Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

A SYSTEMS ENGINEERING FRAMEWORK FOR IMPLEMENTING A SECURITY AND CRITICAL PATCH MANAGEMENT PROCESS IN DIVERSE ENVIRONMENTS (ACADEMIC DEPARTMENTS WORKSTATIONS)

HADI MOHAMMADI
THE GEORGE WASHINGTON UNIVERSITY, USA
hadimoh@gwu.edu

THOMAS MAZZUCHI
THE GEORGE WASHINGTON UNIVERSITY, USA
mazzu@gwu.edu

SHAHRAM SARKANI
THE GEORGE WASHINGTON UNIVERSITY, USA
sarkani@gwu.edu

ABSTRACT

In any networked computing system, particularly in an academic environment, it is not uncommon to administer workstations that have been compromised because of security vulnerabilities and viruses in the operating system or in the software packages that are installed. Applying the Security and Critical Patch Management Process (SCPMP) described in this study at an early stage in a computing system can mitigate the risk of attack as well as decrease the cost of Information Technology (IT) operations. In this paper we propose an optimal systems engineering (SE) framework to improve the current patch management process framework, to prevent an organization's networked computer systems from becoming the target of vulnerabilities that are immediately exploitable and could lead to critical system failure or compromised security. The proposed SE framework comprises a new mechanism for IT Operational Activities that increases network security and critical systems integration. By exerting SE principles with considerable influence on patching, IT departments can manipulate the SCPMP framework to meet patch management requirements, estimate the stages of the SCPMP performance, and determine an affordable portfolio. This article details a patch management framework design and the preliminary implementation of the SCPMP in an academic environment. The focus is on measuring variations in the components of a full patching cycle, and analyzing the data collected from implementing the process in an academic department's workstations as a case study, using Maintenance Optimizing Tools.

Keywords: Patch management, security, systems engineering, vulnerability, maintenance optimization

INTRODUCTION

Contemporary companies, institutions, and even governments maintain their computer operating systems or software packages by applying a Patch Management Process (PMP) to prevent their workstations from being compromised or attacked. Patch management, a main ingredient of IT security management [1] has become one of the prime requisites in every information technology infrastructure to (a) defend networked computer systems from attack, and (b) maintain a secure network. Therefore, it is vital to assure that the latest security and critical patches (SCP) are installed across an entire networked computer system. Patching is done in diverse environments not only to fix security and critical related vulnerabilities, but also to avoid leaking and other disclosure of critical data and security information, known as social loss.[1]

This paper presents an optimal research systems engineering (SE) framework that can improve the current patch management process framework. Our framework has the effect of preventing any organization's networked computer systems from becoming the target of vulnerabilities that are immediately exploitable and could lead to critical system failure or compromised security. According to [2], the process known as SE is to define a system structure based on customer requirements, and to satisfy the customer's need that the system components function correctly and the system overall operates as required. Herein we create a framework for implementing a Security and Critical Patch Management Process (SCPMP), based on SE principles, to identify and allocate patching system requirements and computing systems resources that are embedded. In addition, the SCPMP monitors entire computing systems' design and increases network security and critical systems integration. Current patch management process frameworks are primarily designed using automatic and routine patch and vulnerability management processes [3-9, 18], for maintaining computer operating systems or protecting software packages. The proposed framework accommodates maintenance optimization tools to help IT decision makers decide either to continue deploying patches that are already in process, or to block maintenance and reinitiate their patch management process at the time new critical and security patches (SCP) are released. This framework helps IT decision makers both to cover and include the most recently released patches in their deployment cycle, and to carry out the process regardless of what patches have been released.

The remainder of our paper proceeds as follows: the next section interprets the SCPMP overall structure. Then we propose a framework for implementing the SCPMP in diverse environments and explain how each stage operates. Also in proposal section, we describe how a maintenance optimization tool is accommodated in the proposed framework. Discussion of the case study appears in the case study section, wherein we implement the SCPMP in several academic department workstations by using a Markov Decision Process tool. We conclude our paper and suggest future study in the last section.

SECURITY AND CRITICAL PATCH MANAGEMENT PROCESS OVERALL STRUCTURE

The SCPMP not only focuses on how to deploy operating systems or software package patches on a workstation; it also helps IT decision makers or security operation technicians to use it as a completed patch management cycle to prevent their IT assets from being attacked and to reduce the cost of operations. As it is pointed out in [10], security patch deployment is crucial for managing any IT operation, as it helps to update vulnerable software or operating systems to prevent them from being attacked. At the same time, Dantu, et al., write that not only should IT operation staff deploy patches to end-users' networked computing systems to mitigate the risk of compromise, but they also need to deploy patches across entire networked computer systems [12]. Thus, IT security and patch management are critical in networked computing systems and they are a key methodology for preventing them from becoming compromised. An effective and reliable patch management process is a robust, cyclic protocol for protecting networked computer systems, identifying vulnerabilities, and assessing risk in a given IT department [6] to make computing systems more secure and reliable. A patch management system optimizes security patch deployment for installation and management in multiple software packages and different platforms [11]. As Figure 1 shows, the proposed framework has some components that are initial requirements for the SCPMP; these are a management server center, patch storage, and a client agent (production and beta test workstations).

