Defense Policy Optimization with Linear Temporal Logic Specifications for Interconnected Networks

Armita KazemiNajafabadi*, Derya Aksaray† and Mahdi Imani‡
Northeastern University, Boston, MA, 02115

The increasing interconnection of aerospace systems presents significant security challenges. The protection of these systems must be achieved with minimal disruptions to system operations while ensuring the desired specifications and network security. This paper presents network security through Bayesian attack graphs (BAGs), a powerful class of models that captures stochasticity in attacks and their propagation. The security objective involves dynamically defending the network with limited available resources to ensure the network is secure and stop the propagation of any potential intrusions. Despite the development of several defense and security solutions for BAGs, these methods mostly optimize specific security metrics without providing guarantees in terms of violations of practical security constraints, making them unsafe or inapplicable to sensitive and complex aerospace systems. This paper models security constraints using temporal logic (TL) specifications. Specifically, we formalize a set of specifications regarding resource availability, network security, servers' maintenance requirements, and more as Linear Temporal Logic (LTL) specifications. An automaton-theoretic approach is used to compute feasible policies that guarantee the satisfaction of the desired LTL specifications. Since feasible policies achieve different security performances, this paper develops an efficient security algorithm that selects a policy (among a set of desired feasible policies) yielding the highest expected lookahead security performance in every horizon of a specified length. Numerical experiments demonstrate the effectiveness of the proposed framework in proactively responding to threats and meeting specifications under various conditions.

I. Nomenclature

n	=	number of components in the network
$\mathbf{x}_k$	=	network compromises at time step k
$\mathcal{X}$	=	space of all possible network compromises
$\mathbf{a}_k$	=	defense action at time step k
X	=	space of defense actions or equivalently, the state space of the transition system
α	=	defense failure rate
$\mathcal{N}$	=	BAG nodes
$\mathcal{T}$	=	type of nodes in BAG
$\mathcal{E}$	=	directed edges in BAG
$\mathcal{P}$	=	space of exploit probabilities
$\mathcal{D}_j$	=	the in-neighbor (parent) nodes to the jth node
ρ_{ij}	=	exploit probability from component i to component j
ρ_j	=	success probability of external attack on component j

II. Introduction

Computer networks are the lifelines connecting various components in aerospace systems, from satellites to ground stations [1–4]. These systems are especially vulnerable due to their interconnected nature [5–9]. These networks are vulnerable not only to unauthorized access to sensitive data but also to potential disruptions of critical services.

*PhD Candidate, Department of Electrical and Computer Engineering, Northeastern University
†Assistant Professor, Department of Electrical and Computer Engineering, Northeastern University, and AIAA Member.
‡Assistant Professor, Department of Electrical and Computer Engineering, Northeastern University

Malicious actors can exploit vulnerabilities in these systems to compromise data integrity or disrupt operations. Therefore, there is a critical need for robust solutions that address these security challenges and maintain the integrity and functionality of aerospace networks.

Several security solutions have been developed in recent years [10–14]. These include machine learning techniques that can process huge amounts of multi-model data to detect potential attacks and abnormalities in networks [15, 16], resource optimization techniques to enable monitoring of the vulnerable parts [17–21], as well as defense security solutions to eliminate the possible threat and prevent the propagation of the attacks [4, 22]. Bayesian attack graphs (BAGs) are a powerful class of models that provide a model-based and flexible approach to network security. BAGs depict the intricate dependencies and interconnections between elements of systems, enabling comprehensive security analyses [23–25]. Specifically, BAGs offer a probabilistic representation of attack penetration and propagation in the network by considering the security measures in each network component and their vulnerabilities.

Preventing the propagation of attacks into the network, especially to critical components, necessitates proactive and sequential defense decisions, such as activating firewalls or restoring computers/servers. Several security strategies have been developed for computer networks modeled by BAGs. These include dynamic risk management for the allocation of monitoring resources [26, 27], inference techniques for inferring network vulnerabilities given the available data [28–31], defense strategies using reinforcement learning [32, 33], and representation through logic circuits [34]. The current defense solutions are primarily focused on providing security measures that achieve optimal performance based on specific security metrics. However, in high-stakes aerospace systems, security solutions should not only provide effective security measures, but also ensure that they adhere to practical security constraints. Violations of these constraints could potentially put the entire system at risk.

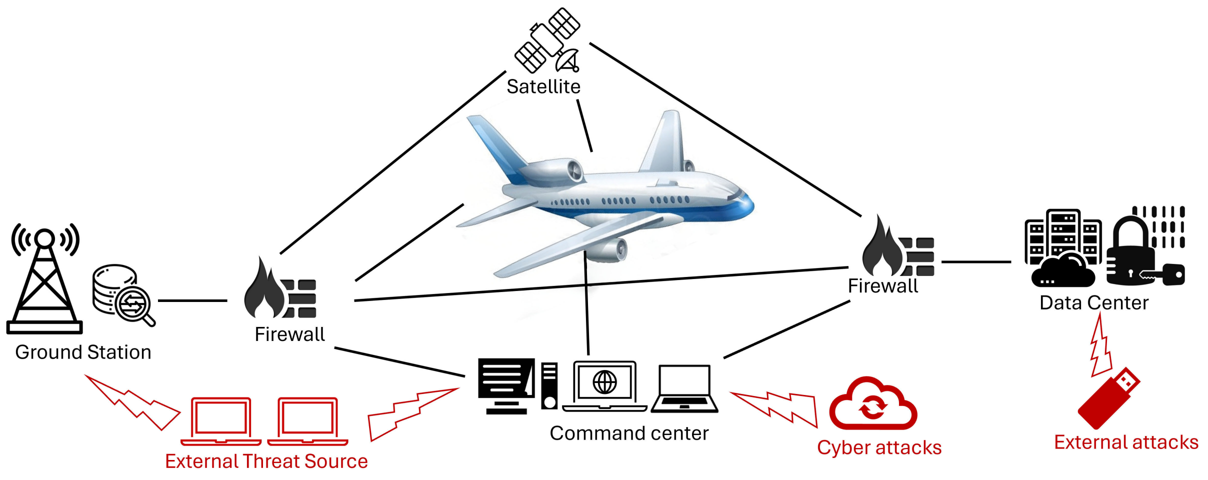


Fig. 1 A diagram illustrating an aerospace network with cybersecurity measures like firewalls and encryption to combat external attacks and insider threats.

An illustrative example of a network in aerospace applications is shown in Figure 1. This figure consists of multiple interconnected elements, including a command center, data center, ground station, and satellite, which help operate the aircraft. The red components demonstrate various types of attacks on this network, including cyber attacks through the internet, external attacks through USB, printers, or other connected devices, as well as other network components that might be connected to potentially external sources. Once an attack penetrates the network, it can potentially propagate among the network elements and put the operation at high risk.

Formal methods is a discipline that provides principled methods to verify systems under complex spatial-temporal-logical specifications [35]. Such complex specifications are typically formalized by specification languages called temporal logic (TL). In the last two decades, there has been a significant interest in using temporal logic to synthesize decision-making policies for complex dynamical systems (e.g., drones [36–38], satellites [39], ground robots [40–42], networked systems [43, 44]). Overall, a provably correct policy is generated by adopting either an automata-theoretic approach [45] or an optimization-based approach [46] that results in formal guarantees on desired TL specification.

This paper explores the use of formal methods for network security of systems modeled by BAGs. A set of practical constraints for network security is considered and formulated using Linear Temporal Logic (LTL) specifications. The constraints range from resource availability, network security requirements, servers' maintenance or shutdown, and more. Security policy must ensure that all LTL specifications are met while yielding proper security performance. We propose a two-step method: 1) the feasibility of security solutions is analyzed via an automata-theoretic approach to ensure the satisfaction of the desired TL specifications; 2) an algorithm is introduced to select a defense policy from the set of feasible policies that yields the highest expected security performance. The resulting policy accounts for the latest network compromises, ensuring adaptability with respect to the attack stochasticity. Using numerical experiments, we demonstrate the performance of the proposed policies in terms of risk mitigation and upholding the specifications and compare them with existing security solutions.

This paper is organized as follows: Section III presents the network model using a Bayesian attack graph. Section IV details a Markov Decision Process (MDP) representation of the network security model. Section V covers the definitions of linear temporal logic (LTL), Büchi automaton, transition system, desired specifications, and product automaton. Section VI discusses the optimal defense strategy among a set of feasible paths in the product automaton, describes the proposed algorithm, and introduces the proposed recursive defense policy, which yields the highest security performance among the selected set. Finally, Sections VII and VIII provide numerical examples and concluding remarks, respectively.

