Received: 14 October 2018 | Revised: 2 September 2019 | Accepted: 2 September 2019

DOI: 10.1002/asmb.2487

RESEARCH ARTICLE

WILEY

Cyber vulnerability maintenance policies that address the incomplete nature of inspection

Enhao Liu[1] | **Theodore T. Allen**[1] | **Sayak Roychowdhury**[2]

[1]Department of Integrated Systems Engineering, College of Engineering, The Ohio State University, Columbus, Ohio

[2]Industrial and Systems Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Correspondence
Enhao Liu, Department of Integrated Systems Engineering, College of Engineering, The Ohio State University, Columbus, OH 43210.
Email: liu.5045@osu.edu

Present Address
Enhao Liu, 210 Baker Systems Building, 1971 Neil Avenue, Columbus, OH 43210-1271.

Funding information
NSF, Grant/Award Number: 1409214 and 1912166

Abstract

In cybersecurity, incomplete inspection, resulting mainly from computers being turned off during the scan, leads to a challenge for scheduling maintenance actions. This article proposes the application of partially observable decision processes to derive cost-effective cyber maintenance actions that minimize total costs. We consider several types of hosts having vulnerabilities at various levels of severity. The maintenance cost structure in our proposed model consists of the direct costs of maintenance actions in addition to potential incident costs associated with different security states. To assess the benefits of optimal policies obtained from partially observable Markov decision processes, we use real-world data from a major university. Compared with alternative policies using simulations, the optimal control policies can significantly reduce expected maintenance expenditures per host and relatively quickly mitigate the most important vulnerabilities.

KEYWORDS

cyber vulnerabilities, incomplete inspection, partially observable Markov decision process (POMDP)

1 | INTRODUCTION

Virtually all types of devices that contain computers have so-called "cyber vulnerabilities," which offer ways for attackers to gain access or at least limit performance. Often, these vulnerabilities are publicly identified. A race then follows between hackers' finding and applying "exploits," and vendors offering patches that are discovered to be needed by scans and implemented by end users. If the hackers win, they cause losses. If the end users or defenders win, the vulnerability is removed or, at least, remediated to become harmless. Also, in many organizations, hosts are in a similar position with respect to exposure. They are protected by an outer firewall but exposed to other hosts within that firewall. Brewster[1] and others reported that Equifax lost 150 million records because hackers exploited a "critical" Apache server patchable vulnerability. The exploitation occurred only a short time after Equifax became aware of the vulnerability through scans. Of course, the impact of cyberattacks can take a financial toll on an organization. Many researchers have focused on the effects of "breaches," which reportedly cost millions of dollars. According to the 2017 Cost of Data Breach Study: United States, conducted by IBM and Ponemon Institute,[2] the average cost for each record lost in a data breach is $225 (which is essentially the same per capita cost for higher education, $200). The resulting total cost paid by organizations averaged $7.35 million.[2] However, the breaches are extremely rare events and a more general class of cybersecurity events might be called "incidents." Although there is typically no legally culpable data breach in these incidents, significant expenses are incurred for investigation and resolution.

Optimization of maintenance policies has been an active topic of research in several different areas. Virtanen[3] derived a control-theoretic optimal maintenance and sale date policy of a generic machine subject to deterioration and random

failure. Kuhn and Madanat[4] proposed robust maintenance policies under model uncertainty for asset management systems. Zhang et al.[5] suggested a Markov decision process (MDP) approach for the derivation of optimal maintenance policies for multicomponent systems under the influence of environmental conditions. In cybersecurity, an important challenge is to derive cost-effective maintenance policies for periodic maintenance for cyber vulnerabilities. In the cyber environment, the data needed for policy development are generally limited due to security concerns such as hosts being offline or incomplete inspection as described by Afful-Dadzie and Allen.[6,7] The common policy for this status quo is referred to as "out-of-sight is out-of-mind" (OSOM),[8] ie, ignoring the hosts that are unobservable despite actual vulnerabilities and associated intrusion risks that could be hidden on them. Afful-Dadzie and Allen[6] developed a data-driven MDP (DD-MDP) model for cyber vulnerability maintenance with parameter uncertainty arising from limited data about maintenance policies for patching vulnerabilities with low and medium severity levels. The states of a single system in their DD-MDP model are defined by severity levels of vulnerabilities and compromised incidents. A Monte Carlo Bayesian Reinforcement Learning (MCBRL)–based approach for cyber preventive maintenance was introduced by Allen et al,[9] which addressed the issue of availability of limited data for estimating transitions but ignored the issue of hosts being turned off or otherwise incomplete scans. Here, we seek to construct a general model for cyber vulnerability maintenance that can mitigate the effect of incomplete inspection and develop cost-effective policies for distinct types of hosts.

The generalized version of MDP, where the state of the system is not fully observable is described as a partially observable Markov decision process (POMDP) by Sondik,[10] Smallwood and Sondik,[11] and Sondik.[12] POMDPs deal explicitly with control of stochastic systems in which the exact state of a system is partially unobservable because of noisy, limited sensors or because of the nature of the problem itself. The POMDP framework[13] is widely exploited in several fields. Byon and Ding[14] proposed a stochastic model for a wind turbine that considers the incomplete inspection from monitoring equipment using a POMDP, and the states of the system are subject to several weather conditions. Kurt and Kharoufeh[15] considered the monotone optimal replacement of a system under Markov deterioration, which is under periodic inspection and operates in a controlled environment. Neves et al[16] also considered a deteriorating system with a discrete-state Markov process under periodic inspection with imperfect condition measurement and a constructed corresponding condition-based maintenance policy. Makis[17] considered a partially observable deteriorating system with continuous-time hidden Markov process and adapted this problem into a POMDP framework to develop an optimal condition-based maintenance policy. Wang et al[18] proposed a model that combines a dependent attack graph with a hidden Markov model to estimate attack states in a network for cost-benefit security hardening. POMDP-based models are particularly useful for cyber vulnerability maintenance problems, as in most cases, breaches remain undetected for a significant time. Roychowdhury[19] proposed a POMDP framework for cyber maintenance taking into account the uncertainty related to the states of a system being compromised or not.

In this article, we propose new parallel POMDPs to periodically execute cost-effective maintenance actions for cyber vulnerabilities based on the internal security state of hosts and external characteristics of hosts. The internal security states of hosts are measured by the severity level of vulnerabilities which are partially scanned or detected by a common vulnerability scoring system (CVSS)[20] in every period. The external characteristics of hosts containing operating systems, critical or ordinary server (including restricted data or not) and the mode of user administrative privilege may not be significantly related to the security state, but they can affect the maintenance costs and result in parallel POMDPs. Based on the DD-MDP model,[6] we generalize the states in Markov process by using only the highest severity level of vulnerabilities on hosts. Since the states could be unobservable because of incomplete inspections, the observations in POMDPs are defined by observable states (also called true states) and unknown states (also called none). However, the observable states might not directly correspond to the true states because of partial scanning or errors of scanning. The sensitivity analysis of observations is performed for cyber vulnerabilities inspired by data from a major university. We also utilize information about compromised hosts associated with "warnings" from the intrusion detection system. Each state-of-host data is associated with the possibility of incurring significant expenses for investigation and resolution. To estimate the probabilities of incurring warnings, several binary logistic regression models are performed. Additionally, the cost function in our proposed model consists of direct costs from maintenance actions and potential incidents costs associated with security states. To assess the benefits of optimal policies obtained from POMDPs, we perform a cyber vulnerability case study from a major university and compare alternative policies (including OSOM, fix, epsilon-greedy, and random) by running simulations for a five-year period.

The remainder of this article is organized as follows. In Section 2, we present the structure of the proposed stochastic model for cyber vulnerabilities. Section 3 documents a computational experiment involving 72 parallel POMDPs from a major university and presents comparisons among alternative maintenance policies using simulations. In Section 4, the implications for decision-makers and opportunities for future research are given.

2 | POMDPS FOR CYBER VULNERABILITIES MAINTENANCE

There are several ways to gain access to cyber vulnerabilities maintenance histories with CVSS information for organizations.[20] In this article, the data of a major university are obtained by using Nessus vulnerability scanning tools from Tenable Network Security. Each entry in the periodic Nessus scans is a discovered vulnerability with vulnerability and host characteristics. Vulnerability characteristics, for instance, include vulnerability type, vulnerability resolution, vulnerability first scanned date, and vulnerability risk. The vulnerability resolution is directly related to an organization's maintenance policy. Moreover, vulnerability risk is measured by CVSS, which evaluates the severity of vulnerabilities on a scale of 0 to 10 with 10 being extremely critical. The CVSS also provides subjective ranges including low (0 to 3.9), medium (4 to 6.9), high (7 to 9.9), and critical (10) severity levels. With respect to host characteristics, each entry also contains hosts information attached to every vulnerability, eg, the host IP address that identifies the unique host, host operating system, organization group ID that identifies the group to which the host belongs, and critical server that describes whether the host has access to restricted data.

Analytically, periodic action would be taken on a host according to a common maintenance policy. The evolution of the security state of hosts after taking action could be tracked based on the periodic Nessus scans. However, hosts could be unobservable because of offline operation or incomplete inspection. An example is described in Table 1. The real states of hosts are untraceable in this case. Hence, we describe the maintenance process for cyber vulnerabilities with a POMDP framework as introduced by Kaelbling et al.[13]

2.1 | States, Actions, Observations

A POMDP for cyber vulnerabilities maintenance is formally defined as a tuple $\langle S, A, T, O, \Theta, R, \gamma \rangle$ where:

- S is a finite set of states of cybersecurity for a type of host. The set of probability distributions over S shall be denoted by $b(S)$. Members of $b(S)$ shall be called belief states. As described by Afful-Dadzie and Allen,[6] the states of a host can be defined by the worst severity level of vulnerabilities on it. As discussed above, CVSS is widely used to evaluate the severity level of vulnerabilities. Hence, elements in S consist of low state, medium state, high state, and critical state, ie, $S = \{Low, Medium, High, Critical\}$. For instance, a host has been scanned in a period and it has several vulnerabilities with different severity levels, ie, two low vulnerabilities and three medium vulnerabilities. Then, the state of this host is medium in this scanned period. However, the "true" vulnerability state of hosts is generally unknown because of factors such as incomplete scans and zero-day vulnerabilities. For instance, the Heartbleed vulnerability (CVE-2014-0160) is

TABLE 1 Scanned records for a subset of hosts over 21 months with no records (0) in bold and the numerical values 1, 2, 3, 4 represent low, medium, high, and critical states, respectively

Host\month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	**0**
2	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
3	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**0**
5	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**0**	**0**	**0**
6	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**0**
7	2	2	2	2	2	2	2	2	2	2	2	4	4	4	4	4	2	2	2	2	2
8	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
9	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
10	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
11	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
12	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
13	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	2	2
14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	**0**	**0**
15	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	2	2
16	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	**0**	**0**
17	**0**	**0**	2	**0**	**0**	**0**	**0**	**0**	**0**	2	2	2	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
18	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
19	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
20	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	**0**	**0**	**0**

evaluated by CVSS as medium level at the beginning, despite it being easy to exploit. Here, our states are approximate in the sense that they are based on known scan results which, in general, do not correspond to the "true" vulnerabilities states of hosts.

