Keep spending: Beyond optimal cyber-security investment

Yunxiang Zhang
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
y.zhang@qmul.ac.uk

Pasquale Malacaria
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, UK
p.malacaria@qmul.ac.uk

Abstract—We introduce an efficient solution for Stackelberg games in the context of a class of Security games and bounded rational attackers. These games model a threat scenario where an attacker can launch multi-stage attacks against a defender who can deploy defensive controls subject to some budget constraints. Because the optimal solution in these games may leave some unspent budget, the question of what to do in this situation arises.

In this work, we suggest investing it iteratively in the closest sub-optimal solutions until possible. Here we develop the needed theory and framework, starting from defining sub-optimality and solving the corresponding optimisations. By using total unimodularity and precise linear programming (LP) relaxation, we provide an efficient computational solution to these games. The security improvement of the proposed approach is illustrated with an AI threat scenario.


I. INTRODUCTION

In the cyber-security framework here studied attackers can mount multi-stage attacks to exploit an organisation’s networks, and an organisation can deploy a security portfolio of defensive controls to mitigate the potential security risks. Examples of cyber-security controls include [1]: identity management and access control, password policy, software update and patch management, anti-malware and anti-phishing software, physical security, etc. Each control could be applied to multiple vulnerabilities to mitigate attacks; however, they are not cost-free, and the organisation has some budget constraints. The problem of finding optimal defensive control security portfolios relates to a special class of games, namely Stackelberg security games [2], [3]. These are two-player games with the defender (the leader) and the attacker (the follower). First, the defender calculates the attacker’s optimal attacking strategy and commits to an optimal defence strategy accordingly. Next, the attacker commits to an attacking strategy based on the observed defender’s strategy (e.g. security portfolio).

Recent research [4]–[11] has addressed how to find optimal security investment to protect those cyber-security targets. However, most of these game-theoretical solutions [5]–[9] assume a perfectly rational attacker who will always seek an optimal strategy to maximise his payoffs. Based on this “worst-case assumption”, the defender’s optimal strategy is to minimise the highest security risk that an optimal attacker can achieve. The worst-case assumption is justifiable because any attacker who deviates from the optimal strategy would result in a lower security risk (e.g. attacking the second weakest path has a lower security risk than attacking the weakest path [7]). However, these “cybersecurity investment” games, involving both security risk and budget constraints, and the optimal min-max solutions (e.g. [5]–[7]) will only focus on minimising the highest security risk, i.e. these approaches will stop investing in defence once they have minimised the highest security risk, even if there is some remaining budget which would allow further (sub-optimal) cyber-security investment.

Hence the key observation motivating this work is that in these games mathematical optimality (i.e. the game solution) may result in “suboptimal” defense in the real world, in the sense that the defense can be improved by a further investment of the remaining budget.

The key question this paper addresses is hence:

“What is the best way to invest the budget remaining after the optimal investment?”

This question is also connected to the classical game-theoretical problem of bounded rationality (and also ϵ-equilibrium 1). This is because a bounded rational attacker or an attacker with incomplete information 2 could launch a sub-optimal attack and hence the remaining budget can be used to counteract such attacks.

At a high level, our strategy is that the defender should first invest as much as possible in the optimal solution, then invest as much as possible of the remaining budget in the best sub-optimal solution, and so on until exhausting the budget or running out of security controls.

Our objective is hence to exhaust available defence resources to minimise the overall security risks of all possible multi-stage attacks.

We use probabilistic attack graphs [12] to model an organisation’s security risks. In an attack graph, the nodes represent

1 an equilibrium that is "nearly" optimal for all players who may have a very small incentive to deviate from the current strategy.

2 note that an attacker can be partially rational due to incomplete information about the implemented security strategy.
the privilege states of the attacker, and each edge represents a vulnerability exploitable by the attacker. Thus, we can use paths from the source node to the target node to represent the attacker’s possible multi-stage attacks, where each path is a sequence of exploitation actions of the organisation’s vulnerabilities. Here the defender’s problem can be expressed as follows: “how can the defender invest all of his budgets in a security portfolio that minimises the overall security risks?”

For the mathematical development, the first step is to define sub-optimality: we study this concept sequentially and parallelly. In the sequential approach, the defender first finds an optimal security portfolio (defence of the weakest path). Next, after deploying the optimal security portfolio, the defender will compute the optimal security portfolio on the updated threat scenario (minimise the security risk of the second weakest path). In the following stages, the defender will continue to use the remaining budgets to minimise the security risks of the next weakest path until exhausting the budgets or controls. By contrast, the parallel approach aims to find an optimal security portfolio to minimise the organisation’s overall security risks (corresponding to \( n \) attacking paths) in one single optimisation. Although the parallel approach seems appealing, we will show that it is problematic (see Section II).

Each stage in the sequential approach is formulated as a Stackelberg game, which is a bi-level optimisation problem (or a min-max problem). We use the term N-Solution for the solution of the N-th stage problem, and N-Solutions for the solutions of the 1 to N stage problems. Anticipating the attacker’s attack on the N-th weakest path (the inner maximisation), the defender (the outer minimisation) finds the optimal (additional) controls within the remaining budgets to minimise the security risk of the N-th weakest path. The inner maximisation problem is a special type of the k-shortest path problem. Although the parallel approach seems appealing, we will show that it is problematic (see Section II).

To exemplify and motivate this work, let’s consider the scenario shown in Figure 1. Node 0 (the source) represents the attacker’s initial state, and the attacker’s target is to exploit the database (node 3). Node 1 represents the state where the attacker has exploited the workstation, and node 2 the state where the attacker has compromised the web server. The edges represent the attacker’s attack steps (i.e. vulnerabilities).

Associated with these edges are security controls. Control Ed is user education and awareness training. Control Ant is anti-malware software for workstations. Control ApS is application isolation and sandboxing for the web server. Control AcC represents the access control to the database.

Controls FW-int, FW-ext, FD-int, and FD-ext are the internal and external firewalls for the web server and the database, respectively.

The baseline success probabilities of the attacker exploiting edge 0 → 1, 0 → 2, 0 → 3, 1 → 2, 1 → 3, and 2 → 3 are L, M, H, M and M respectively, where L = 0.7, M = 0.5, and H = 0.2. That means, that for example the attacker success probability of exploiting edge 0 → 1 is 0.7 if no control is applied on the edge.

The effectiveness of controls are Ed = L, Ant = M, ApS = M, AcC = H, FW-int = M, FW-ext = M, FD-int = M, and FD-ext = H. The effectiveness of a control represents its ability of reducing security risk. Applying control Ed, for example, on edge 0 → 1 reduces the attacker’s success probability by 0.7. Thus, the success probability of the attacker to exploit this edge becomes 0.49 = 0.7 × 0.7.

Suppose that the defender can select at most 7 controls (i.e. the budget is 7).

1-Solution: This solution is the Stalkeberg game solution: the optimal security portfolio is \[ \{ Ed, Ant, ApS, AcC, FW-ext, FD-int, FD-ext \} \] which minimises the highest security to 0.0125 corresponding to the weakest path 0 → 2 → 3. Given the 1-Solution, we find that the security risks corresponding to the remaining paths are 0.01225 for 0 → 1 → 2 → 3, 0.009 for 0 → 1 → 3, and 0.008 for 0 → 3. We notice that there is remaining budget for one control and two unused controls: FD-int and FW-int. Applying these internal firewalls can further reduce security risk of paths 0 → 1 → 2 → 3 and 0 → 1 → 3.

Based on the Stackelberg game setting, the 1-Solution will not add controls FD-int or FW-int to the optimal security portfolio because they cannot further reduce the security risk on the weakest path 0 → 2 → 3.