[1] Social loss represents as the negative impact of security vulnerabilities disclosure on the users, which can lead to critical costs and compromise their computing system [17].

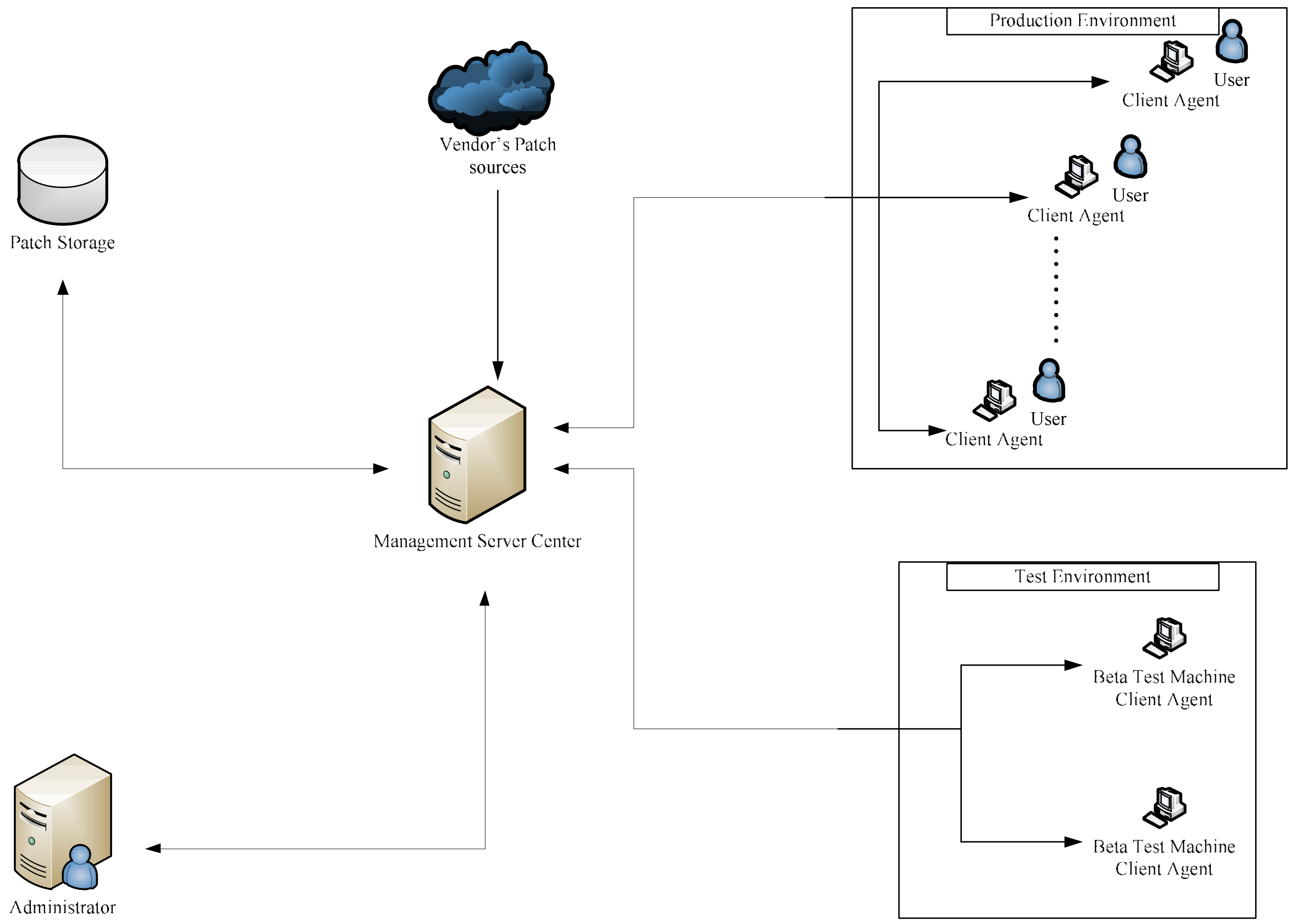

Figure 1: SCPMP overall structure

Management server center: The management server center is the core of the SCPMP, and performs multiple tasks. First, it downloads patches from vendors' patch sources and stores them in patch storage. Second, it provides web-based user interface for IT decision makers or security operation technicians. Third, it communicates with the client agent installed on each networked computer system to detect, monitor, deploy, and distribute patch files and install patches by the client's agent based on each system's configuration. In addition, the management server center can generate a report of the status of the patch installation process. Patch management server center products are offered by such vendors as DELL KACE appliance (KBOX1000, *kace.com*), Lumension (*lumension.com*), and other companies that have the same product that identifies security and critical patches across diverse networked environments and deploys their remediation to be installed through the client agent.[2] Overall, the management server center performs checks for patch definition updates at a specified start time, and downloads all applicable packages based on patch subscription settings. In addition, it limits network activity to off hours, and specifies a stop time for downloading.

Patch storage: Patch storage stores patches that the patch management server center downloads from

[2] The number of companies that offer an automatic patch management server is increasing significantly because of important security disclosures and the need to protect computing systems from attack. The companies named are well known in the IT industry.

vendors. It can contain two kinds of patches: a) patches that include software installers; i.e., the patch file will install a new version of the software; and b) patches that fix security vulnerabilities and breaches or other bugs.

