Optimal Defense Policies for Partially Observable Spreading Processes on Bayesian Attack Graphs

Erik Miehling
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI
miehling@umich.edu

Mohammad Rasouli
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI
rasouli@umich.edu

Demosthenis Teneketzis
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI
teneket@umich.edu

ABSTRACT

The defense of computer networks from intruders is becoming a problem of great importance as networks and devices become increasingly connected. We develop an automated approach to defending a network against continuous attacks from intruders, using the notion of Bayesian attack graphs to describe how attackers combine and exploit system vulnerabilities in order to gain access and progress through a network. We assume that the attacker follows a probabilistic spreading process on the attack graph and that the defender can only partially observe the attacker's capabilities at any given time. This leads to the formulation of the defender's problem as a partially observable Markov decision process (POMDP). We define and compute optimal defender countermeasure policies, which describe the optimal countermeasure action to deploy given the current information.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory, Dynamic programming*; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; G.3 [**Probability and Statistics**]: Markov processes

Keywords

Network security; moving target defense; bayesian attack graphs; stochastic control; POMDP

1. INTRODUCTION

The increasing connectivity of networks and smart devices allows for greater efficiency and flexibility in the operation of complex networked systems. Unfortunately, these conveniences come at the cost of the introduction of multiple vulnerabilities in the network. Particularly concerning is that the operation of critical infrastructure services is becoming increasingly reliant upon (potentially insecure) networked devices, generating significant vulnerabilities in many areas of society. As reported by the Department of Homeland Security's Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), there were 245 reported attacks on critical infrastructure sectors (such as manufacturing, energy, communication, water, transportation, to name a few) in 2014 alone [6]. Also, a recent cyber-attack on the US government resulted in the theft of the personal information of over four million (existing and current) federal employees, raising serious concerns regarding the safety of our networks. These incidents demonstrate that attacks on networked systems have not only remained consistent in frequency over the past few years, but have also evolved in sophistication resulting in more severe consequences on networked systems. This highlights the importance of developing security schemes in order to ensure our nation's systems are protected.

An added complication is that intruders (hereafter referred to as attackers) typically exploit combinations of multiple vulnerabilities to progress through a network. As a result, analyzing and defending against individual vulnerabilities is not sufficient for ensuring the security of the system. The concept of *attack graphs*, first introduced by Schneier [24], offer a useful way to model the interactions and dependencies between vulnerabilities and demonstrates how an attacker can exploit combinations of vulnerabilities to penetrate a network. This allows the network operator (defender) to construct an image of specific paths that attackers could take through the network to reach their goal, allowing for more effective defense decisions.

One issue is that attack graphs can be extremely large even for modestly sized systems. Scaling to realistically sized networks results in attack graphs that contain vast amounts of information, precluding efficient interpretation by a human operator and making it difficult for an appropriate defense action to be taken. This necessitates the development of automated methods that can take full advantage of the information provided by the attack graph, allowing one to define and calculate optimal defense policies.

Since information is continually becoming available as new attacks unfold, we are interested in defining *dynamic* defense policies, that is, policies that prescribe optimal actions given the most recent information available. *Moving target defense* (MTD) schemes constitute a powerful class of dynamic defense policies in which the network operator strives to increase the security of its network by making the system dynamically adaptive to an attacker's behavior and

MTD'15, October 12, 2015, Denver, Colorado, USA.
© 2015 ACM. ISBN 978-1-4503-3823-3/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/280475.28084828.

thus reducing the attacker's information and consequently decreasing the attacker's ability to compromise the system. In essence, given observations of the attacker's behavior, the defender is able to modify characteristics of the network in real-time in order to protect its network.

A consideration that must be taken into account when determining a defense policy is that defense actions have consequences that may interfere with the objective or purpose of the network. To quantify this trade-off (of security and efficient operation) we use three factors, referred to as the *CIA triad*, consisting of *confidentiality*, *integrity*, and *availability*. The first two factors, confidentiality (ensuring data does not get into the wrong hands) and integrity (maintaining accuracy and trustworthiness of the data) are both directly related to maintaining security of a resource, that is, security policies which block the most vulnerabilities from being exploited are the most desirable (under these two factors). However, blocking vulnerabilities (for example, disabling a network service) can negatively impact the connectivity of the network, causing some resources to become *unavailable* for some trusted users. As a result, it is necessary to come up with a trade-off between the three factors which ensures security but also allows for the network to still be usable (or fulfill its objective). This aligns with the philosophy of MTD that one should aim to design networks that are defensible, rather than completely secure [5].

In this paper, we develop an automated approach to defending a network against a continuous stream of attacks. We assume that the defender is primarily concerned with the security of a select set of resources within the network. These resources could correspond to root access on an important machine or gaining access to sensitive data, for example. Dependencies among vulnerabilities are modeled using *Bayesian attack graphs* (first introduced in [17]). Like attack graphs, Bayesian attack graphs model paths that an attacker can take through a network, but additionally allow for exploits to occur with a certain probability. This probability captures the likelihood that the attacker will recognize and carry out a specific exploit. A probabilistic spreading process is thus used to model the attacker's progression through the attack graph. Given a set of capabilities, the attacker performs exploits in order to gain additional capabilities (with a certain probability), allowing it to penetrate further into the network. The network operator (defender) is assumed to only partially observe the attacker's capabilities at any given time and is forced to take a defense action (termed a *countermeasure*) under imperfect information. Countermeasures are assumed to correspond to disabling services that vulnerabilities and exploits depend upon (for example, disabling the port scan service). We capture the trade-offs described by the CIA triad by defining appropriate costs for both the status of the network and the countermeasure actions. The resulting defense problem is formulated as a *partially observable Markov decision process* (POMDP). The computation of defense policies under this approach takes into account not only the previous states of the system, but also the future evolution. In this sense the defense policy is both reactive and anticipatory, forming a belief about the current state of the system based on previous states, actions, and observations as well as taking into account the effect of future possible attacks (and the consequence of defense policies) on the specification of the current defense decision.

1.1 Literature Review

Attack and defense modeling in networks has an extensive history and the resulting body of literature is large. For a good literature survey, the reader is directed to [13].

Attack graphs, which model dependencies between vulnerabilities and exploits, are a popular method for modeling attacks, allowing for specific attack paths to be constructed. A large body of the literature on attack graphs focuses on their construction. In 2005, Lippmann and Ingols [15] published a survey in which they outlined the construction and use of attack graphs in security settings, highlighting the fact that, at the time, the current methods were not scalable to large networks. Since then, the survey by Shandilya et al. [26] has illustrated that progress had been made in the field, making it evident that algorithms now exist that are capable of generating attack graphs for realistically sized networks. Imposing assumptions on the behavior of the attacker has had a large positive influence on the scalability of attack graph generation. In particular, the *monotonicity* assumption (developed by Ammann et al. [1]), which essentially states that the attacker will not give up previously attained capabilities, has greatly reduced the complexity of attack graph generation. Tools such as CAULDRON [11], [12], MulVAL [21], and NetSPA [10] have demonstrated that attack graphs for realistically sized systems can be generated quickly.

The process of augmenting an attack graph with exploit probabilities, in order to define a Bayesian attack graph, is a non-trivial process and an active area of research. Multiple papers discuss the factors that must be considered, as well as offer procedures for computing the probabilities [7], [8], [28]. Our approach assumes that a Bayesian attack graph has already been constructed for the network (via methods described in [7], [8], [28]) and instead focuses on the question of defining and computing an optimal dynamic defense policy.