III. Network Security Model

The systems often contain a wide range of interconnected components that attackers can penetrate from external sources and propagate among the elements. Some of the common components of networks are as follows:

- *Servers*: These devices are the backbone of the network, handling critical services and data. Due to their central role, meticulous security is necessary to ensure uninterrupted operation.
- *Critical Data Devices*: These devices store sensitive or vital data, making them prime targets for attackers. Regular security checks are essential to mitigate the risk of data breaches and ensure data integrity.
- *User Hosts and Devices*: These devices play a crucial role in network security, often connected directly to external sources and servers. They are the primary line of defense, preventing attacks from propagating.
- *Routers*: Routers connect network components and can be vulnerable to attacks that propagate through other connections, transmitting these attacks to other network elements.

The penetration and propagation of network compromises are modeled using Bayesian attack graphs (BAGs) [47]. BAGs are a powerful class of models that capture stochasticity and uncertainty in attack and attack propagation. The nodes of the BAG specify different components in the network, and directed edges specify the connection between different components that can be utilized for the propagation of attack. The weights of each edge specify the probability of attack transmissions, which depends on the type of nodes (e.g., server, data center, user hosts, enterprise servers, etc.) and the security measures in that device. The external attack can be penetrated into the network through components connected to external sources. The external attacks and internal transmissions are modeled through exploit probabilities, which denote the probability that the attack can be successful in compromising the node.

Mathematically, a BAG can be defined by the following tuple:

$$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{E}, \mathcal{P}),$$

where $\mathcal{N} = \{1, \cdots, n\}$ corresponds to the components of the network, $\mathcal{T}$ represents the types of nodes, and $\mathcal{E}$ represents the set of directed edges. We assume that each node belongs to two types: the AND nodes, such as admin servers, and OR nodes, such as SQL servers and user hosts. Each edge $(i, j) \in \mathcal{E}$ is associated with a $\rho_{ij} \in \mathcal{P}$, which shows the probability of exploit through that edge, and $\rho_j \in \mathcal{P}$ denotes the probability of success of the external attack on node j.

IV. Markov Decision Process Representation of Network Security

In this section, we represent network security over the Bayesian attack graph through the Markov decision process. In line with the modeling approach used in our previous work [33], the MDP state can be expressed using a vector, denoted by $\mathbf{x}_k = [\mathbf{x}_k(1), ..., \mathbf{x}_k(n)]$, encapsulating the compromised status of all n components of the network. Here, $\mathbf{x}_k(i)$ takes values of either 0 or 1, with $\mathbf{x}_k(i) = 1$ indicating compromise of the ith component at time step k, and conversely for $\mathbf{x}_k(i) = 0$. The possible network compromises can be expressed through the following 2^n different state vectors: $\{\mathbf{x}^1, \cdots, \mathbf{x}^{2^n}\}$. Attack penetration and propagation rely on factors such as the internal and external exploit probabilities (i.e., ρ_j and ρ_{ij}), node types, and defense strategy. The exploit probabilities are often computed according to the NIST's Common Vulnerability Scoring System (CVSS), which characterizes the severity of vulnerabilities through numerical scores [48]. Fig. 2 depicts the translation of a 6-node BAG into binary strings, representing status changes within the transition dynamics of an MDP.

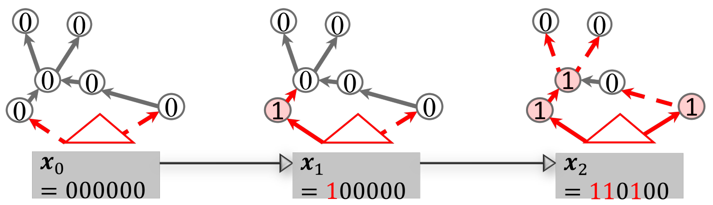


Fig. 2 Translation of collective BAG nodes compromise status into an MDP.

Defense is achieved through restoration, resetting, reimaging, or other means that can eliminate threats at different components of the network. Let $\mathbf{a}_{k-1} \subset \mathcal{N}$ be a subset of nodes selected for defense in time step k. Such defense could often partially eliminate the threats due to the complexity of attack types that might not be eliminated through restoring (i.e., stolen logging or credentials). This paper considers the success probability of attack removal as $(1-\alpha)$, where $0 \leq \alpha \leq 1$ represents the unsuccessful rate of attack removal. The value α depends on the complexity of the attack and reimaging procedure; a more comprehensive defense corresponds to a smaller value α. Let $\mathbf{x}_{k-1}$ be the current state of network compromises. The likelihood of compromise for node j at the time step k upon taking the defense action $\mathbf{a}_{k-1} = \{i_1, \ldots, i_r\} \subset \mathcal{N}$ can be expressed as:

- *AND Nodes:*

$$P(\mathbf{x}_k(j) = 1 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1}) = (1_{j \notin \mathbf{a}_{k-1}} + \alpha 1_{j \in \mathbf{a}_{k-1}}) \left[\rho_j + (1-\rho_j) \prod_{i \in D_j} \rho_{ij} 1_{\mathbf{x}_{k-1}(i)=1} \right], \tag{1}$$

- *OR Nodes:*

$$P(\mathbf{x}_k(j) = 1 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1}) = (1_{j \notin \mathbf{a}_{k-1}} + \alpha 1_{j \in \mathbf{a}_{k-1}}) \left[\rho_j + (1-\rho_j) [1 - \prod_{i \in D_j} \left(1 - \rho_{ij} 1_{\mathbf{x}_{k-1}(i)=1}\right)] \right], \tag{2}$$

where the likelihood is presented if the type of the jth node is AND and OR. AND nodes exhibit resilience against single-parent threats, as compromising an AND node typically necessitates exploits at all of its parent (in-neighbor) nodes. In contrast, the OR nodes can be compromised by just one of their parent nodes being compromised. Higher exploit probabilities and higher external attack frequencies make networks more vulnerable to security breaches. The probability of the jth state variable being uncompromised can also be expressed as: $P(\mathbf{x}_k(j) = 0 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1}) = 1 - P(\mathbf{x}_k(j) = 1 \mid \mathbf{x}_{k-1}, \mathbf{a}_{k-1})$.

V. Problem Formulation for Network Security under Temporal Logical Specifications

This paper utilizes linear temporal logic specifications to express security constraints and derive provably correct security policy. We will first provide the syntax and semantics of the LTL language, and then describe the key elements of the proposed approach.

A. An overview of Linear Temporal Logic (LTL)

The syntax of LTL is built recursively over a set of atomic propositions AP as follows:

$$\phi := ap | \neg ap | \phi \wedge \psi | \phi \vee \psi | \phi U \psi | \phi \rightarrow \psi | X\phi | F\phi | G\phi, \tag{3}$$

where ϕ and ψ are LTL formulae, and $ap \in AP$ is an atomic proposition. An LTL formula can contain the combination of Boolean operators such as negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$) as well as the temporal operators until (U), implication ($\rightarrow$), next (X), eventually (F), and always (G). The semantics of LTL are defined over words w from the alphabet 2^{AP}, i.e., the words consisting of sequences of symbols from the power set of atomic propositions. The infinite trajectories of the system is represented by words. An in-depth discussion of the syntax and semantics of LTL can be found in [35].

An LTL formula has a corresponding graph representation, called Büchi Automaton.

Definition V.1. *(Büchi Automaton) A* Büchi automaton *is a tuple $\mathcal{B} = (Q_{\mathcal{B}}, q_{\mathcal{B}0}, \Sigma, \Delta_{\mathcal{B}}, F_{\mathcal{B}})$, in which $Q_{\mathcal{B}}$ is a finite set of states, $q_{\mathcal{B}0}$ is the initial state, Σ is an alphabet, $\Delta_{\mathcal{B}} : Q_{\mathcal{B}} \times \Sigma \to Q_{\mathcal{B}}$ is a transition function, and $F_{\mathcal{B}} \subseteq Q_{\mathcal{B}}$ is the set of accepting states.*

An accepting state $q \in F_{\mathcal{B}}$ is non-blocking if $\Delta_{\mathcal{B}}(q) \neq \varnothing$. Accordingly, $F^*_{\mathcal{B}} \subset F_{\mathcal{B}}$ denotes the set of all non-blocking accepting states of $\mathcal{B}$. $\Sigma = 2^{AP}$ defines the alphabet. Words from Σ^{ω}, the set of all infinite words from Σ, are accepted by a Büchi automaton if the states in $F^*_{\mathcal{B}}$ are visited infinitely often. Set of all such words forms the language of a Büchi automaton, i.e., $L_{\mathcal{B}}$. There exists a Büchi automaton $\mathcal{B}_{\phi}$ accepting the exact language that satisfies ϕ (i.e., the language $L_{\mathcal{B}_{\phi}} = \left\{w \middle| w \vDash \phi\right\}$) for each formula ϕ over AP.

