamics, with the variations in the budget directly taken into account.

3 System Model

Consider a single component in an organization's IT stack, such as an authentication service, file server, or endpoint device. Each component maintains an attack surface, which represents the **set of currently active vulnerabilities**. For example, in a software release, the attack surface can be defined as the set of unpatched bugs (the definition can also be extended depending on the dependencies to the other systems). In a larger ecosystem, like an enterprise, the IT/OT environment

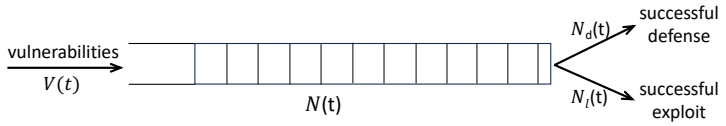


Fig. 1. Attack surface modeled as a queueing system. Vulnerabilities arrive via $V(t)$ and depart through competing defense (patching) and exploit processes.

will have a complex attack surface, composed of the combination of the attack surfaces of each component in the system. The overall attack surface will exhibit an interplay across the network of components, and its size naturally reflects how many vulnerabilities remain exposed at a given time. In this paper, to build the initial foundation, we focus on a single subsystem or component. The single-component model can be naturally extended to multi-component or multi-organizational settings; we briefly discuss such extensions in Section 9.

The size of the associated attack surface at time t is represented by the stochastic process $N(t)$, which we model as the number of jobs in a queue. Here, arrivals correspond to the appearance of new vulnerabilities, and services represent their removal through patching or exploitation[1]. Let $V(t)$ denote the arrival process of vulnerabilities, and $N_d(t)$ and $N_l(t)$ denote the cumulative numbers of defended and exploited vulnerabilities up to time t, respectively, as shown in Fig. 1. The discrete-time evolution of the attack surface size is given by

$$N(t+1) = \big\{N(t) + V(t) - [N_d(t) + N_l(t)]\big\}^+, \tag{1}$$

where $\{\cdot\}^+ = \max\{\cdot, 0\}$. Modeling the attack surface as a queue makes explicit the role of backlog: vulnerabilities accumulate when arrival rates exceed patching capacity, and shrink only when defenses can keep up. This recursion embodies the key intuition: new vulnerabilities enlarge the attack surface, while patching and exploitation act as concurrent removal mechanisms. As shown later in our empirical analysis (Section 6), both the arrival and lifetime processes exhibit burstiness, heavy-tailed persistence, and non-stationarity.

Each vulnerability is subjected to a *race condition* between defensive and offensive actions. Let D_d and D_l denote random variables representing the defense time and exploit time, respectively. For each active vulnerability,

$$D_s = \min\{D_d, D_l\}, \tag{2}$$

determines its completion time, and the winner of the race $s = \arg\min\{D_d, D_l\}$ increments the corresponding counter $N_s(t)$. For instance, if $D_d = 10^{-2}$ and $D_l = 10^{-3}$ for a given vulnerability, the attacker acts ten times faster, leading to an exploit departure. This race captures the operational reality that a vulnerability remains exposed until either it is patched or it is exploited.

We define $\mu_d(t)$ and $\mu_l(t)$ as the instantaneous total mean service rates for the defense and exploitation processes, respectively. These rates quantify how quickly vulnerabilities are removed, either by defenders or attackers, at time t. Both sides may act on multiple vulnerabilities concurrently with the defensive limitation of simultaneous processing across m parallel servers under a fixed total capacity b. In practice, defensive prioritization or limited concurrency can be modeled by reducing m or adjusting per-server service rates. Here, b denotes the organization's defense budget, interpreted as the maximum aggregate patching effort that can be sustained at any time. Hence,

$$\mu_d(t) \le b. \tag{3}$$

[1]Removal of a job upon exploitation is optional in the model. One may assume that a vulnerability can be exploited multiple times, before it is removed/patched from the surface queue.

These characteristics motivate the adoption of a general $G/G/m$–b model rather than simpler memoryless abstractions. Note that in Kendall's notation, the standard $G/G/m/k$ framework [6] uses k to denote the maximum number of jobs allowed in the system (i.e., a queue-length capacity constraint). In contrast, our $G/G/m-b$ notation utilizes m to denote the number of parallel servers and b to represent the aggregate capacity constraint imposed on the total service rate, effectively modeling resource-limited defense operations. This general model is necessary to capture bursty arrivals, heavy-tailed patching times, and hard capacity limits, which are not represented by memoryless queueing models.

Unlike classical queueing models with independent service rates, both $\mu_d(t)$ and $\mu_l(t)$ may depend on the current attack surface size $N(t)$. As $N(t)$ grows, defenders must divide limited resources across more vulnerabilities, while attackers benefit from the expanded surface. This coupling creates a feedback effect: when the number of active vulnerabilities increases, the same defense capacity must be shared across more items, slowing down patching on each vulnerability, while attackers face more exposed targets and thus have more opportunities to succeed. For instance, $\mu_d(t)$ may decrease inversely with $N(t)$, while $\mu_l(t)$ increases proportionally to $N(t)$, reflecting the asymmetric scalability of attack versus defense. The interplay between these processes governs the temporal evolution of the attack surface.

Variations of the Model: Throughout this paper, we consider several specializations derived from our model:

- **Temporal variation analysis:** We use the limiting case $M/G/\infty$, which isolates temporal effects such as heavy-tailed persistence and LRD without capacity constraints, to build a theorem on how the heavy-tailed nature of the arrival and service processes affect the attack surface variations.
- **Data integration:** In Section 6, the model is instantiated as $G/G/m$–b in its full generality to capture bounded defense capacity and bursty vulnerability arrivals observed in the ARVO dataset.
- **Dynamic defense design and optimization:** In Section 8, we use the $G/G/1-b$ variation to build the RL framework, where a single effective defense rate $\mu_d(t)$ is adaptively controlled under resource and switching constraints.

4 Problem Formulation

Building on the stochastic queueing model above, we now formulate the adaptive defense problem. The objective is to allocate limited defense resources over time to minimize long-term exposure and breach costs, while accounting for reconfiguration (switching) overhead.

At each time step t, the defender selects a defense (patching) rate $\mu_d(t)$ subject to the resource-budget constraint $\mu_d(t) \le b$, while the effective exploitation rate $\mu_l(t)$ evolves according to the coupled arrival–service dynamics defined earlier. The resulting queue length $N(t)$ captures the number of active vulnerabilities and thus represents the instantaneous *attack surface size*. The control task is to design a policy $\pi = \{\mu_d(t)\}_{t=1}^{T}$ that balances: (i) risk reduction through faster patching, (ii) efficiency in total resource use, and (iii) stability against frequent reallocations.

We study two core problems that together form the foundation of our framework. The first focuses on data-driven model inference and empirical validation, while the second develops an adaptive control policy for dynamic defense allocation. Each problem highlights a distinct analytical or algorithmic component of the overall approach:

(P1) Data-Driven Characterization and Model Validation.
Given a set of event-level vulnerability data containing discovery and patch timestamps, we aim to find the optimal parameters θ^* that minimize the statistical distance between

the empirical queue-length distribution (QLD), denoted by $\hat{P}$, and the simulated QLD, $P(\theta)$, generated by the candidate model. Formally, we define:

$$\min_{\theta \in \Theta} d(\hat{P}, P(\theta)) \tag{4}$$

where $d(\cdot, \cdot)$ is a divergence metric, specifically the Kullback-Leibler (KL) divergence in our implementation. Θ is the parameter space for the $G/G/m - b$ queueing model, where $\theta = \{m, b, F_{IA}, F_{ST}\}$ includes the number of servers m, total capacity b, and the parametric distributions for inter-arrival (IA) and service times (ST).

Solving (P1) yields a segmented and validated model that captures the non-stationary and heavy-tailed behavior of real-world vulnerability dynamics, providing the empirical foundation for the adaptive defense control in (P2).

(P2) Learning-Based Adaptive Defense. This problem aims to develop a learning policy that adaptively controls $\mu_d(t)$ to minimize cumulative cost, i.e.,

$$\begin{aligned} \min_{\{\vec{\mu}_d(1:T)\}} \sum_{t=1}^{T} \mathbb{E}\Big[& C(\vec{N}_l(t)) + \|\vec{\mu}_d(t)\|_1 \\ & + g(\|\vec{\mu}_d(t) - \vec{\mu}_d(t-1)\|_\infty) \Big] \\ \text{sub.to:} \quad & \vec{\mu}_d(t) \le b, \quad t = 1, \ldots, T, \end{aligned} \tag{5}$$

where $\|\cdot\|_m$ represents the ℓ_m norm. The expectation reflects stochastic variability in attack arrivals and patching delays. The first term $C(\vec{N}_l(t))$ penalizes exploit success proportional to the instantaneous attack surface size, the second term $\|\vec{\mu}_d(t)\|_1$ captures cumulative resource use, and the third term $g(\cdot)$ models the switching cost for defense reconfiguration. Note that in contrast to abstract policy-adaptation cost in existing work, this switching cost is proportional to the *magnitude* of change in executed actions. The ℓ_1 norm in the second term captures the total defense effort expended over time, corresponding to cumulative patching resources. In contrast, the ℓ_∞ norm in the third term measures the largest change in defense rate between consecutive time steps, reflecting the operational cost of reconfiguring defense actions rather than the frequency of policy updates.

