A Systems Engineering Framework for Implementing a Security and Critical Patch Management Process in Diverse Environments (Academic Departments' Workstations)

by Hadi Mohammadi

B.S. in Industrial Management, June 2004, Islamic Azad University
M.S. in Engineering Management, December 2011, The George Washington University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

January 31, 2014

Dissertation directed by

Thomas A. Mazzuchi
Professor of Operations Research and of Engineering Management

Shahram Sarkani
Professor of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Hadi Mohammadi has passed the Final Examination for the degree of Doctor of Philosophy as of December 6th 2013. This is the final and approved form of the dissertation.

A Systems Engineering Framework for Implementing a Security and Critical Patch Management Process in Diverse Environments (Academic Departments' Workstations)

Hadi Mohammadi

Dissertation Research Committee:

Thomas A. Mazzuchi, Professor of Operations Research and of Engineering Management, Dissertation Co-Director

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

E. Lile Murphree, Professor of Engineering Management and Systems Engineering, Committee Member

John Bischoff, Professor of Engineering Management and Systems Engineering, Committee Member

Jason Dever, Professor of Engineering Management and Systems Engineering, Committee Member

© Copyright 2013 by Hadi Mohammadi
All Rights Reserved

Dedication

I would like to thank GOD, first and foremost, for granting me the opportunity to achieve success and happiness. Next, this dissertation is dedicated to the most important people in my life, my family, parents, and in-laws.

To my mother, Shirin– who worked nights to put me through high school and undergrad, and then made sure that I would have the same educational opportunities like the others. Mom, you are my rocked and without your encouragement I wouldn't achieved any things. Thank you for always believing in me.

To my beautiful wife, Fatemeh – You were the most impacted by my studies. None of my Masters and PhD degrees would have been possible without your love, patience, and support. Thank you for inspiring me every day. I appreciate and love you!

.

Acknowledgement

In the fall of 2009, during my first advising session with Dr. Thomas A. Mazzuchi and Dr. Shahram Sarkani, I selected my first courses for the master's degree program at the Department of Engineering Management and Systems Engineering. At that meeting, they asked me about my motivation for moving to the U.S. and I replied, "I came here to change my life." Thinking back to that day has inspired me to work in the following years. Dr. Mazzuchi and Dr. Sarkani have encouraged and guided me through my studies, which enabled me not only to complete the master's degree, but also, the journey of the Ph.D. program.

I would like to thank the chairman of my dissertation defense board Dr. Lile Murphree and members Dr. John Bischoff and Dr. Jason Dever for their reviews. I sincerely thank the graduate faculty members and staff at the Department of Engineering Management and Systems Engineering for their support during my graduate studies. I am grateful to Mr. Raoul Gabiam, the director of Computing Facility Department at GWU, for his support and encouragement during my doctoral endeavors. I would also like to thank Dr. Muhammad Faysal Islam and Ehsan Naranji for the reviews and encouragements.

I would like to thank my family and friends, specially my mother Shirin Afshar, father-in-law Dr. Javad Farahani, brother Shahin Mohammadi, sister Hoda Mohammadi, sister-in-law Zahra Farahani, and my family who always believed and encouraged me to do the best. I am thankful and grateful to my dear wife Fatemeh Farahani for her support and sacrifices during my journey through the Ph.D. program and as always.

Abstract

A Systems Engineering Framework for Implementing a
Security and Critical Patch Management Process in Diverse Environments
(Academic Departments' Workstations)

Use of the Patch Vulnerability Management (PVM) process should be seriously considered for any networked computing system. The PVM process prevents the operating system (OS) and software applications from being attacked due to security vulnerabilities, which lead to system failures and critical data leakage. The purpose of this research is to create and design a Security and Critical Patch Management Process (SCPMP) framework based on Systems Engineering (SE) principles. This framework will assist Information Technology Department Staff (ITDS) to reduce IT operating time and costs and mitigate the risk of security and vulnerability attacks. Further, this study evaluates implementation of the SCPMP in the networked computing systems of an academic environment in order to:

1. Meet patch management requirements by applying SE principles.
2. Reduce the cost of IT operations and PVM cycles.
3. Improve the current PVM methodologies to prevent networked computing systems from becoming the targets of security vulnerability attacks.
4. Embed a Maintenance Optimization Tool (MOT) in the proposed framework. The MOT allows IT managers to make the most practicable choice of methods for deploying and installing released patches and vulnerability remediation.