B. Transition System

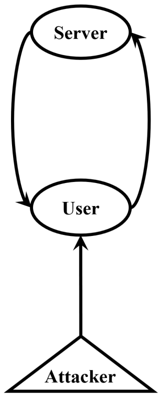
We represent the possible sets of available security actions of the defender and their transitions using the transition system. A general definition of a transition system is as follows:

Definition V.2. *(Transition System) A* Transition System *is a tuple $T = (X, \delta, O, o)$, in which X is a finite set of states, $\delta : X \to 2^X$ is a transition function, O is a set of observations and $o : X \to O$ is an observation map.*

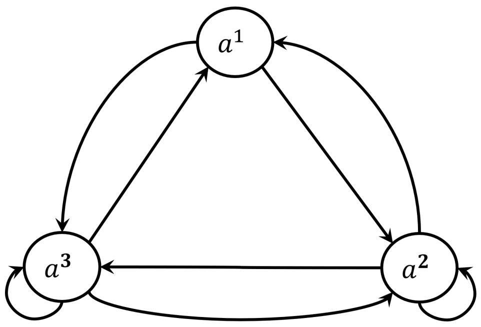
In our setting, the elements of the tuple are as follows:

- X is the set of possible defense actions that can be taken;
- δ denotes the transition relationship between the actions (e.g., the security action $\mathbf{a}^j$ can be performed right after the security action $\mathbf{a}^i$; a self-loop on $\mathbf{a}^i \in X$ indicates the feasibility of consecutively taking the action $\mathbf{a}^i$.
- O is the set of observations that is identical to X.
- o is an identical mapping, i.e., for all $\mathbf{a}^i \in X$, we have $o(\mathbf{a}^i) = \mathbf{a}^i$.

Figure 3 provides an illustrative example of a transition system. On the left plot, you can see a network with server 'a' and device 'b', where user device 'b' is susceptible to external attack. There are three possible actions: $\mathbf{a}^1$: reimage 'a', $\mathbf{a}^2$: reimage 'b', $\mathbf{a}^3$: no reimaging. The transition system consisting of three nodes is presented in Fig. 3(b) (i.e., $X = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$). Edges between nodes indicate the feasible transitions from one security action to another, reflecting the sequential nature of the security process (i.e., $\delta(\mathbf{a}^1) = \{\mathbf{a}^2, \mathbf{a}^3\}, \delta(\mathbf{a}^2) = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}, \delta(\mathbf{a}^3) = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$). It can be seen that the self-loop edges at nodes $\mathbf{a}^1$ do not exist since the consecutive reimaging of the server is not permitted. As mentioned above, the set of observations is $O = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$, and the identical observation mapping is $o(\mathbf{a}^1) = \mathbf{a}^1, o(\mathbf{a}^2) = \mathbf{a}^2, o(\mathbf{a}^3) = \mathbf{a}^3$.



(a) Simple network



(b) transition system

Fig. 3 (a) A simple network with two components: a server 'a', and user host 'b'; (b) the transition system with an action space of size three, $\mathbf{a}^1$: restoring the server, $\mathbf{a}^2$: restoring the user host, $\mathbf{a}^3$: no restoring.

C. Specifications

In this section, we describe the network constraints and how they can be upheld during the defense process. These constraints indicate the desired specifications that should be met by the security policy. Some of the security constraints that can be considered are as follows:

- *Server Restorations*: This specification dictates that servers should not be reimaged in fewer than d_1 consecutive steps to maintain operational continuity. This ensures the prevention of continual disruptions in network operations due to disruptions to critical network components.
- *Data Server Reimaging*: Each data server should be reimaged at least once every d_2 time-steps to minimize the risk of data breaches. By prioritizing these devices, organizations can safeguard sensitive information and prevent unauthorized access.
- *Server Maintenance*: Each server must undergo regular defense processes at least once every d_3 time-steps to ensure the network's overall health and reliability. This specification helps maintain the integrity of network infrastructure and prevents potential vulnerabilities from compromising network security.

We formalize such constraints through LTL specifications. Considering the example problem in Fig. 3(a), let $AP = \{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$ be the set of atomic prepositions. Along with LTL semantics and syntax, the following are examples of LTL specifications that can be expressed for the network:

- After reimaging server 'a', the server must not be reimaged for at least '2' consecutive steps in a row to prevent the disruption of system operations: $G(\mathbf{a}^1 \rightarrow (\neg \mathrm{X}\mathbf{a}^1 \wedge \neg \mathrm{XX}\mathbf{a}^1))$.
- All devices should at some point receive necessary defense: $F(\mathbf{a}^1) \wedge F(\mathbf{a}^2)$.
- Server 'a' should receive a security check at least every four steps: $G(\neg(\neg \mathbf{a}^1 \wedge \neg \mathrm{X}\mathbf{a}^1 \wedge \neg \mathrm{XX}\mathbf{a}^1 \wedge \neg \mathrm{XXX}\mathbf{a}^1))$.

These specifications collectively ensure the desired constraints for the network security system. A wider range of constraints can be defined and formulated in LTL format.

A Büchi automaton for the example described above is provided* in Fig. 4. The nodes of this automaton specify the possible system conditions at any given time. The initial node is specified by 'init'. The edges specify the set of actions that can lead to the transition from one node to another. The states indicated by double circles are the accepting states that need to be visited infinitely often to guarantee the satisfaction of the desired LTL specification (i.e., Büchi acceptance criteria).

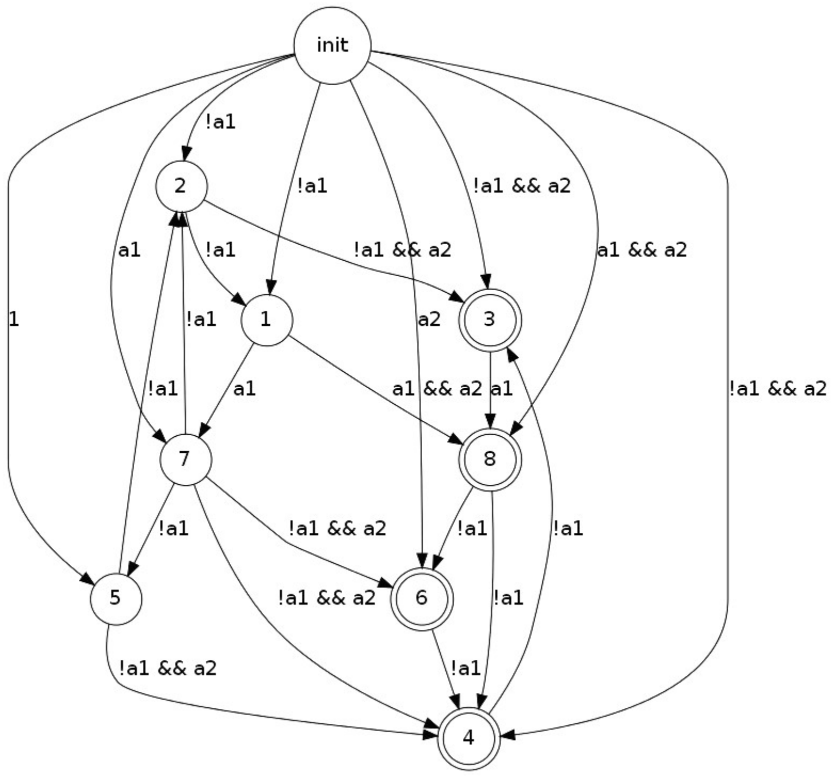


Fig. 4 Büchi automaton for the following specification: $G(\mathbf{a}^1 \rightarrow (\neg \mathbf{X}\mathbf{a}^1 \wedge \neg \mathbf{XX}\mathbf{a}^1)) \wedge F(\mathbf{a}^1) \wedge F(\mathbf{a}^2) \wedge G(\neg(\neg \mathbf{a}^1 \wedge \neg \mathbf{X}\mathbf{a}^1 \wedge \neg \mathbf{XX}\mathbf{a}^1 \wedge \neg \mathbf{XXX}\mathbf{a}^1))$**.**

*The Büchi automaton figure is generated using LTL 2 BA: fast translation from LTL formulae to Büchi automata http://www.lsv.fr/ gastin/ltl2ba/.

