Rational Verification in Repeated Security Games

Surasak Phetmanee[1][0000−0002−8913−1124], Michele Sevegnani[1][0000−0001−6773−9481], and Oana Andrei[1][0000−0002−1306−0219]

School of Computing Science, University of Glasgow, UK
{surasak.phetmanee, michele.sevegnani, oana.andrei}@glasgow.ac.uk

Abstract. Cyber attackers often engage in repeated adaptive attacks, while many existing defensive models are static and lack mechanisms for long-term strategy validation. We introduce a rational verification framework for Repeated Stackelberg Security Games that evaluates the ongoing optimality of defender strategies under rational attacker behaviour. Our framework incorporates discounted payoffs to emphasise early-stage threats and dynamically adjusts strategies in response to evolving conditions. Experimental results show that our approach improves the utility of the defender and supports an effective resource allocation.

Keywords: Rational Verification · Repeated Games · Security Games · Stackelberg Equilibrium

1 Introduction

Cybersecurity defences today often rely on fixed policies or periodically updated strategies that fail to anticipate how attackers adapt over time. In practice, adversaries frequently reuse or evolve their attack techniques in response to visible changes in the defender's stance [9]. For example, when a zero-day exploit is patched, threat actors often pivot quickly to new attack vectors or repackage payloads to bypass mitigations. These security challenges are linked to safety [2]. For example, a maliciously compromised system component can lead to unsafe physical behaviour, so formal security guarantees are crucial to overall safety.

Stackelberg Security Games (SSGs) [20] have been widely used to model such scenarios, where the defender commits to a strategy and the attacker responds optimally. However, SSGs generally do not provide a formal mechanism to assess whether a strategy remains effective as attacker behaviour changes over time [18]. These evolving scenarios can be formalised using repeated games [12]. Rational verification then offers an effective approach in this context; it extends classical model checking to reasoning about equilibrium strategies in multiagent systems [22]. This technique can be applied to improve threat modelling in adaptive adversary scenarios.

In this paper we introduce Repeated Stackelberg Security Games (RSSGs) and rational verification for RSSGs as a formal framework for modelling and verifying defensive strategies over multiple rounds under Stackelberg equilibrium. Our work contributes to research on game-theoretic models for cybersecurity [1,5,13].

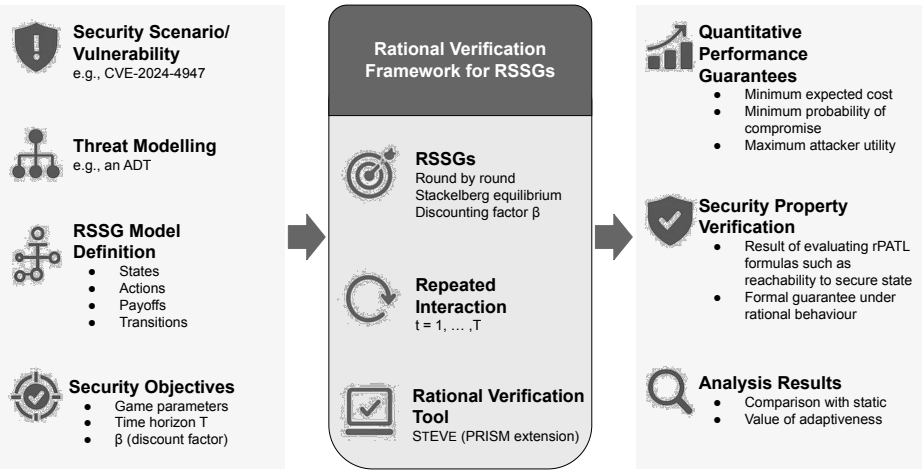

Fig. 1. Overview of the rational verification framework for RSSGs.

In each round, we check whether the defender's strategy remains optimal against a rational attacker. We extend the StEVe tool (v1.0) [15], originally based on the PRISM-games [11] extension, to a new version (v2.0) for round by round Stackelberg equilibrium [19] verification. StEVe is central to our rational verification framework for RSSGs. Figure 1 illustrates its workflow: how inputs such as the security scenario, threat model, and RSSG parameters are processed through a core RSSG model that captures repeated interactions and produces quantitative performance guarantees, formal verification of security properties, and analysis results comparing static and adaptive defender strategies, highlighting the value of adaptiveness through improvements in cost, risk, and response time.

2 Background

2.1 Stackelberg Security Games and Repeated Games

Defenders commit to strategies anticipating informed attacker responses [16]. Informally, a Stackelberg game models a strategic interaction where a defender (the 'leader') first commits to a defensive strategy. An attacker (the 'follower') then observes this strategy and chooses their own best possible response to maximise their utility. An *extensive form game* [12] is a tuple $\mathcal{G} = (N, A, H, Z, \chi, \pi, \gamma, u)$, where $N = \{1, \ldots, n\}$ is the set of players, A the set of actions, H and Z the nonterminal and terminal nodes (with $H \cap Z = \varnothing$), $\chi : H \to 2^A$ the available actions at each nonterminal node, $\pi : H \to N$ the player function, $\gamma : H \times A \to H \cup Z$ the successor function, and $u = (u_1, \ldots, u_n)$ the utility functions, where each $u_i : Z \to \mathbb{R}_{\geq 0}$ maps terminal nodes to non-negative real-valued payoffs. This extensive form captures the sequential nature of Stackelberg games.