Before introducing adaptive defense strategies, we first analyze a few simple scenarios under basic static allocation case to build some intuition on attack surface dynamics.

5 Illustrative Examples: Static Resource Allocation

We begin with a simple baseline that assumes a fixed defense allocation. This example is **not** intended to represent a realistic situation, but rather to build intuition about how defense capacity and vulnerability arrivals interact in a queueing system. In this example, we assume memoryless arrivals and departures from our queue. These insights will help us better interpret the results in the later sections, where we relax the memoryless assumption.

5.1 M/M/∞ Abstraction

The memoryless nature of the arrival and service processes with the M/M/∞ queue removes temporal correlations and capacity interactions, allowing us to focus on how vulnerability arrivals and fixed defense capacity jointly determine attack surface size and exploitation rates. The state of the associated M/M/∞ system can be represented as a Markov chain with countable state-space.

As described in Section 3, in our analyses we assume that organizations have fixed amount of cyber resources and allocate the full amount without a variation from one episode to another. In particular, the rate of the successful defense process $\mu_d = \alpha\lambda$ remains constant and thus independent

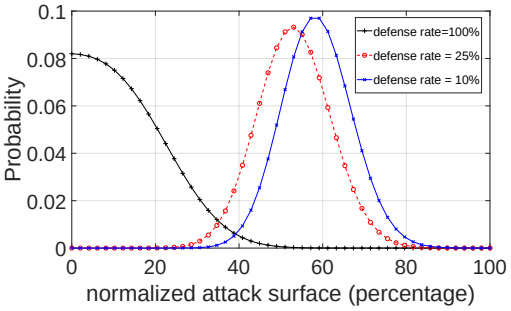


Fig. 2. Probability mass function for the size of the attack surface for different defense rates. Surface size is scaled 0–100% for visual interpretation.

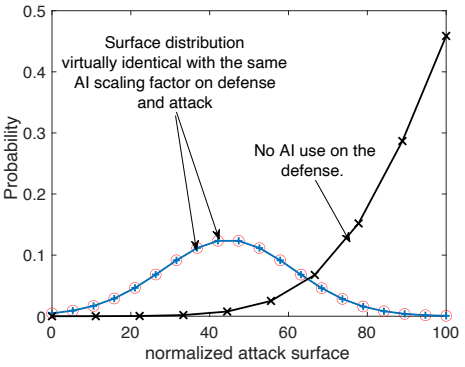


Fig. 3. Probability mass function for the size of the attack surface with AI usage. Distribution remains identical under symmetric AI scaling while a major degradation is observed with no AI use on the offense.

of $N(t)$ where λ is the arrival rate for $V(t)$, the (stationary) vulnerability process, and α is a constant such that $\alpha\lambda \leq b$. Here, μ_d denotes the total defense rate. The variable α signifies the intensity of the defense. As an example, if $\alpha = 1$, we say the defense rate is 100% or if $\alpha = 0.5$, we say the defense rate is 50%.

On the attacker side, we assume the rate of the successful exploitation process grows proportional to $N(t)$ as a larger number of active vulnerabilities attracts more attack attempts targeting the exposed surface, growing proportional to the attack surface size. Hence, $\mu_l(t) = \beta\lambda N(t)$, where β is a constant denoting the intensity of attacks on the organization.

In Fig. 2, we illustrate the probability mass function (PMF) of the state of the attack surface. Here, we picked a constant and normalized the observed queue sizes with respect to that constant. As a result, the queue size is denoted as a "percentage" in the figure, rather than an absolute value. We took the vulnerability arrival rate as $\lambda = 100$ per unit time and $\beta = 0.001$. We have plotted three different PMFs for different values of defense rate: $\alpha = 100\%$, 25%, and 10%.

The curve with the full (100%) defense rate leads to a small expected surface size of $\mathbb{E}[N(t)] = 13.2\%$ and a time-averaged breach rate of $\lim_{t\to\infty} \frac{1}{t}\mathbb{E}[N_l(t)] = 6.79$ breaches per unit time, much lower than the time-averaged defense rate, which happens to be

$$\lim_{t\to\infty} \frac{1}{t}\mathbb{E}[N_d(t)] = \lambda - \lim_{t\to\infty} \frac{1}{t}\mathbb{E}[N_l(t)] = 93.21.$$

As the defense rate decreases, the expected surface size and exploitation rate increase sharply. At $\alpha = 50\%$, the expected surface size grows to 52.1% with a substantial breach rate of 69.69. This means, there are more than twice as many breaches as there are successful defenses. Breach rate grows even further to 84.16% as we further decrease the defense rate to 10%. The interesting thing here is that, the expected attack surface size in this final situation remains at 57.44, very close to the case with $\alpha = 0.25$, despite a significant decrease to $\alpha = 0.1$.

The above situation can be explained by the fact that, as the surface size grows, much of the departures are caused by a breach rather than a successful patch. As a result, the average surface size remains relatively static, however the instantaneous fluctuations above the mean are quickly exploited by the growing set of attackers. As a result, even though one may think that the attack surface is not much higher, the breach rate grows at a much higher pace with reduced defense rate.

Our initial results demonstrate a **phase-transition** phenomenon in the attack surface size distribution. Once the defense rate goes below a certain point, the surface distribution shifts sharply and abruptly to the right. Further reducing the defense rate beyond that shifting point does *not* change the disribution considerably. This observation underlines the importance of keeping a disciplined security posture for an organization and the resources should be allocated to have a rate at least identical to (if not much higher than) the rate of growth in vulnerabilities. For example, if the defense rate is set to $\alpha = 200\%$, the expected attack surface size remains below *a single vulnerability*, while the average breach rate is merely 0.29 per unit time! Our results show that a small increase in resources can substantially improve protection against breaches, whereas insufficient resource allocation leads to sharply degraded security performance.

5.2 AI-driven Dynamics

We continue the illustrative analysis with the memoryless model to study how AI-driven acceleration of vulnerability discovery and response affects attack-surface dynamics, captured through a simple rate-scaling abstraction. Let us introduce an *AI amplification factor* that scales the arrival and service rates and study its effect under symmetric and asymmetric amplification of attack and defense capabilities. We scale the vulnerability arrival rate and the exploitation rate by the same factor a, so $\lambda \to a\lambda$ and $\mu_l \to a\mu_l$. In the second part, we will scale the defense rate μ_d with the same rate, to evaluate the impact of the use of AI on the defensive side, as well as the attackers' side.

In this example, we use the same amplification factor across the three pillars of the model. Our intention here is to illustrate the drastic shift in temporal dynamics even when there is no change in the spatial dynamics of the surface. Also, we show the dynamics under asymmetry in AI amplification between attack and defense, where the asymmetry is in the favor of the attackers.

Similar to Section 5.1, we use $\mu_d(t) = \alpha\lambda$, where α is the defense rate and $\mu_l(t) = \beta\lambda N(t)$. We provide the probability mass function for the attack surface for three different situations:

(1) No AI is used (i.e., $a = 1$) on either the offense or the defense. Here, we choose the vulnerability arrival rate $\lambda = 5$, defense rate $\alpha = 50\%$, and the attack rate $\beta = 0.005$;
(2) AI used on the attack and defense with an AI amplification factor of $a = 4$ on both sides. Here, the $\lambda = 20$, is amplified by a compared to the previous case and both the attack and defense resources are also benefiting the same rate of amplification.
(3) AI is used on the attack side only. As a result, the overall defense resources remain at 2.5 units as in the original situation, while the vulnerability arrival rate and the attack rate scale with the AI amplification factor $a = 4$.

In Fig. 3, we illustrate the attack surface distributions for Cases (1-3) above. For Case 1 (no AI), the expected surface size remains at $\mathbb{E}[N(t)] = 42.48\%$ at a defense rate of 2.5 patches per unit time while the exploit rate is 2.06 exploits per unit time.

Notably, when AI-driven acceleration is applied symmetrically to both attack and defense, the steady-state distribution of the attack surface remains unchanged, even though vulnerabilities arrive and are processed at a faster rate. However, all event rates scale with the amplification factor. In particular, *the exploitation rate increases from* 2.06 *to* 8.24 *exploits per unit time*, i.e., by a factor of a. Thus, while the shape of the attack-surface distribution is preserved, successful exploits occur more frequently due to the accelerated underlying dynamics. This observation highlights that symmetric acceleration primarily compresses the time scale of events rather than altering the distribution itself, and suggests that simply matching attack acceleration with defensive acceleration may be insufficient to reduce exploitation frequency.