Client agent: The client agent is the software-based piece of this patch management system that needs to be installed and deployed on all managed computer systems. The client agent can run full hardware and software inventory reports, scan system information, and detect installed patches. Also, it performs interval scanning of the client's overall workstation system components and communicates with the management server center. The client agent should be installed on production and beta test workstations (test machines built with the same hardware and installed with the same software as the production machines).[3]

SCPMP FRAMEWORK AND DEFINITION

The main goals of the proposed SE practicable framework for implementing the SCPMP are:

1. To provide an effective optimal patch management policy for protecting networked computer systems, in particular academic workstations.
2. To repair software package vulnerabilities.
3. To fix security holes in a computer's operating systems.
4. To enhance IT operation department efficiency.
5. To reduce the cost of IT operations activities.
6. To prevent the loss of critical data and security information from any computing system in diverse environments.

The projected SE framework for implementing the SCPMP; the main stages are defined as follows:

Download: downloading patches by the management server center via a third party and storing them in patch storage, which was discussed in the previous section.

Detect: comprehensive detection across entire networked computer systems (classification and earlier prioritization) to show which patches are already installed, and which patches need to be installed in a new patch management deployment cycle. In other words, by detection, the IT operation staff will figure out which computing system needs to be patched. The detection stage can be seen as initial planning for the deployment stage.

Assess: assessing the downloaded patches and measuring the variations in components of the full patching cycle; also ensuring the downloaded patches will not damage the systems by crashing systems operation and performance. Efficient security patch deployment identifies and remediates security vulnerabilities by deploying and installing patches without negatively impacting the software applications installed on advanced computing systems or crucial servers [6]. Consequently, the impact of precise patch assessment is to help IT managers enhance the efficiency of the patch installation procedure to reduce the system's vulnerabilities.

Test: Testing is one of the essential elements of patch management. Testing the released patches on the same system configuration as the production environment helps IT administrators simulate the installation process and learn the consequences of patch updates and identify any conflicts. For this phase, there should be a few non-production workstations that are designated as beta test environment systems. As noted in [6], patch installation can have unintended results; therefore, our beta test environment systems contain some networked computer systems based on the production workstation's configuration and setting so as to avoid extraneous alternatives.

Incident: An incident might go as follows: After implementing the test phase and ensuring that the downloaded patches do not cause any further interruption of operations, suddenly we notice some highly urgent security and critical patches (USCP) have been released by vendors or a third party such as vulnerability identifiers.[4] IT managers then need to make a decision about this circumstance to ensure that this emergency patch update for software or an operating system platform will not cause any problem and does not increase the system's risk of compromise. The next step is the Maintenance Optimization stage, in which the manager will decide on an efficient plan for continuing the patching process.

Maintenance Optimization (MO): In the decision grid stage or Maintenance Optimization phase the IT manager must decide how to contrive to continue the SCP. We will thoroughly discuss the MO stage in the next segment.

Deploy: the last stage is to deploy and install patches on production workstations. Figure 2 shows the projected SE framework for implementing the SCPMP.

[3] IT operation staffs also use client agent software to install various software, inventory their hardware, script, and report assets.

[4] Vulnerability identifiers or coordinators such as US-CERT (United States Computer Emergency Readiness Team), which develop security solutions (us-cert.gov).