In recent years, there has been a variety of frameworks for security practices in every networked computing system to protect computer workstations from becoming compromised or vulnerable to security attacks, which can expose important information and critical data. I have developed a new mechanism for implementing PVM for maximizing security-vulnerability maintenance, protecting OS and software packages, and minimizing SCPMP cost. To increase computing system security in any diverse environment, particularly in academia, one must apply SCPMP. I propose an optimal maintenance policy that will allow ITDS to measure and estimate the variation of PVM cycles based on their department's requirements. My results demonstrate that MOT optimizes the process of implementing SCPMP in academic workstations.

Table of Contents

Dedication ..... **iv**

Acknowledgement ..... **v**

Abstract ..... **vi**

Table of Contents ..... **viii**

List of Figures ..... **xi**

List of Tables ..... **xiii**

List of Acronyms ..... **xiv**

Chapter 1 — Introduction **1**

1.1 Statement of the Problem **3**

1.2 Relevance and importance **4**

1.3 Contribution to the body of knowledge **4**

1.4 Purpose **5**

1.5 Background **6**

1.6 Significance **7**

1.7 Scope and limitations **9**

1.8 Dissertation organization **10**

Chapter 2 — Literature Review **11**

2.1 Systems Engineering **12**

2.2 Network and Cyber Security **15**

2.3 Patch Vulnerability Management **16**

2.4 Risk Management **18**

2.5 Maintenance Optimization Tool **25**

2.6 IT Operational Cost Analysis **26**

2.7 Patch Deployment Models and Frameworks **29**

2.7.1 Generic patch types: 31

2.7.2 Scholarly patch deployment models and frameworks 33

2.7.2.1 Windows security patch management 33

2.7.2.2 SANS Patch Management Process 35

2.7.2.3 Cooperative software maintenance model 39

2.8 Literature review summary **42**

Chapter 3 — Research Framework **44**

3.1 Download **45**

3.2 Detection **46**

3.3 Test **46**

3.4 Risk Analysis **47**

3.5 Deployment and Installation 48
3.6 Verification 49
3.7 Selection of Hypotheses and Sub-Hypotheses 49
3.8 Overall SCPMP structure and framework 50
3.8.1 Management server center 51
3.8.2 Patch storage 51
3.8.3 Client agent 52
3.8.3.1 Beta tester 52
3.8.3.2 Production environment 53
3.9 Research Framework Summary 53
Chapter 4 — Research Methodology 55
4.1 Maintenance optimization stage 57
4.2 Modeling the Markov Decision Process 58
4.2.1 Inputs 58
4.2.1.1 Patch Deterioration Stages 58
4.2.1.2 Maintenance Actions (MA) 60
4.2.1.3 Cost Estimation 61
4.2.1.4 Equations and optimizing 63
4.3 Research Methodology Summary 65
Chapter 5 — Case Study 67
5.1 Target Environment and Networked Computing Systems 68
5.2 Data collection 69
5.2.1 Management Server Center 69
5.2.2 Operating Systems 71
5.2.3 Software packages and patch counts 73
5.3 Problem Description 75
5.4 Inputs 75
5.5 Problem Formulation and Maintenance Actions 77
5.6 Optimal Policy Solution and results 78
Chapter 6 — Discussion and Conclusions 81
6.1 Conceptual Model 81
6.2 Hypotheses Results 83
6.3 Future study and work 84
6.3.1 Future study of vulnerability disclosure 84
6.3.2 Evaluation and assessment performance method 84

6.4 Conclusion 85
Bibliography 87
Appendix A 95
Appendix B 107

List of Figures

Figure 2-1 Sequence of topics reviewed.....12

Figure 2-2 NASA Systems Engineering Processes and Requirements for executing the tasks of a project (NASA, 2007, pp. 4) 13