- A is a finite set of actions that are applied to a host. The maintenance actions on hosts can be obtained from Nessus scans. Taking the Nessus scans for a major university as an example, there are four types of cyber maintenance actions, ie, auto-patching, research-accept, research-compensate, and remediation. Autopatching means that software vendors are permitted to update their software with vulnerabilities removed without direct local intervention, ie, no IT staffs involved. Research-accept means that IT staff has inspected a certain host and tried to find a solution for it, yet if there is no solution available, the result can be accepted. Research-compensate means that all the highest-level vulnerabilities in a host must be fixed because the result of no solution is unacceptable in this case. Remediation means that the host is taken offline or replaced to avoid more risks.
- $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function. For each $s' \in S$ and for each $a \in A$, $T(s, a, s')$ is the probability from the security state s of a host in a scanned period n to state s' in the next scanned period $n + 1$ under maintenance action a. Thus, for any time period $n \geq 1$, for each pair of states $s' \in S$ and for each action $a \in A$, there holds:

$$T(s, a, s') = \Pr(s^{n+1} = s' | s^n = s, a^n = a) \tag{1}$$

To estimate the parameters in state transition function, we use the simple method mentioned by Afful-Dadzie and Allen.[6] We tabulate the transition probabilities tensor $T(s, a, s')$ for all $s \in S$ which can be viewed as the parameters for a Dirichlet distribution, $N(s, a, 1), N(s, a, 2), \ldots, N(s, a, |S|)$, where $N(s, a, s')$ refers to the count of transitions from state s to state s' under action a. We then estimate $T(s, a, s')$ by using the standard estimation formula:

$$\hat{T}(s, a, s') = \frac{N(s, a, s')}{\sum_{j \in S} N(s, a, j)} \quad \forall s, s' \in S. \tag{2}$$

Examples of counts of transitions and estimated transition probabilities for Linux type of hosts are presented in Tables 2 and 3, respectively.

- O is a finite set of observations obtained from scanned software. As described by Afful-Dadzie and Allen,[6,7] hosts could be fully unobservable in some periodic scans as the result of incomplete inspections. The observations in POMDP are defined by observable states (ie, real states) and unknown states (ie, none). Thus, elements in O consist of observations of none, observations of low state, observations of medium state, observations of high state, and observations of critical state, ie, $O = \{None, Low, Medium, High, Critical\}$.
- $\Theta : S \times A \times O \rightarrow [0, 1]$ is an observation function. For each $s \in S$, for each $a \in A$, and for each $o \in O$, $\Theta(s, a, o)$ is the conditional probability of receiving observation o for a host in a scanned period n given that the state of it is s in that period and the action a has been taken on it in the previous period $n - 1$. Thus, for any time period $n \geq 2$, for each $s \in S$, for each $a \in A$ and for each $o \in O$, there holds:

$$\Theta(s, a, o) = \Pr(o^n = o | s^n = s, a^{n-1} = a) \tag{3}$$

Note that the only different element between observations set O and states set S is $\{None\}$, ie, $O = S \cup \{None\}$. In other words, if a host is scanned, then its observation is equal to its state; otherwise, its observation is *None*. The following assumptions are made for deriving an observation function from cyber vulnerabilities data set.

$$\begin{aligned}
&\Pr(o^n = None | s^n = s, a^{n-1} = a) = \Pr(o^n = None) \quad \text{for } s \in S, a \in A \\
&\Pr(o^n = o | s^n = s, a^{n-1} = a) = 1 - \Pr(o^n = None) \quad \text{for } o = s, o \in O, s \in S, a \in A \\
&\Pr(o^n = o | s^n = s, a^{n-1} = a) = 0 \quad \text{for } o \neq s, o \in O, s \in S, a \in A.
\end{aligned}$$

These assumptions describe that the observations do not depend on maintenance actions because observations are obtained by a vulnerability scanning system and the results of observations depend on whether a host is under incomplete inspection. Also, there is an independent assumption between observations and states due to observation of none. Hence, the estimation of $\Theta(s, a, o)$ depends on the estimation of $\Pr(o^n = None)$, which is the probability of

TABLE 2 Counts of transitions from a major university for Linux hosts under four actions: (a) Autopatching, (b) research-accept, (c) research-compensate, and (d) remediation

(a) Autopatching					(b) Research-accept				
From\to	Low	Medium	High	Critical	From\to	Low	Medium	High	Critical
Low	16 075	1404	35	15	Low	1073	4	0	0
Medium	412	127 519	756	59	Medium	463	610	4	0
High	0	412	127 519	815	High	34	429	610	4
Critical	0	0	412	128 334	Critical	8	53	18	799
(c) Research-compensate					**(d) Remediation**				
From\to	Low	Medium	High	Critical	From\to	Low	Medium	High	Critical
Low	1077	0	0	0	Low	100	100	0	0
Medium	1077	0	0	0	Medium	100	100	0	0
High	34	1043	4	0	High	100	100	0	0
Critical	8	53	817	0	Critical	100	100	0	0

TABLE 3 Estimated transition probabilities from a major university for Linux hosts under four actions: (a) Autopatching, (b) research-accept, (c) research-compensate, and (d) remediation

(a) Autopatching					(b) Research-accept				
From\to	Low	Medium	High	Critical	From\to	Low	Medium	High	Critical
Low	0.9171	0.0801	0.0020	0.0009	Low	0.9963	0.0037	0.0000	0.0000
Medium	0.0032	0.9905	0.0059	0.0005	Medium	0.4299	0.5664	0.0037	0.0000
High	0.0000	0.0032	0.9905	0.0063	High	0.0316	0.3983	0.5664	0.0037
Critical	0.0000	0.0000	0.0032	0.9968	Critical	0.0091	0.0604	0.0205	0.9100
(c) Research-compensate					**(d) Remediation**				
From\to	Low	Medium	High	Critical	From\to	Low	Medium	High	Critical
Low	1.0000	0.0000	0.0000	0.0000	Low	0.5000	0.5000	0.0000	0.0000
Medium	1.0000	0.0000	0.0000	0.0000	Medium	0.5000	0.5000	0.0000	0.0000
High	0.0315	0.9648	0.0037	0.0000	High	0.5000	0.5000	0.0000	0.0000
Critical	0.0091	0.0604	0.9305	0.0000	Critical	0.5000	0.5000	0.0000	0.0000

observing nothing (eg, because the host was turned off). This probability is easily estimated using the average proportion of unobservable hosts in all scanned periods.

Moreover, these assumptions also imply that the actual state is perfectly observed if the host is scanned. We relax these assumptions by considering the scenarios that the actual state is not perfectly captured but is approximately modeled by what is or is not observed. Then, the above assumptions are modified as follows:

$$\begin{aligned}
&\Pr(o^n = None | s^n = s, a^{n-1} = a) = \Pr(o^n = None) \quad \text{for } s \in S, a \in A \\
&\Pr(o^n = o | s^n = s, a^{n-1} = a) = (1 - \Pr(o^n = None)) \cdot (1 - Error) \quad \text{for } o = s, o \in O, s \in S, a \in A \\
&\Pr(o^n = o | s^n = s, a^{n-1} = a) = (1 - \Pr(o^n = None)) \cdot (Error/k) \quad \text{for } o < s, o \in O \setminus \{None\}, s \in S, a \in A \\
&\Pr(o^n = o | s^n = s, a^{n-1} = a) = 0 \quad \text{for } o > s, o \in O \setminus \{None\}, s \in S, a \in A,
\end{aligned}$$

where k refers to the number of instances such that $o < s$. An example of $o < s$ is $o = Low$ and $s = Medium$. Also, *Error* refers to the error rate of scanning. The numerical examples will be described in Section 3.1.

- $R : S \times A \to \mathbb{R}$ is a cost function. For each $s \in S$ and for each $a \in A$, $R(s, a) \in \mathbb{R}$ is the cost obtained by taking action on a host when the state of it is s. Many people view the costs of cybersecurity in terms of rare catastrophic events. Such events could have a major impact on the corporate bottom line, eg, the Sony hack. Other events could cripple power systems for extended time periods. However, our experience suggests that even leaving out the possibility of rare events, there are considerable direct costs that can be modeled in terms of the personnel time required to patch the vulnerabilities. The following equation summarizes the two main costs that we consider:

$$R(s, a) = D(a) + C(s), \tag{4}$$

where $D(a)$ represents expected direct costs of taking actions and $C(s)$ represents expected compromised cost due to potential incidents associated with security states. The estimations of these cost functions are presented in Section 2.3.

- $\gamma \in [0, 1]$ is a discount factor. It represents the difference between future costs and immediate costs for each time step. Smaller discount factors refer to the assumption that future costs are less important than immediate costs. Conversely, for a larger discount factor, which is close to 1, the future costs would count almost as much as the immediate costs. Therefore, the setting of the discount factor is used to control the number of iterations/horizons needed to converge for infinite horizons settings in many POMDP solvers. For example, using a 0.99 monthly discount factor, we are effectively assuming a horizon, which spans many computer hosts. However, such horizon does not represent the actual life span of physical hosts. Instead, it refers to the period in which users will continue with similar or the same software and the same patching policy as they change hosts.

Hence, $\langle S, A, T, O, \Theta, R, \gamma \rangle$ defines a POMDP problem. At the beginning of the control policy implementation, the implied initial belief is $b(s^0)$, which for instance can be assumed as a uniform distribution over S (ie, $(0.25, 0.25, 0.25, 0.25)$). At each time step n, an action a would be applied to the host and an observation o is obtained in the next time step, and then host's belief $b(s^n)$ is updated by Bayes' theorem:

$$b\left(s^{n+1} = s' | a^n = a, o^{n+1} = o\right) = \frac{\Pr\left(o^{n+1} = o | s^{n+1} = s', a^n = a\right)}{\Pr\left(o^{n+1} = o | b^n, a^n = a\right)} \sum_{s \in S} \Pr\left(s^{n+1} = s' | s^n = s, a^n = a\right) b\left(s^n = s\right) \quad \forall s' \in S, \tag{5}$$

where $\Pr(o^{n+1} = o | b^n, a^n = a)$ is a normalizing term, which is given by Equation (6)

$$\Pr\left(o^{n+1} = o | b^n, a^n = a\right) = \sum_{s' \in S} \Pr\left(o^{n+1} = o | s^{n+1} = s', a^n = a\right) \sum_{s \in S} \Pr\left(s^{n+1} = s' | s^n = s, a^n = a\right) b(s^n = s). \tag{6}$$

2.2 | Policies and value function

We use the notation similar to that of Spaan[21] so that a policy $\pi : b(S) \to A$ in POMDP is a mapping from probability distribution over states to actions. It can be defined using a value function $V^\pi : b(S) \to \mathbb{R}$. The value function V^π represents the expected discounted long-term cost by following policy π starting from belief b with finite periods h:

$$V^\pi(b) = \mathbf{E}_\pi \left[\sum_{n=0}^{h} \gamma^n R(b_n, \pi(b_n)) | b_0 = b \right], \tag{7}$$

where $R(b_n, \pi(b_n)) = \sum_{s \in S} R(s, \pi(b_n)) b_n(s)$ and belief $b_n(s)$ is the belief in period n.