Lastly, if the defense does not use AI, while the AI is used on the attack, the expected surface size substantially increases to 80.54%. The exploitation rate is scaled up to 13.23 per unit time, demonstrating a super-linear increase with AI amplification factor. This observation shows that an asymmetry in the AI usage in the favor of the attack side leads to a disproportionately higher increase in the rate of successful exploits. This scenario illustrates how asymmetric acceleration on the attack side can significantly worsen backlog and exploit rates under fixed defense capacity.

The illustrative examples above demonstrate how fixed and AI-amplified defense rates influence the steady-state behavior of the attack surface. We now turn to real-world data to assess whether these modeled dynamics hold in practice. In the next section, we integrate empirical vulnerability data from the open-source software repositories and validate our queueing-theoretic framework against observed attack surface behavior.

6 Data Integration: Software Supply Chain

In this section, we evaluate the suitability of our queueing-based risk model using empirical vulnerability data. Our goal is to assess how accurately the model captures the temporal dynamics and structural properties of attack surface evolution in operational settings. If such validation fails, the utility of the model in guiding practical defense strategies would be limited. Once the fit is validated, the model can be used subsequently in defense and mitigation approaches.

To that end, we implement our framework in the use case of open-source software supply chain. In our implementation, we use the ARVO dataset [19], which aggregates more than 4,000 reproducible vulnerabilities from Google's OSS-Fuzz infrastructure, spanning hundreds of large-scale open-source C/C++ projects. Each record includes rich metadata such as report and fix timestamps, sanitizer type (ASan, MSan, UBSan), crash category (for example, heap buffer overflow or use after free), and severity level (low, medium, or high). This information enables precise event-level tracking of vulnerability discovery and patching. The dataset's granularity makes it particularly well suited for queueing-based modeling: vulnerability disclosures correspond to *arrivals*, while patch completions represent *service completions*.

Using this dataset, we first analyze the empirical dynamics of vulnerability arrival and departures, demonstrating that our queueing-theoretic framework provides an accurate and interpretable representation of real-world attack surface evolution. We then show that the queueing model faithfully reproduces the observed queue size dynamics, confirming its validity as a realistic abstraction of complex software ecosystems. We further characterize the heavy-tailed nature of both arrival and service processes, which reveals a systemic bottleneck that slows patch deployment and motivate the need for adaptive, data-driven defense strategies. In the next section, we build on these findings and propose a RL–based dynamic defense allocation algorithm that optimally distributes defensive effort to manage the attack surface size under resource and switching constraints.

6.1 Validating the Proposed Queueing Model on the ARVO Dataset

The following steps outline our complete empirical pipeline for constructing, segmenting, and validating the queueing model on the ARVO dataset, thereby linking theoretical formulation with real-world vulnerability dynamics.

Step 1. Queue reconstruction and exploratory analysis: We first align vulnerability discovery and patching timestamps to reconstruct the time series of open vulnerabilities $N(t)$. The ARVO dataset used here provides exceptionally high resolution, tracking over 4, 410 vulnerabilities across 260 unique open-source projects from December 2016 to May 2024. Each record includes exact event-level timestamps for vulnerability discovery and patching, alongside critical metadata such as severity (including 1, 150 High-severity cases), detection sanitizer (e.g., asan, msan), and specific crash types like Heap-buffer-overflow and Use-of-uninitialized-value. Figure 4 shows clear

Fig. 4. Temporal evolution of the attack surface size, $N(t)$, in the ARVO dataset, showing bursty discovery, delayed patching, and non-stationary behavior.

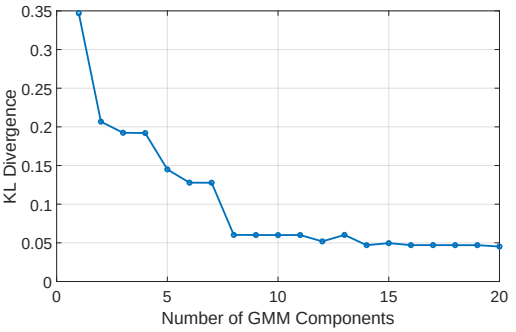


Fig. 5. KL divergence versus the number of Gaussian mixture components. Fit quality improves rapidly up to about ten components, after which additional components yield diminishing returns.

expansion and contraction phases, with bursty arrivals followed by delayed patching, confirming non-stationarity due to capacity limits in patching throughput.

Step 2. Segmentation via Gaussian mixture modeling: To capture these evolving patterns, we employed a *segmented modeling approach*. Segments are defined as mutually exclusive, closed intervals $[t_{start}, t_{end}]$ that collectively partition the entire observation period. Within these intervals, the arrivals and departures are statistically analyzed in isolation from the rest of the dataset to uncover localized distribution shifts and non-stationarities in the time series. A Gaussian mixture model (GMM) fitted to the empirical QLD identified roughly ten quasi-stationary segments, each representing a distinct operational regime. The number of mixture components was selected based on the KL divergence elbow curve shown in Figure 5, which indicates that model fit improves sharply up to around ten components and then saturates. This segmentation enables locally stationary modeling of non-stationary dynamics.

Step 3. Segment-wise parameter estimation: Within each segment, we estimated inter-arrival and service distributions and calibrated the queue parameters (m, b) of a $G/G/m$–b model by minimizing the KL divergence between empirical and simulated QLDs. In this segmented setting, the parameter b represents the *mean available defensive resource* rather than the maximum capacity used in the next section, reflecting the average effective throughput observed in each operational regime. The resulting segmented models accurately reproduced the multimodal and time-varying dynamics of the attack surface, confirming that segmentation is essential for representing the non-stationary evolution observed in ARVO.

Step 4. Statistical characterization of IA and ST: After segmentation, we analyzed the stochastic structure within each stationary window to identify appropriate parametric distributions for inter-arrival (F_{IA}) and service times (F_{ST}). We evaluated a wide range of candidate distributions using five divergence metrics (KL, TVD, L2, JSD, and Wasserstein). Heavy-tailed mixtures consistently outperformed non-heavy-tailed models, such as the exponential distribution, which underestimated tail mass and failed to capture persistence effects. As illustrated in Figure 6, for the first segment (weeks 0–64), the best-fitting non-heavy-tailed model (exponential) yielded significantly higher KL divergences of 1.31 for IA and 1.34 for ST. In contrast, the heavy-tailed loglogistic and Gamma–InverseGaussian distributions achieved much lower KL divergences of approximately 0.77 and 0.42, respectively. These results confirm that both vulnerability discovery and patching processes are fundamentally heavy-tailed, justifying our use of more complex $G/G/m$ abstractions to capture real-world long-lived exposure and temporal clustering.

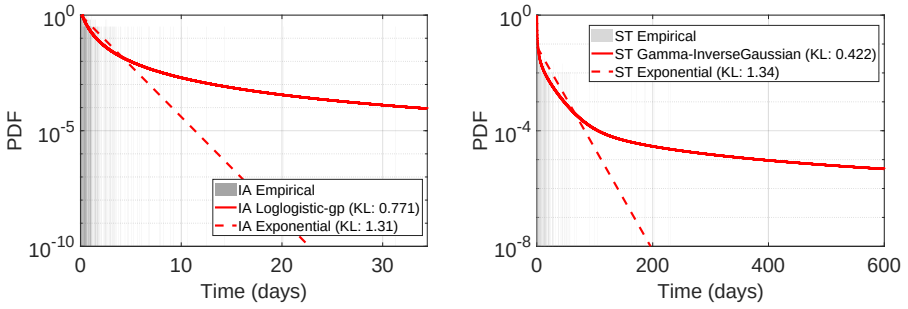


Fig. 6. IA and ST distributions for Component 1 (weeks 0–64). Loglogistic-general Pareto fits IA (KL ≈ 0.77); Gamma–InverseGaussian fits ST (KL ≈ 0.42).

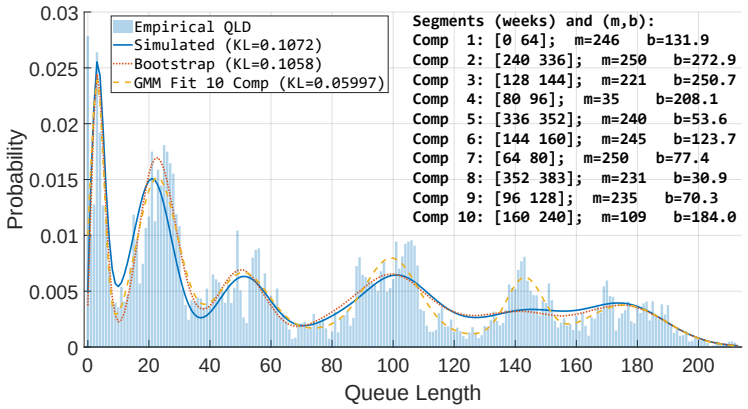


Fig. 7. Final integrated model: empirical QLD compared with segmented, bootstrap, and 10-component GMM fits. In the segmented $G/G/m$–b model, b denotes the mean available defensive resource rather than the maximum capacity used in the global formulation.