2-Solution: Although the 1-Solution has minimised the security risk of the weakest path, there is still remaining budget to protect other paths. The 2-Solution focuses on finding the optimal additional controls to protect the second weakest path. Applying additional control FW-int reduces the security risk of path 0 → 1 → 2 → 3 from 0.01225 to 0.006125. Applying additional control FD-int reduces the security risk of path 0 → 1 → 3 from 0.0098 to 0.0049. Thus, in this simple example the 2-Solution is trivial: adding control FW-int to the optimal security portfolio to protect the second weakest path.
However, in the general case, the 2-Solution is an intractable nonlinear bi-level optimisation problem. Thus, we propose a novel approach (in section III) that makes the problem tractable by decomposing the problem into multiple “edge-forbidden Stackelberg games” (similar to the 1-Solution), so that each game can be converted into an LP problem. Applying the general methodology to the example in Fig 1, the 2-Solution will respectively forbid edge 0 → 2 and edge 2 → 3, resulting in two edge-forbidden Stackelberg games. Each game has a subset of available paths, which are \( \{0 \rightarrow 1 \rightarrow 2 \rightarrow 3, 0 \rightarrow 1 \rightarrow 3, 0 \rightarrow 3\} \) and \( \{0 \rightarrow 1 \rightarrow 3, 0 \rightarrow 3\} \), i.e., the weakest path 0 → 2 → 3 will not be considered in the 2-Solution. Given the 1-Solution, the defender will find the additional controls within budget to minimise the highest security risk of these available paths, in this case FW-int.

**N-Solutions:** the N-Solutions consists of the union of all solutions of the 1 to N stage problems (up to exhausting the budget or controls), which in this example is \([Ed, Ant, ApS, AcC, FW-Ext, FD-Ext, FW-Int]\).

### B. Contributions and Plan of the Paper

The main contribution of this work is a new efficient approach to compute solutions for a class of security games when facing bounded rational attackers.

To this aim, the work starts by discussing the sequential and parallel approaches to sub-optimality (Section II) and it is argued in favour of the former.

In the sequential approach (Section III), we show how to exactly reformulate the original intractable Stackelberg game (a nonlinear bi-level optimisation) as a tractable MILP and thus find the optimal security portfolio that minimises the overall security risks for the organisation.

The inner problem (a k-shortest path problem) is transformed into a simple LP problem and its dual problem is used for the efficient game solution (Section III-B).

The meaningfulness of the methodology and the security improvement it provides is illustrated with a simple Home IoT threat scenario (Section IV), and its efficiency is shown by numerical evaluations (Section V).

### C. Related Works

Security games [3] are a special type of problem in game theory for modelling and reasoning about the strategic interactions between a defender and attackers. Many Stackelberg game-theoretic and Bayesian Stackelberg game-theoretic approaches [13] have been widely used to solve security challenges in public safety [14], wildlife protection [15], [16], and particularly cyber-security [5]–[9]. Recall that a Stackelberg security game has a defender and an attacker. The defender is the leader who first anticipates the attacker’s best strategic response and commits to an optimal defence strategy accordingly. The attacker is the follower who observes the defender’s defence strategy and commits to the optimal attacking strategy. Such a game setting becomes Bayesian if the defender needs to face multiple attacker types and has uncertain knowledge about the attacker type he may face. There are multiple efficient methods to solve general Bayesian Stackelberg games [17]–[19], and specific Bayesian Stackelberg games for cybersecurity decision support [6]. The structure of security games, the interchangeability of Stackelberg and Nash equilibrium, and the restriction on real security scenarios have been well studied in [20]. Recent work [21], inspired by biological evolution, provides a generic approach based on evolutionary algorithm to solve sequential Stackelberg security games. In [22], it extends to a coevolutionary approach to approximate Stackelberg equilibrium.

The closest work to this paper is [7] where the cybersecurity defence game is modelled as a multi-objective bi-level Stackelberg game with a probabilistic attack graph. We use the method in [7] to find the stage-one security portfolio that minimises the security risk of the weakest path (the 1-Solution). The work in [5] extends the mathematical framework to a Markov chain combined with probabilistic attack graphs to reason about time resilience in cyber-security. Our 1-Solution is similar to the works in [8], [9] in terms of the security risks and the budgets; however, these methods only consider single-step attacks rather than multi-stage attack scenarios as in [5]–[7]. Please note that these multi-stage attack scenarios represent attack paths that are formed by multiple privileged states leading to the target. Notice that [5]–[7] only focus on minimising the weakest path, which is similar to our 1-Solution. To the best of our knowledge, this is the first work providing the N-Solutions for \( N > 1 \).

Traditionally, in a Stackelberg security game, we assume a perfectly rational attacker who will always commit to the optimal attacking strategy that maximises his payoff. As discussed in Section I, this is a justifiable assumption. However, in real-world security applications, the defender may face many bounded rational attackers. It has been seen in real-world security applications that these bounded rational attackers may cause degradation in the deployed defence strategy [23] (depending on the problem setting and solutions). Thus, the work in [23], [24] provides robust approaches to address bounded rational human adversaries. Moreover, thanks to the recent advances in machine learning (ML) techniques, many works [25]–[27] have been using learning-based models to predict the attacker’s behaviours based on the collected attacking data. These models have successfully achieved a high prediction accuracy of attacks. However, as mentioned, a “smart” attacker could deceptively change his attack behaviour to mislead the defender whose prediction heavily relies on these collected attacking data [28], [29]. Our approach provides resilience to such attacks.

Our solution closely relates to the k-shortest path problem, a generalisation of the shortest path problem: here the attacker attacks the k-th weakest path (the k-th shortest path) rather than the weakest path (the shortest path). The k-shortest path problem can be solved by extending Dijkstra’s algorithm or the Bellman-Ford algorithm. Moreover, the variations of the k-shortest path problem: the loopy and the loopless variants can also be solved using Eppstein’s algorithm [30] and Yen’s
algorithm [31], respectively. However, our approach needs to transform the attacker’s problem (a maximisation) into its dual problem (a minimisation) and thus relaxes the bi-level optimisation to a tractable single-level optimisation. To the best of our knowledge, we do not know an existing algorithm satisfying such a requirement. Thus, we propose a novel technique to solve the k-th weakest (shortest) path problem. By dividing the problem into multiple edge-forbidden problems (a special type of the shortest path problem where the attacker cannot transit through that forbidden edge), we can transform the k-th weakest (shortest) path problem into an LP problem (a maximisation) with the strong duality holds for it. Thus, it can be further dualised to a minimisation.

Our work also belongs to the cyber-security investment problems that have been studied in a number of papers. One of the first works [4] considers both the costs and benefits of the optimal level of investment. Later, several works [5]–[9] use game-theoretical approaches to optimal investment. Recent works [10], [11] focus on addressing cyber-security investment in supply chains.

D. Background Definitions and Notations

Recall that we use probabilistic attack graphs to model the security risk of an organisation. We use the same notations as [7]. A probabilistic attack graph \( G = (\mathcal{V}, \mathcal{E}, \pi, \mu, s, t) \) is defined as a directed multi-graph: \( \mathcal{V} \) and \( \mathcal{E} \) are the set of nodes and edges. The nodes in the attack graph are the “privilege states” of the attacker, and the edges are vulnerabilities that allow the attacker to exploit and change its privilege states. Thus a multi-stage attack (a sequence of the attacker’s exploitation attempts) can be represented by a path from the source to the target. An edge \( e \), which is directed from node \( i \) to node \( j \), has tail node \( i \) and head node \( j \), and \( \pi(e) \) and \( \mu(e) \) are the functions that return the head node and the tail node of edge \( e \). Function \( p_e(x) \) returns the attacker’s success probability of exploitation attempts on edge \( e \), which is determined by the attacker’s baseline success probability on that edge \( \pi_e \) and the effectiveness coefficient of controls protecting that edge \( p_{ed} \). The source node and the target node are denoted by \( s \) and \( t \).