Our goal is to find the optimal policy $\pi^\star$ that minimizes the value function, ie, the expected long-term cost. Then, the optimal value function $V^\star(b) = \min_\pi V^\pi(b)$. The value function is proved to be piecewise linear and convex in finite-horizon POMDP and can be represented using a set of alpha vectors as shown by Sondik,[10] Smallwood and Sondik,[11] and Sondik.[12] If there is only one period left to apply actions, then we consider only the current cost for a specific belief b. Then, the optimal value function of a single period is $V_0(b) = \min_{a \in A} \sum_{s \in S} R(s, a) b(s) = \min_{a \in A} b \cdot \alpha_0^a$, where $\cdot$ denotes the inner product, and α_0^a is a set of $|A|$ vectors $\alpha_0^a = (\alpha_0^a(1), \ldots, \alpha_0^a(|S|))$, one for each action $a : \alpha_0^a(s) = R(s, a)$. Thus, the optimal value function $V_n(b)$ in period n through value iterations is

$$V_n(b) = \min_{1 \le k \le |V_n|} b \cdot \alpha_n^k \tag{8}$$

V_{n+1} can be calculated using the Bellman backup operator H:

$$\begin{aligned} V_{n+1} &= HV_n = \cup_{a \in A} G_a, \text{ with } G_a = \oplus_{o \in O} G_a^o \text{ and} \\ G_a^o &= \left\{ \frac{1}{|O|} \alpha_0^a + \gamma g_{ao}^k | 1 \le k \le |V_n| \right\}, \end{aligned} \tag{9}$$

where V_n denotes a set of alpha vectors while $V_n(b)$ refers to the inner product of belief b and alpha vectors in V_n. The operator $\oplus$ refers to the cross sum operator. The vector g_{ao}^k is derived from back-projecting the alpha vector α_n^k from V_n

using action a and observation o, which is given by the Equation (10).

$$g_{ao}^{k} = \sum_{s' \in S} \Theta(s', a, o) T(s, a, s') \alpha_n^k(s') \tag{10}$$

Since the actions are associated with alpha vectors, dynamic programming is used for solving POMDP primarily to generate these alpha vectors given the finite horizons or discount factor less than 1. Many of the alpha vectors produced for V_{n+1} through the Bellman backup operator in Equation (9), do not contribute in minimizing the value function for a given belief. To address this issue, Cassandra et al[22] proposed the incremental pruning algorithm to accelerate the Bellman backup operator.

$$HV_n = \mathbf{prune}(\cup_{a \in A} G_a), \text{with } G_a = \mathbf{prune}\left(\mathbf{prune}\left(\bar{G}_a^1 \oplus \bar{G}_a^2\right) \ \ldots \ \oplus \bar{G}_a^{|O|}\right) \text{ and } \bar{G}_a^o = \mathbf{prune}\left(G_a^o\right), \tag{11}$$

where **prune** is the pruning subroutine.

Note that incremental pruning algorithms or other exact optimal POMDP solution methods have been shown to result in high computational complexity because they require computation of the optimal action for the whole space of belief points.[23] This motivates approximate algorithms that search for optimal solutions only for those sets of the belief simplex that could be reachable by interacting with the stochastic environment. Specifically, the point-based value iteration (PBVI) algorithm proposed by Pineau et al[24] is designed to approximately derive the alpha vectors. It applies an approximate backup operator $\widetilde{H}$ to an initial set of belief points B, and then expands the set of belief points and searches for new solutions. Hence, the Equation (9) can be stated as

$$\begin{aligned} V_{n+1} = \widetilde{H} V_n &= \underset{G_a^b, \forall a \in A}{\operatorname{argmin}} \left(G_a^b \cdot b\right), \text{ with } G_a^b = \sum_{o \in O} \underset{\alpha \in G_a^o}{\operatorname{argmin}} (\alpha \cdot b) \quad \forall b \in B \\ \text{and } G_a^o &= \left\{ \frac{1}{|O|} \alpha_0^a + \gamma g_{ao}^k | 1 \le k \le |V_n| \right\}, \end{aligned} \tag{12}$$

where g_{ao}^k is calculated by Equation (10).

2.3 | Estimation of cost function

As mentioned before, the cost function $R(s, a)$ consists of two parts, ie, expected direct cost $D(s)$ of taking maintenance action, and expected compromised cost $C(s)$ due to potential incidents.

The estimates of maintenance action cost mainly depend on the average labor cost per hour and the average time of patching vulnerabilities as suggested by Afful-Dadzie and Allen.[6] The setting for the maintenance actions cost is presented in the cyber vulnerability case study in Section 3.1. In terms of expected compromised cost, more than 90% of actual attacks exploit known vulnerabilities to steal sensitive and confidential information as observed by Cockburn.[25] As discussed before, vulnerabilities are assigned risk ratings by the CVSS. We define the incident rate $p(s)$ as the probability that an incident event is detected by the intrusion detection system for a specific type of host in state s in period n. In other words, the incident event is associated with the highest severity level of vulnerabilities on hosts.

Hence, the expected compromised cost $C(s) = p(s) \times c$, where c is the expected value of loss resulted from the incident. Some researchers studied several prediction methods or classification techniques for cyber risks through different vulnerability databases and different measurements and features such as Soska and Christin,[26] Liu et al,[27] Sabottke et al,[28] Hao et al,[29] Zhang et al,[30] Zhu et al,[31] Liu,[32] and Tavabi et al.[33] Here, we follow the previous work of Liu[32] and exploit multiple logistic regression for predicting incident rate $p(s)$, allowing for several predictor variables related to the cyber environment. The methodology of multivariate logistic regression models typically includes multiple steps involving model selection and modeling diagnostics (see the work of Hosmer et al[34]).

For a set of k independent predictor variables denoted by the vector $\mathbf{x}' = (x_1, x_2, \ldots, x_k)$, let the conditional probability of the outcome being present be denoted by $P(y = 1|\mathbf{x}) = \pi(\mathbf{x})$. The logit of the multiple logistic regression model is given by

$$g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots \ + \beta_k x_k, \tag{13}$$

TABLE 4 Predictor variables sheet

Cyber factors	Factors description	Value and data type	Factors name
Basic factors	Operating Systems	00=Linux 10=OtherOS 01=Windows; categorical	OS Type
	Host Roles (ie, General-Purpose)	True=General-Purpose False=Others; categorical	General-Purpose
	Administrative Privilege	True=User administrative Privilege is allowable False= Management tech is performed to patch; categorical	AdminPrivilege
	Time Trend (ie, Monthly scanned period)	integer	TimeTrend
CVSS factors	The Worst CVSS Severity Level	1=Low Severity Level 2=Medium Severity Level 3=High Severity Level 4=Critical Severity Level; numerical	WorstCVSSLevel
	The Worst CVSS Base Score	[0,10]; numerical	WorstCVSS
	The Average CVSS Base Score	[0,10]; numerical	AvgCVSS
Vulnerabilities Counts factors	Counts of Vulnerabilities in Low Severity Level	integer	VulnCountLow
	Counts of Vulnerabilities in Medium Severity Level	integer	VulnCountMedium
	Counts of Vulnerabilities in High Severity Level	integer	VulnCountHigh
	Counts of Vulnerabilities in Critical Severity Level	integer	VulnCountCritical

in which case the logistic regression model is

$$\pi(x) = \frac{e^{g(x)}}{1 + e^{g(x)}}. \tag{14}$$

The dependent variable is the incident event that is captured by an intrusion detection system. The independent variables are relevant cyber factors obtained from Nessus scans. Typically, these cyber factors group into four sets, ie, basic factors, CVSS factors, and vulnerabilities count factors, which are listed in Table 4.

Here, we study four factors which relate to basic properties of hosts: the operating systems (Windows, Linux, or others), the host roles (general-purpose and non-general-purpose), the privilege status (operator has administrative privilege or not), and the scan period. For the host roles, the general-purpose type of hosts includes computers, database servers and file servers, while the non-general-purpose type of hosts includes printers, routers, hypervisors, etc. For the privilege status, consider that, in student computer labs, students do not have "root access" or administrator privilege because they have no organizational responsibility for host maintenance. Risks are heightened when a host has user administrative privilege. Alternatively, many hosts such as lab computers utilize management techniques to perform patches or to upgrade software applications with fewer resources. For example, a patch of one specific vulnerability will be applied to all the hosts that belong to a managed group.

For the purpose of predicting warnings or incidents, CVSS factors include three different measurements of hosts' security state by CVSS, ie, the worst CVSS severity level, the worst CVSS base score, and the average CVSS base score. Here, the average CVSS base score refers to the mean value of CVSS base scores of all vulnerabilities observed on a given host for a certain scanned period. For instance, if a host has one vulnerability with base score 1.0 and another with base score 5.0, the average CVSS base score of this host is $(1.0 + 5.0)/2$, ie, 3.0. Considering that these three CVSS factors have difference scales, we normalize all of them to lie within the range from -1 to 1.

Since there are three groups of predictor variables, several potential models can be constructed to find the best-fitted models through a series of steps. We first consider only the variable combinations between CVSS factors and basic factors. Selecting three different factors from the group of CVSS factors (ie, WorstCVSSLevel, *WorstCVSS* and *AvgCVSS*) produces three combinations with all basic factors. Only the model with all basic factors will be a benchmark model. This first set of models are first order. Considering the second order of CVSS factors produces three different models that contain all basic factors and three different CVSS factors with their second-order terms. We describe these three

TABLE 5 Potential models across four groups

Models		Factors in model
First order models	Model 1	Basic factors
	Model 2	WorstCVSSLevel + Basic factors
	Model 3	WorstCVSS + Basic factors
	Model 4	AvgCVSS + Basic factors
Second-order models	Model 5	WorstCVSSLevel + WorstCVSSLevel^2 + Basic factors
	Model 6	WorstCVSS + WorstCVSS^2 + Basic factors
	Model 7	AvgCVSS + AvgCVSS^2 + Basic factors
First-order models with vulnerability counts factors	Model 8	VulnCount factors + Basic factors
	Model 9	WorstCVSSLevel + VulnCount factors + Basic factors
	Model 10	WorstCVSS + VulnCount factors + Basic factors
	Model 11	AvgCVSS + VulnCount factors + Basic factors
Second-order models with vulnerability counts factors	Model 12	WorstCVSSLevel + WorstCVSSLevel^2 + VulnCount factors + Basic factors
	Model 13	WorstCVSS + WorstCVSS^2 + VulnCount factors + Basic factors
	Model 14	AvgCVSS + AvgCVSS^2 + VulnCount factors + Basic factors

models as "second-order models". In addition, we construct seven different models by adding all vulnerability counts factors into "first-order model" and "second-order model", respectively. This produces a total of 14 potential models across four groups of models, which are listed in Table 5.

To evaluate which potential models fit the data accurately, a sequence of selection steps is conducted.

- Step 1: For each model, a logistic regression is conducted to get estimated coefficients of predictor variables and corresponding p-value based on Wald test.
- Step 2: From p-value from Step 1, the insignificant variables will be removed from each model at significance level 0.05. The reduced models thus contain only those variables thought to be significant.
- Step 3: A likelihood ratio test generates a relative comparison of each reduced model with the corresponding full model to ensure that the removed variables are not essential to the full model. Then, some full models can be replaced with their corresponding reduced models at significance level 0.05.
- Step 4: The goodness of fit test from Hosmer et al[35] is then applied to generate an absolute evaluation of the model. The Le Cessie-Van Houwelingen normal test statistic[36] for the unweighted sum of squared errors is used for global goodness of fit. The "not good" models are deselected at significance level 0.05.
- Step 5: For those models that have only one variable different, we conduct a likelihood ratio test to remove redundant models, and finally determine the candidate models.

Since this article mainly presents the POMDPs for cyber vulnerabilities maintenance policies, the full details of the selection steps are beyond the scope of this paper. Finally, after model selection and modeling diagnostics, we predict the incident rate $p(s)$ based on the final model (Model 5) whose coefficients and statistics are presented in Table 6. The Wald test statistics are shown in the fourth column and the corresponding p-values are given in the fifth column. Setting significance level at 0.05, we conclude that the variables worst CVSS severity level with its square term, operating systems, general-purpose, administrative privilege, and time trend are possibly significant.

Because the final model involves categorical variables with several levels resulting in multiple degrees of freedom, the Wald statistics can be applied to assess the significance of the coefficients. The Wald statistic test for the coefficients for *OS type* = Other OS is not significant at 0.05 in Table 6. We then conduct the likelihood ratio test to compare the final model to the one without variable operating systems. Under the null hypothesis that the coefficients for OS are equal to zero, the value of the likelihood ratio test is 121.3078 with two degrees of freedom, and the p-value for the test is $P(\chi^2(2) > 121.3078) \approx 0$. Thus, we reject the null hypothesis and conclude that the existence of variable operating systems is also significant to the final model.