We compare and contrast our paper with works in the literature that are concerned with determining dynamic defense policies. The first main area of research in this area is classified as *dynamic risk assessment and management* (see [18] for a short review of the field). The work of Poolsappasit et al. [22], and perhaps the closest paper to our work, formulate a dynamic defense problem as a multi-objective optimization problem and make use of a genetic algorithm to obtain a solution (no comments on the optimality of the obtained solution are provided). In contrast with our work, the theory of stochastic control allows us to formulate the defender's problem and obtain an optimal policy given all of the available information. The second category of dynamic defense approaches fall under the classification of *intrusion response systems* (IRSs); Stakhanova et al. [27] and Shameli-Sendi et al. [25] offer surveys of the field. Within IRS methods, we focus on papers concerned with determining an automated response to threats generate by an *intrusion detection system* (IDS) system, termed *automated response systems*. The work by Gehani and Kedem [9] discusses a real-time risk management system termed Rheostat; however, the approach is limited to risk analysis on a single host. Mu et al. [19] introduced an online risk-assessment model, forming the basis for their work on an intrusion-response system [20]. Liu et al. [16] study a security problem in mobile ad-hoc networks; they use a POMDP to determine the optimal sensor selection in an intrusion detection system. In comparison with the aforementioned literature on IRSs, a distinguishing aspect of our paper is that it makes use of attack graphs to accurately model the vulnerabilities in a system, allowing for a more appropriate defense action to be taken.

POMDPs have been employed in the security setting in a variety of papers [3], [30], [23], [29]; however, the focus in most of these papers has been on modeling the attacker's behavior [3], [30], [23], not determining the optimal defense action. The work of Lu & Brooks [29] formulates the problem of determining the optimal trade-off between the security effectiveness and diversity implementation costs as a POMDP; however, they do not consider an attack graph setting.

Our paper offers one main contribution in relation to the existing literature:

Main contribution: *The formulation (and solution for a small example) of a dynamic defense problem, on a Bayesian attack graph, as a POMDP:* We formulate the problem of determining the optimal defense scheme as a POMDP where intrusions are modeled using a Bayesian attack graph. To the best of the authors' knowledge this is the first time that the defender's problem of determining countermeasures in an attack graph setting has been formulated as a POMDP. In addition, we compute an optimal defense policy for a small attack graph example.

1.2 Outline of Paper

The paper is organized as follows. In Section 3, some motivation and context for our approach is presented. In Section 4, a formal graph-theoretic description of Bayesian attack graphs is described, followed by the probabilistic dynamical model of the attacker in Section 5. The defender's observation space and countermeasures are discussed in Sections 6 and 7, respectively. Section 8 describes the notion of an *information state* for the defender. The defender's problem (formulated as a POMDP) is formulated in Section 9. Finally, a toy numerical example is presented in Section 10 followed by some concluding remarks and directions for future research in Section 11.

2. NOTATION

To distinguish the difference between random variables and their realizations, we use uppercase letters to denote a random variable (i.e. X) and the corresponding lowercase letter to denote its realization (i.e. x). All remaining notation will be made clear as it is needed.

3. THE CONFLICT ENVIRONMENT

We consider an environment in which the network operator is concerned with defending its network against external intrusions in real-time. The defender uses its knowledge of the vulnerabilities and exploits in its system to construct a Bayesian attack graph. This paper analyzes how, given a Bayesian attack graph, an appropriate countermeasure action can be chosen in real-time, under imperfect knowledge of the attacker's capabilities at any given time.

An important consideration in the design of a defense scheme is that attacks may occur on a very fast time-scale; one that may preclude a human operator from being able to efficiently interpret the available information and make appropriate defense decisions. This necessitates the development of a defense system that can be operated independently of a human operator. Consequently, our approach offers a solution in which the monitoring and control of the security status of the network is completely automated. The application domain of the proposed defense system should thus be in networks where the possibility of a (partial) shutdown is preferred to a scenario in which the attacker has (partial) control of the system.

4. BAYESIAN ATTACK GRAPHS

We now describe a graph-theoretic representation of Bayesian attack graphs, restricting attention to *directed acyclic graphs*. The justification for this restriction follows from the same argument as in [22]; the authors disregards cycles by employing a *monotonicity* assumption on the attacker's behavior, developed in [1]. Informally speaking, the monotonicity assumption states that the attacker gains increasing control of the network as time progresses, never giving up previous capabilities (if left to act uninterrupted). It is important to note that when a defender is present (as is the case in our problem), the defender can remove attacker capabilities; the monotonicity assumption simply states that the attacker will not *willingly* give up capabilities. The monotonicity assumption is not very restrictive; in fact, Ou et al. [21] describe an idea behind converting non-monotonic attacks into monotonic attacks by ignoring some of the low-level details of how the attack actually occurs.

The nodes in a (Bayesian) attack graph represent *attributes* whereas edges represent *exploits*. Attributes are defined as *attacker capabilities* and could be interpreted as any of the following system conditions, to name a few: attacker permission levels on a given machine, vulnerabilities of a service or system, or information leakage. Exploits are events that allow the attacker to use their current set of capabilities (attributes) to obtain further capabilities. Formally speaking, a Bayesian attack graph is defined as follows.

Definition 4.1. *[22] A Bayesian attack graph, $\mathcal{G}$, is defined as the (fixed) tuple $\mathcal{G} = (\mathcal{N}, \theta, \mathcal{E}, \mathcal{P})$ where*

- *$\mathcal{N} = \{1, \ldots, N\}$ is the set of nodes, termed* attributes.
- *θ is the set of the node* types. *We assume that each non-leaf*[1] *attribute (node), can be one of two types, $\theta_i \in \{\wedge \text{ (AND)}, \vee \text{ (OR)}\}$. We denote the respective sets of nodes by $\mathcal{N}_\wedge$ and $\mathcal{N}_\vee$.*
- *$\mathcal{E}$ is the set of directed edges, termed* exploits.
- *$\mathcal{P}$ is the set of* exploit probabilities *associated with edges. Each exploit (directed edge), $e = (i, j) \in \mathcal{E}$, has an associated probability $\mathcal{P}(e) = \alpha_{ij} \in [0, 1]$.*

We assume that each attribute $i \in \mathcal{N}$ can either be enabled, meaning the attacker possesses attribute (capability) i, or disabled, meaning that the attacker does not possess attribute i. This gives rise to a definition of a network state, at time t, denoted by $X_t = (X_t^1, \ldots, X_t^N) \in \mathbf{X} := \{0, 1\}^N$, where X_t^i is the state of attribute i at time t,

$$X_t^i \in \{0 \text{ (disabled)}, 1 \text{ (enabled)}\}.$$

The Bayesian attack graph, $\mathcal{G}$, and directed acyclic graphs in general, contain both leaf nodes $\mathcal{N}_L \subseteq \mathcal{N}$, which are nodes

[1]Non-leaf nodes are defined as nodes that have at least one predecessor.

with no predecessors, and root nodes $\mathcal{N}_R \subseteq \mathcal{N}$, defined as nodes with no successors.[2] Leaf nodes are attributes that have not experienced any prior exploit. We interpret leaf nodes as being the connection points to the external world and thus assume that the attacker enters the attack graph at these nodes. On the other hand, root nodes correspond to attributes at the deepest exploit level. Some subset of the root nodes are designated by the defender as being critical attributes, $\mathcal{N}_C \subseteq \mathcal{N}_R$. The defender is primarily concerned with protecting the critical attributes.