The attacker will select a path with the highest probability of success. The defender can “buy” controls to reduce the probability of success of the attacker, \( C \) and \( \mathcal{L}(e) \) denote the set of controls and the set of intensity levels of controls \( c \). Each control at each level has specified costs and effectiveness (reduction of risk). \( \text{Cost}_{cl} \) and \( \text{InCost}_{cl} \) represent the direct costs (e.g. monetary investment) and indirect costs (negative costs, e.g. side-effects on normal operations) of control \( c \) at level \( l \).

II. SUB-OPTIMALITY: SEQUENTIAL AND PARALLEL APPROACHES

Here we discuss sequential and parallel approaches to sub-optimality.

A. Sequential Approach

Assuming a bounded rational attacker, the more the attacker deviates from the optimal strategy, the less the security risk for the defender. The sequential approach follows the idea that the defender first should spend the defence resources to protect the organisation from the most critical attack (e.g. the weakest path). Next, the defender should spend the remaining defence resources on the following most critical attacks and repeat this until he exhausts the defence resources. Thus the sequential approach divides the whole problem into multiple stages. In stage one, the defender minimises the security risk of the weakest path to \( r^*_1 \), and thus the highest security risk is no greater than \( r^*_1 \) regardless of which path the attacker commits. Note that the security risk of the weakest path \( r^*_1 \) cannot be further reduced even if we spend more defence resources on it because it has been minimised. Hence, in stage two, the defender uses the remaining budgets and controls to minimise the security risk of the second weakest path to \( r^*_2 \), and the highest security risk the attacker can achieve is not greater than \( r^*_2 \) once the attacker deviates from the weakest path. The stage-two optimal security portfolio consists of the controls implemented in stage two and the security portfolio implemented in stage one. In the following stages, the defender repeats the process of sequentially minimising the security risk of the following weakest paths until exhausting the budgets or the controls. Therefore, the optimal security portfolio is the stage-one security portfolio augmented with these additional controls found in every following stage.

Remark 1: Notice that in an attack graph, paths may share multiple same edges, and a control may be effective on multiple edges. Thus, a security portfolio of controls typically mitigates the security risks of multiple paths. In other words, a security portfolio reduces the overall security risk of a network, not just of a single path. Moreover, a security portfolio has a corresponding weakest path which refers to the path with the highest security risks in that attack graph. Thus, the weakest path security risk measures the effectiveness of a security portfolio, because any other path has not greater security risk than the weakest path. The lower the weakest path security risk, the better the security portfolio. Thus, the defender’s optimal security portfolio minimises the weakest path security risk, while also mitigating security risks of other paths.

Since the stage-one problem (the 1-Solution) has been addressed in [7], our solution focuses on those stages beyond the first one (the N-Solution with \( N \geq 2 \), First, the stage-two problem can be equivalently considered as the stage-one problem with a forbidden path. Given the stage-one security portfolio, we have “updated” parameters. For example, the budgets are the remaining budgets, the set of controls consists of only the unused controls and the higher levels of used controls (if there are higher levels), and the attacker’s baseline probability of successfully exploiting a vulnerability (an edge) is also based on the stage-one security portfolio. Moreover, the attack graph has been modified, which forbids the attacker from attacking the weakest path. As a result, an optimal attacker can only attack the second weakest path which happens to be equivalently the “weakest” in the stage-two problem. Thus the defender’s objective is to find the
additonal optimal controls to minimise the security risk of that “weakest” path in the stage-two problem. (In other words, the second weakest path is the optimal attack path in the 2-Solution). In a nutshell, the stage-two problem is a similar problem to stage one, but the attack graph has a forbidden path. The following stage problems are similar to the stage-one problem, but the attack graph has more forbidden paths. Later, we show how to solve the stage-two problem in Section III and extend to the following stage problems in Section III-D.

Example: To clarify the basics of the sequential approach, let’s consider the simple attack graph illustrated in Figure 2. Nodes 0 and 1 are the source and the target. Path 1 (top) associates with control $c_1$ and $c_2$, and path 2 (bottom) associates with controls $c_3$ and $c_4$. The baseline probability of path 1 and path 2 are 1 and 0.81, i.e. an attacker has a success probability of 1 and 0.81 to attack “unprotected” path 1 and path 2. Each control has cost one, and its effectiveness coefficient is as shown in the figure. The smaller the coefficient, the more effective the control. Applying the associated controls reduces the success probability of the attacker exploiting the corresponding edges (paths). For example, applying control $c_3$ with effectiveness coefficient of 0.6 on path 2 reduces the attacker’s success probability to $0.486 = 0.60 \times 0.81$. Suppose the defender has a budget of 3 (i.e. at most, three controls can be implemented). We now apply the sequential approach to find the optimal security portfolio. In stage one, the defender will add controls $c_1$ and $c_2$ to the security portfolio. As a result, the security risks of path 1 and path 2 change to 0.912 and 0.81. Based on the worst-case assumption, adding control $c_3$ or $c_4$ cannot further reduce the security risk of path 1. However, it can reduce the security risk of path 2. Thus, path 1 is forbidden in stage two, i.e. only attack path 2 is considered. To minimise the security risk of the “weakest” path (path 2) in stage two, the defender adds control $c_3$ to the security portfolio. As a result, the optimal security portfolio is [$c_1$, $c_2$, $c_3$], and the security risks of path 1 and 2 are minimised to 0.912 and 0.486.

Remark 2: Notice that the $(n+1)$-Solution operates not over the original attack graph but the attack graph resulting from applying the $n$-Solution to the original attack graph; we hence update the remaining budgets, controls, their corresponding intensity levels and effectiveness. In other words, in each stage, we recursively update the attack graph based on the previously implemented security portfolio. For example, the 1-Solution (the stage one solution in the example) selects security portfolio [$c_1$, $c_2$]. In stage two (the 2-Solution), the baseline probabilities of path 1 and path 2 are updated to 0.912 and 0.81, the budget becomes one, and the available controls are now $c_3$ and $c_4$.

Also notice that if a control has multiple intensity levels, and in the $n$-Solution that control has been selected at level $l$, then that control can be selected at a higher level than $l$ in the following optimisations. Please see Equation (5) and the follow up explanation for details.

B. Parallel Approach

Instead of dividing the defender’s problem into multiple stages, the parallel approach aims to find the optimal security portfolio in one single optimisation.

Without loss of generality, let’s consider when the 2-Solutions, i.e. we want to minimise the security risk ($r_1$, $r_2$) for the two weakest paths. The parallel approach can be formulated as follows:

$$\begin{align*}
\min_{x, r_1, r_2} & \quad b_1 r_1 + b_2 r_2 \\
\text{s.t.} & \quad r_1 \geq \max_{\omega_1 \in \Omega} \prod_{e \in \omega_1} p_e(x); \\
& \quad r_2 \geq \max_{\omega_2 \in \Omega \setminus \{\omega_1\}} \prod_{e \in \omega_2} p_e(x);
\end{align*}$$

$\Omega$ is the set of all attacking paths, and $\omega_1$ and $\omega_2$ represent the weakest and the second weakest paths formed by some edges $e$ in the paths. $x$ represents a security portfolio and $p_e(x)$ is attacker’s success probability attacking edge $e$ when portfolio $x$ is deployed. Please note that we omit the budget constraints here for the sake of simplicity.

The maximisation problems represent the attacker best response. In (2), the attacker finds the path with the highest success probability, i.e. the weakest path $\omega_1$. Then, in (3), the attacker wants to find the path with the second highest success probability. Anticipating the attacker’s best response in (1), the defender wants the optimal security portfolio to minimise the weighted sum of the security risks corresponding to these two paths.

Such an optimisation problem is intractable. To find the second weakest path, we need to solve the maximisation problem in (3). Thus, we first need the weakest path by solving the maximisation problem in (2). In other words, (3) is indeed a bi-level optimisation:

$$\max_{\omega_2 \in \Omega \setminus \{\omega_1^*\}} \prod_{e \in \omega_1} p_e(x), \quad \text{s.t.:} \quad \omega_1^* = \arg \max_{\omega_1 \in \Omega} \prod_{e \in \omega_2} p_e(x),$$

which is NP-hard [32]. Moreover, to find the optimal security portfolio $x^*$, we must first solve both optimisations in (2) and (3). However, the solution of these equations depends on the implemented security portfolio $x$, creating another bi-level optimisation problem that is NP-hard [32].