Step 5. Segment-wise queue model fitting and validation: Finally, we validate that the segmented $G/G/m$–b model accurately reproduces the empirical QLD observed in ARVO. Figure 7 compares the empirical QLD with the segmented, bootstrap, and ten-component GMM fits. In the bootstrap model, samples are drawn directly from the empirical data rather than from any fitted distribution, providing a nonparametric characterization. As shown, the segmented $G/G/m$–b model reproduces the empirical QLD with high fidelity, accurately capturing both the multimodal structure and the heavy-tailed persistence observed in practice. Quantitatively, the KL divergence between the empirical and simulated distributions is **0.1072**, comparable to the nonparametric bootstrap (**0.1058**).

Across segments, the number of servers m remains relatively stable (220–250), while the effective resource capacity b varies widely (30–270), reflecting changes in patching throughput and organizational defense posture. Together, these results confirm that *empirically calibrated queueing abstractions replicate real-world attack surface dynamics with sub*–0.11 *KL divergence, demonstrating near-empirical precision.*

6.2 Temporal Characterization of Vulnerabilities of Software Releases

The empirical fits above confirm that vulnerability service times follow a heavy-tailed distribution, with decay exponents (u) in the range $2 < u < 3$. Consequently, vulnerabilities tend to remain active on the attack surface for extended periods, depending on the underlying defense and exploit dynamics. Our results are corroborated by prior measurements [15] on specific networks, in which similar observations were made that the time a vulnerability remains exploitable follows a heavy-tailed law with a tail in D_s that decays slowly in a non-exponential fashion. Building on this empirical evidence, we formally show that when the vulnerability patching process exhibits such heavy-tailed behavior, the attack surface size $N(t)$ develops **long-range dependence (LRD)**, even if the vulnerability arrival process itself is memoryless.

More specifically, let $V(t)$ be memoryless with an arrival rate λ (a homogenous Poisson process for simplicity in this section). Also, we define $F(t)$ as the cumulative distribution function (cdf) of D_s, i.e., the service time[2]. A heavy-tailed distribution is characterized by a decay in $1 - F(t)$, that is slower than t^{-u} for $2 < u < 3$.

Theorem 1. For memoryless $V(t)$ and heavy-tailed $F(t)$, process $N(t)$ is stationary and it exhibits long-range dependence.

Proof. See Appendix A for the complete proof. Here, we provide a sketch: our derivation begins by establishing the autocovariance function of the number of customers in an M/G/∞ queue. We prove that $N(t)$ is wide-sense stationary and for heavy-tailed $F(s)$, the autocovariance function decays slower than $(\tau - t)^2$, leading to a long-range dependent attack surface. □

LRD of the surface size implies that the impact of a vulnerability may last for an *extended* amount of time. The size of the attack surface is not ergodic, making it extremely difficult for the organization to control the cyber risk. That is, two different realizations of the process may give completely different empirical statistics in terms of the attack surface dynamics. Therefore, based on our analytical observation, an organization that regularly performs penetration testing to identify, patch, and mitigate vulnerabilities can avoid long tails on the defense side, eliminating the LRD and its negative side effects. Organizations should set up a security practice to ensure that the vulnerability identification process occurs at regular (preferably deterministic) intervals.

Given that an organization has control over the distribution of inter-defense times, we next develop a RL–based dynamic defense allocation algorithm to regulate the effective service times.

7 A Near-Optimal RL Algorithm for Adaptive Defense

Building on the model (Sec. 3) and the analytical insights into static allocation, temporal dependence, and AI-driven dynamics (Secs. 5 and 6), we now build a systematic approach to the dynamic defense problem. To that end, we present a near-optimal RL algorithm for adaptively allocating constrained resources to dynamic defense, while accounting for switching costs. This learning-based approach is needed because the arrival and service processes governing the attack surface are unknown and may change over time. The RL agent represents an adaptive defender who episodically reallocates limited patching or monitoring resources across a dynamic vulnerability queue. The transition dynamics of the attack surface process are *unknown*. The policy switch for each episode corresponds to an operational reconfiguration (e.g., retuning patching pipelines or reassigning response teams), hence incurring measurable overhead modeled as a switching cost.

[2]For a vulnerability, we have that $D_s = \min(D_d, D_e)$.

7.1 Problem Setting and RL Framework

Specifically, this section provides the solution to the constrained optimization problem stated in (5), where the defender seeks the optimal dynamic defense policy under uncertainty.

Setting: We apply the episodic Markov decision process (MDP) to model the dynamic defending problem. As assumed in standard episodic MDPs, we consider H steps of interaction between the defender and the attack surface in each episode[3] $t = 1, \ldots, T$. At each step $h = 1, ..., H$, based on the observed state $N^h(t)$ of the system, the defender can take a defense action $\mu_d^h(t)$ according to a policy $\pi_t : \mathcal{N} \to \hat{\boldsymbol{\mu}}_d$, where $\mathcal{N} = \{1, \ldots, N\}$ is the system state space and $\hat{\boldsymbol{\mu}}_d = \{\mu_d\}$ is the defense action space. The defense must satisfy the resource-budget constraint $\mu_d^h(t) \leq b$ for all h and t. After the defense action, the state evolves according to Eq. (1). The episodic formulation reflects practical operation cycles, such as periodic defense planning or monitoring windows, during which defense rates are adjusted based on observed backlog.

Given any state N^h at step h, the defense action given by the current policy $\pi_t(\cdot)$ could be different from that given by the policy $\pi_{t-1}(\cdot)$ in last episode, in which case there will be a switching cost

$$\mathcal{S}^h(t, N^h) \triangleq w \cdot \left| \pi_t(N^h) - \pi_{t-1}(N^h) \right|$$

penalizing the change in defense across episodes, e.g., for parameter retuning, budget reallocation, and recontracting. This cost captures the operational overhead of changing defense intensity, rather than merely how often changes occur. Therefore, the goal is to find a desirable algorithm π that optimizes the expected cumulative penalty and defense cost over all steps and time-slots by executing the policies $\pi_{1:T}$, i.e., $\min_{\pi_{1:T}} \sum_{t=1}^{T} [\mathcal{V}^{\pi_t} + \mathcal{S}^{\pi_t}]$. Here, the $\mathcal{V}$-value function is defined to be

$$\mathcal{V}^{\pi_t} \triangleq \mathbb{E}\left[\sum_{h=1}^{H} \left[C\left(N_l^h(t) \right) + \mu_d^h(t) \right] \right],$$

where the expectation is taken with respect to the randomness of the state transition (1) and the race condition, and with a slight abuse of notation, the total switching cost is defined to be

$$\mathcal{S}^{\pi_t} \triangleq \mathbb{E}\left[\sum_{h=1}^{H} \sum_{N^h \in \mathcal{N}} \mathcal{S}^h(t, N^h) \right]. \tag{6}$$

Performance Metric: We use the standard regret as the metric to evaluate the performance of RL algorithm π, which is defined to be

$$\mathrm{Reg}(T) \triangleq \sum_{t=1}^{T} [\mathcal{V}^{\pi_t} + \mathcal{S}^{\pi_t} - \mathcal{V}^*], \tag{7}$$

i.e., the difference between the expected cumulative cost of the RL algorithm π and the expected cumulative cost $\mathcal{V}^*$ of the optimal policy $\pi^* = \arg\min_{\{\pi : \pi(N^h) \leq b, \forall h, N^h\}} \mathcal{V}^{\pi}$. Note that the optimal policy knows all problem parameters and does not change the policy. Thus, there is no switching cost and we drop the round index t. Intuitively, this regret measures the cumulative excess vulnerability exposure incurred by the learning defender relative to an omniscient optimal defense strategy.

Novelties and Challenges: To our knowledge, this work is the first to analyze RL with switching costs that quantify the *magnitude* of policy change rather than merely the frequency of switching. Specifically, the term $\mathcal{S}^h(t, N^h)$ captures the absolute difference between consecutive defense actions, measuring how much the policy changes over time. In contrast, existing RL formulations penalize only whether π_t differs from π_{t-1}, without accounting for the extent of change [2, 10, 26]. Moreover, we address the new challenge arising from the simultaneous presence of switching costs and resource-budget constraints, whose coexistence has not been studied in prior RL literature.

[3] Compared to the fixed allocation in traditional settings, this is a finer-grained setting where the defense action is taken in a more dynamic way.