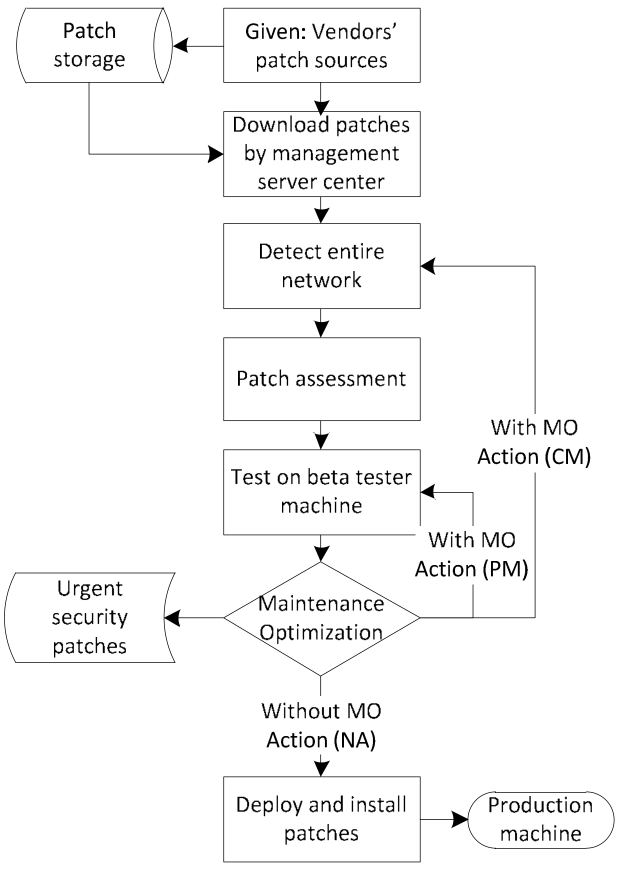

Figure 2: SCPMP framework

Maintenance Optimization Stage

MO increases a decision-maker's power to optimize operation cost and ensure reliable and safe system performance [13]. Hence, involving MO tools will make networked computer systems more secure and protect them from becoming the target of vulnerabilities. The MO stage also keeps the patch management deployment process low-cost, as it defines a more efficient IT operation activity. In this section, we study how IT managers make a decision once some urgent security and critical patches (USCP) are released while they are in the midst of implementing the patching process. Monitoring the receipt and download of security and critical patches (SCP) is a continuous process; therefore, the expectation of receiving USCP is one of the most important priorities for IT department technicians who are responsible for deploying patches throughout networked computer systems. In our proposed SE framework, the likelihood of receiving USCP is considered as a probability.

The patch update process is complex and costly [1]; therefore, it is very important for IT department managers to make an ideal decision not only to (a) cut the costs of the IT department, but also to (b) prevent networked computer systems from receiving any vulnerability attack, where vulnerability is defined as a security or other critical hold or weakness of a software application or a compromised operating system that allows hackers access to the system's root to exploit the flaw [18].

In this paper, the decision-making phase of performing the SCPMP is modeled based on a Markov Decision Process (MDP). Our deterioration model is a finite and countable state space, once the USCP has been released, where the patch management process cycle has already passed the test phase. Accordingly, IT decision makers need to be prepared to deploy and install these newly released patches on the computer systems they manage. Their decision has significant impact on IT operations.

Patch management is a Preventive Maintenance (PM) process; thus, newly released patches need to deploy and install before any damages occur. By applying patch management, the IT department is able to protect its computers from leaking data and becoming the target of security attacks that cause the company or institution to incur major costs, and to avoid maintenance-induced compromise. The MO stage in the proposed SCPMP framework will help IT operation managers decide to continue maintenance with the patches that have already been tested, or block the patching cycle (maintenance) and move back to the detection phase to include the new USCP in the patch packages. Thereupon, they will detect their entire networked computer systems with the new set of patches and proceed with the patching loop. The main contribution of this subsection is to provide 1) the most efficient patch management policy which decreases the cost of IT operational, and 2) the best decision about urgent security and critical circumstances that the IT decision maker is involved with.

Modeling the Markov Decision Process

The proposed framework assumption is based on a seven-day operation cycle, which means the patching cycle process takes a 7-days cycle to implement one loop of the SCPMP as its time epochs. Mazzuchi, et al. [13], Amari, et al. [16], and Okamura, et al. [18] discuss a Markov decision process that is close to the MO process in our framework. The ultimate goal is to find an approach, which in MDP is called a policy, by which the IT decision maker can decide which patch distribution process or maintenance action should be taken in each patching state so as to enhance IT security and decrease the cost of operation. The assumptions of the maintenance optimization phase of the proposed framework follow.

Inputs:

1. We use a Markov Chain[5] to model our multi-stage deterioration.[6] In this model the patching deployment process contains a number of stages and the deterioration proceed in one direction. Here, the patch type condition is classified in four stages, $S=\{S_n,S_v,S_r,S_u\}$, which are determined as the type of patch that can be released; this state space assumption is finite and discrete. The first stage is the perfect condition stage wherein no patch is released and the last stage is the condition wherein a highly urgent patch is released:
   - S_n: no patch is released.
   - S_v: optional version update for the operating system or software installer patch is released
   - S_r: recommended patch is released
   - S_u: urgent security and critical patch is released

Figure 3 illustrates all states of patch types that may be released in the midst of a patching cycle, so that a decision has to be made at the subsequent MO phase. We select a stationary policy because the maintenance decision to be made is based upon the recent type of patches that just released and not on maintenance action chosen time [13]. The criterion for defining the proposed policy is the extent to which it helps IT managers choose an appropriate rule for taking action. States of patch types are:

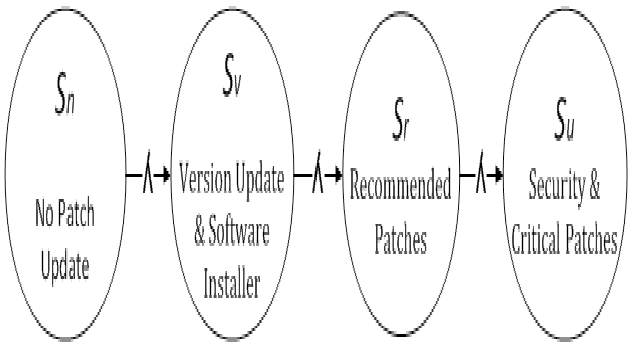
Sn
No Patch Update
λ
Sv
Version Update & Software Installer
λ
Sr
Recommended Patches
λ
Su
Security & Critical Patches

Figure 3: States of patch type

2. We assume that the action space is finite and discrete, and so based upon the patch deterioration type one of the $A=\{a= NA,MM,PM,CM$ four maintenance actions might be chosen (Figure 4); further, the size of the maintenance action is if $s\in S,\ size(Aa)=1$:

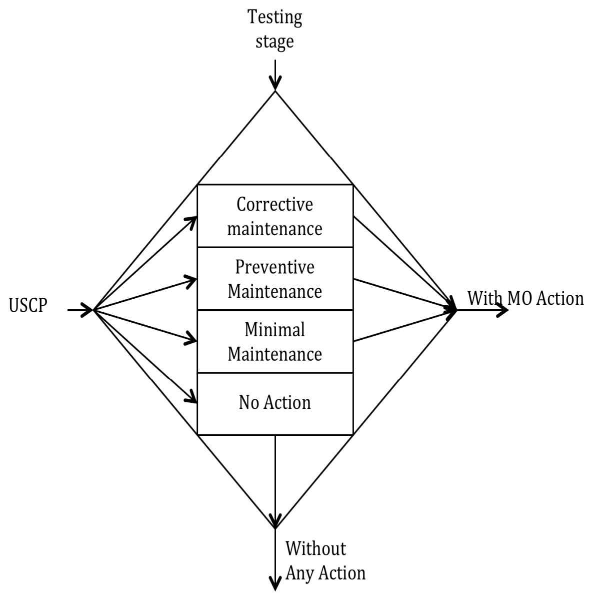

Figure 4: Maintenance decision action

- No Action (NA) is performed; focus on current task regardless of the patches released; no urgent patch has been released.
- Minimal Maintenance (MM) is performed; add USCP to the in-progress patches'

[5] In a deterioration model based on a Markov Chain (MC), all deterioration stages are in one-direction where the beginning stage is the no action stage (good stage) and the last one will be fail stage [16]. The proposed deterioration model follows MC to meet all kinds of patch requirements.

[6] When any operating system or software patch is released, the type of patch will determine the level of deterioration of the patch management process.

package-cycle without performing the detecting and testing procedure on the networked computer systems.
- Preventive Maintenance (PM) is performed; go back to the previous phase, which is to test USCP solely on beta test machines.
- Corrective Maintenance (CM) is performed; go back to the detection phase, include the USCP in the current package of patches, and proceed through the cycle.

3. The total cost of one cycle of implementing the SCPMP is related to the cost of the MO process that is assumed to be bounded. This means that if the cycle proceeds as a regular patching process without any MO decision action, it can change once some patches are released in the middle of our patching cycle. In this paper, we consider the proposed SCPMP while we take any maintenance optimization action. Therefore, the cost formation can be denoted by the following:
   - *cK*: fixed cost of the security and critical patch process for the Management Server Center and departmental cost
   - *cA(a)*: cost of maintenance (patching) action
   - *cF(s)*: cost of damage to a networked computer system per unit time when it becomes the target of vulnerabilities that are immediately exploitable and could lead to critical system failure
   - *cS(s):* Unit of time in state s Cost

Once the SCPMP framework cycle starts to go through each stage to deploy patches and monitor networked computer system performance, the next challenge faced in practice is to determine what policy needs to be applied to increase the efficiency of the patch deployment process. Figure 5 presents a hypothetical deterioration process, *P(t)*, which is a stochastic process that would be tracked over time from each decision stage of the MO phase in the SCPMP framework. As Figure 5, shown *P(t)* starts at approximately zero and tends toward the failure threshold, which is the CM stage. However, if the *P(t)* passes the failure threshold (CM stage), the deterioration process is considered to highly damage the security of the networked computer systems.

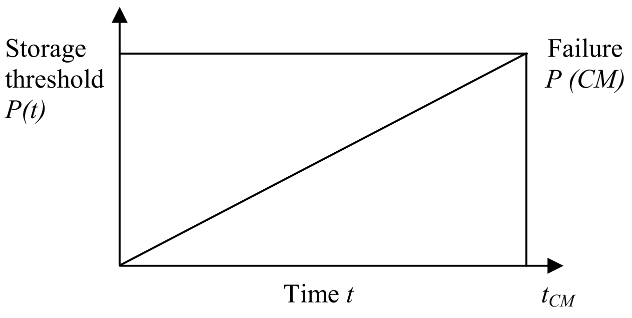

Figure 5: Deterioration process

4. The probability of the maintenance decision action $(a \in A)$ will be chosen in state s and the process transferred to state s' is denoted by $P(s'|s,a)$.

Functional Equation and Optimizing: In this subsection, a probabilistic model for the SCPMP in the MO phase is derived based on the MDP [13, 16, 18, 21-23]. The main goal is to determine the stationary policy that minimizes IT operation costs. We develop and present the iterative policy [21] to determine an optimal maintenance policy, so as to increase the patch management process's efficiency and mitigate the risk of security vulnerabilities. The proposed optimal policy helps IT departments avoid incurring excessive maintenance costs for compromised production machines.