Each attribute $i \in \mathcal{N} \backslash \mathcal{N}_L$ is also assigned a *type*, assumed to be either type **AND**, denoted by $i \in \mathcal{N}_\wedge$, or type **OR**, denoted by $i \in \mathcal{N}_\vee$. The type of the attribute dictates the conditions that the attribute's direct predecessors need to satisfy in order to for the attribute to become enabled. For $i \in \mathcal{N}_\wedge$, there is a non-zero probability of attribute i being enabled at $t+1$ if and only if *all* of attribute i's direct predecessors, denoted by $\bar{\mathcal{D}}_i = \{j \in \mathcal{N} | (j,i) \in \mathcal{E}\}$ (with $\bar{\mathcal{D}}_i = \varnothing$ for $i \in \mathcal{N}_L$), are enabled at time t. For $i \in \mathcal{N}_\vee$, if *any* of the attributes in $\bar{\mathcal{D}}_i$ are enabled at time t, then there is a non-zero probability that attribute i will be enabled at time $t+1$. Attributes with only one direct predecessor are classified as **AND** attributes.

The probabilistic component of a Bayesian attack graph arises due to the probability $\mathcal{P}(e)$ assigned to each exploit e. This probability captures the fact that the attacker may not be fully aware that its current capabilities can be used for an exploit or that the exploit itself may be difficult to carry out and will only succeed with a certain probability. As discussed in [17], domain expert knowledge is required in order to define these probabilities. Fig. 1 represents an instance of a Bayesian attack graph.

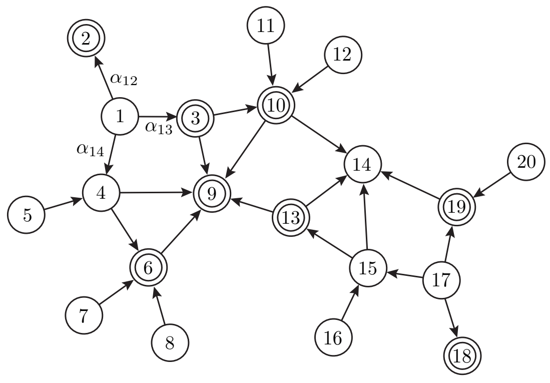

Figure 1: An example of the directed acyclic Bayesian attack graph topology considered in our model. For the above graph, leaf attributes are $\mathcal{N}_L = \{1,5,7,8,11,12,16,17,20\}$, root attributes $\mathcal{N}_R = \{2,9,14,18\}$, critical attributes $\mathcal{N}_C = \{9,14\} \subset \mathcal{N}_R$, $\wedge$ (AND) attributes (illustrated as double encircled nodes), $\mathcal{N}_\wedge = \{2,3,6,9,10,13,18,19\}$, and $\vee$ (OR) attributes, $\mathcal{N}_\vee = \{4,14,15\}$. Notice that leaf attributes are not assigned a type. Probabilities, α_{ij}, are labeled only for attribute 1 for simplicity.

[2]We obey the leaf and root naming convention presented by Schneier [24].

5. ATTACK BEHAVIOR: PROBABILISTIC SPREADING PROCESS

We assume that the attacker's behavior can be modeled by a *probabilistic spreading process*. In this sense, the attacker can be thought of as *nature*, or multiple, coordinating attackers, attempting to reach the critical attribute(s). The attacker is not assumed to follow any specific, intelligent behavior; our model instead assumes that the attacker moves randomly throughout the network.

The contagion[3] spreads in two steps, according to probabilistic dynamics. First, the process is assumed to start at the leaf nodes, $\mathcal{N}_L$, of the attack graph. As mentioned earlier, these leaf nodes represent attributes where the attacker has no capabilities and has performed no prior exploits and thus can be thought of as entry points into the network. At each time-step t, each attribute $i \in \mathcal{N}_L$ becomes enabled with probability $\alpha_i \in [0,1]$, defined as

$$\alpha_i := P(X^i_{t+1} = 1 | X^i_t = 0),\ i \in \mathcal{N}_L.$$

Next, the contagion spreads according to what we term *predecessor rules*. Predecessor rules describe how the process spreads to an attribute as a function of three factors: i) the attribute's type, ii) the states of the attribute's direct predecessors, and iii) the exploit probabilities. Mathematically, for **AND** attributes, $i \in \mathcal{N}_\wedge$, the probability of i transitioning from the disabled state to the enabled state is

$$P(X^i_{t+1} = 1 | X^i_t = 0, X_t) = \begin{cases} \prod\limits_{j \in \bar{\mathcal{D}}_i} \alpha_{ji} & \text{if } X^j_t = 1\, \forall\, j \in \bar{\mathcal{D}}_i \\ 0 & \text{otherwise} \end{cases}.$$

For **OR** attributes, $i \in \mathcal{N}_\vee$, the probability of becoming enabled is

$$\begin{aligned} &P(X^i_{t+1} = 1 | X^i_t = 0, X_t) \\ &\quad = \begin{cases} 1 - \prod\limits_{j \in \bar{\mathcal{D}}_i} (1 - \alpha_{ji}) & \text{if } \exists\, j \in \bar{\mathcal{D}}_i \text{ s.t. } X^j_t = 1 \\ 0 & \text{otherwise} \end{cases}. \end{aligned}$$

These probabilities remain fixed throughout the horizon of the decision problem. An illustration of the spreading process for both attribute types is provided in Fig. 2.

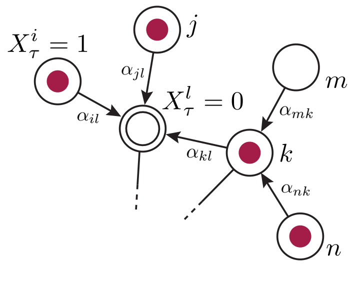

Figure 2: Example network demonstrating the capabilities of an attacker at time $t = \tau$. Node l is an AND attribute whereas node k is an OR attribute. Notice that since node k is an OR attribute, it can be enabled without requiring that node m be enabled.

[3]The term *contagion* is used to describe the spread of attacker's capabilities, that is, attributes that are enabled.

6. DEFENDER'S OBSERVATIONS

The defender receives an observation vector at time t, denoted $Y_t \in \mathcal{Y} = \mathbf{X}$. The defender is assumed to have limited surveillance of the network, that is, it is not aware of the full attacker's capabilities at any given time. We represent this uncertainty through a *probability of detection.* At each time-step, the defender observes that attribute i is enabled with probability β_i, defined as

$$\beta_i := P(Y_t^i = 1 | X_t^i = 1).$$

The assumption of limited surveillance is reasonable – the defender will not know, in general, the full capabilities of the attacker, that is, the attributes in X_t that are enabled, at any given time t. As discussed in the following assumption, we do not allow for false positives to occur.

Assumption 1: We assume that no false positive observations can occur, that is, $P(Y_t^i = 1 | X_t^i = 0) = 0$. This assumption prevents the possibility of the defender observing a positive observation ($Y_t^i = 1$) for an attribute that is actually disabled ($X_t^i = 0$).

As a result of the defender's incomplete observations, the defender is not fully certain of the status of the network at any given time and must form a belief about the network's current condition. The formation of this belief, termed an information state, is discussed in Section 8.

7. DEFENDER'S COUNTERMEASURES

The defender is able to deploy countermeasures in order to remove attacker capabilities and keep its network secure. These countermeasures could take the form of blocking a service (such as port scanning) or shutting down a machine or server, for example. We assume that the defender has access to M binary actions, denoted by $\{u^1, \ldots, u^M\}$. The space of countermeasure actions for the defender at any time is all possible subsets (the power set) of binary actions, denoted by $\mathcal{U} := \wp(\{u^1, \ldots, u^M\})$. Notice that this allows for the possibility of the defender choosing the *empty set* countermeasure action, meaning it allows the system to operate uninterrupted.