As shown in Table 6, the following Equation (15) gives the corresponding estimated logit:

$$\begin{aligned}\hat{g}_R(\mathbf{x}) = &-7.628 + 0.771 \times WorstCVSSLevel + 0.617 \times WorstCVSSLevel^2 \\ &- 0.269 \times OSType_{OtherOS} - 1.540 \times OSType_{Windows} + 1.029 \times GeneralPurpose_{TRUE} \\ &+ 1.900 \times AdminPrivilege_{TRUE} - 0.126 \times TimeTrend.\end{aligned} \tag{15}$$

TABLE 6 Estimated coefficients and statistics for the final logistic regression model

Variable	Coefficient	Standard error	z-value	p-value
Intercept	−7.62843	0.22431	−34.009	$< 2 \times 10^{-16}$
WorstCVSSLevel	0.77125	0.11176	6.901	5.17×10^{-12}
WorstCVSSLevel^2	0.61680	0.15798	3.904	9.45×10^{-5}
OS type=Other OS	−0.26864	0.15960	−1.683	0.0923
OS type=Windows	−1.53934	0.14008	−10.989	$< 2 \times 10^{-16}$
General-Purpose=TRUE	1.02874	0.18904	5.442	5.27×10^{-8}
AdminPrivilege=TRUE	1.89955	0.16854	11.27	$< 2 \times 10^{-16}$
TimeTrend	−0.12638	0.01146	−11.027	$< 2 \times 10^{-16}$

Then, the estimated probability of incident $p(s)$ is given by Equation (16).

$$\hat{P}(y = 1|\mathbf{x}) = \hat{\pi}(\mathbf{x}) = \frac{e^{\hat{g}_R(\mathbf{x})}}{1 + e^{\hat{g}_R(\mathbf{x})}}. \tag{16}$$

3 | CYBER VULNERABILITY CASE STUDY

To derive the optimal policy and associated optimal expected long-term costs, we apply the PBVI algorithm as described in Section 2.2. First, we present the parameter settings of a cyber vulnerability case study for the proposed stochastic models. Next, we present the optimal alpha vectors obtained by proposed parallel POMDPs. Moreover, we conduct simulations for each type of hosts for five-year periods with optimal policy, OSOM policy, fixed policy, random policy, and epsilon-greedy policy. Several comparisons are presented to assess the benefits of optimal policies in terms of the expected costs and hardening security states.

3.1 | Problem description

The cyber vulnerability case study data from a major university includes 21 months of vulnerability scans using Nessus scan tool from Tenable. The total number of entries of 21 scans is almost 2.3 million. The scans contain 2028 distinct vulnerabilities whose risk levels measured by CVSS are low, medium, high, and critical. As discussed in Section 2.1, there are four types of cyber maintenance actions. For simplicity, we index the four actions as $\{1, 2, 3, 4\}$, which represents autopatching (turn on autopatching only), research-accept, research-compensate, and remediation, respectively. Also, the scans span 24 types of 48 495 distinct hosts with fixed IP address categorized by operating systems (Linux, Windows, and others), access to restricted data (yes or no), administrative privilege (yes or no) and general-purpose (yes or no). Most hosts are of the general-purpose type (ie, computers, database servers, and file servers). Therefore, we specifically consider 12 types of general-purpose hosts:

- Operating systems are characterized as OS = {Linux, Windows, Other}. For instance, if a host's operating systems argument contains the keyword of "Linux", then the corresponding OS is designated as Linux.
- Access to restricted data is characterized as Restricted = {True, False}. If a host's access to restricted data argument is true, then the corresponding Restricted is assigned as True, which means that this host has access to restricted data. Otherwise, Restricted is equal to False, which indicates that this host contains only normal data.
- Administrative privileges are characterized as Admin = {True, False}. If a host has administrative privileges (Admin = True), it indicates that the host has users with the ability to install software. Otherwise, Admin is equal to False, which indicates that this host utilizes management techniques to perform patches or to upgrade software applications with fewer resources. For example, a patch of one specific vulnerability will be applied to all the hosts that belong to a managed group. Consequently, it would affect the resulting cost.

Virtually all our hosts are inside the outer firewall of the university and have approximately similar visibility to other hosts. This follows because many hosts (and accounts) are known to be compromised that are inside the firewall. This assumption was used predicting warning incidents in developing Equation (15).

Moreover, we consider six scenarios of error rate of scanning (ie, $Error = \{0\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$) to perform sensitivity analysis by varying the estimations of observation functions as discussed in Section 2.1. Hence, we would perform $12 \times 6 = 72$ parallel POMDPs to derive the optimal policies for each type of hosts given start belief. Each POMDP has a tuple $\langle S, A, T, O, \Theta, R, \gamma \rangle$. The parameters settings are shown as follows:

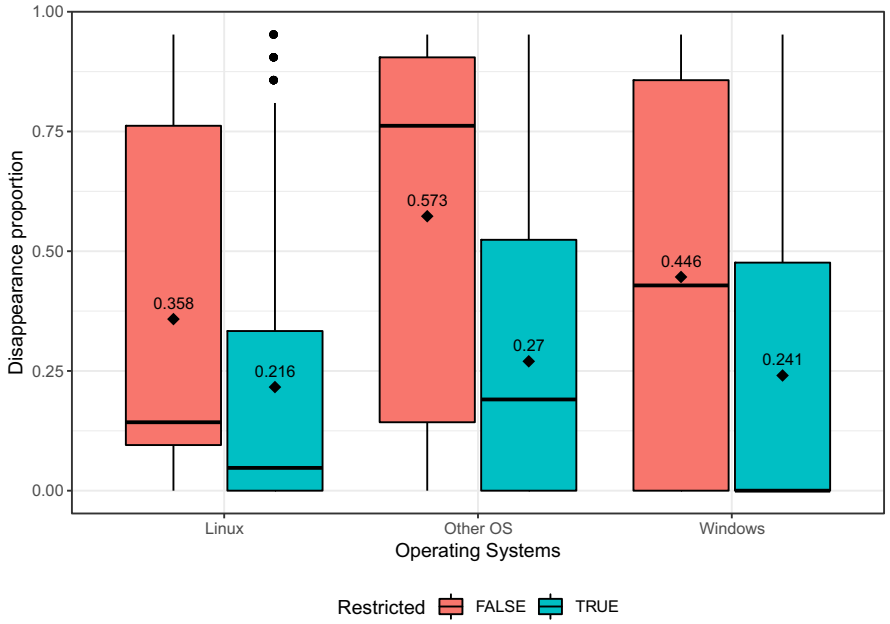

FIGURE 1 Boxplot of disappearance proportion of time for all hosts grouped by *OS* and *Restricted* in 21 scanned periods [Colour figure can be viewed at wileyonlinelibrary.com]

We assume that the transition functions for each type of hosts depend only on OS and actions. That is, there are $3\times4 = 12$ transition functions $T(s, a, s')$. An example of estimated transition probabilities of Linux host under four actions are presented in Table 3.

Similarly, the observation functions $\Theta(s, a, o)$ for each type of hosts depend on OS, Restricted so that there are $3 \times 2 = 6$ observation functions and each of them would be modified given the scenarios of error rate of scanning. According to Section 2.1, the estimation of $\Theta(s, a, o)$ relies on the estimation of $\Pr(o^n = None)$, which could be estimated by the average proportion of disappeared times in the 21 scanned periods for each type of host. We then calculate the proportion of disappeared times for each unique host in the 21 scanned periods. Table 1, for example, shows scanned records for a subset of unique hosts in 21 scanned periods. Each row represents the records of a single host and zero refers to "disappearance." The proportion of disappeared times for a single host is the counts of zeros divided by 21. We then present the distributions of proportions of disappeared times and associated mean value for all hosts classified by OS and Restricted in Figure 1.

For example, $\Pr(o^n = None)$ for Linux type of hosts with restricted data is estimated at 0.216, which means that this type of hosts is not observable about 21.6% time over 21 scanned periods on average. According to the modified assumptions of observation function in Section 2.1, the estimated observation probabilities for each type of hosts would be generated given the estimated $\Pr(o^n = None)$ and error rates of scanning. An example of the estimated observation probabilities matrix with error rate 0% and 5% for Linux hosts is presented in Table 7. Taking the third row ($s = High$) in Table 7(b) as an example. The hosts' type is Linux system and Restricted = True, and the scanning error is 5%. We first find the average proportion of disappeared times for this type of hosts in Figure 1 to estimate $\Pr(o = None|s = High)$, ie, $\Pr(o = None|s = High) = 0.216$. Since there is 5% error, we calculate $\Pr(o = High|s = High) = (1 - 0.216) \cdot (1 - 5\%) = 0.7488$. Also, k is 2 in this case because there are two observations (ie, $o = Low$ and $o = Medium$) "less" than $s = High$. We then compute $\Pr(o = Low|s = High) = \Pr(o = Medium|s = High) = (1 - 0.216) \cdot (5\%/2) = 0.0196$. For $o = Critical > s = High$, we would have $\Pr(o = Critical|s = High) = 0$.

Next, the settings for cost functions $R(s, a)$ are illustrated. Recall that $R(s, a)$ consists of two parts: expected direct cost $D(a)$ and expected compromised cost $C(s) = p(s) \times c$. Also, $p(s)$ refers to the probability that an incident event detected by intrusion detection system for a specific type of host in state s in period n. c is the expected value of loss resulted from the incident and it is related to whether or not a host has access to restricted data. Given that each type of host would render slightly different parameters in these two parts, we elaborate the differences based on the characteristics of hosts (ie, OS, Restricted, and Admin).

If an attack occurred on a host with restricted data (Restricted = True), there would be a relatively large loss than for a host with normal data (Restricted = False). As described by Afful-Dadzie and Allen,[6] conservative estimates of the value of loss c resulting from the incident for these two types of hosts are \$3000 and \$20 000, respectively.

According to Equation (15) and (16), we could predict incident rate $p(s)$ for OS = {Linux, Windows, Other}, Admin = {True, False}, GeneralPurpose = True, TimeTrend = 21 (ie, the current period) and WorstCVSSLevel = {1,2,3,4}, which

TABLE 7 Estimated observation probabilities from a major university for Linux hosts

(a) *Restricted = True, Error = 0%*						(b) *Restricted = True, Error = 5%*					
State\Obs	**None**	**Low**	**Medium**	**High**	**Critical**	**State\Obs**	**None**	**Low**	**Medium**	**High**	**Critical**
Low	0.216	0.784	0	0	0	Low	0.216	0.7840	0	0	0
Medium	0.216	0	0.784	0	0	Medium	0.216	0.0392	0.7448	0	0
High	0.216	0	0	0.784	0	High	0.216	0.0196	0.0196	0.7448	0
Critical	0.216	0	0	0	0.784	Critical	0.216	0.0131	0.0131	0.0131	0.7448
(c) *Restricted = False, Error = 0%*						**(d) *Restricted = False, Error = 5%***					
State\Obs	**None**	**Low**	**Medium**	**High**	**Critical**	**State\Obs**	**None**	**Low**	**Medium**	**High**	**Critical**
Low	0.358	0.642	0	0	0	Low	0.358	0.642	0	0	0
Medium	0.358	0	0.642	0	0	Medium	0.358	0.0321	0.6099	0	0
High	0.358	0	0	0.642	0	High	0.358	0.0161	0.0161	0.6099	0
Critical	0.358	0	0	0	0.642	Critical	0.358	0.0107	0.0107	0.0107	0.6099

TABLE 8 The estimated incident rate $p(s)$ for each type of hosts in the current scanned period

(a) Hosts with administrative privilege (Admin = True)				(b) Hosts with management (Admin = False)			
State	**Windows**	**Linux**	**Other OS**	**State**	**Windows**	**Linux**	**Other OS**
Low	0.0072%	0.0349%	0.0257%	Low	0.0010%	0.0049%	0.0036%
Medium	0.0114%	0.0530%	0.0405%	Medium	0.0017%	0.0079%	0.0061%
High	0.0190%	0.0885%	0.0677%	High	0.0028%	0.0133%	0.0101%
Critical	0.0550%	0.2558%	0.1956%	Critical	0.0082%	0.0384%	0.0293%

TABLE 9 The estimated labor costs $L(a)$ and estimated direct costs $D(a)$ for three operating systems under four actions

(a) The estimated labor costs $L(a)$				(b) The estimated direct costs $D(a)$			
Action	**Windows**	**Linux**	**Other OS**	**Action**	**Windows**	**Linux**	**Other OS**
Action 1	\$0	\$0	\$0	Action 1	\$0	\$0	\$0
Action 2	\$45	\$35	\$25	Action 2	\$68	\$53	\$38
Action 3	\$215	\$170	\$135	Action 3	\$323	\$255	\$203
Action 4	\$1000	\$1000	\$1000	Action 4	\$1500	\$1500	\$1500

is presented in Table 8. For instance, a Linux host of general type with administrative privilege is scanned with critical state (ie, state 4) in period 21. The predicted incident rate is then computed by setting $WorstCVSSLevel = 1$ (normalize 4 into $[-1, 1]$), $OS = Linux$, $Admin = True$, $GeneralPurpose = True$, $TimeTrend = 21$; $g(\mathbf{x}) = -7.628 + 0.771 * 1 + 0.617 * 1 + 1.029 + 1.9 - 0.126 * 21 = -5.957$, and then $P(y = 1|\mathbf{x}) = e^{-5.957}/(1+e^{-5.957}) \approx 0.26\%$ which is the same as in Table 8 (a).