We assume that states, time, and actions are discrete. Then, all possible maintenance actions are as follows:

λ: Transition from state s to state s'

A: Sets of possible and finite maintenance actions in state S if:

$$a \in A, size(A_a) \quad (1)$$

When the patches are released in state s at time t=0,1,2… and maintenance action $a \in A$ is taken, the patching deployment operation can be in stage s' after one unit of time with probability:

$$P(s'|s,a) = \Pr\left[X_{t+1} = s' | X_t = s\ , a_t = a\right] \quad (2)$$

$$T(s'|s,a) = E[T_{t+1}|X_t = a, a_t = a, X_{t+1} = s'] \quad (3)$$

Equation (3) comprises the time spent in state s′ when the initial state is s and action a has been taken. For computing the expected cost $C(s,a)$ while we are in state s and make decision a, we have:

$$C(s,a) = cK + cM(s,a) + \sum_{s,s' \in S} P(s'|s,a) \,.\, cF(s') + \sum_{s,s' \in S} P(s'|s,a) \,.\, T(s'|s,a) . cS(s') \qquad (4)$$

In MDP the probability of transition from one state to another does not depend upon the patch state history if one of the maintenance actions is selected; therefore, the patch deployment process is rely on the type of patches which have been urgently released and doesn't depend on the time of the maintenance action is taken [13]. Based upon [13, 22], *V(s)* is a value function that represents the expected objective value obtained from the discounted cost incurred. For discounted object function:

$$V(s) = C(s,a) + \sum_{s,s' \in S} P(s'|s,a) \,.\, \alpha . V(s') \qquad (5)$$

Where $\alpha \in (0,1)$ represents the discount factor for one unit of time, $\alpha = 1/1 + \lambda$ which attempts to minimize the expected discounted cost for one unit of time, ↙$_{s,s'}$ is the rate of transition time from state s to state s', and *V(s)* is the value function using ⇨. While we start the maintenance process, we choose to perform maintenance action in state *s* and derive the cost of $C(s,a)$. Then, the total probability of transition to state $s'$where a maintenance action $A = \{a = NA, MM, PM, CM\}$ is employed and initial state is *s* will multiply to discounted factor ⇨ and value function of state s'. The value of a state is the minimum expected cost that will be incurred in that state, plus the expected discounted value of all possible states s'. A fixed policy of the *V,C,P*, and *T* functions, can be rewritten in matrix-vector form as equation (5). Based upon the Bellman equation [22] the optimal policy can be derived:

$$\pi(s) = \arg min_a\{C(s,a) + \sum_{s,s' \in S} P(s'|s,a) \,.\, \alpha . V(s')\}$$
(6)

We can set an optimal policy $\pi(s)$ for minimizing $C(s,a)$ and re-evaluate V and C, and repeat for each state. This is called policy iteration, and it guarantees that the selected policy minimizes the patching operation cost [16]. In the other words, a policy $\pi(s) = a$ is selected such that the decisions based upon optimal policy of the MO stage are taken according to chose the maintenance decision which selected from limited actions *A(a)* and the costs of patching process *C=(s,a)* that are incurred once the transition is in state *s* and action $a \in A$ is taken. As Mazzuchi, et al., point out, the cost of deploying patches at each stage, plus the probability that each state multiplies to the value of performing action in the next state, $Va(s')$, changes Equation (5) to become recursive. Equation (5) represents the overall value of state *s*, which determines which action is opted to be taken [13]. After counting the cost of all possible states *s* from Equation (6), we derive that the optimal-cost decision of the MO phase would be to minimize the result of Equation (6), and the IT manager needs to choose the policy based on each action taken in each state of a patch type.

CASE STUDY

The proposed method is demonstrated through a case study that implements the SCPMP through an academic networked computers system. The process is assumed to go on indefinitely, and the problem is to opt for an optimal policy, which minimizes the average cost security and critical patching process.

Inputs: as we mentioned before, the patching process takes a 7-day cycle. Then the mean time between two cycles of patching is 168 (7/24) hours, λ*=0.168.* Further,

1. Cost of the IT operation, cK = $1442
2. Cost of being valuable and caused system failure: cF(s)= $891
3. No patching Action Cost: cM(NA)=0
4. Minimal Maintenance Cost: cM(MM)= $26
5. Preventive Maintenance Cost: cM(PM)= $20
6. Corrective Maintenance Cost: cM(CM)= $618

Table 1 shows that the total state transition probabilities $\sum_{s,s' \in S} P(s'|s,a)$ and expected time functions matrix based on this case study are:

Table 1: Transition probabilities and time function

$\sum_{s,s' \in S} \mathbf{P(s'\|s,a)}$	1	0	0	0
	0	1	1	0
	0	0	2/3	1
	0	0	0	1
$\mathbf{T(s'\|s,a)}$	24	0	0	0
	0	24	0	0
	0	24	24	48
	0	0	0	72

Problem formulation and optimal policy solution: we assume that the action space is finite and discrete, and so, based on patch deterioration type, one of the following four maintenance decision actions $A = \{a =$ *NA,MM,PM,CM* can be taken [16]. Further, the size of maintenance actions is, if:

$s \in S, \qquad size(Aa) = 1$
then

$$A_1 = \{NA\}, A_4 = \{CM\} \tag{7}$$

The maintenance actions in the S_V and S_R states are determined as follows:

$$A_2 = A_3 = \{NA, MM, PM\} \tag{8}$$

If the IT decision maker takes no action (NA), there is no change in the patching cycle:

$$P_{s,s'}(NA) = \begin{cases} 1 & for \quad s = s' \\ 0 & for \quad s \neq s' \end{cases} \tag{9}$$