D. Product automaton

A product automaton is an essential representation that encodes all feasible transitions of the transition system that ensures the satisfaction of the desired LTL specification. To this end, the formal definition is as follows:

Definition V.3. *(Product automaton) Given a transition system $T = (X, \delta, O, o)$, and a Büchi automaton $B = (Q_{\mathcal{B}}, q_{\mathcal{B}0}, \Sigma, \Delta_{\mathcal{B}}, F_{\mathcal{B}})$, we define the product automaton as a tuple $P = T \times B = (X_P, P_{init}, \Delta_P, F_P)$, where*

- *$X_P = X \times Q_{\mathcal{B}}$ is a finite set of states;*
- *$P_{init} = X \times q_{\mathcal{B}0} \subseteq X_P$ is the set of initial states;*
- *$\Delta_P : X_P \to 2^{X_P}$ is the transition function such that for any two states, $p = (x, q) \in X_P$ and $p' = (x', q') \in X_P$, we have: $x' \in \delta(x)$, and $\Delta_{\mathcal{B}}(q, o(x)) = q'$;*
- *$F_P = (X \times F_{\mathcal{B}}) \subseteq X_P$ is the set of accepting states.*

An accepting state $p \in F_P$ is non-blocking if $\Delta_P(p) \neq \varnothing$. Accordingly, $F_P^* \subset F_P$ denotes the set of all non-blocking accepting states of P. Satisfactory policies are found by using the product automaton and finding accepting trajectories on it. Similarly to the Büchi acceptance criteria, the acceptance criteria for the product automaton also involve visiting the states in F_P^* infinitely often. To represent infinite trajectories with the acceptance condition, we will consider the following structure. Any infinite trajectory over the product automaton will be described in H†*-prefix-suffix* form, where the prefix considered in this format is a finite trajectory of non-repeating nodes that reaches a state in F_P^* in at most H time steps and its corresponding suffixes are all trajectories from the visited accepted state to an element of the set F_P^* in at most H time steps. Overall, once the prefix part of the trajectory is realized, repeating each of the suffixes at consecutive H-steps will ensure visiting a state in F_P^* infinitely often. By planning every H step, we narrow the set of all accepting trajectories to the policies with a *H-prefix-suffix* form.

VI. Obtaining an Efficient Defense Strategy from a Desired Set of Paths in the Product Automaton

After defining all feasible security policies of *H-prefix-suffix* form, the next step is choosing the one yielding the highest security performance in a H-lenght horizon given the latest information (e.g., network compromises). To this end, we propose a systematic method.

Let the security performance be characterized by the reward function $R : \mathcal{X} \times X_P \times \mathcal{X} \to \mathbb{R}$, where $R(\mathbf{x}, (\mathbf{a}, q_{\mathcal{B}}), \mathbf{x}')$ indicates the security gain when the network compromises move from state $\mathbf{x}$ to $\mathbf{x}'$ under defense action $\mathbf{a}$ (disregarding $q_{\mathbf{B}} \in Q_{\mathcal{B}}$). Note that the costs of actions are also incorporated in the reward.

Let $\mathbf{x}_k$ be the current network compromises at time step k, and v_k be the current state of the product automaton. The past sequence of defense actions can be used to compute the current state of the product automaton, denoted by v_k. Performing a graph search, the feasible paths from v_k to a state in F_P^* with a maximum length of H can be expressed as $\Psi(v_k) := \Psi(v_k, H)$, where all paths in $\Psi(v_k)$ are either prefixes or suffixes of a desired *H-prefix-suffix* policy. Let $\psi = \{\psi^0, ..., \psi^{H'-1}\} \in \Psi(v_k)$, where $H' < H$, be a single path for an H horizon. The best (highest expected reward) among all such policies in $\Psi(v_k)$ at time step k can be obtained as a result of the following optimization problem:

$$\psi_k^* = \underset{\psi \in \Psi(v_k)}{\operatorname{argmax}} E\left[\sum_{h=0}^{H'-1} \gamma^h R(\mathbf{x}_{k+h}, \psi^h, \mathbf{x}_{k+h+1}) \mid \mathbf{x}_k\right], \tag{4}$$

where the maximization is over $\Psi(v_k)$ at time step k, and $\gamma \in (0, 1]$ is the discount factor prioritizing the short-term security over the future horizon, and the expectation takes into account the stochasticity in the state and defense processes. The solution in (4) yields the optimal finite-horizon defense policy, ensuring the maximum average accumulated reward over that horizon.

Learning the above policy given the current network compromises $\mathbf{x}_k$ provides a set of actions for the next H' steps and ensures the feasibility of the solution. However, the policy is optimal given the horizon, and the information about the network compromises at time step k. As defense actions are taken, the stochasticity of the process might lead to different network compromises, which can be taken into account to adjust the policy further. Therefore, this paper performs optimal sequential defense policy that adapts itself given the latest network compromises and ensures satisfying the specification in an infinite horizon.

†In the context of this paper, H is a pre-defined parameter provided by the user.

We further expand the expectation in (4) to compute the following policy:

$$\psi_k^* = \underset{\psi \in \Psi(v_k)}{\operatorname{argmax}} \left[\sum_{h=0}^{H'-1} \gamma^h \sum_{\mathbf{x}_{k+h}, \mathbf{x}_{k+h+1}} p(\mathbf{x}_{k+h}, \mathbf{x}_{k+h+1} \mid \mathbf{x}_k, \psi^{0:h}) R(\mathbf{x}_{k+h}, \psi^h, \mathbf{x}_{k+h+1}) \right], \tag{5}$$

where the joint distribution of the states is used instead of expectation in (4) and the optimization solution ensures selecting the path with the highest expected accumulated reward.

The posterior distribution of the network compromises at time step $k+h$, $h>0$, given the last observation $\mathbf{x}_k$ and following ψ can be expressed as:

$$\mathbf{\Pi}^{\psi}_{k+h|k}(i) = P\left(\mathbf{x}_{k+h} = \mathbf{x}^i \mid \mathbf{x}_k, \psi^{0:h}\right), i = 1, ..., 2^n, \tag{6}$$

where the distribution of network compromises at time step k can be expressed as:

$$\mathbf{\Pi}^{\psi}_{k|k}(i) = 1_{\mathbf{x}^i = \mathbf{x}_k}, i = 1, ..., 2^n, \psi \in \Psi(v_k), \tag{7}$$

with the indicator $1_{\mathbf{x}^i=\mathbf{x}_k}$ assigns 1 to a single element of the posterior distribution.

We define the transition matrix $M(\mathbf{a})$ under the defense action $\mathbf{a}$ as:

$$(M(\mathbf{a}))_{ij} = P(\mathbf{x}_k = \mathbf{x}^i \mid \mathbf{x}_{k-1} = \mathbf{x}^j, \mathbf{a}_{k-1} = \mathbf{a}) = \prod_{l=1}^{n} \left(\eta^j_{l,\mathbf{a}} 1_{\mathbf{x}^i(l)=1} + (1-\eta^j_{l,\mathbf{a}}) 1_{\mathbf{x}^i(l)=0} \right), \tag{8}$$

for $i, j = 1, \dots, 2^n$; where $1_{\mathbf{x}^i(l)=1}$ is the indicator, taking 1 if $\mathbf{x}^i(l) = 1$ and 0 otherwise, and

$$\begin{aligned} \eta^j_{l,\mathbf{a}} = {} & (1_{l \notin \mathbf{a}} + \alpha 1_{l \in \mathbf{a}}) \left[\rho_l + (1-\rho_l) \prod_{r \in D_l} \rho_{rl} 1_{\mathbf{x}^j(r)=1} \right] 1_{\mathcal{N}_l = \text{AND}} \\ & + (1_{l \notin \mathbf{a}} + \alpha 1_{l \in \mathbf{a}}) \left[\rho_l + (1-\rho_l) \left(1 - \prod_{r \in D_l} (1 - \rho_{rl} 1_{\mathbf{x}^j(r)=1}) \right) \right] 1_{\mathcal{N}_l = \text{OR}}. \end{aligned} \tag{9}$$

Note that $1_{\mathcal{N}_l=\text{AND}}$ is 1 if the lth node type is AND and 0 otherwise. The transition probabilities in (8) and (9) are derived from the conditional probabilities for AND and OR nodes in (1) and (2).