There are four types of cyber maintenance actions. Each action is associated with a direct cost $D(a)$ that includes labor cost $L(a)$ and nonlabor expenditures including utility expenses, administration cost, etc. From the chief IT staff of the major university, they provide a list of estimated labor cost $L(a)$, which is shown in Table 9 (a). We assume an overhead of 50% of the labor cost as the nonlabor expenditures. Hence, we would have $D(a) = 1.5L(a)$, which is shown in Table 9 (b). Additionally, applied management techniques by a single host could reduce total action costs because the action of that host would execute this action on all hosts with management techniques (Admin = False). Thus, we assign a multiplier with 1/50 to reduce the direct action cost for hosts with management. The denominator of multiplier is determined by the average number of hosts in the same management group. Consequently, the cost function $R(s, a)$ is summarized in Equation (17).

$$R(s,a) = \begin{cases} D(a) + p(s) \times c & \text{for } Admin = True \\ D(a) \times \text{multiplier} + p(s) \times c & \text{for } Admin = False \end{cases} \tag{17}$$

Also, the discount factor γ is set to be 0.99 monthly. The tolerance level used in PBVI solver is 0.1.

3.2 | POMDP results and implications

With the selected parameters settings, 72 parallel POMDPs are solved using the PBVI algorithm in which 10 000 belief points are generated from 100 steps after 100 random initial points. From this we obtain the 72 sets of alpha vectors associated with actions. Given a starting belief point, the optimal action would be selected with minimal dot product of alpha vectors and belief according to Equation (8). Examples of optimal alpha vectors obtained for various factor levels Admin is True and False and error rate is 0% and 5% are described in Table 10.

TABLE 10 Alpha vectors for Linux type of hosts with restricted data

(a) Four alpha vectors (Admin = True, Error = 0%)					(b) Four alpha vectors (Admin = False, Error = 0%)				
Action	**coef. 1**	**coef. 2**	**coef. 3**	**coef. 4**	**Action**	**coef. 1**	**coef. 2**	**coef. 3**	**coef. 4**
3	1378.5	1382.1	1426.9	1579.6	2	142.0	142.6	146.6	156.6
3	1368.1	1371.7	1419.5	1580.6	2	138.0	140.3	145.7	163.9
2	1165.8	1193.4	1295.5	1641.1	2	137.8	140.3	145.9	164.0
1	1117.2	1159.1	1303.4	1623.5	1	137.0	140.6	147.3	164.4
(c) Five alpha vectors (Admin = True, Error = 5%)					**(d) Six alpha vectors (Admin = False, Error = 5%)**				
Action	**coef. 1**	**coef. 2**	**coef. 3**	**coef. 4**	**Action**	**coef. 1**	**coef. 2**	**coef. 3**	**coef. 4**
3	1379.2	1382.8	1427.6	1581.1	3	141.7	142.3	145.7	155.7
3	1368.8	1372.4	1420.2	1582.2	3	141.0	141.6	145.7	155.7
2	1166.5	1194.1	1296.8	1644.9	2	137.6	139.6	144.8	163.4
1	1152.5	1161.1	1304.9	1627.7	2	137.0	139.3	144.8	163.4
1	1117.9	1159.8	1305.1	1627.6	2	136.8	139.3	145.0	163.5
					1	136.0	139.7	146.5	163.9

TABLE 11 The optimal actions for a Linux host with restricted data under five belief scenarios (first two horizon)

(a) Admin = True, Error = 0%						
Scenario	**Start**	**None**	**Low**	**Medium**	**High**	**Critical**
0	1	1	1	1	2	3
1	1	1	1	1	2	3
2	1	1	1	1	2	3
3	2	1	1	1	2	3
4	3	1	1	1	2	3
(b) Admin = False, Error = 0%						
Scenario	**Start**	**None**	**Low**	**Medium**	**High**	**Critical**
0	2	2	1	2	2	2
1	1	2	1	2	2	2
2	2	2	1	2	2	2
3	2	2	1	2	2	2
4	2	2	1	2	2	2

We then consider five scenarios for starting belief points, ie, uniform, low state, medium state, high state, and critical state. The uniform distribution of belief $(0.25, 0.25, 0.25, 0.25)$ describes a host with full uncertainty on its true state at the beginning, which coincides with the unobservability discussed before. If a host is observed in low state, then the associated belief is $(1, 0, 0, 0)$. Similarly, belief $(0, 1, 0, 0)$ refers to medium state, belief $(0, 0, 1, 0)$ refers to high state, and belief $(0, 0, 0, 1)$ refers to critical state. We recode the five scenarios of start beliefs as $\{0, 1, 2, 3, 4\}$, which denote uniform, low state, medium state, high state, and critical state, respectively.

Next, we present the optimal actions (first two horizons) for Linux hosts and Windows hosts when the error rate of scanning is 0%. We consider that both contain restricted data and that Admin is True and False. In Table 11(a), the second column describes the optimal actions taken in the current period. These optimal actions associated with belief scenarios are selected with the minimal dot product of alpha vectors in Table 10(a) and the corresponding belief. The third to seventh columns describe the optimal actions taken in the next period under five scenarios based on the observations scanned. Take the row of scenario 0 (ie, start belief is uniform) as an example. A Linux host with restricted data and administrative privilege is fully unobservable (also known as incomplete inspection) in the current period. Then, the optimal action in the current period is to execute autopatching ($a = 1$) in order to minimizing the immediate cost and future cost. In the next period, there are five possible observations scanned according to the third to seventh columns. If this host is still unobservable ($o = None$) in the next scanned period, the belief would be updated based on Equation (5) and Equation (6). The next optimal action is then selected with minimal dot product of alpha vectors and updated belief. In this case, the next optimal action is still autopatching ($a = 1$). If this host is observed in the next scanned period and it has low or medium vulnerabilities ($o = Low, Medium$), then the next optimal action is autopatching ($a = 1$). If it is observed with high vulnerabilities ($o = High$) or critical vulnerabilities ($o = Critical$), the optimal actions are research-accept ($a = 2$) and research-compensate ($a = 3$), respectively.

Table 11 (a) and (b) compares the action difference between two types of Linux hosts since the optimal policy is different for each type of host. When there is no scanning error, compared with actions taken on a Linux host with restricted data

TABLE 12 The optimal actions for a Windows host with restricted data under five belief scenarios (first two horizon); The NA value means that it is impossible to observe "Low" when taking action 1 on scenario 3

(a) Admin = True, Error = 0%						
Scenario	**Start**	**None**	**Low**	**Medium**	**High**	**Critical**
0	1	1	1	1	1	2
1	1	1	1	1	1	2
2	1	1	1	1	1	2
3	1	1	NA	1	1	2
4	2	1	1	1	1	2
(b) Admin = False, Error = 0%						
Scenario	**Start**	**None**	**Low**	**Medium**	**High**	**Critical**
0	1	2	1	1	2	2
1	1	1	1	1	2	2
2	1	2	1	1	2	2
3	2	2	1	1	2	2
4	2	2	1	1	2	2

and administrative privilege, a Linux host with restricted data and management techniques (ie, Admin = False) tends to choose inferior actions (lower numbering code) for critical vulnerabilities while it selects superior actions when the observation is that the host is turned off or missing or medium vulnerabilities are observed. This is explained by the process in the stochastic model. According to the transitions, observations, and cost functions, POMDPs for each type of hosts would "learn" the performance of actions taken in terms of hardening the security state and minimizing associated costs. According to Equation (17), the mode of administrative privilege could affect the cost function. When Admin = False, the costs of taking actions would be dramatically reduced as the result of management technique. Then, it is still cost-effective to take superior actions on this Linux host with management techniques when no observations detected or medium vulnerabilities observed. From Table 12 (a) and (b), we could see that such comparisons for Windows type of hosts are quite similar.

Table 11 (a) and Table 12 (a) compare the action difference between Linux hosts and Windows hosts with administrative privilege. For simplicity, we designate these two types of hosts as Linux and Windows because the only difference in characteristics of these two hosts is the operating system. In Table 12 (a), compared with the actions taken on Linux, Windows tends to choose inferior actions (lower numbering code) for high and critical vulnerabilities. As we discussed before, POMDPs would "learn" the performance of each action and then choose the best one. Hence, autopatching is chosen for Windows indicating that this action is enough to deal with vulnerabilities with low, medium, and high levels. When the magnitudes of cost functions for different type of hosts are comparable (eg, they have the same mode of administrative privilege), if an advanced or superior action is chosen, it implies that this type of hosts is vulnerable to higher level vulnerabilities. Also, since a more vulnerable host would evidently lead to relatively larger costs, a Linux system is more vulnerable than a Windows system when high and critical vulnerabilities have been scanned, and superior actions such as research-compensate ($a = 3$) should be taken on it.