Countermeasures directly influence the state of attributes. Consider a countermeasure $u \in \mathcal{U}$. Each binary action u^m in u disables a set of attributes, denoted by $\mathcal{W}_{u^m} \subseteq \mathcal{N}$. For example, action u^1 could correspond to the set of nodes $\mathcal{W}_{u^1} = \{1, 19\}$ in Fig. 1. A countermeasure $u \in \mathcal{U}$, which is a combination of the actions in $\{u^1, \ldots, u^M\}$, thus disables the attributes in the set $\mathcal{W}_u = \{i \in \mathcal{N} | i \in \mathcal{W}_{u^m}, u^m \in u\}$.[4]

Assumption 2: We assume that all leaf attributes, $\mathcal{N}_L$, are covered by at least one binary action. That is, for each $i \in \mathcal{N}_L$ there exists at least one binary action u^m such that $i \in \mathcal{W}_{u^m}$.

7.1 Cost Function

The cost of taking countermeasure $u \in \mathcal{U}$ while in state $x \in \mathbf{X}$ is denoted by $C(x, u)$. This cost takes into account the three factors that make up the CIA triad. The first property of the cost function we impose captures the defender's desire to keep the critical attributes out of the attacker's control (disabled). Consider two states, $x, \hat{x} \in \mathbf{X}$. State x is one where all critical attributes are disabled, that is $x^i = 0$ for all $i \in \mathcal{N}_C$, whereas state $\hat{x}$ has at least one critical attribute enabled, that is $\hat{x}^i = 1$ for some $i \in \mathcal{N}_C$. The cost function should reflect that $\hat{x}$ is more costly from a confidentiality and integrity perspective, that is, for any given countermeasure $u \in \mathcal{U}$, $C(\hat{x}, u) > C(x, u)$. To take into account the availability factor we impose a second property of the cost function. Countermeasures have negative impacts on availability; for example, shutting down a server or blocking a specific protocol/service has the adverse effect of the loss of some network connectivity for trusted users. If one countermeasure $\hat{u}$ has a more significant negative impact on availability than another countermeasure u, the cost function should obey $C(x, \hat{u}) > C(x, u)$, for any state x.

[4]Depending on the graph topology, each countermeasure action also disables additional nodes (we term this the *reach* of the countermeasure action). We discuss this point further in Section 8 and Appendix A.

8. DEFENDER'S INFORMATION STATE

For simplicity of analysis, we now make a restriction regarding the attribute types present in the attack graph.

Assumption 3: We assume, for the remainder of the paper, that the attack graph only contains **AND** attributes, that is, $\mathcal{N}_\wedge = \mathcal{N} \setminus \mathcal{N}_L$ and $\mathcal{N}_\vee = \varnothing$.

We define the *history* at time t, H_t, as all of the defense countermeasure actions up to and including time $t-1$ and the observations up to and including time t, that is

$$H_t = (\pi_0, U_0, Y_0, U_1, Y_1, \ldots, U_{t-1}, Y_t).$$

An *information state*, I_t, is a summary of the history, H_t, that satisfies two conditions (p. 81, [14]): 1) I_t can be evaluated from H_t, and 2) There must exist a function, $\mathcal{F}_t$, such that $I_{t+1} = \mathcal{F}_t(I_t, Y_{t+1}, U_t)$, where Y_{t+1} and U_t comprise the new information received between time t and $t+1$. An information state based on the above two criteria need not be sufficient for making an optimal decision.[5] In search of a sufficient information state, one could use the history itself, H_t; however, the history is unbounded in time. In the setting of POMDPs, an alternative sufficient information state is the random vector $\Pi_t = (\Pi_t^1, \ldots, \Pi_t^K)$ (as discussed in [2], [14]), with

$$\Pi_t^i = P(X_t = x_i | H_t)$$

where $x_i \in \mathcal{X}$. The space $\mathcal{X}$ represents all *feasible* states (graphs) and includes only the $|\mathcal{X}| = K \leq 2^N$ graphs that satisfy the monotonicity assumption discussed at the beginning of Section 4. Under the restriction of the attack graph only containing **AND** attributes (assumption 3), the monotonicity assumption can be more formally stated as follows.

Assumption 4: We assume that the only feasible states are *monotone*, denoted by the set $\mathcal{X}$. A state (graph) $x = (x^1, \ldots, x^N) \in \mathbf{X}$ in an attack graph with only **AND** attributes, that is $\mathcal{N}_\wedge = \mathcal{N} \setminus \mathcal{N}_L$ and $\mathcal{N}_\vee = \varnothing$, is termed *monotone* if for every i with $x^i = 1$, all of i's predecessors are enabled, that is, $x^j = 1$ for all $j \in \mathcal{D}_i$.

The above assumption allows us to exploit the connectivity of the attack graph and restrict attention to only monotone states instead of all 2^N states in the space $\mathbf{X}$. An example of some monotone (feasible) states can be seen in Fig. 3. Monotonicity greatly reduces the dimensionality of

[5]For a trivial example of this consider the information state $I_t = 0$ for all t.

the state space and allows for tractable numerical analyses on moderately sized networks.

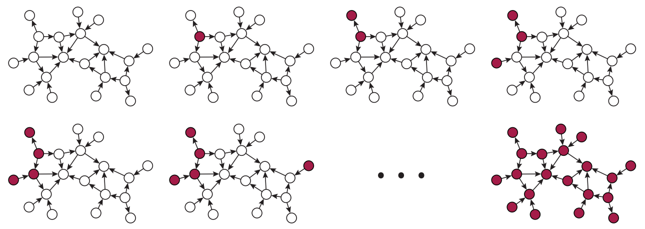

Figure 3: Illustration of some monotone states, $x_i \in \mathcal{X}$. Notice that every enabled attribute has all predecessors enabled as well. The defender's information state at any given time is defined as a PMF over the monotone states (graphs) in the attack graph.

The (realized) information state π_t represents a probability mass function over states (i.e. graphs) for a given realized history h_t and thus lives in the $(N-1)$-simplex defined over the space of feasible states $\mathcal{X}$, that is $\pi_t \in \Delta(\mathcal{X})$. There exists a function $\mathcal{T} : \Delta(\mathcal{X}) \times \mathcal{Y} \times \mathcal{U} \to \Delta(\mathcal{X})$ such that

$$\pi_{t+1} = \mathcal{T}(\pi_t, y_{t+1}, u_t).$$

This function takes the defender's current belief, π_t, and the new (realized) information obtained between time-steps t and $t+1$, $\{y_{t+1}, u_t\}$, and forms updated belief, π_{t+1}. The precise expression for the function $\mathcal{T}$ can be found in Appendix A.

9. DEFENDER'S PROBLEM

We are now ready to define the defender's optimization problem. The defender wishes to apply a countermeasure action $u_t \in \mathcal{U}$ at each time-step t in order to keep the critical attributes disabled (i.e. out of control of the attacker) while maintaining availability. This is achieved by determining an appropriate defense policy, g, defined below.