Algorithm 1 Dynamic Defense Under Resource Constraints and Switching Costs

1: **Parameters:** $\eta = \frac{1}{2H(H+1)}$ and $c > 0$
2: **Initialization:** Q-value functions $\tilde{Q}^h(N, \mu_d) = H$ and $Q^h(N, \mu_d) = \tilde{Q}^h(N, \mu_d)$, state-action visitation counts $\mathcal{N}^h(N, \mu_d) = 0$, where N represents the number of vulnerabilities in the queue, and μ_d represents the defense action
3: **for** $t = 1 : T$ **do**
4: **for** $h = 1 : H$ **do**
5: Take defense action
$$\mu_d^h(t) = \arg\max_{\{\mu_d\}} Q^h(N^h(t), \mu_d)$$
6: Based on the arrivals of vulnerabilities and race condition, the queue state evolves to $N^{h+1}(t)$ according to Eq. (1)
7: Update the defense changing parameter
$$k = \mathcal{N}^h(N^h(t), \mu_d^h(t)) + 1$$
8: Update bonus $\mathcal{B}(k) = c\sqrt{H^3/k}$ for defense exploration
9: Update the estimate-$\tilde{Q}$-value function according to Eq. (8).
10: Update the estimate-$\tilde{V}$-value function
$$\tilde{V}^h(N^h(t)) = \min\left\{H, \max_{\{\mu_d\}} \tilde{Q}^h(N^h(t), \mu_d)\right\}$$
11: **if** $t \in \{t_n\}_{n\geq 1}$ **then**
12: Update the belief-Q-value
$$Q^h(N^h(t), \cdot) = \tilde{Q}^h(N^h(t), \cdot)$$
13: **end if**
14: **end for**
15: **end for**

7.2 Algorithm Design

Our algorithm maintains two Q-value estimates. One is updated continuously to learn from new data, while the other is updated less frequently to determine defense actions and limit switching costs, which is detailed in Algorithm 1. The algorithm outlines the core RL update under delayed policy switching. The algorithm maintains an optimistic Q-estimate, updated periodically according to a geometrically increasing triggering sequence to balance responsiveness and stability. From a high-level point of view, we take the defense action according to an optimistic belief-Q-value function. Specifically, at each step, our algorithm first updates an estimate-$\tilde{Q}$-value function, which represents the value of taking a certain action at a state (line 9). An action with larger estimate-$\tilde{Q}$-value function output is preferred. Intuitively, the belief-Q-value function should be updated more frequently when the sample size is small (i.e., the uncertainty is large), and it should be updated less and less frequently when the sample size becomes larger and larger. To achieve this, a delayed belief-Q-value function is updated when it has not been updated sufficiently long and triggers a switching threshold. Hence, to achieve effective defense with low switching costs, we need to carefully construct an effective belief-Q-value function to guide the defense action and construct an elegant triggering time sequence $\{t_n\}_{n\geq 1}$ for updating the belief-Q-value function (lines 11-12), as well as for changing defense actions. Operationally, this design avoids frequent reconfiguration while still allowing rapid adaptation when uncertainty is high.

Specifically, in Line 7 of Algorithm 1, we first update the number of times the state $N^h(t)$ and defense action $\mu_d^h(t)$ are visited simultaneously, which will generate the bonus term in Line 8. This bonus term essentially captures the level of uncertainty after collecting k samples, such

that according to the concentration inequality (e.g., Hoeffding's inequality), the estimate-$\tilde{Q}$-value function is an optimistic estimate of the optimal true Q-value with high probability. To guarantee this, we update the estimate-$\tilde{Q}$-value function as follows,

$$\begin{aligned}\tilde{Q}^h(N^h(t),\mu_d^h(t)) = (1-\alpha(t))\tilde{Q}^h(N^h(t),\mu_d^h(t)) \\ + \alpha(t)\left[(\bar{C}+b-C(N_l^h(t))+\mu_d^h(t))/(\bar{C}+b)\right. \\ \left.+\tilde{V}^{h+1}(N^{h+1}(t))+\mathcal{B}(k)\right],\end{aligned} \tag{8}$$

where $\bar{C} = \sup\{C(\cdot)\}$. This estimate-$\tilde{Q}$-value is an weighted average between the old estimate-$\tilde{Q}$-value $\tilde{Q}^h(N^h(t),\mu_d^h(t))$ (for exploiting the knowledge learned from historical samples) and the newly learned knowledge from currently visited state-action pair

$$\begin{aligned}\left[(\bar{C}+b-(C(N_l^h(t))+\mu_d^h(t)))/(\bar{C}+b)\right] \\ +\tilde{V}^{h+1}(N^{h+1}(t)),\end{aligned} \tag{9}$$

together with a bonus term $\mathcal{B}(k)$ (for encouraging exploration of potentially better defense strategies).

Finally, lines 11 and 12 determine whether or not to change the belief-Q-value function that will directly determine the defense action $\mu_d(t)$. Let $\tau(i) = \lceil (1+\epsilon)^i \rceil$ for $i = 1, 2, ...$, and define the triggering time sequence as

$$\{t_n\}_{n\geq 1} = [1, \tau(i_0)] \cup \{\tau(i_0+1), \tau(i_0+2), \ldots\}, \tag{10}$$

where ϵ and $i_0 = \left\lceil \frac{\log(10H^2)}{\log(1+\eta)} \right\rceil$ are hyper–parameters chosen by the algorithm. For all $t \in \{1, 2, \ldots\}$, $\tau_{\text{last}}(t) := \max\{t_n : t_n \leq t\}$ and $\alpha(t) = \frac{H+1}{H+t}$. The triggering time sequence (10) allows policy switch every time-slot at the beginning, and then the delay for policy switch keeps exponentially-increasing after a certain amount $\tau(i_0)$ of samples has been collected. For example, the policy switches as follows. Given state $N^h(t)$ at step h, we take some particular defense $\mu_d^h(t)$ for time t, and update both the estimate-$\tilde{Q}$-value and the belief-Q-value immediately. After the time-slot $\tau(i_0)$, we still update $\tilde{Q}$ immediately. However, we only update Q when t is in the triggering time sequence. This exponentially delayed update schedule enables high responsiveness early on and stability as uncertainty decreases, effectively balancing adaptation and switching cost.

7.3 Theoretical Regret Bound: Near-Optimality

We show that the proposed algorithm achieves near-optimal regret with high probability. In particular, with high probability $1-p$, the theoretical regret of our algorithm is upper-bounded by $\tilde{O}(\sqrt{T})$, which is optimal. Recall that the regret is defined to compare our RL performance with the optimal policy, which is an oracle defender with full knowledge of system parameters and no switching penalty.

Theorem 2. (Regret Upper-Bound) With high probability $1-p$, $p \in (0,1)$, the regret of Algorithm 1 is upper-bounded by $\tilde{O}\left(\sqrt{H^3\bar{C}^4bT}\right)$ for any horizon $T = \tilde{\Omega}\left(H^6\bar{C}^2b^2\right)$.

Proof. See Appendix B for the complete proof. Here, we provide a sketch. The proof follows optimism-based analysis for episodic RL, with new developments to handle the delayed defense switching and resource budgets. Let $\tilde{Q}^h(t)$ denote the *estimate-$\tilde{Q}$-value function*, which is continuously updated from new samples, and let $Q^h(t)$ denote the *belief-Q-value function*, a stabilized version used for policy decisions and updated only at triggering times $\{t_n\}_{n\geq 1}$. The proof involves the following key ideas.

(i) Optimism and Concentration: Each $\tilde{Q}^h(t)$ update uses a step size $\alpha(t) = \frac{H+1}{H+t}$ and an exploration bonus $\mathcal{B}(k) = c\sqrt{H^3 \log(1/p')/k}$. By standard concentration arguments (e.g., Azuma–Hoeffding inequality), with probability at least $1 - p'$, the estimate satisfies

$$0 \le \tilde{Q}^h(t) - Q^{*,h} \le \mathcal{B}(k) = \tilde{O}\left(\sqrt{\frac{H^3}{k}}\right),$$

uniformly over all (h, N, μ, t), where $Q^{*,h}$ is the optimal Q value. This ensures *optimism*: the learned Q-values upper bound the true optimal values within $\mathcal{B}(k)$.

(ii) Regret Decomposition: Let $\delta^h(t) = \tilde{V}^h(t) - V^{\pi_t,h}$ denote the instantaneous regret at step h in episode t. Since the policy π_t is derived from the most recently updated $\tilde{Q}^h(k')$ at trigger $k' \le t$, we decompose

$$\delta^h(t) \le |(Q^h(k') - Q^{\pi_t,h})| + |\tilde{Q}^h(k') - Q^h(k')|,$$

where the first term behaves as in standard optimistic RL, while the second term measures the deviation caused by delayed updates.

(iii) Controlling the Delay via Triggering Sequence: Between two triggers, $\tilde{Q}$ evolves according to small step sizes and bounded bonuses. Under the geometrically increasing triggering schedule $t_n = \lceil (1+\epsilon)^n \rceil$, the cumulative deviation $\sum_t |\tilde{Q}^h(k') - Q(^h k')|$ grows at most by a constant factor $1 + O(1/H)$ relative to non-delayed updates. Hence, the delay introduces only a multiplicative $O(1/H)$ overhead.

(iv) Error Propagation over the Horizon: Summing the per-step inequalities and propagating value errors through the horizon yields

$$R^h \;\le\; (1 + O(1/H))R^{h+1} + \tilde{O}(\sqrt{H^3 T}),$$

where $R^h = \sum_t \delta^h(t)$. Unrolling across $h = 1, \ldots, H$ gives $\sum_h R^h = \tilde{O}(\sqrt{H^3 T})$. Including bounded per-step costs $\bar{C}$ and feasible budget b scales the bound to $\tilde{O}(\sqrt{H^3 \bar{C}^4 b T})$.