If the IT decision maker takes minimal maintenance (MM):

$$P_{s,s'}(MM) = P_{s,s'}(PM) = \begin{cases} 1 & for \quad s = s' + 1 \\ 0 & for \quad s \neq s' + 1 \end{cases}$$
(10)

Where the IT decision maker takes Corrective Maintenance (CM) the deterioration stage reaches the perfect stage:

$$P_{s,s'}(CM) = \begin{cases} 1 & for \quad s' = 1 \\ 0 & for \quad s' \neq 1 \end{cases} \tag{11}$$

In this case study we focus not only on the diversity of networked computer systems, but also on the different usage of computers throughout a university by staff, faculty, students, and researchers, and in laboratories. The optimal cost policy can be to minimize Equation (5) with respect to any action that will be taken [13, 16]. Table 2 shows the solution of a stabilized policy for patch management in an academic environment:

Table 2: Optimal policy

State	Maintenance Action	Value
S_n	NA	5492
S_v	MM	7004
S_r	PM	4493
S_u	CM	4277

The proposed policy helps IT decision makers to reduce the cost of patch management by avoiding systems failure that would cause the institution to hire one more technician to rebuild or reimage a compromised computer with a cost of 19 to 25 US dollars per hour [20]. We tried to mitigate the risk of systems failure and security vulnerabilities that create more cost for an IT department, which would have to hire more technical staff to capture users' profiles and rebuild the system that has been compromised. With the proposed SCPMP framework, an IT department can save 10 percent of its operational budget because of the resulting highly efficient level of its employees' performance.

CONCLUSION

In this paper, we have proposed an SE framework for an SCPMP in diverse environments based on a case study of academic networked computer systems. We focused on an optimal research SE framework that can improve the current patch management process framework for preventing an organization's networked computer systems from becoming the target of vulnerabilities that are immediately exploitable and could lead to critical system failure or security compromise. We studied an academic environment with nonhomogeneous software packages installed on complex computing systems. Indeed, implementing the patch management process in different computational environments using the Bayesian stochastic theorem could be the subject of future study.

In this work, after we modeled our MO phase by applying the MDP model, we developed a new security and critical patch management framework to help IT managers make decisions not only to make the patch management process more efficient, but also to keep the patch management deployment process a low-cost part of the operational IT infrastructure.

REFERENCES

[1] Cavusoglu, H., H. Cavusoglu, and J. Zhang. "Security Patch Management: Share the Burden or Share the Damage?" *Management Science*, Volume 54, Number 4, 2008, pp.657-670.

[2] Nikolaidou, M., N. Alexopoulou, A. Tsadimas, A. Dais, and D. Anagnostopoulos. "A Consistent Framework for Enterprise Information System Engineering" *Presentation at 10th IEEE International Enterprise Distributed Object Computing Conference*, doi: 10.1109/EDOC.2006.6 (2006), October 2006, pp.492-496.

[3] Tian, H. T., L. S. Huang, Z. Zhou, and Y. L. Luo. "Arm Up Administrators: Automated Vulnerability Management." *Presentation at 7th International Symposium on Parallel Architectures, Algorithms and Networks*, 2004, doi: 10.1109/ISPAN.2004.1300542 (2004), May 10-12, pp.587-593.

[4] Wu, W., F. Yip, E. Yiu, and P. Ray. "Integrated Vulnerability Management System for Enterprise Networks." *Presentation at the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005, doi: 10.1109/EEE.2005.83 (2005), March 29-April 1, pp.698-703.

[5] Zhao, D., S. M. Furnell, and A. Al-Ayed. "The Research on a Patch Management System for Enterprise Vulnerability Update." *Presentation at the WASE International Conference on Information Engineering*, doi: 10.1109/ICIE.2009.233 (2009), July 10-11, 2009, pp.250-253.

[6] Liu, S., R. Kuhn, and H. Rossman. "Surviving Insecure IT: Effective Patch Management." *IT Professional* 11, no.2, doi: 10.1109/MITP.2009.38, March-April 2009, pp.49-51

[7] Park, Jung-jin, Jin-sub Park, Jeong-gi Lee, Bong-hoi Kim, Geum-boon Lee, and Beom-joon Cho. "Windows Security Patch Auto-Management System Based on XML." *Presentation at the 9th International Conference on Advanced Communication Technology*, doi: 10.1109/ICACT.2007.358382 (2007), February 12-14, 2007, pp.407-411.

[8] Ramaswamy, A., S. Bratus, S. W. Smith, and M. E. Locasto. "Katana: A Hot Patching Framework for ELF Executables." *Presentation at the International Conference on Availability, Reliability, and Security*, 2010. doi: 10.1109/ARES.2010.112 (2010), February 15-18, pp.507-512.

[9] Chang, Chuan-Wen, Dwen-Ren Tsai, and Jui-Mi Tsai. "A Cross-Site Patch Management Model and Architecture Design for Large Scale Heterogeneous Environment." *Presentation at the 39th Annual International Carnahan Conference on Security Technology*, doi: 10.1109/CCST.2005.1594837 (2005), October 11-14, 2005, pp.41-46

[10] Yang B., Aran N. A., Zeng S., and Puri R. "SLA-Driven Applicability Analysis for Patch Management." *Presentation at the IFIP/IEEE International Symposium on Integrated Network Management*, doi: 10.1109/INM.2011.5990544 (2011), May 23-27, 2011, pp.438-445

[11] Seo, Jung-Taek, Yun-ju Kim, Eung-Ki Park, Sang-won Lee, Taeshik Shon; and Jongsub Moon. “Design and Implementation of a Patch Management System to Remove Security Vulnerability in Multi-Platforms.” *Fuzzy Systems and Knowledge Discovery* 4223 doi: 10.1007/11881599_87, 2006, pp.716-724.