The defender's defense policy should not only take into account the current belief of the network, but also the future evolution of the system (due to both the possible attacker actions and the defense policy). Formally, the defender is attempting to determine a defense policy g which maps its current belief of the network, $\pi_t \in \Delta(\mathcal{X})$, to a countermeasure action, $u \in \mathcal{U}$, that is, $g : \Delta(\mathcal{X}) \to \mathcal{U}$, such that g minimizes the infinite-horizon discounted expected cost, as shown by Problem (P_D).

$$\min_{g \in \mathcal{G}} \mathbb{E}^g \left\{ \sum_{t=0}^{\infty} \rho^t C(\mathit{\Pi}_t, U_t) \big| \mathit{\Pi}_0 = \pi_0 \right\} \qquad (P_D)$$

$$\text{subject to } U_t = g(\mathit{\Pi}_t)$$
$$\mathit{\Pi}_{t+1} = \mathcal{T}(\mathit{\Pi}_t, Y_{t+1}, U_t)$$

where $\mathbb{E}^g\{\cdot\}$ denotes the expectation with respect to the probability measure induced by the defense policy $g \in \mathcal{G}$, $\mathcal{G}$ is the space of admissible control (defense) policies, π_0 is the initial information state, and $C(\pi_t, u_t)$, for any $\pi_t \in \Delta(\mathcal{X})$, $u_t \in \mathcal{U}$, is the expected instantaneous cost, written as

$$C(\pi_t, u_t) = \sum_{x_i \in \mathcal{X}} \pi_t^i C(x_i, u_t).$$

The quantity $\rho \in (0, 1)$ is termed the *discount factor* and represents how much of the future to take into account for the current decision (with a higher ρ corresponding to a longer view).

9.1 Dynamic Programming Equations

Dynamic programming is used in order to obtain a solution to the infinite-horizon discounted expected cost problem, (P_D). We first define the cost, $V^g(\pi)$, associated with policy g and initial information state π.

$$V^g(\pi) := \mathbb{E}^g \left\{ \sum_{t=0}^{\infty} \rho^t C(\mathit{\Pi}_t, g(\mathit{\Pi}_t)) \big| \mathit{\Pi}_0 = \pi \right\} \qquad (1)$$

The optimal defense policy g^* minimizes Eq. (1) over $g \in \mathcal{G}$. Let $V^*(\pi) = V^{g^*}(\pi)$; the value function $V^*(\pi)$ satisfies the dynamic programming (Bellman) optimality equation [14],

$$V^*(\pi) = \min_{u \in \mathcal{U}} Q^*(\pi, u)$$

for all $\pi \in \Delta(\mathcal{X})$, where $Q^*(\pi, u)$ is defined as

$$Q^*(\pi, u) := C(\pi, u) + \rho \sum_{y \in \mathcal{Y}} P_y^{\pi, u} V^*(\mathcal{T}(\pi, y, u))$$

with $P_y^{\pi,u} = P(Y = y|\pi, u) = \sum_{x_i \in \mathcal{X}} \pi^i P(Y = y|X = x_i, u)$ (where $P(Y = y|X = x_i, u)$ is the observation probability). The optimal defense policy is defined by

$$g^*(\pi) = \operatorname*{argmin}_{u \in \mathcal{U}} Q^*(\pi, u)$$

for all $\pi \in \Delta(\mathcal{X})$. The solution method used in the following section to obtain an optimal defense policy makes use of the above dynamic programming equations.

10. A SAMPLE NETWORK

We demonstrate our problem on a small Bayesian attack graph, seen in Fig. 4.

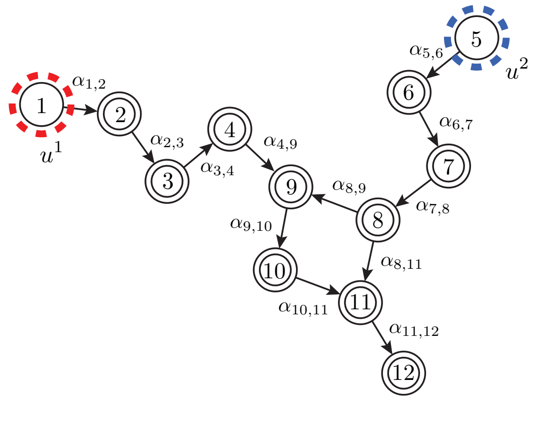

Figure 4: The sample Bayesian attack graph used for our numerical example. All nodes are considered to be AND attributes. We have one critical attribute, $\mathcal{N}_C = \{12\}$, and two countermeasures. Countermeasure actions u^1 and u^2 correspond to sets $\mathcal{W}_{u^1} = \{1\}$ and $\mathcal{W}_{u^2} = \{5\}$, respectively.

The toy example consists of 12 attributes containing two leaf attributes (nodes 1 and 5) and one critical attribute (node 12). The attributes in this test network are interpreted as follows.

Attributes

1. Vulnerability in WebDAV on machine 1
2. User access on machine 1
3. Heap corruption via SSH on machine 1
4. Root access on machine 1
5. Buffer overflow on machine 2
6. Root access on machine 2
7. Squid portscan on machine 2
8. Network topology leakage from machine 2
9. Buffer overflow on machine 3
10. Root access on machine 3
11. Buffer overflow on machine 4
12. Root access on machine 4

The defender is assumed to have access to two binary actions, defined as follows.

Actions

u^1: Block WebDAV service
u^2: Disconnect machine 2

With two binary actions, we have $2^2 = 4$ countermeasure actions, that is $\mathcal{U} = \{\varnothing, \{u^1\}, \{u^2\}, \{u^1, u^2\}\}$.

The cost function in the example is assumed to be additively separable, that is $C(x, u) = C(x) + D(u)$ for all $x \in \mathcal{X}$, $u \in \mathcal{U}$. State costs, $C(x)$, are defined as

$$C(x) = \begin{cases} 1 & \text{if } x^i = 1 \text{ for } i \in \mathcal{N}_C \\ 0 & \text{otherwise} \end{cases}.$$

The availability cost of a countermeasure $u \in \mathcal{U}$, denoted $D(u)$, for the purposes of this example, is defined to be proportional to the number of attributes that the countermeasure action disables (that is, the reach of the countermeasure, defined in Eq. A.1). The rationale for this is that if a countermeasure disables more attributes then it likely has a larger impact on availability and should be more expensive to implement. Since $|\mathcal{R}_{u^1}| = |\mathcal{R}_{u^2}|$, we set $D(u^1)$ and $D(u^2)$ to be equal, specifically $D(u^1) = D(u^2) = 1$. We assign a high cost for the countermeasure action that resets all attributes in the graph, $D(\{u^1, u^2\}) = 5$, since this reduces availability to zero.

The assumptions on attribute types and attacker behavior allow us to greatly reduce the dimensionality of the sample problem. Without any assumptions, the number of states in the sample network, illustrated in Fig. 4, is $2^{12} = 4096$. However, under the restriction of **AND** attributes (assumption 3) and monotone states (assumption 4) the number of feasible states, $\mathcal{X}$, is reduced to $|\mathcal{X}| = 29$, a state-space size reduction of over 140 times.

The dimensionality of the observation space can also be significantly reduced. Even though we cannot change what we observe (any of the possible 2^{12} combinations of enabled attributes in the observation set $\mathcal{Y}$) we can change how we interpret the observations. Since we have assumed no false positives (no possibility of seeing an enabled attribute when it is, in fact, disabled) a given observation is always a subset of the true underlying state $x \in \mathcal{X}$. Therefore, we can map a given observation $y \in \mathcal{Y}$ to an observation $\hat{y} \in \hat{\mathcal{Y}} = \mathcal{X}$, where $\hat{\mathcal{Y}}$ is a reduced space consisting only of informationally useful observations.