3.3 | Simulations and comparisons

In this section, we evaluate the benefits of applying optimal policy derived from the proposed POMDPs in terms of expected individual costs and the evolution process of security states by repeating simulations for five-year periods (60 monthly horizons) in 1000 trials. The simulation procedures of each trial are illustrated as follows:

- *Step 0*: *Initialization*. Setting the initial number of horizons as 1. An initial belief is randomly selected from the five scenarios of starting belief (ie, uniform, low, medium, high, and critical). Also, the corresponding initial observation could be generated based on the initial belief. For example, initial observation would be *None* if the initial belief is uniformly distributed. An initial state is then sampled from S based on the distribution of the initial belief. The initial total costs is assigned as 0, and the initial discount factor γ is assigned as 1.
- *Step 1*: *Action mapping*. For the simulation of optimal policy, an action is then taken with the minimal dot product of alpha vectors from the solved POMDP and the current belief. For other policies, this *Action mapping* would be modified, which we will discuss later.
- *Step 2*: *Updates of costs, states, observations, and belief*. An immediate cost is obtained based on the cost function given the pair of state and action. Then, the new total costs would be the sum of the previous total costs and the immediate cost multiplied by discount factor γ. Next, a new state is sampled from S based on the distribution of transition probabilities given the pair of previous state and action. Also, a new observation is sampled from O based on the distribution of

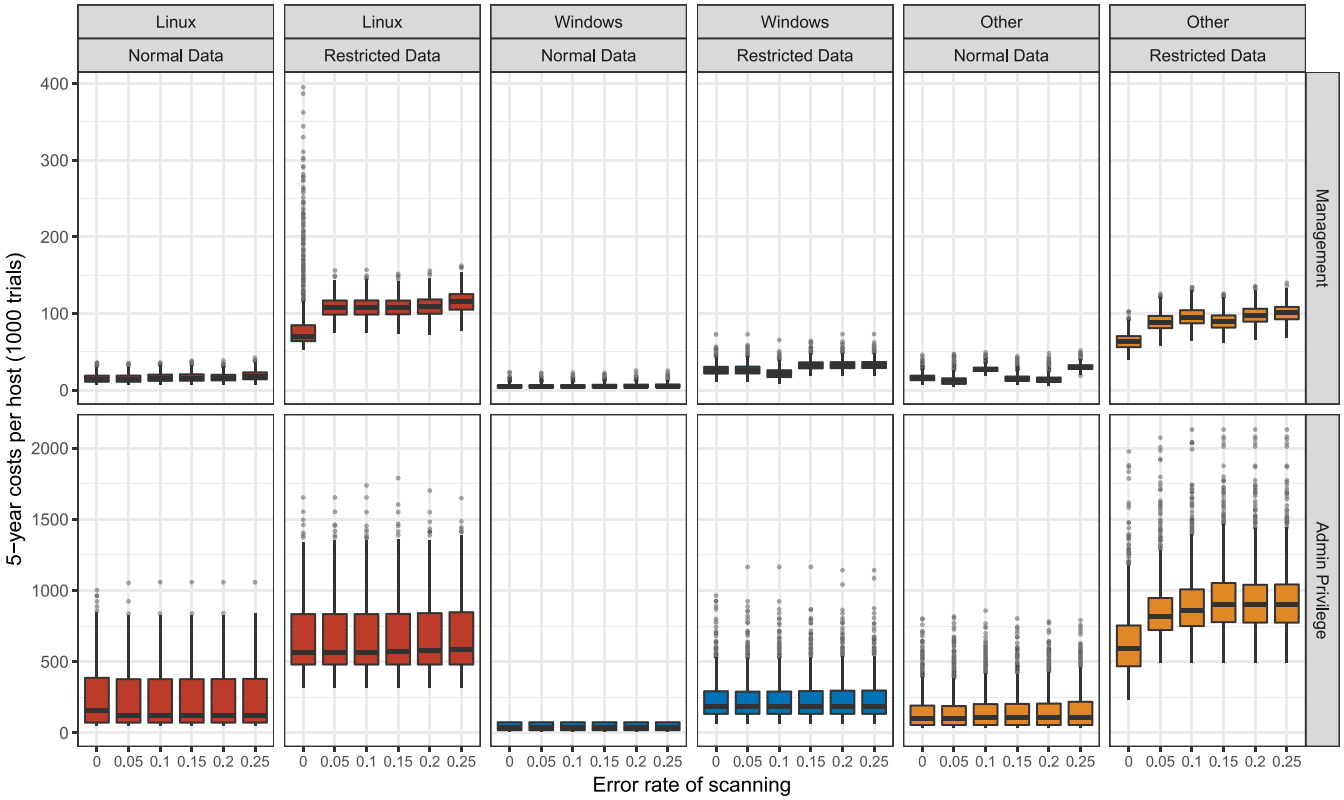

FIGURE 2 Simulation of five-year costs per host for 12 types of hosts under optimal policy across different scanning errors [Colour figure can be viewed at wileyonlinelibrary.com]

observation probabilities given the pair of new state and action taken. Then, a new belief is updated given the pair of observation and action according to Equation (5) and Equation (6). Last but not least, we update discount factor γ multiplied by 0.99, and update the number of horizons by adding 1.

- *Step 3*: *Stop criteria*. If the number of horizons is less than or equal to 60, continue to *Step 1* and *Step 2*. Otherwise, the trial of simulation stops and returns the total costs and a sequence of state evolution.

We then simulate four different policies for comparisons. Those are OSOM policy, fixed policy, epsilon-greedy policy, and random policy. The key difference among these policies is in taking of actions. Hence, the *Action mapping* step in the simulation procedure could be modified based on different policy.

- OSOM policy: Remember that OSOM policy stands for "out-of-sight is out-of-mind", which is the common policy for cyber vulnerabilities maintenance. Specifically, it ignores hosts when there are no observations ($o = None$). This "ignore" action implies that we do not have knowledge about what the host's state is, and then the state transition probabilities of this action would be assumed to be a uniform distribution. Moreover, when low or medium observations are captured, the OSOM policy would take autopatching action ($a = 1$) for hosts; when high or critical observations are captured, this policy would take research-accept action ($a = 2$) for hosts. In short, the OSOM policy is observation-based *Action mapping* instead of alpha-vector-based *Action mapping* for optimal policy.
- Fixed policy: It is a simple version of maintenance policy. It does not follow the alpha vectors. Instead, it would take research-accept action ($a = 2$) for hosts when the index of the maximum value of belief corresponds to high or critical state; otherwise, it would take auto-patching action ($a = 1$). In short, the fixed policy is a simple belief-based *Action mapping*,
- Epsilon-greedy policy: It is a greedy version of the optimal policy. It takes an optimal action a based on alpha vectors with a high probability $1 - \epsilon$ and a random action $a' \in A \setminus \{a\}$ with a low probability ϵ. Here, we set the ϵ as 5%.
- Random policy: It refers to randomly taking action $a \in A$ for hosts.

We first present in Figure 2 the simulation of costs for 12 type of hosts under optimal policy across six scenarios of scanning errors. The simulation results show that the individual host costs have been slightly altered by increasing the

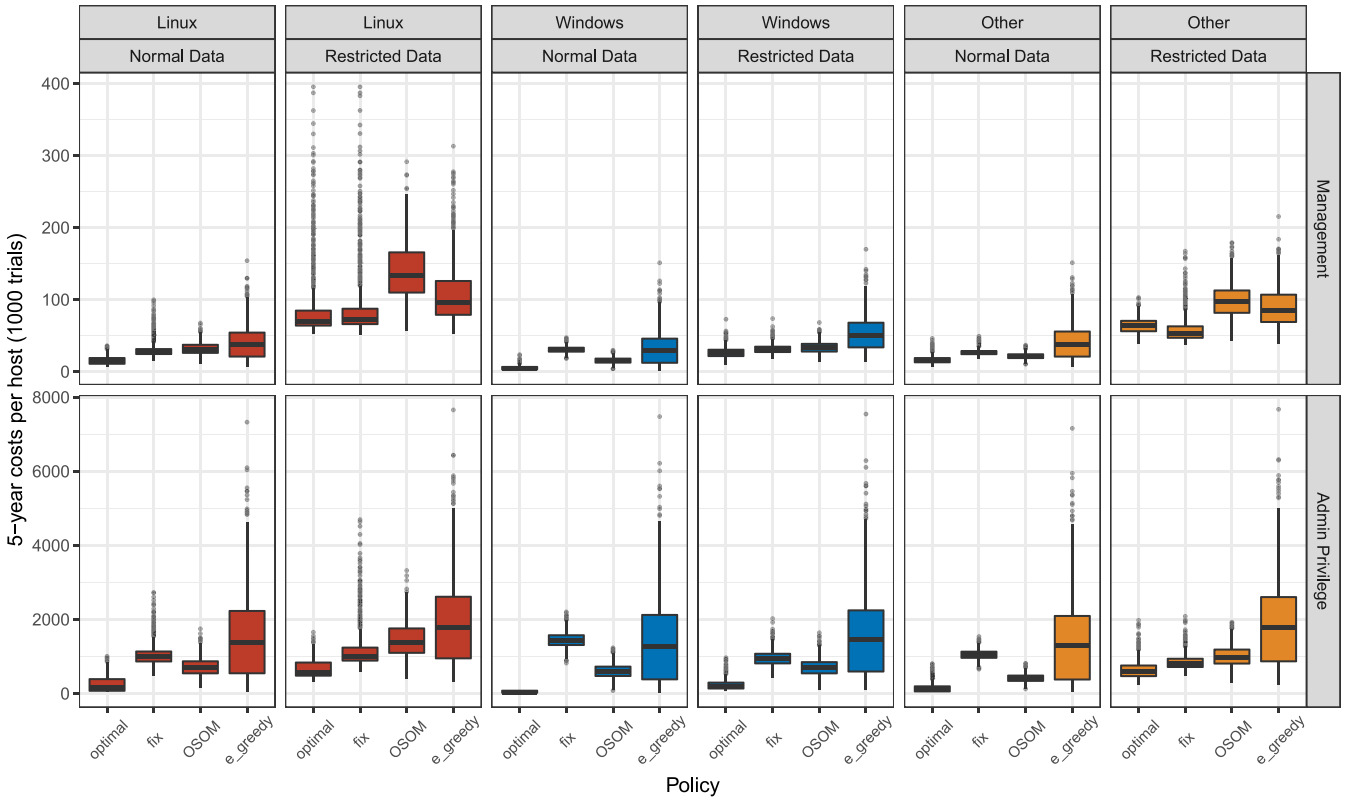

FIGURE 3 Simulation of five-year costs per host for 12 types of hosts under four different policies [Colour figure can be viewed at wileyonlinelibrary.com]

scanning errors for some types of hosts such as Linux type hosts with restricted data and management. This is because the alpha vectors of hosts of this type are slightly different for diverse scanning errors. Table 10 (b) and (d) present the alpha vectors of Linux hosts with restricted data and management for scanning errors 0% and 5%. The actions associated with the alpha vectors for scanning error 5% are seen to be superior to those for scanning error 0%, thereby resulting in a slight increase on the costs. This also implies that an advanced action might be needed for some types of hosts when a host's state is inaccurately known because of scanning errors. Nevertheless, for many types of hosts, the individual costs over 5 years are hardly affected by increasing the scanning errors from 0% to 25%.

Next, we present the simulation of costs for 12 types of hosts under different policies (ie, optimal policy, fixed policy, OSOM policy and epsilon-greedy policy) when the scanning error is 0% in Figure 3. Note that we remove the random policy from the results because its outstandingly large costs (its average costs exceed $400 for all hosts with management and exceed $20 000 for all hosts with administrative privilege) are out of scale with others. Figure 3 shows that, in general, optimal policy is the most cost-effective policy among others for all types of hosts whereas the epsilon-greedy policy is the worst strategy in terms of costs. Compared with the fixed policy, intuitively, optimal policy significantly reduces the costs for hosts with administrative privilege while the reduction is insignificant for hosts with management. The main reason is that the mode of management would considerably reduce the cost compared with the mode of administrative privilege according to Equation (17).