[12] Dantu, R., P. Kolan, R. Akl, and K. Loper. "Classification of Attributes and Behavior in Risk Management Using Bayesian Networks." *Presentation at 2007 IEEE Conference on Intelligence and Security Informatics*, doi: 10.1109/ISI.2007.379536 (2007), May 23-24, 2007, pp.71-74.

[13] Mazzuchi, T. A., J. M. van Noortwijk, and M. J. Kallen. “Maintenance Optimization.” In *Encyclopedia of Statistics in Quality and Reliability*. New York: Wiley, 2007. doi: 10.1002/9780470061572.eqr109.

[14] van der Weide, J. A. M., M. D. Pandey, and J. M. van Noortwijk. “Discounted Cost Model for Condition-Based Maintenance Optimization.” *Reliability Engineering & System Safety* 95, Issue 3 ISSN 0951-8320, 10.1016/j.ress.2009.10.004, March 2010, pp.236-246.

[15] van Noortwijk, J. M., A. Dekker, R. M. Cooke, and T. A. Mazzuchi. "Expert Judgment in Maintenance Optimization." *IEEE Transactions on Reliability*, Volume 41, Number 3, doi: 10.1109/24.159813, September 1992, pp.427-432.

[16] Amari, S. V., L. McLaughlin, and H. Pham. "Cost-Effective Condition-Based Maintenance Using Markov Decision Processes." *Presentation at the Annual Reliability and Maintainability Symposium*, doi: 10.1109/RAMS.2006.1677417 (2006), January 23-26, 2006, pp.464-469.

[17] Cavusoglu, H. and S. Raghunathan. "Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge." *IEEE Transactions on Software Engineering*, Volume 33, Number 3d, doi: 10.1109/TSE.2007.26, March 2007, pp.171-185.

[18] Okamura, H., M. Tokuzane, and T. Dohi. "Optimal Security Patch Release Timing under Non-homogeneous Vulnerability-Discovery Processes." *Presentation at 20th International Symposium on Software Reliability Engineering*, doi: 10.1109/ISSRE.2009.19 (2009), November 16-19, 2009, pp.120-128.

[19] Murphy, Kevin. ”Markov Decision Process (MDP) Toolbox for Matlab.”, http://www.cs.ubc.ca/~murphyk/Software/MDP/mdp.html, 1999, Last updated: October 23, 2002.

[20] Salary Wizard, “a division of Kenexa”, URL = http://swz.salary.com/SalaryWizard/PC-aintenance-Technician-I-Salary-Details-20052.aspx, January 2013 Salary.com, last updated April 2013.

[21] Marie-Josee Cros. “Markov Decision Processes (MDP) Toolbox”,

http://www.mathworks.com/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox, 09 Nov 2009, last updated 31 Oct 2012.

[22] Bellman, R. “On the Theory of Dynamic Programming.” *Proceedings of the National Academy of Sciences of the USA*, PNAS, Volume 38, Number 8, URL = {http://www.pnas.org/content/38/8/716.short}, August 1, 1952, pp.716-719.

[23] Ross M. S., *Applied Probability Models with Optimization Applications* (Courier Dover Publications 1970), ISBN 0-486-67314-6, 1970, pp.119-132

AUTHOR BIOGRAPHIES

Hadi Mohammadi is a doctoral student in the Engineering Management and Systems Engineering Department in the School of Engineering and Applied Science at the George Washington University, Washington, D.C. He has held software deployment engineer position at Department of Computing Facility.

Thomas Mazzuchi, D.Sc. received a B.A. (1978) in Mathematics from Gettysburg College, Gettysburg, PA, a M.S. (1979) and a D.Sc. (1982), both in Operations Research from the George Washington University, Washington DC. Currently, he is a Professor of Engineering Management and Systems Engineering in the School of Engineering and Applied Science at the George Washington University, Washington, D.C. He is also the Chair of the Department of Engineering Management and Systems Engineering at the George Washington University where he has served as the Chair of the Operations Research Department and as Interim Dean of the School of Engineering and Applied Science.

Shahram Sarkani, Ph.D., P.E., is a Professor of Engineering Management and Systems Engineering at The George Washington University. Professor Sarkani has engaged in engineering research, technology development, and engineering education since 1980. He is author of over 150 technical publications and presentations. He remains engaged with important ongoing research in the fields of engineering management, systems engineering, civil engineering, and logistics management. Since 1987, he has conducted sponsored research with such organizations as NASA, the National Institute of Standards and Technology, the National Science Foundation, the U.S. Agency for International Development (in association with Zagazig University, Egypt), the U.S. Department of Interior, the U.S. Department of Navy, the U.S. Department of Transportation, and Walcoff and Associates Inc.