Moreover, this approach depends on the defender’s beliefs (weights $b_1$, $b_2$) about the attacker’s rationality. Let’s consider the example in the previous section (see Figure 2). Since the example is simple, it is trivial to find the solution. Suppose
\( b_1 = 0.8 \) and \( b_2 = 0.2 \), i.e. the defender believes the attacker is partly rational. The parallel approach will select the security portfolio \([c_1, c_3, c_4]\): this provides less security on the weakest path than the sequential solution, i.e. paths 1 and 2 have security 0.950 and 0.296, respectively.

The parallel approach has hence the following problems: a) some values used in some constraints depend on the prior solution of other constraints and b) the result depends on the weights in the objective functions (the \( b_i \)) and it is not clear how such weights ought to be computed.

Another important difference between the two approaches is the following:

**Proposition 1:** The Stackelberg game solution is always part of the sequential approach (N-solutions), but it is not, in general, part of the parallel approach solution.

To see why, let \( A_1 \) be the portfolio corresponding to the game solution (i.e. \( A_1 \) is the 1-Solution). Suppose \( B_1 \) is a suboptimal portfolio that is \( \epsilon \) close to \( A_1 \) but such that it can be extended to \( B_2 \) which provides good protection of the two weakest paths, whereas \( A_1 \) can only be extended to a portfolio \( A_2 \) which provides inferior protection of the two weakest paths (compared to \( B_2 \)). Then the parallel approach will select \( B_2 \) and this is suboptimal on the weakest path, i.e. the game solution is not part of \( B_2 \).

Motivated by these considerations this work will hence study the sequential approach.

**C. Disjoint suboptimality**

A different notion of suboptimality not yet discussed would be one where the 2-Solution would look for these paths which are disjoint (i.e. has no shared edge) with the previous weakest path. This is a weaker notion (in terms of security guarantees) than the one we adopted: in the example (see Figure 1) this approach will ignore the second weakest path \( 0 \to 1 \to 2 \to 3 \) and select control \( FD-int \) to reduce the security risk of path \( 0 \to 1 \to 3 \).

**III. N-SOLUTIONS: SEQUENTIAL APPROACH**

We use the method in [7] as the optimal solution (the 1-Solution). Thus, we now focus on the problem with \( N = 2 \) (i.e. the stage-two problem). The defender aims to find the optimal additional security portfolio within the remaining budgets, which minimises the security risk corresponding to the second weakest path. Such a security portfolio can be expressed as follows:

\[
\forall c \in \mathcal{C}, l \in \mathcal{Z}(c) : \pi_{cl} \in \{0, 1\} \text{ and } \forall c \in \mathcal{C} : \sum_{l \in \mathcal{Z}(c)} \pi_{cl} \leq 1, \tag{5}
\]

where \( \mathcal{C} \) and \( \mathcal{Z}(c) \) denote the set of controls and the set of intensity levels of controls which are available after the previous optimisation. For example, suppose control \( c \) has two intensity levels \( L_1 < L_2 \). If control \( c \) at level \( L_1 \) has been selected in stage one, then \( c \) is in \( \mathcal{C} \), but only the higher level \( L_2 \) of \( c \) can be selected in stage two, i.e. \( \mathcal{Z}(c) = \{L_2\} \). In this case, the effectiveness and the costs of control \( c \) at level \( L_2 \) should be updated according to the applied \( L_1 \). Let \( p_{c,L_1} \) (resp. \( p_{c,L_2} \)), \( \text{Cost}_{c,L_1} \) (resp. \( \text{Cost}_{c,L_2} \)), and \( \text{InCost}_{c,L_1} \) (resp. \( \text{InCost}_{c,L_2} \)) denote the original effectiveness and costs of control \( c \) at level \( L_1 \) (resp. \( L_2 \)). The updated effectiveness and costs of \( c \) at level \( L_2 \) are \( (p_{c,L_2}, \text{Cost}_{c,L_2}, \text{InCost}_{c,L_2}) \), and \( \text{InCost}_{c,L_2} - \text{InCost}_{c,L_1} \), given \( c \) at level \( L_1 \) has been selected in stage one. If control \( c \) is not selected in stage one, then both levels of control \( c \) can be selected in stage two. Finally if \( c \) has been selected at the highest level \( L_2 \) in the 1-Solution then control \( c \) has been used and thus is not in the set \( \mathcal{C} \). Selection of a control is modelled by the binary indicators: \( \pi_{cl} = 1 \) if controls \( c \) at level \( l \) is selected; otherwise, \( \pi_{cl} = 0 \). Since at most one level of a control can be implemented, the sum of the binary indicators of all levels of a control is less than or equal to one, i.e.: \( \forall c \in \mathcal{C} : \sum_{l \in \mathcal{Z}(c)} \pi_{cl} \leq 1 \).

Moreover, the security controls are not cost-free; thus, the selected portfolio must be subject to the following budget constraints:

\[
\sum_{c \in \mathcal{C}, l \in \mathcal{Z}(c)} \pi_{cl} \text{Cost}_{cl} \leq \mathcal{B}_D; \quad \sum_{c \in \mathcal{C}, l \in \mathcal{Z}(c)} \pi_{cl} \text{InCost}_{cl} \leq \mathcal{B}_I, \tag{6}
\]

where \( \text{Cost}_{cl} \) and \( \text{InCost}_{cl} \) represent the updated direct costs (e.g. monetary investment) and indirect costs (e.g. side-effects on normal operations) of remaining control \( c \) at level \( l \), respectively. The sum of direct costs and indirect costs must be not greater than the remaining direct and indirect budgets \( \mathcal{B}_D \) and \( \mathcal{B}_I \).

The “attacker’s problem” is now to select a path with the second-highest security risk, namely the second weakest path. Such a problem can be decomposed into multiple edge-forbidden problems. In each edge-forbidden problem, we forbid one single edge of the weakest path \( \omega_1 \); hence the attacker will find the optimal attack path (with the highest security risk) that does not use this forbidden edge. In other words, the optimal attack path must not be the weakest path \( \omega_1 \). To guarantee the attacker’s problem feasibility, we assume the edge-forbidden attack graph has at least one complete path from the source and the target. Moreover, we will later show that this assumption can be lifted in the dual problem. Finally, the attacker will find the second weakest path with the highest security risk from all optimal attack paths found in the edge-forbidden problems.

Thus the 2-Solution (\( N = 2 \)) can be expressed as follows:

\[
\min_{\pi} r, \quad \text{s.t. (5), (6)}; \quad (\forall e_f \in \omega_1 : r \geq \max_{e_{f} \rightarrow \ell(E \setminus \{e_f\})} \prod_{e \in \omega_{r \rightarrow \ell(E \setminus \{e_f\})}} p_e(\pi)), \tag{7}
\]

where \( r \) denotes the security risk of the second weakest path, and \( \omega_{r \rightarrow \ell(E \setminus \{e_f\})} \) represents a path from the source to the target, formed by edges in the edge set \( \mathcal{E} \setminus \{e_f\} \) where \( e_f \) is the forbidden edge. Note that the edge-forbidden problems are represented as the maximisation problems in Equation (7). The overall success probability of the attacker on edge \( e \) given
the security portfolio $\pi$ is denoted by $p_e(\pi)$, which can be expressed as:

$$p_e(\pi) = \pi_e \prod_{c \in C(l,t) \in Z(e)} (p_{c,d} \pi_{c,d} + (1 - \pi_{c,d})), \quad (8)$$

where $\pi_e$ is the “baseline” success probability of attacker exploitation attempts on edge $e$, and $p_{c,d} \in (0, 1)$ is the effectiveness coefficient of control $c$ at level $l$ applied on edge $e$. Moreover, $C(e)$ is the set of remaining controls associated to edge $e$. Thus, the security risk of a path $\omega_{s-t}(C(e) \setminus \{e\})$ is the product of $p_e(\pi)$ for all $e \in \omega_{s-t}(C(e) \setminus \{e\})$. Note that the values of $\pi_e$ and $p_{c,d}$ should be updated based on the previously implemented security portfolio (i.e. here is the optimal security portfolio found in the 1-Solution).