A *Stackelberg Security Game (SSG)* is a tuple $(\mathcal{G}, AP, L)$ where: $\mathcal{G}$ is a two-player extensive form game, with $N = \{1, 2\}$ representing the defender (leader) and attacker (follower) respectively, AP is a set of atomic propositions representing system properties, $L : H \cup Z \to 2^{AP}$ is a labelling function assigning propositions to nodes.

Let Σ_1 and Σ_2 denote the sets of all strategies available to the defender and attacker, respectively, where a strategy $\sigma_i \in \Sigma_i$ is a function that selects an action for player i at each nonterminal node. We allow these strategies to be either deterministic or mixed (probabilistic). That is, a strategy σ_i may select a specific action or a probability distribution over the set of available actions A_i. Formally, this is defined as $\sigma_i : H_i \to \Delta(A_i)$, where $\Delta(A_i)$ denotes the set of discrete probability distributions over A_i. This generalisation enables strategies to be expressed as stochastic mappings, as formalised in Section 3.2.

Let $u_i : \Sigma_1 \times \Sigma_2 \to \mathbb{R}_{\geq 0}$ be the utility function for player i, where $u_1(\sigma_1, \sigma_2)$ is the defender's payoff and $u_2(\sigma_1, \sigma_2)$ is the attacker's payoff. This function represents the expected utility for player i, derived from the fundamental payoffs at the terminal nodes of the game. Any given strategy profile (σ_1, σ_2) induces a probability distribution over the terminal nodes, and $u_i(\sigma_1, \sigma_2)$ is the resulting expected value. The set of *best responses* available to the attacker against a defender strategy σ_1 is defined as:

$$\mathrm{BR}(\sigma_1) = \{\sigma_2 \in \Sigma_2 \mid u_2(\sigma_1, \sigma_2) \geq u_2(\sigma_1, \sigma_2') \text{ for all } \sigma_2' \in \Sigma_2\}.$$

A *Stackelberg equilibrium* [20] is a strategy profile (σ_1^*, σ_2^*) such that:

$$\sigma_1^* = \arg\max_{\sigma_1 \in \Sigma_1} u_1(\sigma_1, \mathrm{BR}(\sigma_1)), \quad \text{and} \quad \sigma_2^* \in \mathrm{BR}(\sigma_1^*).$$

Here, the defender commits to a strategy σ_1^* that maximises their utility assuming the attacker responds optimally, and the attacker chooses a best response σ_2^* to that committed strategy. SSGs are one-shot models, with the game ending after a single attacker response, which makes it difficult to model persistent and adaptive threats, where attackers return with potentially new strategies after observing defence actions. In such scenarios, a defence that was once optimal, i.e., a strategy that maximised the expected utility of the defender under Stackelberg equilibrium assumptions, may no longer remain effective as the attacker's behaviour evolves. This limitation motivates the need for models that support repeated interactions.

A *repeated game* [12] is one in which a base game is played multiple times by the same players. The base game can either be played for a finite number of rounds $T \geq 1$ producing a finitely repeated game, or played indefinitely producing an infinitely repeated game. In such settings, each round yields a *reward* r_i^j to player i, which corresponds to the payoff received in round j. A *discounted utility* assigns decreasing weight to future rewards using the formula $\sum_{j=1}^{T} \beta^j r_i^j$, where $0 \leq \beta^j \leq 1$ is a discount factor. Since $\beta^j > \beta^{j+1}$ for all j when $\beta^j < 1$, future rewards are weighted less heavily than immediate ones. In repeated games, the utility for player i accumulates across rounds based on the payoff received in each round. This allows a player's payoff to reflect the entire history of the game.

2.2 Rational Verification

Rational verification [22] is a formal analysis technique that asks whether a given property ϕ formalised in Probabilistic Alternating-time Temporal Logic (rPATL) [4] holds under the assumption that all agents in the system behave rationally. Each agent selects a strategy that maximises their own utility, taking into account the strategies of others. In our framework, we use a modified form of the temporal logic rPATL introduced in our previous work [15]. Formally, given a game $\mathcal{G}$ and an rPATL property ϕ, rational verification checks whether ϕ holds when players follow some or all equilibrium strategies in $\mathcal{G}$, such as Stackelberg equilibria. We verify the ongoing optimality of a defender's strategy under the Stackelberg assumption and expressed it in rPATL as the formula:

$$\langle\langle C\rangle\rangle \mathsf{R}^{r}_{\mathsf{SE}=?}[\,\mathsf{F}\phi\,] \stackrel{def}{=} \sup_{\sigma_1 \in \Sigma_1} \sup_{\sigma_2 \in BR(\sigma_1)} \mathbb{E}^{\sigma_1,\sigma_2}[\,\mathsf{F}\phi\,]$$

Here, $\langle\langle C\rangle\rangle$ refers to the coalition of the two players, in our case, the defender and the attacker; $\mathsf{R}^{r}_{\mathsf{SE}=?}$ is the reward operator under the Stackelberg equilibrium; F is the temporal operator specifying a reachability goal; ϕ denotes a desirable system property, for example *the system not being compromised*. Therefore, the formula evaluates the expected cumulative reward for the defender to eventually reach a state where ϕ holds, assuming rational attacker behaviour. A state in the RSSG model summarises the history up to the current round such as which mitigations have been applied, which attacker steps have succeeded, and whether the system is vulnerable or compromised. In addition, σ_1 is the defender's strategy and σ_2 the attacker's best response. Defenders must evaluate whether their strategies remain effective when facing adaptive and strategic attackers.