The predictive posterior distribution of network compromises in (6) can be expressed using the transition matrices following ψ as:

$$\mathbf{\Pi}^{\psi}_{k+h+1|k} = M(\psi^h) ... M(\psi^0) \, \mathbf{\Pi}^{\psi}_{k|k}, \tag{10}$$

Note that for $\psi^i = (\mathbf{a}^i, q^i_{\mathcal{B}}) \in X_p$, we define $M(\psi^i) = M(\mathbf{a}^i)$. The predictive posterior can also be computed recursively as:

$$\mathbf{\Pi}^{\psi}_{k+h+1|k} = M(\psi^h) \, \mathbf{\Pi}^{\psi}_{k+h|k}, \tag{11}$$

Similarly, the joint distribution of network compromises in two consecutive times $k+h$ and $k+h+1$ can be expressed as:

$$M(\psi^h)^T \odot \left(1_{1 \times 2^n} \otimes \mathbf{\Pi}^{\psi}_{k+h|k} \right), \tag{12}$$

where " ⊗ " is the Kronecker product, and "⊙" denotes the component-wise multiplication of two matrix.

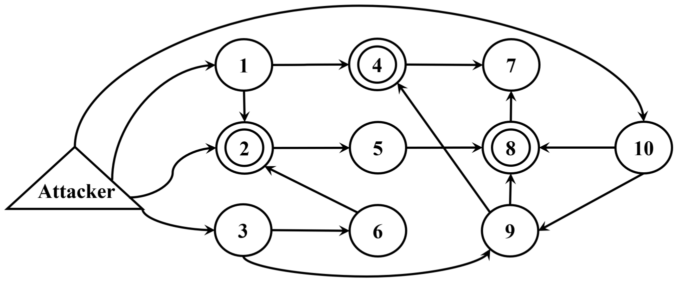
For a given state $\psi^l = (\mathbf{a}, q_{\mathcal{B}}) \in X_P$, the matrix-form of the reward function can be expressed as:

$$(\mathbf{R}(\psi^l))_{ij} = R\left(\mathbf{x}^j, \psi^l, \mathbf{x}^i\right), \text{ for } i, j = 1, .., 2^n. \tag{13}$$

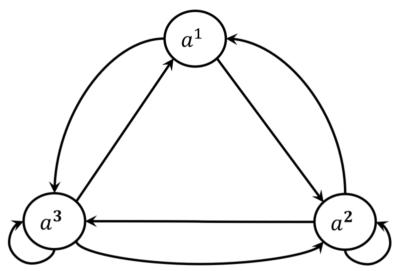
Then, given (5) and (10)-(13), the desired optimal policy at time k can be expressed as:

$$\begin{aligned} \psi_k^* = {} & \underset{\psi \in \Psi}{\operatorname{argmax}} \left[\sum_{h=0}^{H'-1} \gamma^h \left\| \left[M(\psi^h)^T \odot (1_{1 \times 2^n} \otimes \mathbf{\Pi}^{\psi}_{k+h|k}) \right] \odot \mathbf{R}^T(\psi^h) \right\|_1 \right], \\ & \text{s.t. } \mathbf{\Pi}^{\psi}_{k+h+1|k} = M(\psi^h) \mathbf{\Pi}^{\psi}_{k+h|k}, h = 0, ..., H-1. \end{aligned} \tag{14}$$

Upon selecting the best path among the H-length feasible paths of desired format, the corresponding defense action sequence should be applied to the system over the next H' horizon as: $[\mathbf{a}_k, ..., \mathbf{a}_{k+H'-1}]$, where



(a) Simple network



(b) transition system

Fig. 5 (a) A computer network consisting of ten nodes/components; (b) the transition system consisting of three actions, $\mathbf{a}^1$: restoring node 1, $\mathbf{a}^2$: restoring node 2, $\mathbf{a}^3$: no restoring.

$\psi_k^* = [(\mathbf{a}_k, q_{\mathcal{B}_k}), ..., (\mathbf{a}_{k+H'-1}, q_{\mathcal{B}_{k+H'-1}})]$. Performing such policies every H (or H') step to reach a trajectory of the form *H-prefix-suffix*, ensures the specification is upheld while optimizing for security rewards. The last network compromises upon performing the policy can be used to select the policy on the next horizon. Starting from $\mathbf{x}_0$, and considering an initial state of P for the automaton, based on the search space definition, the goal in the first H steps is to reach a non-blocking accepting state, termed a desired prefix in the *H-prefix-suffix* format. For the repetition of suffixes, let $\mathbf{x}_{k+H}$ be the network state at time step $k + H$ and v_{k+H} be the corresponding accepting state of P at time $k + H$. Then, we can use $\Psi(v_{k+H})$ and compute the optimal path according to the new initial posterior distribution $\mathbf{\Pi}_{k+H|k+H}(i) = 1, \mathbf{x}^i = \mathbf{x}_{k+H}$. The selection of the best path among *H-prefix-suffix* guarantees meeting specifications over an infinite horizon. Note that when the optimal path length is smaller than the pre-specified value H, re-planning occurs after H' time steps, where $H' < H$.

The detailed steps of the proposed security framework are provided in Algorithm 1. The transition system and specifications (Büchi automaton) are used to compute the product automaton for the network. The feasible paths/policies are computed according to the initial status of the automaton. During the execution, given the posterior distribution of the network compromises, the expected security reward of each feasible path of the desired format is computed, and the policy with the maximum expected security rewards in a horizon is followed. The selected action sequence is applied to the network, and specification is not violated over a finite or infinite horizon. The network compromises at the end of the horizon can be used for the selection of the next best sequence of actions among the same feasible paths.

The desired path finder algorithm (offline step) worst-case time complexity with a modified Depth-first search (DFS) is $O(|X_P|^{H+1})$, as it obtains all simple paths of maximum length H starting from every node $v \in X_P$ and ending in a non-blocking accepting state in F_P^*. Since generating the presumable trajectory requires calculating the next step posteriors, the time complexity of choosing an optimal path for the next H steps in the online step is $O(|X_P|^H \times H \times 2^{2n})$. This represents the worst-case complexity; however, the implementation can be optimized based on the graph's structure and the specific problem conditions.

VII. Numerical Experiments

The numerical experiments in this section assess the performance of the defense strategy. The experiment is run on a network comprising 10 nodes, shown in Figure 5(a). The network contains three AND nodes and seven OR nodes. The nodes 1, 2, 3, and 10 are under external attacks. The network vulnerabilities indicated by ρ_{ij} are considered as: $\rho_{12} = 0.7, \rho_{14} = 0.6, \rho_{25} = 0.6, \rho_{36} = 0.55, \rho_{39} = 0.7, \rho_{47} = 0.7, \rho_{58} = 0.7, \rho_{62} = 0.7, \rho_{87} = 0.7, \rho_{94} = 0.6, \rho_{98} = 0.7, \rho_{10\,8} = 0.7, \rho_{10\,9} = 0.7$, with the external attack probability as: $\rho_1 = 0.65, \rho_2 = 0.6, \rho_3 = 0.2, \rho_{10} = 0.6$. A uniform prior is considered for the initial network compromise. Also, the reimaging miss-rate is set as $\alpha = 0.1$. All results of the numerical experiments are averaged over 100 independent runs/trajectories.

The defense can be applied on nodes 1 and 2 through reimaging or restoring. The space of defense action can be expressed as follows:

$$X = \{\mathbf{a}^1 = \{1\}, \mathbf{a}^2 = \{2\}, \mathbf{a}^3 = \{\}\}. \tag{15}$$

The defender's transition system has three nodes, where corresponding actions and components are described in Fig. 5(b). The following desired specifications are considered for the network: 1) After reimaging of node '1', this node must not be reimaged for at least '2' consecutive steps in a row: $G(\mathbf{a}^1 \rightarrow (\neg X\mathbf{a}^1 \wedge \neg XX\mathbf{a}^1))$; 2) devices in node 1