Each edge-forbidden problem in (7) can be equivalently expressed as:

$$\max_{y_{e,f}} \prod_{e \in E} (y_{e,f} \times p_e(\pi) + 1 - y_{e,f,e}), \quad (9)$$

s.t.: $\forall e \in E, y_{e,f,e} \in \{0, 1\}$;

$$\sum_{e \in C(l,t) \in Z(e) = i} y_{e,f,e} - \sum_{e \in C(l,t) \in Z(e) = i} y_{e,f,e} = \begin{cases} -1 & i = t, \\ 1 & i = s, \\ 0 & i \in V \setminus \{s,t\}, \end{cases}$$

$$\forall e \in E \setminus \{e\} : y_{e,f,e} \leq 1 \text{ and } y_{e,f,e} \leq 0; \quad (10)$$

where binary variables $y_{e,f,e}$ indicates whether edge $e$ is selected by the attacker to form the optimal attack path, and (10) represents flow conservation constraints to ensure that the optimal attack path connects the source and the target. Moreover, constraint (11) ensures the edge is forbidden.

Remark 3: Notice that following [7], we assume controls to be independent. In [7], the authors have justified the assumption that controls are composed independently. Since different controls have distinct defensive “mechanisms” based on distinct operations or features, it is natural to assume that controls are independent. Moreover, the model is capable of extending to partially dependent controls. For example, suppose controls $c_1$ and $c_2$ are partially dependent, i.e. the overall effectiveness of implementing both controls is different from the product of the effectiveness of each of them. To incorporate such controls, we can introduce an auxiliary control, named $c_{12}$, with a specific overall effectiveness, and an additional constraint that $x_{c_1} + x_{c_2} + x_{c_{12}} \leq 1$. Thus the optimisation can choose controls $c_1$, $c_2$ or both of them appropriately. On the other hand, if one control appears to be a higher implementation level than another control (e.g. “intrusion prevention system” is a higher implementation level than “intrusion detection system”), we can select only one of two controls.

A. Total Unimodularity of the Edge-forbidden Problem

Next we will show the polytope formed by constraints (10) and (11) has integral vertices. Therefore, the integer constraints (9) can be precisely relaxed to linear constraints: $0 \leq y_{e,f,e} \leq 1, \forall e \in E$. This is an important step because it will allow for a dramatic improvement of the computational time of the optimisation.

**Definition 1:** A matrix $A$ is totally unimodular if every square submatrix of $A$ has determinant $\{-1, 0, +1\}$.

The flow constraints (10) can be expressed as $F y_{e,f} = c_F$, where matrix $F$ represents the incidence matrix of the directed attack graph and integral vector $c_F$ represents the right-hand-side of constraint (10). Following Example 2, Chapter 19 in [33], the incidence matrix of a directed graph is totally unimodular, i.e. matrix $F$ is totally unimodular.

**Theorem 1:** An integral matrix $A$ is totally unimodular if and only if, for all integral vectors $b, b, ub, lb$, the vertices of the polytope $\{x : b \leq Ax \leq b; lb \leq x \leq ub\}$ are integral.

**Proof:** This can be derived from the Hoffman-Kruskal Theorem. For the detailed discussion, please refer to Corollary 19.2a and Equations (4) and (5) in Chapter 19 in [33].

Since integral matrix $F$ is totally unimodular and constraints (11) has an integral vector of upper bounds, the vertices of the polytope formed by constraints (10) and (11) are integral. Following Theorem 1, the integer constraints (9) can be relaxed to $0 \leq y_{e,f,e} \leq 1, \forall e \in E$.

B. Edge-forbidden Dual Problem

Recall the objective function of the edge-forbidden problem is in the product form. However, we can equivalently maximise the log of the objective since $\log(x)$ is strictly monotone for $x > 0$: $\log(\prod_{e \in E} y_{e,f} \times p_e(\pi) + 1 - y_{e,f,e}) = \sum_{e \in E} \log(y_{e,f,e} \times p_e(\pi) + 1 - y_{e,f,e})$. Moreover, if $y_{e,f,e} = 0$, then $\log(y_{e,f,e} \times p_e(\pi) + 1 - y_{e,f,e}) = 0$; otherwise, it is $\log(p_e(\pi))$. Hence, the objective function can be further relaxed to $\sum_{e \in E} \log(y_{e,f,e} \times p_e(\pi))$. Note that we treat $\log(p_e(\pi))$ as constants in the attacker’s problem, given a security portfolio $\pi$.

First, we prove the strong duality of the edge-forbidden problem using refined Slater’s Theorem (Section 5.2.3 in [34]), which provides sufficient conditions for strong duality to hold. Namely, if the edge-forbidden problem is convex and feasible when the constraints are linear and the domain of objective function is open, then the strong duality holds for it. Note the relaxed edge-forbidden problem is linear, therefore convex. Indeed, the domain of the relaxed objective function is open. Moreover, following our assumption that there is at least a path connecting the source and the target, the problem is also feasible. Therefore, the strong duality holds for the edge-forbidden problem, and its dual problem is attained.

Here, we associate the Lagrange function $L(y_{e,f}, \lambda, \nu)$ to
find the dual edge-forbidden problem:

\[ L(y_{ef}, \lambda, \nu) = \sum_{e \in E} y_{ef,e} \log(p_e(\pi)) - \sum_{i \in V} \lambda_i (\sum_{e \in \mathcal{E}(e) = i} y_{ef,e} - \sum_{e \in \mathcal{N}(e) = i} y_{ef,e} - m_i) - \sum_{e \in E} \nu_{ub,e}(y_{ef,e} - d_{ef,e}) = \sum_{e \in E} y_{ef,e} \nu_{ub,e} - \nu_{ub,e} \log(p_e(\pi)) - \lambda(e) + \lambda_{N(e)} + \sum_{e \in E} \nu_{ub,e} \times d_{ef,e} + \sum_{i \in V} \lambda_i \times m_i \]  

(12)

where \( \lambda_i, \nu_{ub,e}, \nu_{ub,a} \) are the Lagrange multipliers for the flow constraints and inequality constraints. Moreover, \( m_i \) and \( d_{ef,e} \) are constants: if \( i = s \), \( m_i = 1 \); else if \( i = t \), \( m_i = -1 \); otherwise \( m_i = 0 \). Also, \( d_{ef,e} = 0 \) and \( d_{ef,e} = 1 \) for \( e \in \mathcal{E} \setminus \{ef\} \). By definition, the Lagrange dual function is as follows:

\[ g(\lambda, \nu) = \sup_{y_{ef}} L(y_{ef}, \lambda, \nu) \]  

(13)

\[ = \begin{cases} 
\lambda_s - \lambda_t + \sum_{e \in \mathcal{E} \setminus \{ef\}} \nu_{ub,e}, & \text{if } \lambda(e) - \lambda_{N(e)} - \log(p_e(\pi)) + \nu_{ub,e} = 0, \forall e \in \mathcal{E} \setminus \{ef\} \\
\nu_{ub,e}, & \text{if } \lambda(e) - \lambda_{N(e)} - \log(p_e(\pi)) + \nu_{ub,e} > 0, \forall e \in \mathcal{E} \setminus \{ef\} \\
\infty, & \text{otherwise.}
\end{cases} \]

The dual problem is feasible only if \( \nu_{ub,e} \geq 0 \) and \( \nu_{ub,a} \geq 0 \), \( \forall e \in \mathcal{E} \) hold, i.e. the dual feasibility conditions as given in the KKT (Karush-Kuhn-Tucker) conditions. Therefore, the dual edge-forbidden problem is as follows:

\[ \min_{\lambda, \nu_{ub,e}} \lambda_s - \lambda_t + \sum_{e \in \mathcal{E} \setminus \{ef\}} \nu_{ub,e}, \]  

s.t.: \( \forall e \in \mathcal{E} \setminus \{ef\} \): \( \nu_{ub,e} + \lambda(e) - \lambda_{N(e)} \geq \log(p_e(\pi)), \nu_{ub,e} \geq 0 \),

(14)

Moreover, we can relax the dual problem by letting \( \nu_{ub,e} = 0, \forall e \in \mathcal{E} \setminus \{ef\} \) using complementary slackness in the KKT conditions.