(v) Switching Costs: The number of belief-Q-value updates, and hence policy changes, is logarithmic in time. Thus, the cumulative switching cost contributes at most $\tilde{O}(\log T)$ to regret, absorbed by the main term.

Combining the above, the total regret then follows.

□

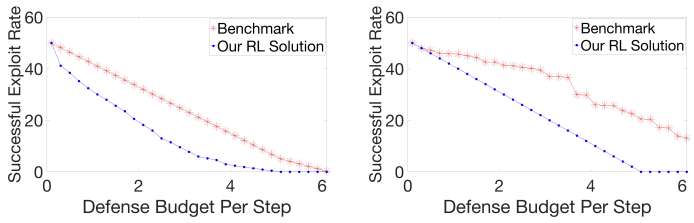
To the best of our knowledge, this is the first regret bound established for dynamic defense under the coexistence of resource-budget constraints and switching costs for amount of changes.

8 Numerical Evaluation

We now evaluate the proposed framework through a series of numerical experiments that combine model-based simulations and data-driven evaluations. These experiments evaluate how the RL defense policy performs under both synthetic and real-world conditions, focusing on attack surface size and exploitation rates. The analysis proceeds in three parts: (i) controlled model-based simulations to verify core dynamics, (ii) trace-driven evaluation using the ARVO dataset, and (iii) aggregate-budget simulations that examine RL reallocation under realistic resource constraints.

8.1 Model-based RL evaluation

We begin with controlled simulations based on the analytical model introduced in Section 4. These simulations evaluate the RL defense policy in a simplified environment where all system parameters are known. The results highlight two main effects: (i) the RL policy's ability to reduce successful exploits compared to fixed allocations, and (ii) its smoothing behavior under nonstationary vulnerability arrivals.



(a) Stochastic arrivals. (b) Adversarial arrivals.

Fig. 8. Successful exploit rate vs. per-step defense budget (patches per unit time) under Model-based RL simulations.

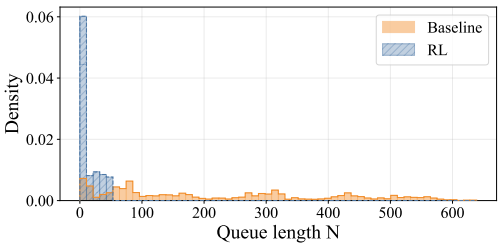


Fig. 9. Trace-driven comparison of queue-length probability densities on ARVO (RL vs. baseline) for per-step budget $b = 1.0$ patches per unit time.

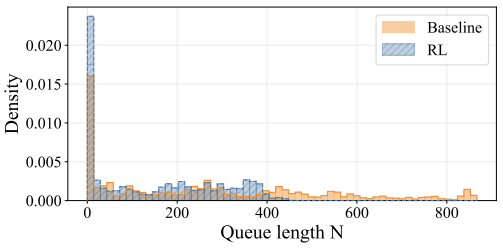


Fig. 10. Queue-length histogram under the empirical aggregate baseline budget and the RL-reallocated policy. Statistics are reported in the text.

Setup: We consider $T = 10^4$ rounds, each with $H = 10$ steps, and use the model $\mu_\ell(t) = 3N(t)$ for exploitation rate in these runs. Vulnerability arrivals are drawn under two regimes: (i) *stochastic* arrivals with rate $\lambda = 5$ at every step, and (ii) *adversarial* arrivals that vary arbitrarily in $[0, 10]$. We compare a fixed defending policy (constant per-step μ_d) to the learned policy produced by our RL procedure, sweeping the per-step defense budget b (measured in patches per unit time) and plotting the resulting successful exploit rates.

Results: Figure 8 shows successful exploit rate versus defense budget for the two arrival regimes. The learned policy reduces successful exploits substantially across budgets and attains up to a 55% reduction for certain budget points in the stochastic regime. In the adversarial regime, the learned policy both reduces the mean exploit rate and smooths high-frequency fluctuations compared to the fixed allocation, reducing both the mean exploit rate and its variability compared to the fixed allocation. The reduction in variability improving predictability of performance, thereby enables an organization to better plan their budget to achieve a specific goal against the attacks.

8.2 Trace-driven RL evaluation

We now evaluate the RL defense policy using a trace-driven simulator built from the ARVO dataset. Vulnerabilities arrive at each time step according to the ARVO records. The empirical per-step defended counts from ARVO define the trace-driven baseline. At each time step, the RL agent observes the current queue length (the number of active vulnerabilities) as the state and selects a defense-rate action. The simulator maps this rate to an integer number of defended vulnerabilities and updates the next state accordingly. We then compare the RL policy against the ARVO baseline.

Setup: We preprocess the ARVO data by binning records into 6-minute intervals; each bin is one time step and ten consecutive bins form an episode (one hour). This produces $T = 64{,}395$ episodes.

Table 1. Trace-driven comparison statistics: queue-length mean & variance (ARVO, RL vs. baseline).

Policy / Budget (patches per unit time)	Mean	Variance
Baseline (data)	219.9	30,930
RL (b=0.5)	59.4	1,602
RL (b=1.0)	13.0	219
RL (b=1.5)	4.7	56
RL (b=2.0)	1.2	6
RL (b=2.5)	0.1	0.4
RL (b=3.0)	0.1	0.3

Here, the defense budget b represents the maximum number of patches that can be applied per unit time. We do not split the data into separate training and test sets; the full trace is used for learning and we observe the result of the learning. For each bin, the number of reported vulnerabilities is used as the stepwise arrival count, and the empirical defended counts in ARVO serve as the baseline defense events. The RL agent observes only the arrivals and selects a defense-rate action; the simulator maps that rate to an integer defended count via a Poisson draw and updates the next state accordingly. The selected rate is converted to an integer defended count by drawing from a Poisson distribution and truncating to the nearest nonnegative integer. At each step, both the baseline and the RL defended counts (together with the arrivals) are applied to update the queue state according to Eq. (1). The ARVO trace does not include separate attacker exploit events, so our trace-driven evaluation considers only vulnerability arrivals and defenses.

The RL action space is $\{0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ and at each step the agent selects a defense-rate action μ from this set. Because the agent is table-based, we set the maximum indexed queue size for the Q-table to $N_{\max} = 300$; if a larger queue is observed the agent still uses the $N_{\max}$ index for action selection, while we record the actual queue length for statistics. We use a per-step budget b (not an episode-level budget) and sweep $b \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ patches per unit time across simulations. As each time step represents a 6-minute interval, a budget of $b = 1.0$ corresponds to an average capacity of 10 patches per hour. The optimistic bonus in the Q-value update is $\mathcal{B}(k) = c_{\text{bonus}}\sqrt{H^3/k}$ with $c_{\text{bonus}} = 0.1$. To ensure a fair comparison, we tune a weight on the RL defense cost so that the RL policy's total defended count over the full trace matches the empirical baseline's total defended count. All other algorithmic settings follow Section 7.2.

Results: Figure 9 shows the queue-length density when per-step budget is $b = 1.0$ patches per unit time. For each budget b, we average results over 5 random seeds and report the queue-length mean and variance for the RL policy and the baseline in Table 1. The RL policy consistently reduces the queue length compared to the baseline, and the gains become larger as the per-step budget increases.

8.3 Aggregate-budget RL reallocation

We next compare the baseline and RL when both use the same aggregate defense budget estimated from the data in Figure 7. Specifically, we ask how queue-length performance changes if the RL policy is allowed to reallocate, over time, the total defense rates used by the baseline.

Setup: Following Figure 7, we segment the ARVO trace into ten regimes and estimate the baseline per-segment defense rates. In this subsection, we keep the ARVO arrivals unchanged and generate the baseline defended counts by drawing Poisson samples with the estimated per-segment rates. We then sum the baseline's defense rates over the full period to obtain its aggregate defense budget. The RL operates as in Section 8.2 but is constrained so that its total defense effort over the full

period does not exceed this aggregate budget. This allows us to evaluate RL reallocation across steps under an equal total defense resource.

Results (representative run): Figure 10 plots the queue-length densities for the baseline (with the estimated defense rates) and for RL (with aggregate-budget reallocation). RL substantially reduces large queue lengths compared to the baseline. The summary statistics are mean$(N) = 146.63$, $N_{95} = 379$, and $N_{99} = 412.2$ for RL. mean$(N) = 267.52$, $N_{95} = 772$, and $N_{99} = 852$ for baseline. Thus, RL reallocation substantially reduces the mean queue length and shrinks the high-percentile tails of the distribution.