2.3 Threat Modelling

Threat modelling [17] is a structured process for identifying, enumerating, and prioritising potential threats. In practice [7,9,21], we can use vulnerability information to formalise threat models such as attack trees or attack defence trees. We use Attack Defence Trees (ADTs) [8] – a well-known graphical model used to represent security threats and countermeasures. ADTs are a restricted form of attack defence modelling in which all attacker and defender actions appear at a single level beneath the root. This restriction is suitable for our modelling purposes, as it focuses on immediate countermeasures rather than complex defence in depth structures, and it simplifies the translation to game states. The syntax consists of four ADTerm patterns: $\mathtt{cd}(b', b)$ $\mathtt{cd}(b', f(b_1, \ldots, b_k))$ $\mathtt{cd}(f'(b'_1, \ldots, b'_{k'}), b)$ $\mathtt{cd}(f'(b'_1, \ldots, b'_{k'}), f(b_1, \ldots, b_k))$ where $b'_j \in B_d$ represent individual defender actions, $b_i \in B_a$ denote attacker actions, and $f' \in \{\vee^d, \wedge^d\}$ and $f \in \{\vee^a, \wedge^a\}$ are Boolean operators used to model disjunctive or conjunctive strategies for defenders and attackers respectively. This formal syntax allows us to precisely capture the strategic relationship between defender mitigations and attacker objectives. We use this structure to model the CVE-2024-4947 vulnerability as a concrete example in our framework.

Example 1. CVE-2024-4947 [14] is a type confusion vulnerability in Chrome's V8 JavaScript engine, allowing remote code execution via crafted web pages. Hacker groups have exploited this flaw by deploying deceptive sites to trick users into visiting malicious content. In the ADT, the defender may apply mitigation actions such as updating Chrome to version 125.0.6422.60 (`PatchChrome`), blocking known malicious domains (`BlockMaliciousSites`), or enabling enhanced script monitoring (`EnableEnhancedMonitoring`); we denote these actions by b'_1, b'_2, and b'_3, respectively. The attacker must complete a multi-step process: first, luring the victim to a malicious site (`AttemptPhishing`, b_1); second, exploiting the V8 vulnerability upon visit (`AttemptExploitV8`, b_2); and finally, executing a malicious payload to achieve code execution (`ExecutePayload`, b_3).

This scenario is expressed as the ADTerm $\texttt{cd}(\vee^d(b'_1, b'_2, b'_3), \wedge^a(b_1, b_2, b_3))$, where the attacker must complete all three steps – phishing, exploiting, and payload execution – to compromise the system, while the defender can prevent the attack by applying any one of the mitigation actions. Although $\wedge^a$ typically denotes conjunction without sequence, in this case the steps are inherently sequential due to technical dependencies.

In our RSSG model, these actions are mapped directly from the ADT: defender actions `PatchChrome`, `BlockMaliciousSites`, and `EnableEnhancedMonitoring` correspond to b'_1, b'_2, b'_3, with an additional `DoNothing` option. The attacker actions comprise `AttemptPhishing`, `AttemptExploitV8`, and `ExecutePayload`, which map to b_1, b_2, b_3, along with `Wait` as a passive choice. This alignment ensures consistency between the threat model and the repeated game (see Table 1).

The ADT model is translated into the RSSG. The defender and attacker actions identified in the ADT such as `PatchChrome` and `AttemptExploitV8` directly define the action sets A_1 and A_2 used in the game. The logical structure of the ADT, which dictates the necessary steps for a successful compromise, informs the state space such as `Vulnerable_Unpatched`, `Attacker_Exploiting`, `Compromised`, and the transition probabilities between them. Finally, the security outcomes are quantified as reward (or cost) structures. For example, a successful attack corresponds to a significant negative effect for the defender (`incident_cost` in Table 2), while defender actions have their own associated costs (`defence_cost`).

3 Repeated SSGs and Rational Verification

3.1 Modelling and Equilibrium in RSSGs

An RSSG extends the one-shot SSG $(\mathcal{G}, AP, L)$ by allowing the game $\mathcal{G}$ (the stage game) to be played over a finite sequence of T rounds. Although the original extensive form game is defined over nodes $H \cup Z$, the repeated game requires a richer notion of state. We therefore define a derived state space S, where each state $s_t \in S$ captures contextual information at round t, such as past actions and system conditions. Unlike nodes in $H \cup Z$, which represent single decision points or outcomes, states in S summarise the evolving game history and support reasoning about strategy adaptation.