Following the complementary slackness conditions, we must have \( \nu_{ub,e} = 0 \) for \( e \in \mathcal{E} \setminus \{ef\} \). Recall that \( y_{ef,e} = 1 \) if edge \( e \) is selected to form the optimal attacking path; otherwise \( y_{ef,e} = 0 \). Suppose \( \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\}) \) is the optimal attacking path, then we must have \( \nu_{ub,e} = 0 \) for any \( e \not\in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\}) \) and \( e \in \mathcal{E} \setminus \{ef\} \) to satisfy the complementary slackness conditions. Thus, for any optimal solution, the objective function equals to \( \lambda_s^* - \lambda_t^* + \sum_{e \in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\})} \nu_{ub,e}^* \) where we let \( \lambda_s^*, \lambda_t^* \) and \( \nu_{ub,e}^* \) denote the optimal values.

Next, let’s focus on the optimal attacking path. By constraints (14), we have

\[ \lambda_s^* - \lambda_t^* + \sum_{e \in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\})} \nu_{ub,e}^* \sum_{e \in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\})} \log(p_e(\pi)). \]  

(15)

In other words, the optimal objective function equals to \( \sum_{e \in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\})} \log(p_e(\pi)) \). If we let \( \nu_{ub,e} = 0 \) for all \( e \in \mathcal{E} \setminus \{ef\} \), by constraints (14), Equation (15) still holds, i.e. the optimal objective function still equals \( \sum_{e \in \omega_{\ast \rightarrow t}(\mathcal{E} \setminus \{ef\})} \log(p_e(\pi)) \).

Thus, we can relax the dual edge-forbidden problem by letting \( \nu_{ub,e} = 0 \) for all \( e \in \mathcal{E} \setminus \{ef\} \).

We also notice that constraint \( \nu_{ub,e} + \lambda(e) - \lambda_{N(e)} \geq \log(p_e(\pi)) \) always holds if \( \nu_{ub,e} \) is a large positive number \( M \). Therefore, the dual edge-forbidden problem can be relaxed as an LP as follows:

\[ \min_{\lambda, \nu_{ub,e}} \lambda_s - \lambda_t, \]  

s.t.: \( M + \lambda(e) - \lambda_{N(e)} \geq \log(p_e(\pi)) \);  

(16)

\( \forall e \in \mathcal{E} \setminus \{ef\} : \lambda(e) - \lambda_{N(e)} \geq \log(p_e(\pi)) \).

(17)

Next, let’s justify the relaxed dual edge-forbidden problem. For all possible paths from the source to the target (i.e. feasible solutions), the objective function must follow such inequalities (derived from constraints (16) and (17)):

\[ \lambda_s - \lambda_t \geq \sum_{e \in \omega_{\ast \rightarrow t}} \log(p_e(\pi)); \text{ or equivalently} \]  

\[ \exp(\lambda_s - \lambda_t) \geq \prod_{e \in \omega_{\ast \rightarrow t}} p_e(\pi), \]  

(18)

if the path does not contain the forbidden edge \( ef \); otherwise, we have

\[ \lambda_s - \lambda_t \geq \sum_{e \in \omega_{\ast \rightarrow t}} \log(p_e(\pi)) - M; \text{ or equivalently} \]  

\[ \exp(\lambda_s - \lambda_t) \geq \prod_{e \in \omega_{\ast \rightarrow t}} p_e(\pi)/\exp(M). \]  

(19)

The minimisation will push the value of \( \lambda_s - \lambda_t \) down to the smallest value that satisfies both (18) and (19), and the corresponding path is the attacker’s optimal path. However, the attacker never selects a forbidden “path” that contains the forbidden edge since the security risk will result in an extremely small value with a large \( M \), i.e. selecting such a path must violate (18). In other words, the attacker will only select a path that does not contain the forbidden edge. Moreover, if there is no complete path that does not contain the forbidden edge (i.e. inequalities (18) do not exist), then the minimisation will return a path with the forbidden edge, resulting in a low value satisfying (19). Thus, we can lift the feasibility assumption that there is at least a complete path for the dual edge-forbidden problem.

C. 2-Solution

Here we combine the defender’s and the attacker’s problems to formulate a mixed-integer linear programming (MILP) to
find the optimal additional security portfolio for the defender:

$$\min_{R, \pi} \{ \lambda_{e_f, s} \forall e_f \in \omega_1 \} \quad R$$

s.t.: (5); (6):

$$\forall e_f \in \omega_1 : \quad R \geq \lambda_{e_f, s} - \lambda_{e_f, t},$$

$$M + \lambda_{e_f, \pi(x)} - \lambda_{e_f, \pi(c)} \geq \log(p_{e}(x)),$$

$$\forall e \in \mathcal{E} \setminus \{ e_f \} : \quad \lambda_{e_f, \pi(x)} - \lambda_{e_f, \pi(c)} \geq \log(p_{e}(x)),$$

where $R = \log(r)$ and $\lambda_{e_f}$ is a dual vector for the edge-forbidden problem $e_f$. Note that we relax the constraint $R \geq \min_{\pi} (\lambda_{e_f, s} - \lambda_{e_f, t})$ to there exists $\lambda_{e_f}$ such that (20) holds, subject to the corresponding constraints. Moreover, $\log(p_{e}(x)) = \log(p_{e}(\pi)) + \sum_{c \in \mathcal{C}(x)} \log(p_{ec}(\pi_c + (1 - \pi_c)))$; equivalently $\log(p_{e}(\pi)) = \log(\pi_{e}) + \sum_{c \in \mathcal{C}(x)} \log(p_{ec}(\pi_c + \pi_{ec}) log(p_{e}))$, which is a linear function.

The formal development and solution for the generalised sequential approach ($N \geq 3$) is similar to the 2-Solution.

D. N-Solution for $N \geq 3$

In the case $N = 2$, we added constraints to force the choice of one edge not in the previous weakest path. In the general case each edge-forbidden problem can be indexed using a tuple $(e_{f,1}, \ldots, e_{f,N-1}) \in \omega_1 \times \cdots \times \omega_{N-1}$ (where $e_{f,i}$ denotes an edge being forbidden on the $i$-th weakest path). Therefore, in that edge-forbidden problem, the attacker is forbidden to transit through any edge $e_i \in \{ e_{f,1}, \ldots, e_{f,N-1} \}$. For the sake of simplicity, we let $\pi_f = (e_{f,1}, \ldots, e_{f,N-1})$ denote a tuple and $\mathcal{E}_f = \{ e_{f,1}, \ldots, e_{f,N-1} \}$ denote the corresponding set.

The formal development and solution for the general case will then follow the one presented in the previous sections, i.e. total unimodularity, LP relaxation, and duality.