Algorithm 1 Defense Policy Optimization with Linear Temporal Logic Specifications for Bayesian Attack Graphs

```
Inputs: Defense action space $X$, number of network components $n$, transition matrices $M(\mathbf{a})$, desired specifications $\Phi$, horizon $H$, transition system $T$, initial network compromises $\mathbf{x}_0$, security reward function $(\mathbf{R}(\mathbf{a}))_{ij} : R(\mathbf{x}^i, \mathbf{a}, \mathbf{x}^j)$.
Offline Step
 1: Construct Büchi automaton $B$ from $\Phi$.
 2: Compute the product of Büchi automaton $B$ and the transition system $T$ to obtain the product automaton $P$.
 3: Obtain $F_P^* \subset F_P$, the non-blocking accepting states of $P$.
 4: For all $v \in X_P$ compute: $\Psi(v) = \Psi(v, H)$.
Online Step: Real-time Defense
 5: Initial network compromises, $\mathbf{x}_0$, initial product automaton state $v_0$, and $k = 0$.
 6: repeat
 7:     $\mathbf{\Pi}_{k|k}(i) = 1_{\mathbf{x}^i = \mathbf{x}_k}, i = 1 ..., 2^n$.
 8:     $V(\psi) = 0, \psi \in \Psi(v)$.
 9:     for $\psi = [\psi^1, ..., \psi^{H'}] \in \Psi(v)$ do
10:         for $h \in \{0, ..., H' - 1\}$ do
11:             $V(\psi) = V(\psi) + \gamma^h \left\| \left[ M(\psi^h)^T \odot (1_{1\times 2^n} \otimes \mathbf{\Pi}^{\psi}_{k+h|k}) \right] \odot \mathbf{R}(\psi^h) \right\|_1$
12:             $\mathbf{\Pi}^{\psi}_{k+h+1|k} = M(\psi^{h+1}) \mathbf{\Pi}^{\psi}_{k+h|k}$
13:         end for
14:     end for
15:     $\psi_k^* = \text{argmax}_{\psi \in \Psi(v_k)} V(\psi)$
16:     Take security actions $[\mathbf{a}_k, ..., \mathbf{a}_{k+H'-1}]$ following $\psi_k^*$, update $v_{k+H'}$ (states of the automaton) accordingly, and get the last network compromises $\mathbf{x}_{k+H'}$.
17:     $k \leftarrow k + H'$
18: until stopping criterion is met
```

and node 2 should at some point receive necessary defense: $F(\mathbf{a}^1) \wedge F(\mathbf{a}^2)$; 3) node 1 should receive a security check at least every four steps: $G(\neg(\neg\mathbf{a}^1 \wedge \neg\mathrm{X}\mathbf{a}^1 \wedge \neg\mathrm{XX}\mathbf{a}^1 \wedge \neg\mathrm{XXX}\mathbf{a}^1))$.

The entire specification can be represented as follows: $G(\mathbf{a}^1 \to (\neg\mathrm{X}\mathbf{a}^1 \wedge \neg\mathrm{XX}\mathbf{a}^1)) \wedge F(\mathbf{a}^1) \wedge F(\mathbf{a}^2) \wedge G(\neg(\neg\mathbf{a}^1 \wedge \neg\mathrm{X}\mathbf{a}^1 \wedge \neg\mathrm{XX}\mathbf{a}^1 \wedge \neg\mathrm{XXX}\mathbf{a}^1))$. The Büchi diagram and product automaton are constructed given the transition system and specifications.

We compare the performance of the proposed method with two common defense policies in network security: 1) Dynamic programming (DP), which provides the infinite horizon defense solution as a form of Markovian policy. This policy does not account for specification, but it is guaranteed to yield the highest expected security results. 2) Greedy defense policy includes taking the defense action that yields the best expected security solution in the next time step. Our proposed defense policy approach is similar to a finite-horizon dynamic programming solution in the sense that both methods utilize an H-step lookahead to propose a policy. For the proposed method, the best solution is then selected from a subset of all possible policies that can uphold the given specifications.

In the first result shown in Figure 6, the average accumulated reward, or equivalently negative of the average number of compromises, for three methods are presented. One can see that dynamic programming yields the lowest average accumulated reward since its search space is for all policies rather than a set of feasible policies similar to our method. The Greedy policy performed worse than DP but was slightly better than our method. This comes from the Greedy policy's lack of lookahead consideration. Finally, the proposed policy has shown more fluctuation and higher average compromises, which results from the restrictions that are imposed by the proposed method to ensure specifications are met.

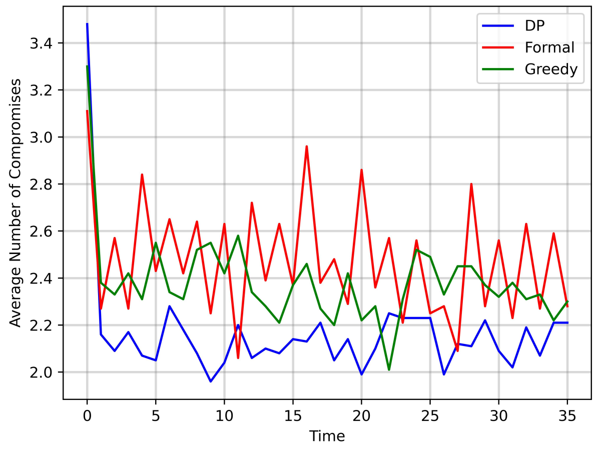


Fig. 6 Average number of compromises per step obtained by different defense policies.

To better understand the shortcomings of existing approaches to account for specifications, Figure 7 represents the average number of violations of four specifications per time step. In practice, these specifications are hard constraints that should not be violated during decision-making. The Markovian aspects of the DP policy prevent accounting for such constraints defined over time, which can be seen in the largest violation in Figure 7. In fact, 1.75 out of 4 constraints are violated at each step, which indicates the unsuitability of such policy in practice. Meanwhile, the better performance of the DP method in terms of an average number of compromises can also be justified using this. Similar to DP, the Greedy policy violates the specifications, while it is less than the DP method. As expected, the proposed method yields zero violation of the constraints since the defense policy is only selected among the existing paths in the product automaton. Therefore, the larger number of compromises by the proposed method in Figure 6 comes with guaranteeing upholding the specifications during the defense process.

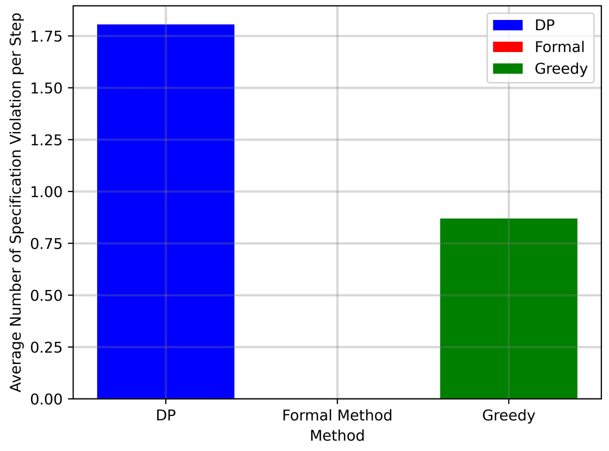


Fig. 7 Average number of violations of specifications per step for different defense policies.

The impact of the horizon used by the proposed method is considered in this section. Horizon 4 is considered the smallest value since it guarantees to reach the accepted state in the product automaton and to circle that state for a long period. Since this horizon also specifies the lookahead aspect of the defense process by the proposed policy, we investigate the impact of the horizon on security performance. The results are compared with the optimal finite-horizon

Table 1 Average number of compromises and violations per step for proposed policy and finite-horizon dynamic programming method with different horizons.

Method	$H = 4$	$H = 4$	$H = 6$	$H = 6$
	#Compromises	#Violations	#Compromises	#Violations
Formal Method	2.417	0	2.594	0
Finite Horizon DP	2.161	1.8	2.152	1.8
	$H = 8$	$H = 8$	$H = 10$	$H = 10$
	#Compromises	#Violations	#Compromises	#Violations
Formal Method	2.537	0	2.477	0
Finite Horizon DP	2.120	1.8	2.107	1.8

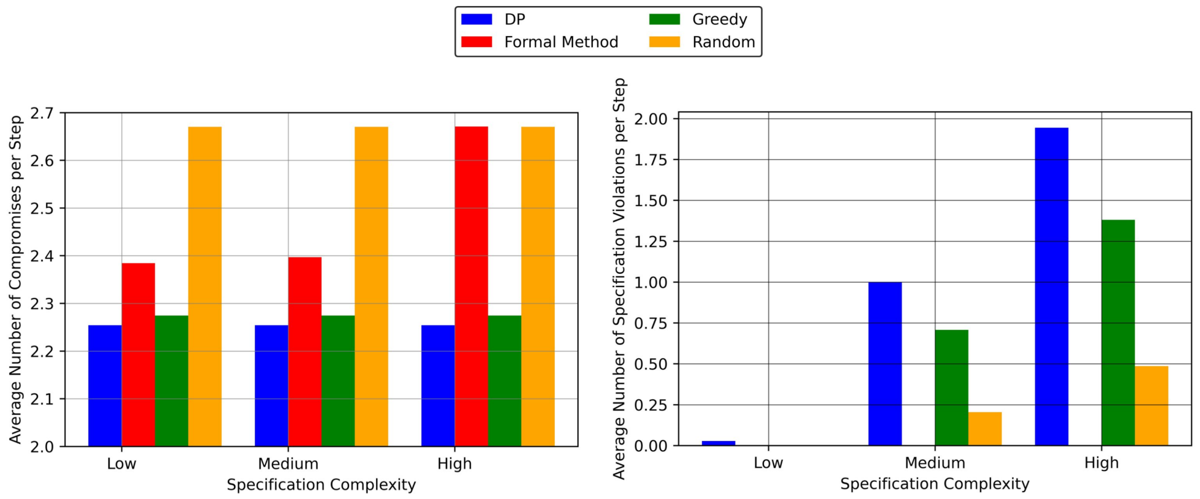


Fig. 8 (a) Average compromises and (b) average specification violations per step across different defense methods for specifications of varying complexity.