Table 1. States and Actions

Category	Details / Variables
States	Key states capture security posture and attacker progress: `Initial`, `Vulnerable_Unpatched`, `Attacker_Phishing`, `Attacker_Exploiting`, `Compromised`, `Patched`, `SiteBlocked`, `EnhancedMonitoringActive`, `Game_Over` (terminal).
Defender Actions (A_1)	Available actions for the defender: `DoNothing`, `PatchChrome`, `BlockMaliciousSites`, `EnableEnhancedMonitoring`.
Attacker Actions (A_2)	Available actions for the rational attacker (follower): `Wait`, `AttemptPhishing`, `AttemptExploitV8`, `ExecutePayload`.

The RSSG framework is stateful unlike repeated one-shot SSGs. The optimal strategy at round t depends on the current state s_t, reflecting past interactions. This enables adaptive defence against evolving attacks, where static strategies are insufficient. This example is a simplified two rounds ($T = 2$) of our CVE-2024-4947 case study, using the parameters defined in Table 1 and Table 2. The defender chooses between `DoNothing` (cost 0) or `BlockMaliciousSites` (cost -3), and an attacker who must first `AttemptPhishing` and then `AttemptExploitV8` to succeed. A successful compromise costs the defender -200, and the probabilities of phishing and exploit success are 0.4 and 0.75, respectively. In a single round game, the attacker can only phish, so the risk of compromise is zero. The defender's optimal one-shot strategy is therefore `DoNothing` to avoid the certain cost of -3 from blocking sites. The static and dynamic strategies are presented below.

- Static Strategy: The defender plays `DoNothing` in both rounds. The attacker's only path to victory is a successful phish in Round 1 followed by a successful exploit in Round 2. The probability of this sequence is $0.4 \times 0.75 = 0.3$. The expected utility is:
$$U_{\text{static}} = 0.3 \times (-200) = -60$$
- Dynamic Strategy: The defender plays `DoNothing` in Round 1. In Round 2, their action depends on the state. If the phish succeeded in Round 1 (a 40% probability), the defender plays `BlockMaliciousSites` at a cost of -3 to prevent the possible exploit. If the phish failed (a 60% probability), they again do nothing. The expected utility is:
$$U_{\text{dynamic}} = \underbrace{(0.6 \times 0)}_{\text{Phish fails}} + \underbrace{(0.4 \times -3)}_{\text{Phish succeeds, so Block}} = -1.2$$

Round by Round Stackelberg Equilibrium. During each round $t \in \{1, \ldots, T\}$ of an RSSG, the defender (leader) chooses an action a_1^t, based on the current state s_t which encapsulates the history. The attacker (follower) observes a_1^t and chooses a best response action a_2^t to maximise their own utility. We focus on verifying properties under the assumption that the attacker plays a best response in every round, given the defender's action and the current state. To reason about utility in a repeated setting, we extend the notion from one-shot games.

Table 2. Transition Probabilities and Payoffs

Element	Value / Description
Transition Probabilities	– Patch Success Probability: 0.98 per round (if `PatchChrome` chosen). – Phishing Success Probability: 0.4 (if `Vulnerable_Unpatched` and attacker chooses `AttemptPhishing`). – Exploit Success Probability: 0.75 (if phishing successful and attacker chooses `AttemptExploitV8`). – Site Blocking Effectiveness (prevents phishing): 0.9 (if `BlockMaliciousSites` chosen). – Compromise occurs if `ExecutePayload` follows successful exploitation. – `Game_Over` reached after $T = 10$ rounds or upon compromise.
Defender Rewards/Costs	Defined via reward structures: – "incident_cost": -200 (upon reaching `Compromised`). – "defence_cost" (per round): `PatchChrome` $= -5$, `BlockMaliciousSites` $= -3$, `EnableEnhancedMonitoring` $= -2$. – "total_cost": Sum of "incident_cost" and accumulated "defence_cost". – "steps": Accumulates 1 per round.
Attacker Utility	Defined via "attacker_utility" structure: – Success: $+100$ (upon reaching `Compromised`). – Failed Attempt: -1 (for failed `Phishing` or `Exploit`).

In the extensive form model, the utility function $u_i : Z \to \mathbb{R}_{\geq 0}$ assigns payoffs to terminal nodes. We define $U_i(s_t, a_1^t, a_2^t)$ as the expected cumulative utility for player i starting from state s_t, given the actions a_1^t and a_2^t taken in round t, and assuming equilibrium strategies are followed thereafter. This generalises the terminal node utility to account for accumulated outcomes across multiple rounds. We also define a reward function $R_i : S \times A_1 \times A_2 \to \mathbb{R}_{\geq 0}$, where $R_i(s_t, a_1^t, a_2^t)$ gives the immediate reward received by player i in round t, based on the current state s_t, the defender's action a_1^t, and the attacker's action a_2^t. These per round rewards form the basis for computing expected cumulative utility. To model temporal preferences, we adopt the notion of discounted utility. Given a discount factor $\beta \in [0, 1]$, the total discounted utility for player i over horizon T is:

$$R_i^T = \sum_{t=1}^{T} \beta^{t-1} R_i(s_t, a_1^t, a_2^t)$$

This prioritises earlier rewards when $\beta < 1$, which reflects practical urgency in security contexts. Formally, a strategy profile $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_i = (\sigma_i^1, \ldots, \sigma_i^T)$ dictates the actions a_i^t chosen by player i in round t, constitutes *a round by round Stackelberg equilibrium* if for every round $t \in \{1, \ldots, T\}$:

1. The attacker's action a_2^t maximises their expected utility, given the defender's action a_1^t and the current state s_t:

   $$a_2^t \in \arg\max_{a \in A_2(s_t)} \mathbb{E}[U_2(s_t, a_1^t, a)]$$

   where $A_2(s_t)$ are the attacker's available actions in state s_t, and U_2 represents the attacker's expected utility function which might consider immediate reward $R_2(s_t, a_1^t, a)$ or future discounted rewards.