Thus, the problem can be represented as a MILP to find the N-Solution as follows:

$$\min_{R, \pi} \{ \lambda_{\pi_f, s} \forall \pi_f \in \omega_1 \times \cdots \times \omega_{N-1} \} \quad R$$

s.t.: (5), (6):

$$\forall \pi_f \in \omega_1 \times \cdots \times \omega_{N-1} : \quad R \geq \lambda_{\pi_f, s} - \lambda_{\pi_f, t},$$

$$\forall e \in \mathcal{E}_f : \quad M + \lambda_{\pi_f, \pi(x)} - \lambda_{\pi_f, \pi(c)} \geq \log(p_{e}(\pi)),$$

$$\forall e \in \mathcal{E}_f : \quad \lambda_{\pi_f, \pi(x)} - \lambda_{\pi_f, \pi(c)} \geq \log(p_{e}(\pi)),$$

where in (5), (6) the set of controls, levels and budgets are the ones remaining after the (N-1)-Solution.

E. Tie-breaking

Some attention is due to the cases where there are multiple equivalent weakest paths or multiple equivalent portfolios. In these cases the optimisation will return one among the equivalent paths or portfolios and this may result in different solutions for the following stages. This problem can be mitigated (albeit inefficiently) by using some backtracking strategy, or it can be “avoided” by using some $\epsilon$-values in the parameters so to force unicity of paths and portfolios. Also it is worth saying that while these cases are easy to build in small artificial graphs, a real attack graph will be large and complex, and it will be unlikely to have multiple equivalent paths or portfolios.

IV. HOME IOT THREAT SCENARIO – A CASE STUDY

We use the sequential approach here developed for adversarial machine learning (AdvML) threat scenarios against smart-home IoT devices, in particular a smart cleaning robot (SCR). Based on the recently developed ATLAS\(^3\) [35] and reported attacks on SCRs, the attack graph for the scenario is in Figure 3. This simple case study is rather high-level with only a few controls and attacks, so designed to be easily understood. However as shown in the evaluation section, the framework scales up to a large attack graph and set of controls.

Here we consider a scenario where the attacker’s aims are to cause a smart cleaning robot (SCR) to malfunction or take control of the robot camera. Nowadays, SCRs are often embedded with AI techniques to detect and avoid obstacles. They often use MTTQ, an IoT connectivity protocol acting as a bridge between the SCR, the backend servers, and the user’s apps. Conventionally, the attacker may exploit underlying vulnerabilities in local network, user’s apps or the SCR itself (e.g. lacks of proper authentication mechanisms, hardcoded username and passwords, and unencrypted connections) to attempt an array of attacks. In addition, this scenario takes into account AdvML attacks on IoT devices powered with AI technologies. Possible attack scenarios and taxonomy can be found in ATLAS [35] and the NIST’s AdvML report [36], respectively. In particular, we consider evasion attacks, a type of AdvML attacks that allows the attacker to craft adversarial examples or masks to make SCR’s ML model to misclassify.

The attack scenario is as follows:

- **edge $s \rightarrow 1$:** the attacker may first gain a foothold in the user’s local network or home router through traditional attacks, for example, the attacker uses phishing techniques to manipulate the user to give away sensitive information (e.g. username and passwords) of the local network and the home router.
- **edge $1 \rightarrow 2$:** with a foothold in the local network, the attacker can directly exploit the SCR through its underlying vulnerabilities. For example, the remote code vulnerability, CVE-2018-10987, allows the attacker to obtain SCR’s admin privileges. Using a default username and password combo, the authenticated attacker can send a specially crafted UDP (User Datagram Protocol) packet and execute commands on the SCR as root. The attacker can cause the SCR to malfunction (edge $2 \rightarrow T$), or further perform evasion attacks.
- **edges $2 \rightarrow 3/4/5 \rightarrow 6 \rightarrow T$:** once the attacker has root privileges, he can perform an evasion attack: the attacker

\(^3\) A knowledge base of AdvML tactics, techniques, and real-world case studies published by the MITRE association.
finds adversarial examples with small perturbations causing a large change in loss function and results in output misclassification [36]. Thus, the attacker needs to collect and infer information about the ML model. There are multiple techniques:

1) edges $2 \rightarrow 3 \rightarrow 6$: the attacker may collect ML model artefacts from publicly available resources, since organisations often use open source models and architecture to train with additional proprietary data in production, or he may steal the model. Complete knowledge of the underlying ML model in the SCR allows the attacker to craft adversarial examples using white-box optimisation. However, it is difficult to obtain complete knowledge of the ML model.

2) edges $2 \rightarrow 4 \rightarrow 6$: if the attacker cannot find complete knowledge to perform a white-box optimisation, the attacker could monitor the traffics and obtain input-output pairings to develop a substitute model that operates much like the target model. Using the substitute model, the attacker can craft adversarial examples. This technique is also known as Oracle Attacks. [36]

3) edges $2 \rightarrow 5 \rightarrow 6$: the attacker may have no knowledge of the inner working of the underlying ML model in the SCR. The attacker may still craft black-box adversarial examples based on a proxy model. Such a technique is generally less effective but requires much less access to the model.

- edges $1 \rightarrow 7 \rightarrow 9$: instead of causing the SCR to malfunction, the attacker can stealthily spy on users by controlling the surveillance camera on the SCR. The attacker who has a foothold in the local network can monitor the traffics between the SCR, user’s apps and the MQTT server. Because of lacks of authentication mechanisms, the attacker can steal the certificate inside the app which has hardcoded username and passwords.

With the certificate, the attacker can impersonate the user who can connect to the server$^4$.

- edges $s \rightarrow 8 \rightarrow 9$: alternatively, the attacker can use social engineering attacks to let the user install malicious apps on the smartphone and steal the user’s valid account to connect to the server.

- edges $9 \rightarrow 10 \rightarrow T$: once connected to the server, the attacker can continue an array of attacks. Here the attacker takes control of the surveillance cam to spy on the user.

In this case study we consider the following controls as potential mitigation for the vulnerabilities (edges in the graph):

- **User Education** (Ed): in attack steps $s \rightarrow 1$ and $s \rightarrow 8$, the attacker use social engineering attacks to obtain sensitive information from the user. Good education and awareness training encourage the user to keep good cyber hygiene habits and be aware of ongoing cyber attacks. For example, change the home router’s default password, and awareness of local network anomalies. We consider two levels of user education: Ed 1 – basic education; Ed 2 – advanced user awareness.

- **Firewall** (FeW): the incoming and outgoing network traffic should be monitored and controlled based on predetermined security rule to prevent the attacker obtain the foothold inside the trusted network (edge $s \rightarrow 1$). We consider two levels of firewall: FeW 1 – stateless firewall; FeW 2 – stateful firewall for network traffic filtering.

- **Anti-malware** (AnT): the user’s smartphone should be protected by anti-malware or anti-virus that use signatures or heuristics to detect malicious software, e.g. a malicious app (edge $s \rightarrow 8$).

- Perform automatic application patch management (PaM): in attack steps $1 \rightarrow 2$ and $1 \rightarrow 7$, the attacker exploits the underlying vulnerabilities in the SCR. Automatic application patch management updates the smart cleaning

$^4$This attack is reported at https://threatpost.com/unpatched-security-flaws-open-connected-vacuum-to takeover/153142/
robot to patch underlying vulnerabilities frequently. We consider two levels of patch management: PaM 1 – Patching policy of long provisioning time; PaM 2 – Patching policy of short provisioning time

- **Access control to information about the ML model** (PuB): in attack step $2 \rightarrow 3$, the attacker attempts to find the underlying ML model in the smart cleaning robot from publicly available resources and the product provider. The details of the ML model (e.g. parameters, architecture and training process) of a commercial product should not be available to the public to prevent white-box AdvML attacks. We consider two levels of access control: PuB 1 – detailed restrictions on publications; PuB 2 – organisation-wide ban on publication of datasets and AI models and techniques

- **Data protection** (DaP): in attack step $1 \rightarrow 7$ and $2 \rightarrow 4$, the attacker monitors the traffic coming and going from the SCR. The loss of control over data can unravel sensitive information about the SCR. Encryption of data stored and in transit provides Data protection.