dynamic programming, which optimally accounts for security reward over the entire policies. The average number of network compromises for both methods is presented in Table 1. The number of violations by the proposed policy is zero at all conditions since the feasible path is achievable for horizons larger than 4. As the horizon gets larger, the performance of the proposed method becomes slightly worse. This comes from the fact that the re-planning takes place after performing the actions on the horizon. Thus, for smaller horizons, such re-planning occurs more often, leading to defense actions that are more aligned with the network compromises. For the finite horizon DP policy, one can see the improvement in defense performance as the horizon length increases. This comes from more lookahead aspects of the policy to account for future network crises when selecting defense action. However, one can see that such better security performance comes with a high cost of 1.8 out of 4 violations of constraints at each step. Thus, the results further indicate the capability of the proposed policy to find the security solution that guarantees holding specifications and achieving proper security performance.

In the final experiment, we analyzed three distinct specifications, each imposing a different level of constraints on reimaging actions: minimal, moderate, and high. The minimal constraint specification is represented as: $F(\mathbf{a}^1) \wedge F(\mathbf{a}^2)$. This ensures that both nodes $\mathbf{a}^1$ and $\mathbf{a}^2$ must receive necessary defense actions at some point within the planning horizon. There are no restrictions on the order or timing of these actions. The moderate constraint specification adds limitations on the reimaging process, requiring that once a node is reimaged, it cannot be reimaged again in the subsequent step. This can be expressed as: $F(\mathbf{a}^1) \wedge F(\mathbf{a}^2) \wedge G(\mathbf{a}^1 \rightarrow \neg \mathrm{X}\mathbf{a}^1) \wedge G(\mathbf{a}^2 \rightarrow \neg \mathrm{X}\mathbf{a}^2)$. The high-constraint specification introduces additional rules, focusing on more stringent requirements for reimaging node $\mathbf{a}^1$. It mandates a minimum two-step delay before $\mathbf{a}^1$ can be reimaged again and requires that $\mathbf{a}^1$ is reimaged at least once every four steps. This specification can be written as: $F(\mathbf{a}^1) \wedge F(\mathbf{a}^2) \wedge G(\mathbf{a}^1 \rightarrow (\neg \mathrm{X}\mathbf{a}^1 \wedge \neg \mathrm{XX}\mathbf{a}^1)) \wedge G(\mathbf{a}^2 \rightarrow \neg \mathrm{X}\mathbf{a}^2) \wedge G(\neg(\neg \mathbf{a}^1 \wedge \neg \mathrm{X}\mathbf{a}^1 \wedge \neg \mathrm{XX}\mathbf{a}^1 \wedge \neg \mathrm{XXX}\mathbf{a}^1))$.

In the final experiment, we evaluate the infinite-horizon dynamic programming approach, our proposed Formal method (with $H = 4$), as well as the Greedy and Random algorithms. The evaluation focuses on two metrics: the average number of compromises the network experiences over 36 time-steps and the average number of specification violations for each method. Figure 8(a) illustrates the average number of compromises. As expected, the DP approach, explicitly designed to minimize compromises over an infinite horizon, achieved the best performance with an average of 2.25 compromises. The Greedy algorithm performed slightly worse, while our proposed Formal method kept compromises below 2.4 under low and moderate constraint levels. However, under highly constrained specifications, the number of compromises for our method increased slightly to 2.66. The Random algorithm demonstrated the weakest performance, averaging 2.66 compromises. The DP, Greedy, and Random methods displayed insensitivity to changes in specification complexity, maintaining consistent compromise rates regardless of the imposed constraints.

Figure 8(b) depicts the average number of specification violations. Under minimal constraints, almost all methods adhere to the specification, ensuring at least one reimaging action per node within the horizon. However, even under these light constraints, the DP approach occasionally violates the specification. The strength of our proposed method becomes evident as the complexity of the specification increases: it ensures zero violations, regardless of the constraints. In contrast, other methods, particularly DP and Greedy, show a significant increase in violations under stricter constraints. For highly constrained specifications, these approaches average more than one violation per time step. Interestingly, the Random approach sometimes results in fewer specification violations, although this is achieved at the cost of a higher number of compromises due to its lack of strategic decision-making.

In summary, the results of this experiment demonstrate the advantages of our proposed Formal method. It guarantees adherence to constraints on action sequences and dependencies, with zero specification violations across all scenarios. When constraints are minimal, its performance is closer to DP, as the method has greater flexibility. However, as constraints tighten, our approach diverges to satisfy the constraints and maintain compliance. In contrast, non-formal methods like DP and Greedy do not explicitly account for constraints. Although the number of compromises remains consistent regardless of specification complexity, the number of violations increases significantly as the constraints become stricter. This highlights the importance of integrating formal methods into decision-making processes for scenarios with complex constraints.

VIII. Conclusion

This paper develops a defense policy for the security of networks modeled by Bayesian attack graphs (BAGs) with a set of practical security constraints. Our approach utilizes linear temporal logic specifications to formally represent critical system requirements, including resource limitations, network security, and maintenance needs. By constructing a product automaton and employing Linear Temporal Logic (LTL), we compute feasible defense policies that guarantee adherence to these constraints. Furthermore, we developed an efficient defense policy that selects the policy among a set of feasible policies with a specific format, yielding the highest expected lookahead security performance in every horizon of length H. Numerical experiments validate the effectiveness of our framework in terms of meeting security specifications and security performance.

Acknowledgments

Research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-24-2-0166. This work was also supported by the National Science Foundation award IIS-2311969, ARMY Research Laboratory award W911NF-23-2-0207, ARMY Research Office award W911NF-21-1-0299, and Office of Naval Research award N00014-23-1-2850.

References

[1] Dave, G., Choudhary, G., Sihag, V., You, I., and Choo, K.-K. R., "Cyber security challenges in aviation communication, navigation, and surveillance," *Computers & Security*, Vol. 112, 2022, p. 102516.

[2] Swinney, C. J., and Woods, J. C., "A review of security incidents and defense techniques relating to the malicious use of small unmanned aerial systems," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 37, No. 5, 2022, pp. 14–28.

[3] Alsulami, A. A., and Zein-Sabatto, S., "Resilient cyber-security approach for aviation cyber-physical systems protection against sensor spoofing attacks," *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2021, pp. 0565–0571.

[4] Zhuo, M., Liu, L., Zhou, S., and Tian, Z., "Survey on security issues of routing and anomaly detection for space information networks," *Scientific Reports*, Vol. 11, No. 1, 2021, p. 22261.

[5] Ahmad, R. W., Hasan, H., Yaqoob, I., Salah, K., Jayaraman, R., and Omar, M., "Blockchain for aerospace and defense: Opportunities and open research challenges," *Computers & Industrial Engineering*, Vol. 151, 2021, p. 106982.

[6] O'Mahony, G. D., Curran, J. T., Harris, P. J., and Murphy, C. C., "Interference and intrusion in wireless sensor networks," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 35, No. 2, 2020, pp. 4–16.

[7] Lin, Y., Ghoreishi, S. F., Lan, T., and Imani, M., "High-level human intention learning for cooperative decision-making," *2024 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2024, pp. 209–216.

[8] Ukwandu, E., Ben-Farah, M. A., Hindy, H., Bures, M., Atkinson, R., Tachtatzis, C., Andonovic, I., and Bellekens, X., "Cyber-security challenges in aviation industry: A review of current and future trends," *Information*, Vol. 13, No. 3, 2022, p. 146.

[9] Zhou, D., Sheng, M., Li, J., and Han, Z., "Aerospace integrated networks innovation for empowering 6G: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[10] Maleh, Y., "Machine learning techniques for IoT intrusions detection in aerospace cyber-physical systems," *Machine Learning and Data Mining in Aerospace Technology*, 2020, pp. 205–232.

[11] Gupta, B. B., Gaurav, A., Marín, E. C., and Alhalabi, W., "Novel graph-based machine learning technique to secure smart vehicles in intelligent transportation systems," *IEEE transactions on intelligent transportation systems*, 2022.