2. The defender's action a_1^t is chosen as part of a strategy σ_1 that maximises their total expected discounted utility over the horizon T, assuming the attacker plays a best response strategy $\sigma_2^* \in BR(\sigma_1)$, where $BR(\sigma_1)$ denotes the set of best responses to σ_1. Formally:

$$\sigma_1 \in \arg\max_{\sigma_1'} \mathbb{E}_{\sigma_1', \sigma_2^*}\left[R_1^T\right] \quad \text{where} \quad R_1^T = \sum_{t=1}^{T} \beta^{t-1} R_1(s_t, a_1^t, a_2^t)$$

Here, $R_1(s_t, a_1^t, a_2^t)$ is the immediate reward for the defender in round t; (s_t, a_1^t, a_2^t) belongs to the sequence of states and actions resulting from the strategy profile (σ_1', σ_2^*), where σ_1' is a candidate defender strategy and $\sigma_2^* \in BR(\sigma_1')$; and $\beta \in [0, 1]$ is the discount factor.

This definition implies that the defender commits to a sequence of actions determining actions based on state that is optimal over the horizon T, under the constraint that the attacker will rationally counter the defender's action within each round.

The Role of Discounting (β). The discount factor $\beta \in [0, 1]$ is critical as it models the defender's temporal preferences. One option is *exponential decay* for $\beta < 1$ which prioritises near-term outcomes; future rewards are valued less by β^{t-1}, reflecting urgency, uncertainty, or the time value of security investments. The expression β^{t-1} means that a reward received in round t is multiplied by β^{t-1}, so rewards later in time are given less weight. This is common in cybersecurity where immediate threat mitigation is often paramount. Another option is *no discounting* for $\beta = 1$, when all rounds are valued equally. In this case the total payoff for each player i is the sum of rewards $R_i^T = \sum_{t=1}^{T} R_i(s_t, a_1^t, a_2^t)$. It is suitable for objectives focused on average performance over the fixed horizon T.

Computational Verification and Stability. Verifying if a given strategy profile constitutes a round by round Stackelberg equilibrium, or synthesising the optimal defender strategy, typically involves methods like backward induction or value iteration [12] adapted for finite repeated Stackelberg games. These methods compute the expected cumulative discounted rewards and identify the optimal actions at each decision point.

3.2 Strategy Synthesis

Our framework supports reasoning about and synthesising defender strategies within the RSSG context. The goal is to find a defender strategy σ_1 that maximises the total expected discounted utility R_1^T under the round by round assumption.

The *optimal static strategy* corresponds to finding a single strategy $\mu_1 : S \to \Delta(A_1)$, where μ_1 maps each state $s \in S$ to a probability distribution over defender actions. Here, S is the set of states in the repeated game, and $\Delta(A_1)$ denotes the set of probability distributions over the defender's action set A_1. The same strategy μ_1 is applied in every round $t = 1, \ldots, T$, and the defender's expected utility R_1^T is evaluated assuming that, in each round, the attacker selects a best

Table 3. Notation summary for strategy types in RSSGs.

Symbol	Description
Σ_1, Σ_2	Strategy sets for the defender (1) and attacker (2)
$\sigma_1 \in \Sigma_1$	Defender's strategy across all T rounds
σ_1^t	Defender's action (or decision rule) in round t
$\mu_1 : S \to \Delta(A_1)$	Static Strategy: maps states to action distributions, reused each round
$\sigma_2^* \in BR(\sigma_1)$	Best response strategy of the attacker to σ_1

response to the action chosen according to $\mu_1(s_t)$. While the strategy itself is fixed, its performance reflects the full sequence of interactions.

The *optimal dynamic strategy* corresponds to finding an optimal sequence of actions $\sigma_1 = (\sigma_1^1, \ldots, \sigma_1^T)$, where the action σ_1^t applied in round t is determined by optimising the remaining discounted utility from round t onwards, depending on the round t and the current state s_t. Table 3 summarises the notation used for defender and attacker strategies in the repeated game.

3.3 Formalising and Verifying Security Properties with rPATL

Rational verification allows us to check if desired temporal properties hold under the assumption that players adhere to an equilibrium concept. We use rPATL syntax [4], assuming the defender maximises expected discounted utility over T rounds, while the attacker plays a best response in each round.

In our framework, rPATL formulae include state formulae (ϕ), path formulae (ψ), and reward path formulae (ρ), but their semantics are interpreted under the round by round Stackelberg equilibrium assumption. Temporal operators relevant to our analysis include both reward and probability expressed in rPATL.