- **Robustness improvement** (RoI) of the ML model: in evasion attacks (edge $6 \rightarrow T$), the attacker crafts adversarial examples to mislead the ML model in the SCR. Robustness improvement includes multiple techniques, such as Adversarial Training, Gradient Masking, Defensive Distillation, and Ensemble Methods [36]. For example, data containing adversarial perturbations but with corrected labels are injected into the training dataset (Adversarial Training), the model’s sensitivity to a small perturbation is reduced (Gradient Masking and Defensive Distillation), or multiple classifiers are trained and combined together to improve robustness (Ensemble Methods).

- **Randomization** mechanism (RaD) [36]: the attacker attempts to collect data from the smart cleaning robot to analyse the model ($2 \rightarrow 4$). Randomization mechanisms, for example differential privacy, protect the privacy of inputs and thus the ML model from analysing. However, such techniques may cause degradation of the SCR performance.

For the values of the parameters given below, the Stackelberg game solution (the 1-Solution) will provide mitigation for these attacks. However with the strategy introduced in this work, as there is some remaining budget, further mitigation will also be provided in the following stages (the 4-Solutions).

Please note these values are educated guesswork and meant to illustrate the methodology. They can be estimated using surveyed data. For example, the estimated effectiveness coefficients in [7], [37]. Moreover, [38] provides a formulation (based on the CVSS) to estimate the baseline probability of exploiting a vulnerability. Effectiveness coefficients and costs of controls are presented in Table I, where $\forall H = 0.1, H = 0.3, M = 0.5, L = 0.7, \forall L = 0.9$, and VL, L, M, H, VH = 1, 2, 3, 4, 5.

The baseline probabilities of the edges are $\pi_{0,1} = M, \pi_{0,8} = M, \pi_{1,2} = M, \pi_{1,7} = M, \pi_{2,7} = 1, \pi_{2,3} = VH, \pi_{2,4} = H, \pi_{2,5} = M, \pi_{3,6} = M, \pi_{4,6} = H, \pi_{5,6} = VH, \pi_{6,7} = M, \pi_{7,9} = L, \pi_{8,9} = L, \pi_{9,10} = M, \pi_{10,T} = L$. Recall that the baseline probabilities here represent the difficulty of an attacker of successfully perform the corresponding attack step when no control is applies.

Suppose both direct and indirect budgets are 20 and 20. The Stackelberg game solution in [7] (the 1-Solution) returns the optimal security portfolio [Ed 2, FeW 2, PaM 1, AnT], which has direct and indirect cost 10 and 11 and hence leaves 10 and 9 units of budget unspent. The weakest path is $s \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow T$ with the highest security of 0.04288. There are multiple unused controls. Although adding more controls cannot further mitigate the security risk the weakest attack path, it do help mitigate other attacks.

The N-Solutions is as follows:

- 2-Solution: replace PaM 1 by PaM 2. The second weakest path is $s \rightarrow 1 \rightarrow 2 \rightarrow T$ with security risk reduced from 0.02625 to 0.01575. The remaining budgets are 9 and 8.
- 3-Solution: add DaP. The third weakest path is $s \rightarrow 1 \rightarrow 7 \rightarrow 9 \rightarrow 10 \rightarrow T$ with security risk reduced from 0.006431 to 0.001158. The remaining budgets are 6 and 4.
- 4-Solution: add RoI. The fourth weakest path are two paths: $s \rightarrow 1 \rightarrow 2 \rightarrow 3/5 \rightarrow 6 \rightarrow T$ with both the security risks reduced from 0.0006562 to 0.0001969. The remaining indirect budget is zero, hence we stop.

With the sequential approach developed in this work, the solution becomes [Ed 2, FeW 2, PaM 2, AnT, DaP, RoI] with significantly improved mitigation of the overall security risk.

### V. Evaluation

In this section, we illustrate the scalability of the N-Solutions using random attack graphs with similar topology to real attack graphs. In [39], the ATT&CK Enterprise Matrix provides the attacker’s possible techniques to mount multi-stage attacks. A complete attack path includes the following 14 steps: Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact.

For the evaluation, we use the following attack graph setting as shown in Figure 4: Each attack graphs consists of $h$ layers,
each represents an attack step. In each layer, there are $m$ nodes which represent possible attack vectors. Nodes 0 and $(h \times m + 1)$ denote the source and the target. The source connects to all nodes in layer 1, and all nodes in layer $h$ connect to the target. Each node in layer $i$ has a probability $3/m$ of forming an edge with a node in layer $(i+1)$. In addition, each node has a probability of $1/(h \times m)$ of forming an edge with any node in layer $j \geq i$. The optimisation was programmed in Python using Gurobi solver. All computations were performed on a PC with a 3.80 gigahertz AMD Ryzen 7 5800X and 64 gigabytes RAM.

![Fig. 4. Attack graph setting.](image)

1) Scenario One: In scenario one, there are 37 controls randomly associated with the edges (attack steps); each control has two levels, and direct and indirect costs are randomly selected between 1 and 10. Direct and indirect budgets are set to 100. We let $h = 14$, i.e. the same number of attack steps presented in the ATT&CK Enterprise Matrix [39]. Each layer has $m = 50$ nodes, so there are 700 nodes in an attack graph. The optimisation computation time for the 2,3,4-Solutions is shown in Figure 5.

![Fig. 5. Results of the evaluations (700 nodes). The red line is the least-square line of the scatter plot. Each experiment is repeated 50 times so to collect sufficient data.](image)

Next, we expand the number of attack steps from 14 to 25. The computation time is shown in Figure 6.

2) Scenario Two: Notice that in scenario one, most of the controls are implemented in the 1-Solution, leaving few controls for subsequent stages.

To make the optimisation more challenging, we devised Scenario 2, where we directly compute the 2-Solution, the 3-Solution, and the 4-Solution by defining the weakest, the second-weakest, and the third-weakest paths as arbitrary paths in an attack graph. Each attack path has 15 edges.

We consider 10 controls, each with two levels, and direct and indirect costs are randomly selected between 1 and 10. Both direct and indirect budgets are 10. We let $h = 14$ and $m = 50$ (700 nodes); the computation time is shown in Figure 7. Each experiment is repeated 15 times.

![Fig. 7. Results of the evaluations (700 nodes).](image)

3) Discussion and Limitations: We evaluated two scenarios. In scenario 1, we use the 1-Solution to find the stage-one optimal security portfolio and the weakest path. Then, we compute the 2-Solution, the 3-Solution, and the 4-Solution with the remaining budgets and controls to find the additional controls for the second, the third and the fourth weakest paths. In scenario two, we directly compute the 2-Solution, the 3-Solution, and the 4-Solution by defining the corresponding weakest paths as arbitrary paths in the attack graph.

We notice that the main bottleneck in the computation is the length of the attack path, more than the size of the graph. In scenario 1, we consider attack paths with lengths up to 26 edges, and in scenario two, we consider lengths up to 15 edges. These are reasonable attack lengths as supported by the ATT&CK Enterprise Matrix [39].

VI. Conclusion

This work introduced a novel and efficient solution for cyber-security Stackelberg games where attackers may act with
bounded rationality. The context and justification of these games and of the N-Solutions have been presented, and the potential application to security is illustrated with a simple Home IoT threat scenario. We evaluated the scalability of this approach using random attack graphs with a topology similar to those in real-world scenarios. We showed that our method can efficiently compute the 4-Solutions. Although the main bottleneck in computation is the length of the attack path, it is worth noting that real-world attack graphs generally have short attack paths, which our approach can handle effectively.

The game theoretical approach presented here and its worked short attack paths, which our approach can handle effectively.

is worth noting that real-world attack graphs generally have
short attack paths, which our approach can handle effectively.

The game theoretical approach presented here and its worked
short attack paths, which our approach can handle effectively.

Future work will focus on estimating parameters and collecting data and validating the solution in experiments with real-world scenarios.

REFERENCES