[12] Ravari, A., Jiang, G., Zhang, Z., Imani, M., Thomson, R. H., Pyke, A. A., Bastian, N. D., and Lan, T., "Adversarial Inverse Learning of Defense Policies Conditioned on Human Factor Models," *Proceedings of the 57th Asilomar Conference on Signals, Systems, and Computers*, 2024.

[13] Asadi, N., Hosseini, S. H., Imani, M., Aldrich, D. P., and Ghoreishi, S. F., "Privacy-preserved federated reinforcement learning for autonomy in signalized intersections," *International Conference on Transportation and Development 2024*, 2024, pp. 390–403.

[14] Garcia, A. B., Babiceanu, R. F., and Seker, R., "Artificial intelligence and machine learning approaches for aviation cyber-security: An overview," *2021 Integrated Communications Navigation and Surveillance Conference (ICNS)*, IEEE, 2021, pp. 1–8.

[15] Ouiazzane, S., Addou, M., and Barramou, F., "A multiagent and machine learning based denial of service intrusion detection system for drone networks," *Geospatial Intelligence: Applications and Future Trends*, 2022, pp. 51–65.

[16] Baig, Z., Syed, N., and Mohammad, N., "Securing the smart city airspace: Drone cyber attack detection through machine learning," *Future Internet*, Vol. 14, No. 7, 2022, p. 205.

[17] Yazıcıoglu, A. Y., Egerstedt, M., and Shamma, J. S., "Communication-free distributed coverage for networked systems," *IEEE Transactions on Control of Network Systems*, Vol. 4, No. 3, 2016, pp. 499–510.

[18] Zhang, Z., Imani, M., and Lan, T., "Modeling other players with Bayesian beliefs for games with incomplete information," *arXiv preprint arXiv:2405.14122*, 2024.

[19] Alali, M., Kazeminajafabadi, A., and Imani, M., "Deep reinforcement learning sensor scheduling for effective monitoring of dynamical systems," *Systems Science & Control Engineering*, Vol. 12, No. 1, 2024, p. 2329260.

[20] Boldyrikhin, N. V., Safaryan, O. A., Razumov, P. V., Porksheyan, V. M., Smirnov, I. A., Korochentsev, D. A., Cherckesova, L. V., and Romanov, A. M., "Controlling the Resources of the Intrusion Detection System at Network Objects Monitoring," *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, 2020, pp. 1–6.

[21] Alali, M., and Imani, M., "Kernel-Based Particle Filtering for Scalable Inference in Partially Observed Boolean Dynamical Systems," *IFAC-PapersOnLine, 20th IFAC Symposium on System Identification (SYSID 2024)*, Elsevier, 2024.

[22] Yoon, S., Cho, J.-H., Kim, D. S., Moore, T. J., Free-Nelson, F., and Lim, H., "Attack graph-based moving target defense in software-defined networks," *IEEE Transactions on Network and Service Management*, Vol. 17, No. 3, 2020, pp. 1653–1668.

[23] Wohlfart, E., Schauer, S., and Holz, T., "Bayesian attack graphs: An advanced probabilistic model for security risk assessments," *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ACM, 2015, pp. 783–794.

[24] Wohlfart, E., Schauer, S., and Holz, T., “Bayesian attack graphs: Security risk assessment and probabilistic graphical models,” *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ACM, 2013, pp. 621–632.

[25] Li, K., and Koutsoukos, X., “Bayesian Attack Graphs: A New Approach for Modeling Security Risks in Computer Networks,” *IEEE Transactions on Dependable and Secure Computing*, Vol. 16, No. 3, 2019, pp. 386–399.

[26] Poolsappasit, N., Dewri, R., and Ray, I., “Dynamic security risk management using Bayesian attack graphs,” *IEEE Transactions on Dependable and Secure Computing*, Vol. 9, No. 1, 2011, pp. 61–74.

[27] Kazeminajafabadi, A., Ghoreishi, S. F., and Imani, M., “Optimal Detection for Bayesian Attack Graphs under Uncertainty in Monitoring and Reimaging,” *2023 American Control Conference (ACC)*, IEEE, 2024.

[28] Muñoz-González, L., Sgandurra, D., Barrère, M., and Lupu, E. C., “Exact inference techniques for the analysis of Bayesian attack graphs,” *IEEE Transactions on Dependable and Secure Computing*, Vol. 16, No. 2, 2017, pp. 231–244.

[29] Ravari, A., Ghoreishi, S. F., and Imani, M., “Implicit Human Perception Learning in Complex and Unknown Environments,” *American Control Conference (ACC)*, IEEE, 2024.

[30] Ravari, A., Ghoreishi, S. F., and Imani, M., “Optimal Inference of Hidden Markov Models Through Expert-Acquired Data,” *IEEE Transactions on Artificial Intelligence*, 2024.

[31] Asvija, B., Eswari, R., and Bijoy, M., “Bayesian attack graphs for platform virtualized infrastructures in clouds,” *Journal of Information Security and Applications*, Vol. 51, 2020, p. 102455.

[32] Miehling, E., Rasouli, M., and Teneketzis, D., “Optimal defense policies for partially observable spreading processes on Bayesian attack graphs,” *Proceedings of the second ACM workshop on moving target defense*, 2015, pp. 67–76.

[33] Kazeminajafabadi, A., and Imani, M., “Optimal Joint Defense and Monitoring for Networks Security Under Uncertainty: A POMDP-Based Approach,” *IET Information Security*, 2024.

[34] Matthews, I., Mace, J., Soudjani, S., and van Moorsel, A., “Cyclic Bayesian attack graphs: a systematic computational approach,” *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, 2020, pp. 129–136.

[35] Baier, C., and Katoen, J.-P., *Principles of model checking*, MIT press, 2008.

[36] Aksaray, D., Yazıcıoğlu, Y., and Asarkaya, A. S., “Probabilistically guaranteed satisfaction of temporal logic constraints during reinforcement learning,” *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 6531–6537.

[37] Pant, Y. V., Yin, H., Arcak, M., and Seshia, S. A., “Co-design of control and planning for multi-rotor uavs with signal temporal logic specifications,” *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 4209–4216.

[38] Büyükkoccak, A. T., Hu, Y., Taheri, A., Aksaray, D., and Gebre-Egziabher, D., “State-Estimation-Aware Planning for Autonomous Systems with Temporal Logic Specifications,” *AIAA SCITECH 2023 Forum*, 2023, p. 2665.

[39] Pantazides, A., Aksaray, D., and Gebre-Egziabher, D., “Satellite mission planning with signal temporal logic specifications,” *AIAA SCITECH 2022 Forum*, 2022, p. 1091.

[40] Buyukkocak, A. T., Aksaray, D., and Yazıcıoğlu, Y., “Energy-Aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks with Temporal Logic Specifications,” *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 8659–8665.

[41] Buyukkocak, A. T., Aksaray, D., and Yazıcıoğlu, Y., “Sequential control barrier functions for mobile robots with dynamic temporal logic specifications,” *Robotics and Autonomous Systems*, Vol. 176, 2024, p. 104681.

[42] Aksaray, D., “Resilient satisfaction of persistent and safety specifications by autonomous systems,” *AIAA Scitech 2021 Forum*, 2021, p. 1124.

[43] Buyukkocak, A. T., Aksaray, D., and Yazıcıoğlu, Y., “Planning of heterogeneous multi-agent systems under signal temporal logic specifications with integral predicates,” *IEEE Robotics and Automation Letters*, Vol. 6, No. 2, 2021, pp. 1375–1382.

[44] Coogan, S., Gol, E. A., Arcak, M., and Belta, C., “Traffic network control from temporal logic specifications,” *IEEE Transactions on Control of Network Systems*, Vol. 3, No. 2, 2015, pp. 162–172.

[45] Belta, C., Yordanov, B., and Gol, E. A., *Formal methods for discrete-time dynamical systems*, Vol. 89, Springer, 2017.

[46] Pant, Y. V., Abbas, H., Quaye, R. A., and Mangharam, R., “Fly-by-logic: Control of multi-drone fleets with temporal logic objectives,” *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 186–197.

[47] Kazeminajafabadi, A., and Imani, M., “Optimal monitoring and attack detection of networks modeled by Bayesian attack graphs,” *Cybersecurity*, Vol. 6, No. 1, 2023, p. 22.

[48] Mell, P., Scarfone, K., Romanosky, S., et al., “A complete guide to the common vulnerability scoring system version 2.0,” *Published by FIRST-forum of incident response and security teams*, Vol. 1, 2007, p. 23.