The operator $\langle\langle D\rangle\rangle \mathsf{R}^r_{\mathsf{SE}=?}[\rho]$ denotes the expected cumulative reward that coalition D can guarantee under Stackelberg equilibrium, for a given reward structure r and reward path formula ρ. In our setting, the coalition is always $D = \{1\}$, representing the defender (player 1). The formula ρ typically takes the form $\mathsf{F}\phi$, meaning that the reward is accumulated along a path until a state satisfying ϕ is reached.

The operator $\langle\langle D\rangle\rangle \mathsf{P}_{\sim p}[\psi]$ expresses whether the defender can ensure that the probability of satisfying path formula ψ meets a bound $\sim p$, where $\sim \in \{<, \leq, \geq, >\}$. For example, $\mathsf{P}_{\geq 0.9}$ asks whether the probability is at least 90%. Valid path formulae ψ include temporal operators such as $\mathsf{X}\phi$ (next state), or $\phi_1 \, \mathsf{U}^{\leq T} \, \phi_2$ (bounded until within T steps). The query $\mathsf{P}_{=?}[\psi]$ computes the optimal probability the defender can guarantee.

In our rational verification framework, these queries are interpreted under Stackelberg equilibrium. That is, the defender selects a strategy to maximise their outcome, while the attacker responds rationally in each round. This equilibrium constraint is automatically enforced by our extension to the PRISM-games tool (StEVe), which evaluates the rPATL queries accordingly.

Table 4. Formalisation of security metrics using rPATL within a rational verification framework for RSSGs. This table maps metrics to security properties and their corresponding rPATL representation under round by round Stackelberg equilibrium.

Metric	**Security Property Formalisation**
Cost Reduction	**Question:** What is the minimum expected total cost for the defender over T rounds, assuming optimal defence against a rational attacker? **Formula:** $\langle\langle D\rangle\rangle \mathsf{R}^{\text{“}total\_cost\text{”}}_{\mathsf{SE}=?}[\mathsf{F}\ \text{“}game\_over\text{”}]$
Value of Adaptiveness	**Description:** Quantifies the benefit (e.g., cost reduction) of an optimal adaptive defence strategy compared to the optimal static strategy over T rounds. Requires comparing the results of $\mathsf{R}_{\mathsf{SE}=?}$ calculations under adaptive versus static assumptions.
Mitigation Time	**Question:** What is the minimum expected time required to mitigate a specific critical vulnerability, assuming optimal defence against a rational attacker? **Formula:** $\langle\langle D\rangle\rangle \mathsf{R}^{\text{“}steps\text{”}}_{\mathsf{SE}=?}[\mathsf{F}\ \text{“}vulnerability\_mitigated\text{”}]$ *Note: Assumes a reward/cost structure named “steps” incrementing by 1 per time step.*
Incident Reduction	**Question:** What is the minimum probability of critical system compromise within T rounds, assuming optimal defence against a rational attacker? **Formula:** $\langle\langle D\rangle\rangle \mathsf{P}_{min=?}[\mathsf{F}^{<=T}\ \text{“}compromised\text{”}]$

A key strength of rational verification is its ability to translate high level security objectives into formally verifiable properties. We align our templates with the PRISM syntax. Table 4 illustrates the mapping between the metrics and the corresponding security properties that are formalised in our approach. The semantics assigned to these rPATL formulae within our StEVe tool are computed under the assumption that players follow the round by round Stackelberg equilibrium defined in Section 3.1.

These rPATL formulae are specified and model checked using our extended PRISM-games environment, implemented in the StEVe tool. The tool performs the computation when evaluating these properties, providing formal guarantees and rational verification.

4 Experimental Results and Analysis

4.1 Datasets

We created a dataset that reflects realistic vulnerabilities, particularly those exploited by Advanced Persistent Threats (APTs) in multi-step attacks, to assess the effectiveness of our rational verification framework. Cyber attackers often reuse known exploits across different campaigns, adapting their strategies

over time, which requires security models capable of reasoning about such repeated interactions. This list builds upon the work by Kuppa et al. [10] and has been updated with more recent information.[1] Our dataset comprises 126 CVEs associated with these APT activities. Data for each CVE was gathered programmatically using scripts interacting with publicly available resources, primarily the NVD API 2.0 for technical details and CVSS scoring, and the CISA KEV catalogue JSON feed to identify actively exploited vulnerabilities. Information regarding associated APT groups was gathered from public reporting and sources like MITRE ATT&CK. The collected data for each CVE includes attributes for game modelling, such as the attack sequences, mitigation, and severity.[2] We analyse CVE-2024-4947 as a representative case study due to its recency, critical severity, confirmed exploitation in the wild, and its nature as a browser based exploit involving multiple potential attacker steps and defender responses, making it highly relevant for modelling repeated interactions within our framework.

4.2 Experimental Setup

The experiments were conducted using the StEVe tool (v2.0), which can compute round by round Stackelberg equilibria in RSSGs and verify rPATL properties under this equilibrium assumption. We modelled the CVE-2024-4947 scenario, detailed in Example 1, as an RSSG in StEVe, in particular the dynamic interaction between the defender and attacker over multiple rounds.[3] Experiments were run for a horizon of $T = 10$ rounds with discount factor $\beta = 0.9$, reflecting a preference outcomes consistent with typical cybersecurity urgency. A 10-round horizon provides a sufficient window to observe a multi-stage attack and the defender's adaptive responses, whilst still ensuring the analysis remains computationally tractable. The high discount factor emphasises the critical importance of mitigating threats quickly, which is a standard assumption in security operations. Comparative analyses were also conducted for $\beta = 1.0$ (no discounting) and $\beta = 0.7$ to assess sensitivity to preferences.