9 Discussion and Future Directions

We introduced a spatio-temporal queueing abstraction for the attack surface that models incoming vulnerabilities as arrivals and departures as either successful exploits or successful patches, and used this framework to derive several analytic insights. In particular, we highlight (i) a highly non-linear relationship between defense-resource shortfall and the rate of successful exploits, (ii) the emergence of long-range temporal dependence in the attack surface process when vulnerability lifetimes are heavy-tailed, and (iii) the fact that an aggregate AI-amplification of arrival and exploit rates can increase breach rates even when the attack surface distribution remains qualitatively similar.

While our analysis primarily focuses on a single-component system for clarity, the framework naturally extends to multi-component and multi-organizational settings. An organization's total attack surface can be viewed as a collection of correlated queues, each representing a subsystem such as authentication, storage, or cloud services. At a larger scale, an ecosystem of organizations can be modeled as a network of statistically dependent queueing systems, capturing interdependencies arising from shared software libraries, third-party integrations, or supply-chain relationships. For tractability, we restrict our formal analysis to the single-queue case, which already exhibits the essential dynamics of heavy-tailed persistence, feedback coupling, and resource constraints that characterize real-world attack surface evolution.

Building on these foundations, natural directions for future work include extending the queueing abstraction to **multiple, dependent queues** that reflect component structure; modeling the ecosystem of multiple organizations (and their interactions) as an interconnected queueing system; exploring collaborative defense formulations (e.g., multi-agent approaches) under resource constraints; and expanding empirical data collection to strengthen and validate the modeling assumptions.

10 Conclusion

We introduced a novel queueing-theoretic formulation to model the evolving cyber-attack surface, capturing both its temporal dynamics and spatial structure. Rather than following a narrow perspective that would focus on isolated attack vectors or specific vulnerabilities, our approach provides a holistic framework that reveals how systemic behaviors—such as heavy-tailed patching times—lead to long-range temporal dependencies in vulnerability exposure. The model also accommodates AI-induced amplification effects, allowing us to quantify the spatio-temporal dynamics of the attack surfaces under AI-generated threats and defenses. This dynamic queueing abstraction also lays the foundation for a principled defense allocation strategy, which we cast as a constrained sequential control problem and solve using reinforcement learning. We validated the framework using large-scale open-source vulnerability traces and show that an RL-based adaptive defense policy with near-optimal regret can reduce the mean attack surface size and the high-percentile tails under the same aggregate budget.

References

[1] Sanae Amani, Christos Thrampoulidis, and Lin Yang. 2021. Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning*. PMLR, 243–253.

[2] Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. 2019. Provably efficient q-learning with low switching cost. *Advances in Neural Information Processing Systems* 32 (2019).

[3] Berkay Berabi, Alexey Gronskiy, Veselin Raychev, Gishor Sivanrupan, Victor Chibotaru, and Martin Vechev. 2024. DeepCode AI Fix: Fixing Security Vulnerabilities with Large Language Models. arXiv:2402.13291 [cs.CR]

[4] Richard Fang, Rohan Bindu, Akul Gupta, Qiusi Zhan, and Daniel Kang. 2024. LLM Agents can Autonomously Hack Websites. arXiv:2402.06664 [cs.CR]

[5] Andrew Feutrill, Matthew Roughan, Joshua Ross, and Yuval Yarom. 2020. A queueing solution to reduce delay in processing of disclosed vulnerabilities. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 1–11.

[6] Natarajan Gautam. 2012. Analysis of queues. *CRC Press, LLC, Boca Raton, Florida, United States* 10 (2012), 2222496.

[7] Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. 2024. Coercing LLMs to do and reveal (almost) anything. arXiv:2402.14020 [cs.LG]

[8] Kaushik Haldar and Bimal Kumar Mishra. 2017. Mathematical model on vulnerability characterization and its impact on network epidemics. *International Journal of System Assurance Engineering and Management* 8, 2 (2017), 378–392.

[9] Charles Harry, Ido Sivan-Sevilla, and Mark McDermott. 2025. Measuring the size and severity of the integrated cyber attack surface across US county governments. *Journal of Cybersecurity* 11, 1 (2025), tyae032.

[10] Jiawei Huang, Jinglin Chen, Li Zhao, Tao Qin, Nan Jiang, and Tie-Yan Liu. 2022. Towards deployment-efficient reinforcement learning: Lower bound and optimality. *arXiv preprint arXiv:2202.06450* (2022).

[11] Xin Jin, Charalampos Katsis, Fan Sang, Jiahao Sun, Elisa Bertino, Ramana Rao Kompella, and Ashish Kundu. 2023. Prometheus: Infrastructure Security Posture Analysis with AI-generated Attack Graphs. arXiv:2312.13119 [cs.CR]

[12] Jack A. Jones. 2011. FAIR - Factor Analysis of Information Risk. *Risk Management Insight LLC* (2011). https://www.risklens.com/resources/fair-risk-analysis-model

[13] M. Khosravi-Farmad and A. Ghaemi-Bafghi. 2020. Bayesian Decision Network-Based Security Risk Management Framework. *Journal of Network and Systems Management* 28 (2020), 1794–1819. doi:10.1007/s10922-020-09558-5

[14] Igor Kotenko, Diana Gaifulina, and Igor Zelichenok. 2022. Systematic Literature Review of Security Event Correlation Methods. *IEEE Access* 10 (2022), 43387–43420. doi:10.1109/ACCESS.2022.3168976

[15] Richard Lippmann and Daniel J Fried. 2005. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation. In *Recent Advances in Intrusion Detection*. Springer Berlin Heidelberg, 162–184.

[16] Mingyan Liu. 2021. *Embracing Risk: Cyber Insurance as an Incentive Mechanism for Cybersecurity*. Springer.

[17] Qisi Liu, Liudong Xing, and Chencheng Zhou. 2019. Probabilistic modeling and analysis of sequential cyber-attacks. *Engineering Reports* 1, 4 (2019). doi:10.1002/eng2.12065

[18] Pratyusa K. Manadhata and Jeannette M. Wing. 2011. An Attack Surface Metric. *IEEE Transactions on Software Engineering* 37, 3 (2011), 371–386. doi:10.1109/TSE.2010.60

[19] Xiang Mei, Pulkit Singh Singaria, Jordi Del Castillo, Haoran Xi, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, Hammond Pearce, Brendan Dolan-Gavitt, et al. 2024. Arvo: Atlas of reproducible vulnerabilities for open source software. *arXiv preprint arXiv:2408.02153* (2024).

[20] Daniel Miessler. [n. d.]. AI Agents. Link for the LinkedIn Post. Accessed: 2024-11-07.

[21] Sobhan Miryoosefi and Chi Jin. 2022. A simple reward-free approach to constrained reinforcement learning. In *International Conference on Machine Learning*. PMLR, 15666–15698.

[22] Federal Bureau of Investigation. 2024. FBI Warns of Increasing Threat of Cyber Criminals Utilizing Artificial Intelligence. FBI Posting Link.

[23] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. 2012. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing* 9, 1 (2012), 61–74. doi:10.1109/TDSC.2011.34

[24] Julie J.C.H. Ryan and Scott D. Dexter. 2009. A Bayesian Network Model for Predicting Cyber Security Threats. *Journal of Information Assurance and Security* 4, 2 (2009), 105–114. https://www.mirlabs.org/jias/Volume4_2_2009/Vol4_2.html

[25] Abdulhakim Sabur, Ankur Chowdhary, Dijiang Huang, and Adel Alshamrani. 2022. Toward scalable graph-based security analysis for cloud networks. *Computer Networks* 206 (2022), 108795. doi:10.1016/j.comnet.2022.108795

[26] Ming Shi, Yingbin Liang, and Ness Shroff. 2023. Near-optimal Adversarial Reinforcement Learning with Switching Costs. In *Eleventh International Conference on Learning Representations*.

[27] Ming Shi, Yingbin Liang, and Ness Shroff. 2023. A near-optimal algorithm for safe reinforcement learning under instantaneous hard constraints. In *International Conference on Machine Learning*. PMLR, 31243–31268.

[28] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2024. Large Language Models are Advanced Anonymizers. arXiv:2402.13846 [cs.AI]

[29] Christopher Theisen, Nuthan Munaiah, Mahran Al-Zyoud, Jeffrey C Carver, Andrew Meneely, and Laurie Williams. 2018. Attack surface definitions: A systematic literature review. *Information and Software Technology* 104 (2018), 94–103.