Similar observations apply in the comparison between optimal policy and OSOM policy except for Linux type hosts with restricted data (the second "column" in Figure 3). The cost reduction of this exceptional type of Linux host, however, is still not that significant for the mode of management compared with administrative privilege, which is shown in Figure 4. Here, we use the same scale in Figure 4 and see that the magnitude of cost reduction for the "row" of management is dwarfed by the "row" of administrative privilege. Therefore, one can apply the fixed policy or OSOM policy for those hosts which are managed in terms of the maintenance costs. For instance, if a university faculty or a student is using a lab computer in their department which is managed for patching vulnerabilities or upgrading software, the current maintenance policy (OSOM) or a simple policy (fix) is sufficient to protect it. Nonetheless, some computers or servers are not using management technique, and users then have administrative privilege to install software on them. In this case, the OSOM or fixed policy is not able to maintain the devices with minimum costs, and the optimal policy is needed to

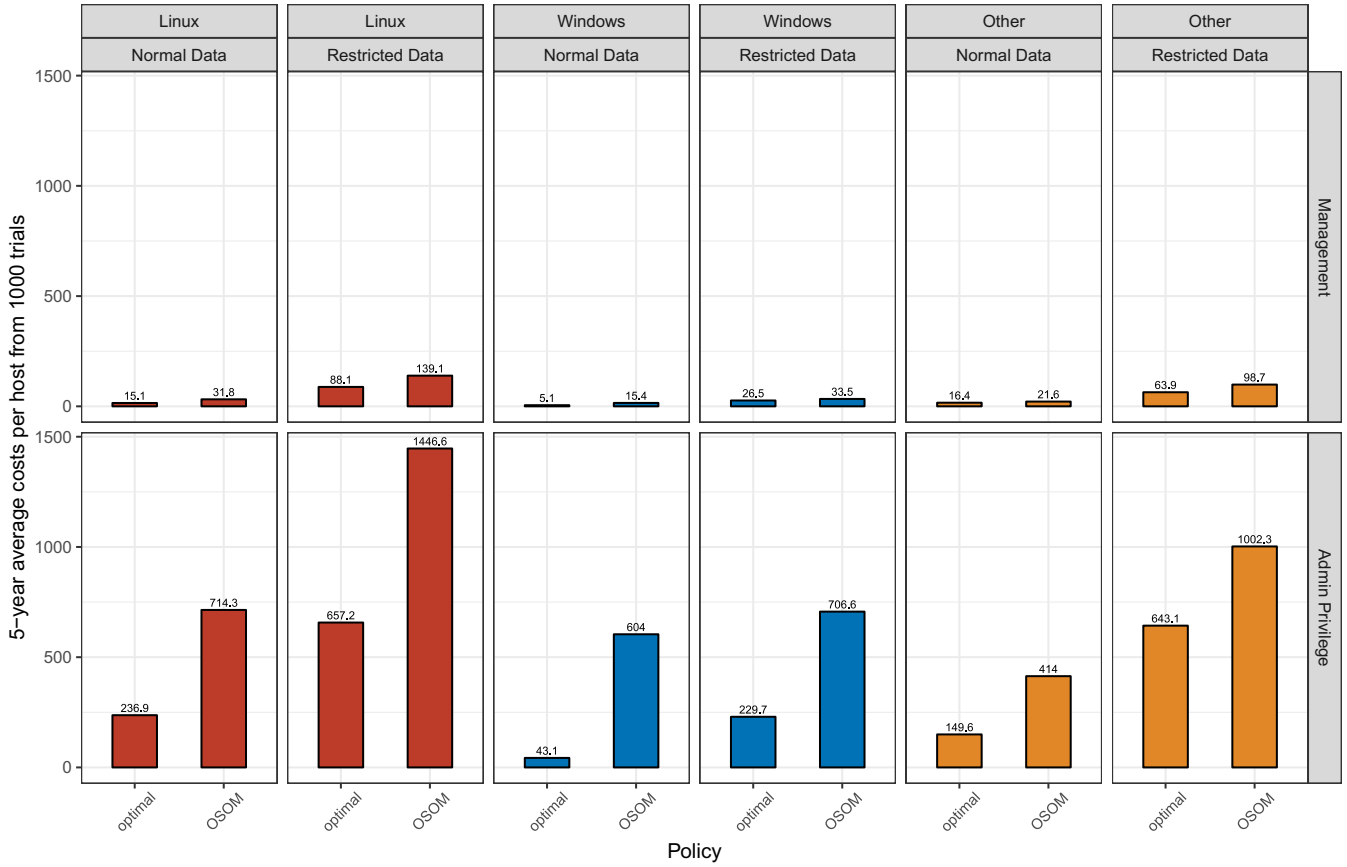

FIGURE 4 Optimal policy versus "out-of-sight is out-of-mind" policy: simulation of five-year average costs per host for 12 types of hosts [Colour figure can be viewed at wileyonlinelibrary.com]

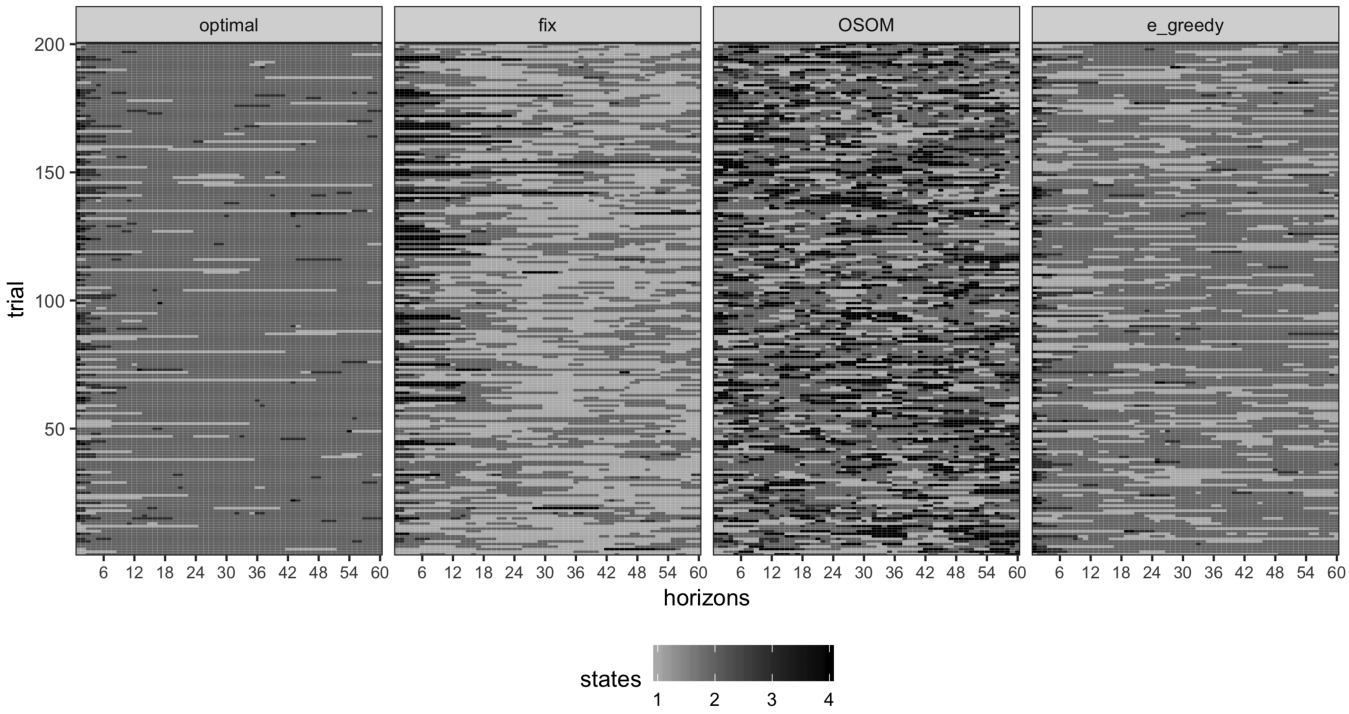

FIGURE 5 State evolution of first 200 simulation trials for Linux type of hosts with restricted data and administrative privilege under different policies

replace the current policy for hosts with administrative privilege. Consequently, the optimal policy derived from POMDP would be the most cost-effective maintenance strategy for all types of hosts, whereas the OSOM policy or fixed policy is still relatively cost-effective for most hosts if management techniques are applied to them.

Additionally, Figure 4 shows that the five-year average individual costs for a Windows type of host under optimal policy is much lower than that of a Linux type of host if both have the same characteristics on Restricted and Admin. This is because a Windows system is more defendable than a Linux system as described in Section 3.2.

We further evaluate the benefits of taking optimal policy in terms of hardening security states by visualizing the evolution process of states of five years (60 monthly horizons) under different policies. Figure 5 shows the state evolution of the first 200 simulation trials for Linux type of hosts with restricted data and administrative privilege under optimal policy, fixed policy, OSOM policy and epsilon-greedy policy when the scanning error is 0%. Each row represents the state evolution over 60 horizons in a simulation trial. Here, we recode the four states $\{Low, Medium, High, Critical\}$ as $\{1, 2, 3, 4\}$ and the associated color range is from bright to dark. From Figure 5, the panel of OSOM policy is much darker than others, which means that the OSOM policy is not able to control this type of hosts' states in a lower severity level. Compared with the OSOM policy, the fixed policy is capable of maintaining hosts when the starting state is not critical, whereas the fixed policy is not able to mitigate critical state within six months, which results in long dark streams in the second panel. Compared with the fixed policy, although optimal policy controls the most of states in the medium level, it could quickly mitigate the critical states within two or three months. Remember that the optimal actions associated with alpha vectors for this type of hosts contain action 1, 2 and 3 (see Table 10 (a)). With respect to epsilon-greedy policy, its performance is similar to the optimal policy but it is slightly brighter than the optimal policy because it has a 5% chance to take other actions such as remediation action ($a = 4$) which have high probabilities of mitigating severe states but costly expenditure. In summary, optimal policy could quickly remove critical states and maintain the hosts states in low (rarely) or medium (mostly) levels with minimal costs for the Linux type of hosts with restricted data and administrative privilege.

As we discussed before, OSOM policy or fixed policy might be allowed for those hosts that are managed because the cost reduction for such type of hosts is not significant compared with optimal policy. Here, we also present an example of the state evolution for Linux type hosts with restricted data and management to assess whether or not OSOM or fixed policy is sufficient to maintain hosts states. Figure 6 shows the state evolution of the first 200 simulation trials for such type of hosts. OSOM policy is still the darkest one, which is not able to maintain hosts states. The optimal policy in Figure 6 does not have the advantage of quickly mitigating critical states within two or three months, but it is still capable of mitigating critical states within one year and maintaining the hosts states in low (mostly) or medium level (rarely). In addition, the

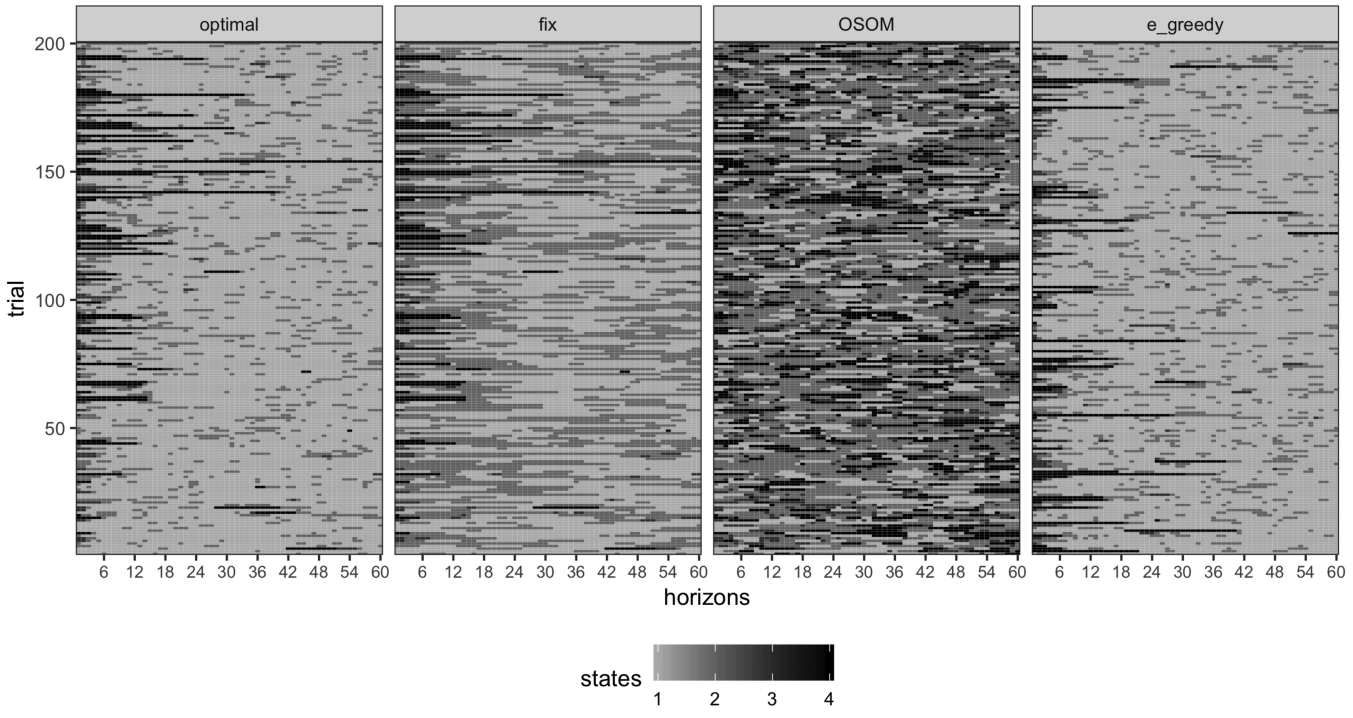

FIGURE 6 States evolution of first 200 simulation trials for Linux type of hosts with restricted data and management under different policies

optimal policy is slightly brighter than the fixed policy which is also able to mitigate severe states and to keep the states in low (less) or medium (mostly). Therefore, we restate that only fixed policy might be allowable for hosts which are managed for maintenance considering the insignificant cost reduction compared with optimal policy.