The parameters for this problem are: probabilities of detection: $\boldsymbol{\beta} = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.7, 0.6, 0.7, 0.85, 0.95)$; attack probabilities: $\alpha_1 = 0.5$, $\alpha_5 = 0.5$; and spread probabilities: $\alpha_{1,2} = 0.8$, $\alpha_{2,3} = 0.8$, $\alpha_{3,4} = 0.9$, $\alpha_{4,9} = 0.8$, $\alpha_{5,6} = 0.8$, $\alpha_{6,7} = 0.9$, $\alpha_{7,8} = 0.8$, $\alpha_{8,9} = 0.8$, $\alpha_{8,11} = 0.8$, $\alpha_{9,10} = 0.9$, $\alpha_{10,11} = 0.8$, $\alpha_{11,12} = 0.9$. For simulation purposes, we assume a discount factor of $\rho = 0.85$ and an initial belief vector of $\pi_0 = (1, 0, \ldots, 0)$. The resulting defense problem for the sample network in Fig. 4 is a 29-state/observation, 4-action POMDP.

10.1 Results and Discussion

We make use Cassandra's C-software package *pomdp-solve* [4] in order to obtain the optimal defense policy. Provided that the solver converges, it generates an output specifying the optimal solution to the POMDP. This solution takes the form of a (high-dimensional) partition of the information state simplex $\Delta(\mathcal{X})$ (a 28-dimensional simplex, in our example) and is thus difficult to represent via text or a figure. We instead show the optimal countermeasure action for a selection of information states, seen in Fig. 5.

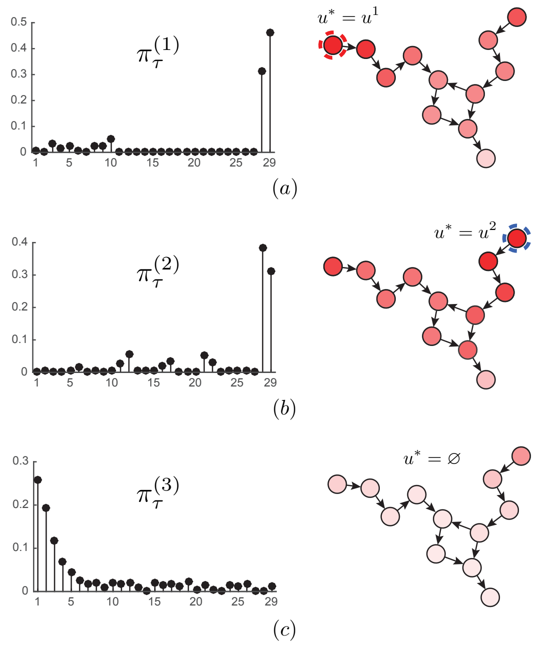

Figure 5: The left column represents a selection of three information states, at time τ, with different optimal countermeasure actions. The optimal policy prescribes: (a) countermeasure $u^* = u^1$ for information state $\pi_\tau^{(1)}$; (b) countermeasure $u^* = u^2$ for information state $\pi_\tau^{(2)}$; and (c) countermeasure $u^* = \varnothing$ for information state $\pi_\tau^{(3)}$. Graphs in the right column represent *heat maps* of the probability that each attribute is enabled under the current information state (a darker shade corresponds to a higher probability of being enabled).

For each information state a network *heat map* is computed (see the right-hand column in Fig. 5). The heat map, determined from the current information state, offers a graphical representation of the probability that each attribute is enabled.

The optimal policy is intuitive. It can be seen from the heat maps that optimal countermeasure is the one that disables the attributes that have a sufficiently high probability of being enabled (see Fig. 5(a) and (b)). When the probability of enabled attributes is low, no action is chosen, as can be seen in Fig. 5(c). The countermeasure $u = \{u^1, u^2\}$ was not prescribed by the optimal policy for any information state. We believe this is because the critical attribute, $i = 12$, is in the reach of both countermeasure actions $u = u^1$ and $u = u^2$, and thus there is no need to take the more expensive countermeasure $u = \{u^1, u^2\}$.

11. CONCLUSION & FUTURE WORK

As networks grow in size and attacks become more frequent, reliance on human operators to make defense decisions is becoming increasingly unrealistic. This paper introduces a formal model for protecting a network (specifically a critical subset of resources) against attacks in real-time. The approach uses Bayesian attack graphs to model dependencies between vulnerabilities and exploits. The attacker is assumed to move randomly throughout the graph and is thus modeled by a probabilistic spreading process. The defender is assumed to have imperfect information regarding the current status of the network at any given time and, consequently, is required to make a decision based on its *belief* of the network's status. This leads to the formulation of the defense problem as a POMDP. An optimal policy, which maps the current belief to the optimal countermeasure action, is obtained for a small sample network.

Future work will consist of, i) enriching the set of attribute types in the Bayesian attack graph (to include **OR** attributes), ii) including the possibility of false positive observations, and iii) developing scalable solution methods in order to obtain policies in very large-scale attack graphs.

12. ACKNOWLEDGMENTS

This work was supported in part by NSF grant CNS-1238962 and ARO MURI grant W911NF-13-1-0421.

13. REFERENCES

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.

[2] K. J. Astrom. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174, 1965.

[3] L. Carin, G. Cybenko, and J. Hughes. Cybersecurity strategies: The QuERIES methodology. *Computer*, 41(8):20–26, 2008.

[4] T. Cassandra. pomdp-solve: POMDP solver software, v5.4, 2003–2015. [Online; accessed 2-March-2015].

[5] Department of Homeland Security. Moving target defense. [Online; accessed 19-April-2015].

[6] Department of Homeland Security. Industrial control systems cyber emergency response team (ICS-CERT) year in review, 2014. [Online; accessed 10-April-2015].

[7] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graphs. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 698–703, July 2008.

[8] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, pages 23–30. ACM, 2008.

[9] A. Gehani and G. Kedem. Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection*, pages 296–314. Springer, 2004.

[10] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 117–126. IEEE, 2009.

[11] S. Jajodia and S. Noel. Topological vulnerability analysis. In *Cyber Situational Awareness*, pages 139–154. Springer, 2010.

[12] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. Cauldron mission-centric cyber situational awareness with defense in depth. In *Military Communications Conference, 2011-MILCOM 2011*, pages 1339–1344. IEEE, 2011.

[13] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13:1–38, 2014.

[14] P. R. Kumar and P. Varaiya. *Stochastic systems: Estimation, identification, and adaptive control.* Prentice Hall Englewood Cliffs, NJ, 1986.

[15] R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[16] J. Liu, F. R. Yu, C. H. Lung, and H. Tang. Optimal combined intrusion detection and biometric-based continuous authentication in high security mobile ad hoc networks. *Wireless Communications, IEEE Transactions on*, 8(2):806–815, 2009.

[17] Y. Liu and H. Man. Network vulnerability assessment using bayesian networks. In *Defense and Security*, pages 61–71. International Society for Optics and Photonics, 2005.

[18] D. López, O. Pastor, and L. García Villalba. Dynamic risk assessment in information systems: state-of-the-art. In *Proceedings of the 6th International Conference on Information Technology, Amman*, pages 8–10, 2013.

[19] C. Mu, X. Li, H. Huang, and S. Tian. Online risk assessment of intrusion scenarios using DS evidence theory. In *Computer Security-ESORICS 2008*, pages 35–48. Springer, 2008.

[20] C. Mu and Y. Li. An intrusion response decision-making model based on hierarchical task network planning. *Expert systems with applications*, 37(3):2465–2472, 2010.

[21] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345. ACM, 2006.

[22] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.

[23] C. Sarraute, O. Buffet, and J. Hoffmann. POMDPs make better hackers: Accounting for uncertainty in penetration testing. *arXiv preprint arXiv:1307.8182*, 2013.