[30] H. Wang, D. Zhang, and S. Jajodia. 2008. An Attack-Graph Based Probabilistic Security Metric. In *IFIP Data and Applications Security '08*. Springer, 109–124. https://link.springer.com/chapter/10.1007/978-0-387-09699-5_8

[31] Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. GradSafe: Detecting Unsafe Prompts for LLMs via Safety-Critical Gradient Analysis. arXiv:2402.13494 [cs.CL]

[32] Qi Zhang, Chunjie Zhou, Yu-Chu Tian, Naixue Xiong, Yuanqing Qin, and Bowen Hu. 2018. A Fuzzy Probability Bayesian Network Approach for Dynamic Cybersecurity Risk Assessment in Industrial Control Systems. *IEEE Transactions on Industrial Informatics* 14, 6 (2018), 2497–2506. doi:10.1109/TII.2017.2768998

A Proof of Theorem 1

Let us define the indicator variable:

$$I_i(t) \triangleq \begin{cases} 1, & \text{a vulnerability arrives in } ((i-1)\delta, i\delta) \\ & \quad \text{and still in the system at time } t \\ 0, & \text{otherwise} \end{cases}.$$

for δ being the amount of temporal increment. Therefore,

$$N(t) = \sum_{i=-\infty}^{t/\delta} I_i(t). \tag{11}$$

For any t, τ such that $t \leqslant \tau$,

$$\begin{aligned} \mathbb{E}\left[N(t)N(\tau)\right] &= \mathbb{E}\left[\sum_{i=-\infty}^{t/\delta}\sum_{j=-\infty}^{\tau/\delta} I_i(t)I_j(\tau)\right] \\ &= \sum_{\{i\leqslant t/\delta, j\leqslant\tau/\delta \mid i\neq j\}} \mathbb{E}\left[I_i(t)I_j(\tau)\right] + \sum_{i=-\infty}^{t/\delta} \mathbb{E}\left[I_i(t)I_i(\tau)\right] \\ &= \sum_{\{i\leqslant t/\delta, j\leqslant\tau/\delta \mid i\neq j\}} \mathbb{E}\left[I_i(t)\right]\mathbb{E}\left[I_j(\tau)\right] + \sum_{i=-\infty}^{t/\delta} \mathbb{E}\left[I_i(t)I_i(\tau)\right], \end{aligned} \tag{12}$$

where Eq. (12) follows since $I_i(t)$ and $I_j(\tau)$ are independent for $i \neq j$. We can also write

$$\begin{aligned} \mathbb{E}\left[N(t)\right]\mathbb{E}\left[N(\tau)\right] &= \mathbb{E}\left[\sum_{i=-\infty}^{t/\delta} I_i(t)\right]\mathbb{E}\left[\sum_{j=-\infty}^{\tau/\delta} I_j(\tau)\right] \\ &= \sum_{\{i\leqslant t/\delta, j\leqslant\tau/\delta \mid i\neq j\}} \mathbb{E}\left[I_i(t)\right]\mathbb{E}\left[I_j(\tau)\right] \\ &\quad + \sum_{i=-\infty}^{t/\delta} \mathbb{E}\left[I_i(t)\right]\mathbb{E}\left[I_i(\tau)\right]. \end{aligned} \tag{13}$$

Combining Eq. (12) and (13) we get,

$$\begin{aligned} \operatorname{cov}\left(N(t), N(\tau)\right) &= \mathbb{E}\left[N(t)N(\tau)\right] - \mathbb{E}\left[N(t)\right]\mathbb{E}\left[N(\tau)\right] \\ &= \sum_{i=-\infty}^{t/\delta} \left\{\mathbb{E}\left[I_i(t)I_i(\tau)\right] - \mathbb{E}\left[I_i(t)\right]\mathbb{E}\left[I_i(\tau)\right]\right\} \qquad (14) \\ &= \sum_{i=-\infty}^{t/\delta} \left\{\lambda\delta[1 - F(\tau - i\delta)] - o(\delta)\right\}, \qquad (15) \end{aligned}$$

where the first term in (15) is due to $\mathbb{E}\left[I_i(t)I_i(\tau)\right]$ and the $o(\delta)$ is due to $\mathbb{E}\left[I_i(t)\right]\mathbb{E}\left[I_i(\tau)\right]$ and that the probability of two vulnerabilities in the same instant is $o(\delta)$. As $\delta \to 0^+$,

$$\operatorname{cov}\left(N(t), N(\tau)\right) = \int_{-\infty}^{t} \lambda\left[1 - F(\tau - s)\right] ds = \lambda \int_{|\tau - t|}^{\infty} \left[1 - F(s)\right] ds. \tag{16}$$

Since the above covariance is a function of the time difference $t - \tau$ only, $\{N(t)\}$ is a stationary process. Thus, for heavy-tailed $F(s)$, the autocovariance function decays slower than $(\tau - t)^2$, leading to a long-range dependent attack surface.

B Proof of Theorem 2

Proof. For readability, we remove some indices when the context is clear. Recall that $\tilde{Q}^h(t)$ denotes the *estimate-Q-value function*, continuously updated from new samples at step h and time t, while $Q^h(t)$ denotes the *belief-Q-value function*, a stabilized version used to determine the defense action and updated only at triggering times $\{t_n\}_{n\geq 1}$.

Step 1: Instantaneous regret decomposition: Let $\tilde{\delta}^h(t)$ be the instantaneous regret at step h and time t. Following the standard optimistic decomposition,

$$\begin{aligned}\tilde{\delta}^h(t) &= \Big(\max\left\{\tilde{Q}^h(t_\uparrow),\, \tilde{Q}^h(t)\right\} - Q^{*,h}\Big)\big(N^h(t), \mu_d^h(t)\big) \\ &\leq |\tilde{Q}^h(t_\uparrow) - Q^{*,h}| + |\tilde{Q}^h(t) - \tilde{Q}^h(t_\uparrow)|,\end{aligned} \tag{17}$$

where $t_\uparrow = \tau_{\text{last}}(t) + 1$ denotes the most recent triggering time before t. The first term corresponds to the standard value-estimation error, and the second term represents the additional deviation introduced by the delayed belief update.

Step 2: Recursive relation for $\tilde{Q}^h(t)$: The update of the estimate-$g\tilde{Q}$-value at visit t can be expanded as

$$\begin{aligned}&\tilde{Q}_t^h(N, \mu_d) - Q^{\pi,h}(N, \mu_d) \\ &= \alpha(t_i)\Big(H - Q^{\pi,h}(N, \mu_d)\Big) + \sum_{i=1}^{k} \alpha(t_i)\Big[C^h(N, \mu_d) \\ &\quad + \tilde{V}_{t_i}^{h+1}\big(N_d^{h+1}(t_i)\big) - V^{\pi,h+1}\big(N^{h+1}(t_i)\big) \\ &\quad + \big(\widehat{P}_{t_i}^h - P^h\big)V^{\pi,h+1}(N, \mu_d) + \mathcal{B}(t_i)\Big],\end{aligned} \tag{18}$$

where $\alpha(t_i)$ is the step size, $B(t_i)$ is the exploration bonus, and $\widehat{P}_{t_i}^h$ is the empirical transition model. This expression separates the stochastic update noise, transition deviation, and optimism term.

Step 3: Bounding the delayed perturbation: For any $t > t_\uparrow$, the cumulative drift between two consecutive triggers satisfies

$$\begin{aligned}&|\tilde{Q}^h(t) - \tilde{Q}^h(t_\uparrow)|\big(N^h(t), \mu_d^h(t)\big) \\ &\leq \phi_k + \sum_{i=\tau_{\text{last}}(t)+1}^{t} \alpha(t)\, \tilde{\zeta}_{t_i}^{h+1} + \bar{\zeta}_t^h,\end{aligned} \tag{19}$$

where $\phi_k = O\big(\sqrt{H^3/k}\big)$ and $\bar{\zeta}_t^h = O\big(\sqrt{H^3/t}\big)$ hold uniformly with high probability. Both terms can be absorbed into a constant multiple of ϕ_k, since $\phi_{t_\uparrow} \leq (1 + O(1/H))\phi_k$ under the geometric triggering rule.

Step 4: Coefficient aggregation: When summing Eq. (19) over all time steps, each successor-state error $\tilde{\zeta}_{t_\uparrow}^{h+1}$ is weighted by the accumulated step-size coefficients. Using the triggering sequence $t_n = \lceil (1+\varepsilon)^n \rceil$ with $\varepsilon = \frac{1}{2H(H+1)}$ and initial index $r_0 = \lceil \frac{\log(10H^2)}{\log(1+\varepsilon)} \rceil$, we obtain

$$\sum_t \Big(\mathbf{1}\{\text{no trigger at } t\}\, \alpha(\tau_{\text{last}}(t)) + \mathbf{1}\{\text{trigger at } t\}\, \alpha(t) \Big) \leq 1 + \tfrac{3}{H}, \tag{20}$$

showing that the delay inflates the propagation factor by at most $(1 + 3/H)$.

Step 5: Regret recursion and final bound: Let $R^h = \sum_t \tilde{\delta}^h(t)$. Combining the above results yields

$$R^h \leq (1 + O(1/H)) R^{h+1} + \tilde{O}(\sqrt{H^3 T}).$$

Unrolling the recursion over $h = 1, \ldots, H$ gives $\sum_{h=1}^{H} R^h = \tilde{O}(\sqrt{H^3 T})$. Accounting for the bounded per-step cost $\bar{C}$ and defense-cap budget b scales the bound to $\tilde{O}(\sqrt{H^3 \bar{C}^4 b T})$. The number of belief-$Q$-value updates, and hence policy changes, is logarithmic in time. Thus, the cumulative switching cost contributes at most $\tilde{O}(\log T)$ to regret, absorbed by the main term.

□