4.3 Evaluation Objectives and Metrics

We defined a set of objectives to quantitatively measure the effectiveness of the defender's strategies. Our primary goal is to demonstrate the concrete advantages of an adaptive defence, computed via our framework, over a static one. We focus on metrics that capture defender cost, system risk, and mitigation efficiency. These metrics are directly mapped to verifiable properties within our model, allowing for a formal comparison of different strategic approaches.

The expected key results (KR) are as follows: (KR1) Achieve a significant reduction (target: $\geq 20\%$) in the defender's minimum expected total cost using

[1] Updates were gathered from sources including the MITRE ATT&CK .

[2] The complete dataset is archived and available for download from this repository .

[3] The key elements are defined in Table 1, and the involving transitions and payoffs are summarised in Table 2.

the optimal dynamic Stackelberg equilibrium strategy compared to the best static strategy. (KR2) Achieve a measurable reduction (target: $\geq 15\%$) in the minimum probability of compromise using the dynamic strategy compared to the static strategy. (KR3) Quantify the minimum expected time to mitigation (target: ≤ 3 rounds) achievable with the dynamic strategy under Stackelberg equilibrium. (KR4) Compute the specific minimum expected total cost achievable by the defender over $T = 10$ rounds ($\beta = 0.9$) under Stackelberg equilibrium. (KR5) Compute the specific minimum probability of compromise within $T = 10$ rounds ($\beta = 0.9$) under Stackelberg equilibrium. (KR6) Demonstrate the ability to model preferences by showing the quantitative impact of varying the discount factor (β) on optimal strategies and costs. These targets, such as achieving a $\geq 20\%$ reduction in cost, a $\geq 15\%$ reduction in compromise probability, or mitigation within 3 rounds, serve as illustrative benchmarks to demonstrate the benefits of adaptive strategies. The 20% cost reduction is motivated by economic considerations in cybersecurity investment [6], which emphasise proportional and cost-effective defence spending. The other targets are not based on formal standards but help contextualise the improvements observed in our case study.

4.4 Case Study Results: CVE-2024-4947

Adaptive vs. Static Strategy Performance (KR1, KR2, KR3). We first evaluated the core benefit of using an adaptive strategy derived from the round by round Stackelberg equilibrium compared to the best possible static defence strategy. For the defender's minimum expected total cost (KR1), the optimal dynamic strategy achieved a value of -28.5, representing a 36.7% reduction compared to the best static strategy's cost of -45.0. This significantly exceeds the target reduction $\geq 20\%$. Regarding the minimum probability of compromise (KR2), the dynamic strategy reduced this probability to 0.075 (i.e., 7.5%), a reduction of 25% compared to the static strategy's probability of 0.10 (10%). This meets the target reduction $\geq 15\%$. Furthermore, the minimum expected time to mitigation defined as reaching a 'Patched' or 'SiteBlocked' state preventing compromise for the current attack vector using the dynamic strategy (KR3) was quantified as 2.8 rounds using the "steps" reward structure. This achieves the target of ≤ 3 rounds. Thus, the adaptive strategy successfully met all three of our primary comparative objectives. These results demonstrate the performance and agility of the adaptive approach in terms of cost, risk, and responsiveness for this scenario.

Equilibrium and Impact of Discounting (KR4, KR5, KR6). Our framework provides formal guarantees on security posture under the Stackelberg equilibrium assumption. For the baseline parameters ($T = 10, \beta = 0.9$), the specific minimum expected total cost (KR4) was computed as -28.5. The corresponding minimum probability of compromise (KR5) was 7.5%. We also analysed the impact of temporal preferences (KR6) by varying the discount factor, changing β significantly influenced the optimal strategy and outcomes.

5 Conclusion and Future Work

This paper introduces a framework integrating rational verification with RSSGs to tackle the challenge of repeated, adaptive attacks in cybersecurity scenarios. Our experimental evaluation based on a real life case study of the CVE-2024-4947 vulnerability, demonstrates the benefits of this approach. We showed that the optimal dynamic strategy computed through our StEVe framework outperforms the best static strategy, leading to lower expected costs, a reduced probability of compromise, and diminished attacker utility. Our results show that adaptive strategies grounded in game-theoretic reasoning can offer formal guarantees on security performance against rational adversaries. This contribution enhances the security and safety of modern, interconnected systems by helping prevent malicious exploitation.

The accuracy of our findings depends on the alignment of the RSSG model with the real-world CVE-2024-4947 scenario. The chosen states, actions, transition probabilities, and payoff values are based on available public information and security principles but inevitably involve simplifications and estimations. The assumption of perfect attacker rationality might not hold in all real-world cases, as attackers can be limited by resource constraints, make mistakes, or have different utility functions. The results are derived from a single case study. The round by round Stackelberg equilibrium concept assumes the attacker observes and best responds to the defender's action each round, which may not capture all modes of interaction such as simultaneous moves. While StEVe enabled analysis for $T = 10$, scaling rational verification for RSSGs to significantly larger state spaces or very long horizons remains computationally challenging.