4 | CONCLUSIONS AND FUTURE WORK

In this article, we construct new parallel stochastic models to derive cost-effective cyber maintenance actions and belief-dependent dynamic optimal policies for different characteristics of hosts. A common definition of cybersecurity state is measured by the CVSS. The potential intrusions (generally called incidents) associated with security states are considered by predicting the probability of incidents using multi variable logistic regression, thereby resulting in a new cost function estimation for POMDP models based on hosts' features.

From our case study, the practical conclusion highlights are as follows (see Figure 3):

1. Hosts with restricted data should have their images managed centrally such that users should not have administrative privilege because the cost of maintaining hosts with administrator privileges is generally 10× higher than for managed hosts.
2. The OSOM policy with restricted data on the host is likely unacceptable but a simple probability-based policy (ie, the fixed policy) is likely acceptable avoiding unnecessary scans and the complication of the optimal POMDP policy.
3. For hosts with administrative privilege, the optimal policy is needed to keep costs low given that alternatives can generate hundreds of dollars more in costs over five years.
4. Hosts with Linux operating systems and restricted data tend to cost the most to maintain, particularly when they are not maintained optimally or, at least, with a policy based on their belief state value (ie, the fixed policy).

In our simulations, we consider six conditions of scanning errors to conduct sensitivity analysis for the assumptions of estimating observation probabilities. We validate that the expected individual costs of the majority of hosts are hardly influenced by increasing the scanning errors from 0% to 25%. Moreover, we compare the optimal policy with four alternative policies (OSOM, fix, epsilon-greedy, and random) using simulations to evaluate the benefits of taking optimal policy. To evaluate the performance on hardening security states, we present a heat map of evolution process of states of 60 months to visualize differences among alternative policies. The findings include that optimal policy is capable of alleviating critical vulnerabilities within a few months and generally maintaining hosts states in low or medium states, while OSOM policy is not able to maintain hosts states at lower levels because it would ignore those unobservable hosts whose underlying states could be high or critical.

Many topics remain for future research. Clearly, similar studies can be conducted at other universities and organizations. Studies can be repeated to see how findings change over time. Further, taking limited resources into account, the option of taking suboptimal action could be chosen when there is an action unavailable at the current time. Additionally, since the key limitations of the proposed model relate to the parameter estimation and associated assumptions, a MCBRL could be applied to address the parameter uncertainty and limited data issues. Moreover, social media data could be used to identify "celebrity" vulnerabilities and update the model building on work in Allen et al.[37] Resilience metrics can be used to incentivize rapid patching or remediation of various classes of vulnerabilities, eg, building on models of resilience in Woods et al.[38] Elicitation of data from experts can also be used in estimating the costs and priorities of vulnerabilities (eg, as in Allen and Maybin[39]). A variable fidelity modeling approach could be used to mix local high fidelity data (scan and incident data) with data from other sources, eg, using modeling approaches like in Huang and Allen[40] or Allen et al.[41]

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grants #1409214 and #1912166. The authors would also like to thank Helen Patton, Steven Romig, and Rajiv Ramnath for the general support for this and related research.

ORCID

Enhao Liu https://orcid.org/0000-0002-9281-900X
Theodore T. Allen https://orcid.org/0000-0002-9522-3252

REFERENCES

1. Brewster T. How Hackers Broke Equifax: Exploiting A Patchable Vulnerability. https://www.forbes.com/sites/thomasbrewster/2017/09/14/equifax-hack-the-result-of-patched-vulnerability/#1c1b89d55cda. Accessed September 13, 2018; 2017.
2. Ponemon Institute LLC. 2017 Cost of Data Breach Study: United States. https://www.ponemon.org/blog/2017-cost-of-data-breach-study-united-states. Accessed September 13, 2018. 2017.
3. Virtanen I. Optimal maintenance policy and planned sale date for a machine subject to deterioration and random failure. *Eur J Oper Res*. 1982;9(1):33-40.
4. Kuhn KD, Madanat SM. Robust maintenance policies for Markovian systems under model uncertainty. *Comput-Aided Civ Infrastructure Eng*. 2006;21(3):171-178.
5. Zhang Z, Wu S, Li B, Lee S. Optimal maintenance policy for multi-component systems under Markovian environment changes. *Expert Syst Appl*. 2013;40(18):7391-7399.
6. Afful-Dadzie A, Allen TT. Data-driven cyber-vulnerability maintenance policies. *J Qual Technol*. 2014;46(3):234-250.
7. Afful-Dadzie A, Allen TT. Control charting methods for autocorrelated cyber vulnerability data. *Quality Engineering*. 2016;28(3):313-328.
8. Allen TT, Liu E. Forecasting cyber maintenance costs with improved scan analytics using simulation. Paper presented at: 2018 Winter Simulation Conference (WSC)IEEE; 2018; Gothenburg, Sweden.
9. Allen TT, Roychowdhury S, Liu E. Reward-based Monte Carlo-Bayesian reinforcement learning for cyber preventive maintenance. *Comput Ind Eng*. 2018;126:578-594.
10. Sondik EJ. *The Optimal Control of Partially Observable Markov Processes* [PhD thesis]. Stanford, CA: Stanford University; 1971.
11. Smallwood RD, Sondik EJ. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*. 1973;21(5):1071-1088.
12. Sondik EJ. The optimal control of partially observable Markov processes over the infinite horizon: discounted costs. *Operations Research*. 1978;26(2):282-304.
13. Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*. 1998;101(1-2):99-134.
14. Byon E, Ding Y. Season-dependent condition-based maintenance for a wind turbine using a partially observed Markov decision process. *IEEE Trans Power Syst*. 2010;25(4):1823-1834.
15. Kurt M, Kharoufeh JP. Monotone optimal replacement policies for a Markovian deteriorating system in a controllable environment. *Oper Res Lett*. 2010;38(4):273-279.
16. Neves ML, Santiago LP, Maia CA. A condition-based maintenance policy and input parameters estimation for deteriorating systems under periodic inspection. *Comput Ind Eng*. 2011;61(3):503-511.
17. Makis V. Optimal condition-based maintenance policy for a partially observable system with two sampling intervals. *Int J Adv Manuf Technol*. 2015;78(5-8):795-805.
18. Wang S, Zhang Z, Kadobayashi Y. Exploring attack graph for cost-benefit security hardening: a probabilistic approach. *Comput Secur*. 2013;32:158-169.
19. Roychowdhury S. *Data-Driven Policies for Manufacturing Systems and Cyber Vulnerability Maintenance* [PhD thesis]. Columbus, OH: The Ohio State University; 2017.
20. Mell P, Scarfone K, Romanosky S. A complete guide to the common vulnerability scoring system version 2.0. In: Published by FIRST Forum of Incident Response and Security Teams. 1; 2007; Gaithersburg, MD.
21. Spaan MT. Partially observable Markov decision processes. In: *Reinforcement Learning*. Basel, Switzerland: Springer; 2012:387-414.
22. Cassandra A, Littman ML, Zhang NL. Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes. In: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence; 1997; Providence, RI.
23. Madani O, Hanks S, Condon A. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*. 2003;147(1-2):5-34.
24. Pineau J, Gordon G, Thrun S. Point-based value iteration: an anytime algorithm for POMDPs. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03); 2003; Acapulco, Mexico.
25. Cockburn E. Websites here, websites there, websites everywhere … , but are they secure? *Quaestor Q*. 2009;4(3):1-4.
26. Soska K, Christin N. Automatically detecting vulnerable websites before they turn malicious. Paper presented at: 23rd USENIX Security Symposium (USENIX Security 14); 2014; San Diego, CA.
27. Liu Y, Sarabi A, Zhang J, Naghizadeh P. Cloudy with a chance of breach: forecasting cyber security incidents. Paper presented at: 24th USENIX Security Symposium (USENIX Security 15); 2015; Washington, DC.
28. Sabottke C, Suciu O, Dumitras T. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. Paper presented at: 24th USENIX Security Symposium (USENIX Security 15); 2015; Washington, DC.
29. Hao S, Kantchelian A, Miller B, Paxson V, Feamster N. PREDATOR: proactive recognition and elimination of domain abuse at time-of-registration. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security; 2016; Vienna, Austria.
30. Zhang S, Ou X, Caragea D. Predicting cyber risks through national vulnerability database. *Inf Secur J Glob Perspect*. 2015;24(4-6):194-206.
31. Zhu X, Cao C, Zhang J. Vulnerability severity prediction and risk metric modeling for software. *Applied Intelligence*. 2017;47(3):828-836.

32. Liu E. *Logistic Regression Model for Predicting Warning "Incident" Rates and Implications for the Common Vulnerability Scoring System* [master's thesis]. Columbus, OH: The Ohio State University; 2017.
33. Tavabi N, Goyal P, Almukaynizi M, Shakarian P, Lerman K. Darkembed: exploit prediction with neural language models. Paper presented at: 32nd AAAI Conference on Artificial Intelligence; 2018; New Orleans, LA.
34. Hosmer DW, Lemeshow S, Sturdivant RX. *Applied Logistic Regression*. Vol 398. Hoboken, NJ: John Wiley & Sons; 2013.
35. Hosmer DW, Hosmer T, Le Cessie S, Lemeshow S. A comparison of goodness-of-fit tests for the logistic regression model. *Statist Med*. 1997;16(9):965-980.
36. Le Cessie S, Van Houwelingen J. A goodness-of-fit test for binary regression models, based on smoothing methods. *Biometrics*. 1991:1267-1282.
37. Allen TT, Sui Z, Parker NL. Timely decision analysis enabled by efficient social media modeling. *Decision Analysis*. 2017;14(4):250-260.
38. Woods DD, Schenk J, Allen TT. An initial comparison of selected models of system resilience. In: *Resilience Engineering Perspectives*. Vol 2. Boca Raton, FL: CRC Press; 2016:95-116.
39. Allen TT, Maybin KM. Using focus group data to set new product prices. *J Prod Brand Manag*. 2004;13(1):15-24.
40. Huang D, Allen TT. Design and analysis of variable fidelity experimentation applied to engine valve heat treatment process design. *J Royal Stat Soc Ser C (Appl Stat)*. 2005;54(2):443-463.
41. Allen TT, Xiong H, Afful-Dadzie A. A directed topic model applied to call center improvement. *Appl Stoch Models Bus Ind*. 2016;32(1):57-73.

How to cite this article: Liu E, Allen TT, Roychowdhury S. Cyber vulnerability maintenance policies that address the incomplete nature of inspection. *Appl Stochastic Models Bus Ind*. 2019;1–21. https://doi.org/10.1002/asmb.2487