[24] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[25] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, and M. Dagenais. Intrusion response systems: survey and taxonomy. *Int. J. Comput. Sci. Netw. Secur*, 12(1):1–14, 2012.

[26] V. Shandilya, C. B. Simmons, and S. Shiva. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014.

[27] N. Stakhanova, S. Basu, and J. Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1-2):169–184, 2007.

[28] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy. Using bayesian networks for cyber security analysis. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 211–220. IEEE, 2010.

[29] L. Yu and R. R. Brooks. Applying pomdp to moving target optimization. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, page 49. ACM, 2013.

[30] Y. Zhang, X. Fan, Z. Xue, and H. Xu. Two stochastic models for security evaluation based on attack graph. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, pages 2198–2203. IEEE, 2008.

APPENDIX

A. INFORMATION STATE UPDATE

We now derive the information state update for Bayesian attack graphs with only **AND** attributes. Consider the following timing diagram, Fig. 6, illustrating the order of events in our model.

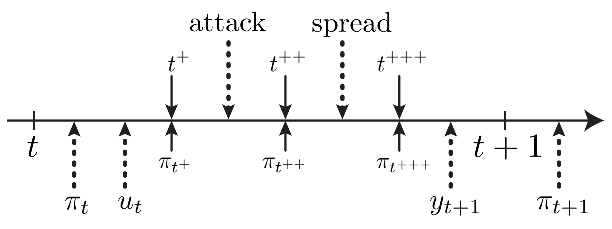

Figure 6: Event and update timing in our model.

We decompose the information state update into four steps, represented by the composition $\mathcal{T} = \mathcal{T}_4 \circ \mathcal{T}_3 \circ \mathcal{T}_2 \circ \mathcal{T}_1$. Then,

$$\pi_{t+1} = \mathcal{T}(\pi_t, y_{t+1}, u_t) = \mathcal{T}_4(\mathcal{T}_3(\mathcal{T}_2(\mathcal{T}_1(\pi_t, u_t))), y_{t+1}).$$

$\underline{\mathcal{T}_1 \textit{ update}}$:

First, we define the the *reach* of a countermeasure action $u \in \mathcal{U}$, denoted by $\mathcal{R}_u$, as the set of nodes that are disabled as a result of deploying u, that is,

$$\mathcal{R}_u := \{i \in \mathcal{N} | i \in \mathcal{S}_j, j \in \mathcal{W}_u\} \tag{A.1}$$

where S_j is the set of successors of j. The update to the intermediate information state, π_{t^+}, is written as

$$\begin{aligned}\hat{\pi}^i_{t^+} &= \mathcal{T}^i_1(\pi_t, u_t) \\ &= P(X_{t^+} = x_i | \pi_t, u_t) \\ &= \begin{cases} 0 & \text{if } \exists j \text{ s.t. } x^j_i = 1 \text{ and } j \in \mathcal{R}_{u_t} \\ \pi^i_t & \text{otherwise} \end{cases}\end{aligned}$$

followed by the normalization $\pi^i_{t^+} = \hat{\pi}^i_{t^+}/\mathbf{1}^\top \hat{\pi}_{t^+}$.

$\underline{\mathcal{T}_2 \textit{ update}}$:

Recall that it is assumed that the attacker can compromise each leaf node $i \in \mathcal{N}_L$ with probability α_i at each time step. This step, denoted by the label *attack* in Fig. 6, results in the following intermediate information state update $\pi_{t^{++}}$.

$$\begin{aligned}\pi^i_{t^{++}} &= \mathcal{T}^i_2(\pi_{t^+}) \\ &= P(X_{t^{++}} = x_i | \pi_{t^+}) \\ &= \sum_{x_k \in \boldsymbol{X}} P(X_{t^{++}} = x_i, X_{t^+} = x_k | \pi_{t^+}) \\ &= \sum_{x_k \in \boldsymbol{X}} P(X_{t^{++}} = x_i | X_{t^+} = x_k) P(X_{t^+} = x_k | \pi_{t^+})\end{aligned}$$

Some of the probability terms, $P(X_{t^{++}} = x_i | X_{t^+} = x_k)$, of the above summation are zero; these correspond to transitions from $X_{t^+} = x_k$ to $X_{t^{++}} = x_i$ that are not feasible. As a result, we can reduce the complexity of the above update by only including feasible states in the summation, that is, states $X_{t^+} = x_k$ that could possibly lead to state $X_{t^{++}} = x_i$. This feasible set of states that can lead to x_i, denoted by $\mathcal{X}^i_{t^+}$, is defined as

$$\mathcal{X}^i_{t^+} = \Big\{ x_j \in \mathcal{X} \;\Big|\; \mathcal{I}(x_j^{\mathcal{N}\setminus\mathcal{N}_L} = 1) = \mathcal{I}(x_i^{\mathcal{N}\setminus\mathcal{N}_L} = 1), \\ \mathcal{I}(x_j^{\mathcal{N}_L} = 1) \subseteq \mathcal{I}(x_i^{\mathcal{N}_L} = 1) \Big\}$$

where $\mathcal{I}(x_i^{\mathcal{N}\setminus\mathcal{N}_L} = 1)$ denotes the set of indices, $j \in \mathcal{N} \setminus \mathcal{N}_L$ such that $x^j_i = 1$. The conditional probability terms are defined, for $x_k \in \mathcal{X}^i_{t^+}$, as

$$\begin{aligned}&P(X_{t^{++}} = x_i | X_{t^+} = x_k) \\ &\qquad = \Bigg(\prod_{l \in \mathcal{I}(x_{i\setminus k}^{\mathcal{N}_L} = 1)} \alpha_l \Bigg) \Bigg(\prod_{l \in \mathcal{I}(x_i^{\mathcal{N}_L} = 0)} (1 - \alpha_l) \Bigg).\end{aligned}$$

The set $\mathcal{I}(x_{i\setminus k}^{\mathcal{N}_L} = 1)$ is shorthand for $\mathcal{I}(x_i^{\mathcal{N}_L} = 1) \setminus \mathcal{I}(x_k^{\mathcal{N}_L} = 1)$ and represents the indices of attributes in the set $\mathcal{N} \setminus \mathcal{N}_L$ that have become enabled at time t^+. The set $\mathcal{I}(x_i^{\mathcal{N}_L} = 0)$ represents the indices, $j \in \mathcal{N}_L$, where $x^j_i = 0$. Finally, the update from π_{t^+} to $\pi_{t^{++}}$ is

$$\pi^i_{t^{++}} = \mathcal{T}^i_2(\pi_{t^+})$$
$$= \sum_{x_k \in \mathcal{X}^i_{t^+}} \left[\left(\prod_{l \in \mathcal{I}(x^{\mathcal{N}_L}_{i \setminus k}=1)} \alpha_l \right) \left(\prod_{l \in \mathcal{I}(x^{\mathcal{N}_L}_i=0)} (1-\alpha_l) \right) \pi^k_{t^+} \right].$$

<u>*$\mathcal{T}_3$ update*</u>:

After time t^{++}, the attacker exploits further attributes using its current attributes via the probabilistic spreading process described in Section 5. This step is denoted by the label *spread* in Fig. 6 and results in the next update to intermediate information state $\pi_{t^{+++}}$.