In future work we will refine the parameters of the underlying RSSG model to better capture real-world complexities. This includes extending our approach to handle more general, multi-level ADTs. While the current framework uses a simplified ADT structure for tractability, more complex ADTs would introduce significant computational overhead. Developing efficient verification algorithms or techniques to apply our framework to these larger, more realistic security scenarios is a key research challenge. Theoretical extensions could also explore different equilibrium concepts beyond round by round Stackelberg, potentially offering insights into interactions with different commitment or information structures, alongside deeper analysis of infinite games. A second promising direction concerns scalability and practical adoption. Developing more efficient verification algorithms is necessary for applying the framework to larger, more appropriate security scenarios. Future work includes integrating survivability and recoverability analysis [3] to evaluate system resilience against component failures and service degradation. Finally, empirical studies across diverse vulnerabilities and security domains are needed to assess the generalisability and effectiveness of rational verification in repeated security games. The datasets, models, and artifacts supporting this study are available in an online repository.[4]

[4] https://zenodo.org/records/13338608

Acknowledgments. S.P. is supported by a Royal Thai Government Scholarship. M.S. is supported by an Amazon Research Award on Automated Reasoning.

References

1. Aslanyan, Z., Nielson, F., Parker, D.: Quantitative verification and synthesis of attack defence scenarios. In: CSF'16. pp. 105–119. IEEE Computer Society (2016)
2. Avizienis, A., Laprie, J., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE TDSC **1**(1), 11–33 (2004)
3. Calder, M., Sevegnani, M.: Stochastic model checking for predicting component failures and service availability. IEEE TDSC **16**(1), 174–187 (2019)
4. Chen, T., Forejt, V., Kwiatkowska, M.Z., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. FMSD **43**(1), 61–92 (2013)
5. Do, C.T., Tran, N.H., Hong, C.S., Kamhoua, C.A., Kwiat, K.A., Blasch, E., Ren, S., Pissinou, N., Iyengar, S.S.: Game theory for cyber security and privacy. ACM Comput. Surv. **50**(2), 30:1–30:37 (2017)
6. Gordon, L., Loeb, M.: The economics of information security investment. ACM Trans. Inf. Syst. Secur. **5**(4), 438–457 (2002)
7. Konsta, A.M., Lafuente, A.L., Spiga, B., Dragoni, N.: Survey: Automatic generation of attack trees and attack graphs. Comput. Secur. **137**, 103602 (2024)
8. Kordy, B., Mauw, S., Radomirovic, S., Schweitzer, P.: Attack Defence Trees. J. Log. Comput. **24**(1), 55–87 (2014)
9. Kulik, T., Dongol, B., Larsen, P.G., Macedo, H.D., Schneider, S., Tran-Jørgensen, P.W., Woodcock, J.: A survey of practical formal methods for security. Form. Asp. Comput. **34**(1), 1–39 (2022)
10. Kuppa, A., Aouad, L.M., Le-Khac, N.: Linking CVE's to MITRE ATT&CK techniques. In: Proc. 16th Int. ARES Conf. pp. 21:1–21:12. ACM (2021)
11. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In: Proc. of CAV'2020. LNCS, vol. 12225, pp. 475–487. Springer (2020)
12. Leyton-Brown, K., Shoham, Y.: Essentials of game theory: A concise multidisciplinary introduction. Morgan & Claypool (2008)
13. Ng, C.Y., Hasan, M.K.B.: Cybersecurity serious games development: A systematic review. Comput. Secur. **150**, 104307 (2025)
14. NVD: CVE-2024-4947. https://nvd.nist.gov/vuln/detail/CVE-2024-4947
15. Phetmanee, S., Sevegnani, M., Andrei, O.: StEVe: A rational verification tool for Stackelberg security games. In: IFM'24. vol. 15234, pp. 267–275. Springer (2024)
16. Pita, J., Bellamane, H., Jain, M., Kiekintveld, C., Tsai, J., Ordónez, F., Tambe, M.: Security applications: Lessons of real world deployment. ACM SIGecom Exch. **8**(2), 1–4 (2009)
17. Shostack, A.: Threat modeling: Designing for security. John Wiley & Sons (2014)
18. Sinha, A., Fang, F., An, B., Kiekintveld, C., Tambe, M.: Stackelberg security games: Looking beyond a decade of success. In: Proc. IJCAI18 (2018)
19. von Stackelberg, H.: Market structure and equilibrium. Springer (2011)
20. Tambe, M.: Security and game theory. Cambridge University Press (2012)
21. Widel, W., Audinot, M., Fila, B., Pinchinat, S.: Beyond 2014: Formal methods for attack tree-based security modeling. ACM Comput. Surv. **52**(4), 75:1–75:36 (2019)
22. Wooldridge, M., Gutierrez, J., Harrenstein, P., Marchioni, E., Perelli, G., Toumi, A.: Rational verification: From model checking to equilibrium checking. Proc. of the AAAI Conf. on Artificial Intelligence **30**(1) (2016)