$$\pi^i_{t^{+++}} = \mathcal{T}^i_3(\pi_{t^{++}})$$
$$= P(X_{t^{+++}} = x_i | \pi_{t^{++}})$$
$$= \sum_{x_k \in \mathcal{X}^i_{t^{++}}} P(X_{t^{+++}} = x_i, X_{t^{++}} = x_k | \pi_{t^{++}})$$
$$= \sum_{x_k \in \mathcal{X}^i_{t^{++}}} P(X_{t^{+++}} = x_i | X_{t^{++}} = x_k) P(X_{t^{++}} = x_k | \pi_{t^{++}}) \quad \text{(A.2)}$$

We have restricted the sum to feasible states, $\mathcal{X}^i_{t^{++}}$, that is, the set of states that can *spread* to $X_{t^{+++}} = x_i$. This set is defined as

$$\mathcal{X}^i_{t^{++}} = \Big\{ x_j \in \mathcal{X} \;\Big|\; \mathcal{I}(x^{\mathcal{N}_L}_j = 1) = \mathcal{I}(x^{\mathcal{N}_L}_i = 1),$$
$$\mathcal{I}(x^{\mathcal{N}\setminus\mathcal{N}_L}_j = 1) \subseteq \mathcal{I}(x^{\mathcal{N}\setminus\mathcal{N}_L}_i = 1),$$
$$\mathcal{I}(x^{\mathcal{N}\setminus\mathcal{N}_L}_i = 1) \setminus \mathcal{I}(x^{\mathcal{N}\setminus\mathcal{N}_L}_j = 1) \subseteq \bar{\mathcal{D}}(x_j) \Big\}$$

where $\bar{\mathcal{D}}(x_j)$ is defined as the set of attributes in x_j that have all direct predecessors enabled, that is, $\bar{\mathcal{D}}(x_j) := \{k \in \mathcal{N} \,|\, x^l_j = 1, l \in \bar{\mathcal{D}}_k\}$. Notice that $x_i \in \mathcal{X}^i_{t^{++}}$. The conditional probabilities in the summation in Eq. (A.2) are written, for $x_k \in \mathcal{X}^i_{t^{++}}$, as

$$P(X_{t^{+++}} = x_i | X_{t^{++}} = x_k)$$
$$= \left(\prod_{m \in E(x_k) \cap \mathcal{I}(x_i=1)} \prod_{n \in \bar{\mathcal{D}}_m} \alpha_{nm} \right)$$
$$\cdot \left(\prod_{m \in E(x_k) \cap \mathcal{I}(x_i=0)} (1 - \prod_{n \in \bar{\mathcal{D}}_m} \alpha_{nm}) \right) \quad \text{(A.3)}$$

where $E(x_k) := \mathcal{I}(x_k = 0) \cap \bar{\mathcal{D}}(x_k)$ denotes the set of attributes that are *eligible* to be enabled in $X_{t^{++}} = x_k \in \mathcal{X}^i_{t^{++}}$. The set $E(x_k) \cap \mathcal{I}(x_i = 1)$ denotes the set of eligible attributes that have become enabled at t^{+++} whereas the set $E(x_k) \cap \mathcal{I}(x_i = 0)$ denotes the set of eligible attributes that remained disabled. It is clear that for some x_k there may be no eligible attributes, that is $E(x_k) = \varnothing$. In this case, the product is empty and the conditional probability is set to $P(X_{t^{+++}} = x_i | X_{t^{++}} = x_k) = 1$. The final update becomes

$$\pi^i_{t^{+++}} = \mathcal{T}^i_3(\pi_{t^{++}})$$
$$= \sum_{x_k \in \mathcal{X}^i_{t^{++}}} \left[\left(\prod_{m \in E(x_k) \cap \mathcal{I}(x_i=1)} \prod_{n \in \bar{\mathcal{D}}_m} \alpha_{nm} \right) \cdot \left(\prod_{m \in E(x_k) \cap \mathcal{I}(x_i=0)} (1 - \prod_{n \in \bar{\mathcal{D}}_m} \alpha_{nm}) \right) \right] \pi^k_{t^{++}}.$$

<u>*$\mathcal{T}_4$ update*</u>:

Finally, the defender records an observation, $y_{t+1} \subseteq \mathcal{Y}$, and the information state is updated to π_{t+1}.

$$\pi^i_{t+1} = \mathcal{T}^i_4(\pi_{t^{+++}}, y_{t+1})$$
$$= P(X_{t+1} = x_i | \pi_{t^{+++}}, Y_{t+1} = y_{t+1})$$
$$= \frac{P(X_{t+1} = x_i, Y_{t+1} = y_{t+1} | \pi_{t^{+++}})}{P(Y_{t+1} = y_{t+1} | \pi_{t^{+++}})}$$
$$= \frac{P(Y_{t+1} = y_{t+1} | X_{t+1} = x_i) P(X_{t+1} = x_i | \pi_{t^{+++}})}{\sum_{x_j \in \mathcal{X}} P(Y_{t+1} = y_{t+1}, X_{t+1} = x_j | \pi_{t^{+++}})}$$
$$= \frac{P(Y_{t+1} = y_{t+1} | X_{t+1} = x_i) P(X_{t+1} = x_i | \pi_{t^{+++}})}{\sum_{x_j \in \mathcal{X}} P(Y_{t+1} = y_{t+1} | X_{t+1} = x_j) P(X_{t+1} = x_j | \pi_{t^{+++}})}$$

The conditional probability terms can be written as

$$P(Y_{t+1} = y_{t+1} | X_{t+1} = x_i) P(X_{t+1} = x_i | \pi_{t^{+++}}) \quad \text{(A.4)}$$
$$= P(Y_{t+1} = y_{t+1} | X_{t^{+++}} = x_i) P(X_{t^{+++}} = x_i | \pi_{t^{+++}}) \quad \text{(A.5)}$$
$$= P(Y_{t+1} = y_{t+1} | X_{t^{+++}} = x_i) \pi^i_{t^{+++}} \quad \text{(A.6)}$$

where we have used the fact that no state transition is possible between times t^{+++} and $t+1$ to obtain Eq. (A.5) from Eq. (A.4). Eq. (A.6)) is obtained by using the definition of the information state.

Taking into account the defender's probability of detection, the conditional probability term in Eq. (A.6) is

$$P(Y_{t+1} = y | X_{t^{+++}} = x)$$
$$= \begin{cases} \left(\prod_{j \in \mathcal{O}(x,y)} \beta_j \right) \left(\prod_{j \in \bar{\mathcal{O}}(x,y)} (1-\beta_j) \right) & \text{if } \mathcal{M}(x,y) = \varnothing \\ 0 & \text{otherwise} \end{cases} \quad \text{(A.7)}$$

The set $\mathcal{O}(x,y) := \mathcal{I}(x = 1) \cap \mathcal{I}(y = 1)$ denotes the set of attributes that were observed to be enabled. Similarly, the set $\bar{\mathcal{O}}(x,y) := \mathcal{I}(x = 1) \cap \mathcal{I}(y = 0)$ denotes the attributes that were enabled but were not observed. Finally, the set $\mathcal{M}(x,y) := \mathcal{I}(x = 0) \cap \mathcal{I}(y = 1)$ represents the incompatible combination of observing an enabled attribute when it is not in fact enabled (by assumption 1). Finally, the update is written using Eq. (A.7) as

$$\pi^i_t = \mathcal{T}^i_4(\pi_{t^{+++}}, y_{t+1})$$
$$= \frac{P(Y_{t+1} = y_{t+1} | X_{t^{+++}} = x_i) \pi^i_{t^{+++}}}{\sum_{x_j \in \mathcal{X}} P(Y_{t+1} = y_{t+1} | X_{t^{+++}} = x_j)) \pi^j_{t^{+++}}}.$$