Figure 2-3 NASA Systems Engineering engine, (NASA, 2007, pp. 5) 14

Figure 2-4 Vulnerability life-cycle model (Okamura et al., 2009) 18

Figure 2-5 Timeline for vulnerability discovery (Cavusoglu et al., 2007).....18

Figure 2-6 IT risk management (Grob et al., 2008) 24

Figure 2-7 A generic model of patch management (Adams, 2007).....30

Figure 2-8 System architecture of an automatic Windows patch management process based on XML methods (Park et al., 2007).....34

Figure 2-9 SANS's patch management process (Medzich, 2004).....36

Figure 2-10 A cooperative software maintenance framework (Gupta et al., 2011).....40

Figure 2-11 Systems engineering Vee Model (Forsberg and Mooz, 1991) 42

Figure 3-12 Circular patch management method.....45

Figure 3-13 Modified model of circular deployment of the patch management process..48

Figure 3-14 SCPMP overall structure (Mohammadi et al., 2013).....51

Figure 4-15 SCPMP framework (Mohammadi et al., 2013) 56

Figure 4-16 Patch type deterioration stages (Mohammadi et al., 2013) 59

Figure 4-17 Maintenance decision actions (Mohammadi et al., 2013).....61

Figure 4-18 Deterioration process (Mohammadi et al., 2013) 62

Figure 5-19 Patching workflow (Dell KACE K1000, kace.com).....70

Figure 5-20 Optimal policy for deploying patches based on MDP calculation.....80

Figure 6-21 Conceptual model and research roadmap.....82

List of Tables

Table 1-1 Stakeholder Significance and Interest 7

Table 2-2 Risk impact categories (MITRE Corporation, 2013) 20

Table 2-3 Security and critical patch management risk matrix 21

Table 2-4 Risk assessment martix 23

Table 2-5 Patch management process comparison based on SE principles 43

Table 5-6 List of managed OSs and computer statistics for SEAS (Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu) 71

Table 5-7 SEAS departmental networked computing systems (seascf.seas.gwu.edu) 72

Table 5-8 Software packages for applying patches (Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu) 73

Table 5-9 Patch bulletin information (SEAS; Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu) 74

Table 5-10 costs breakdown list (Mohammadi et al., 2013) 76

Table 5-11Transition probabilities and time functions (Mohammadi et al., 2013) 77

Table 5-12 Optimal Policy based on cost analysis (Mohammadi et al., 2013) 79

List of Acronyms

CM	Corrective Maintenance
INCOSE	International Council of Systems Engineering
IT	Information Technology
ITDS	Information Technology Department Staff
ITL	Information Technology Laboratory
ITM	Information Technology Management
MA	Maintenance Action
MDP	Markov Decision Process
MM	Minimal Maintenance
MOP	Maintenance Optimization Phase
MOT	Maintenance Optimization Tool
NA	No Action
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
OS	Operating System
OVAL	Open Vulnerability Assessment Language
PDM	Patch Deployment Model
PM	Preventive Maintenance
PMP	Patch Management Process “Not same as the Project Management Professional credential.”
PVM	Patch and Vulnerability Management
SCP	Security and Critical Patch
SCPMP	Security and Critical Patch Management Process
SDMS	System and Deployment Management Server
SE	Systems Engineering
SUS	Soft Update Service
USP	Urgent Security Patches

Chapter 1 — Introduction

In recent years, the major underlying challenge facing Information Technology Department Staff (ITDS) is to protect networked computer systems, which must be defended from becoming compromised or vulnerable to security attacks. An overwhelming aspect of this problem is to deploy and install OS and software patches across the entire network. By establishing and ensuring network security, the network environment becomes more reliable. Patch management is the core duty of network security management (Cavusolgu, Cavusolgu and Zhang 2008), which not only has been addressed by corporations and firms, but also by government departments and federal agencies. Network engineers and IT decision makers try to reduce network vulnerabilities and software flaws by applying proper patches at the appropriate times. Performing the Security and Critical Patch Management Process (SCPMP) (1) defends networked computing systems from becoming compromised (Mohammadi, Mazzuchi, and Sarkani, 2013), (2) mitigates the risk of security attacks, and (3) decreases IT operating costs, which are among the most significant expenses for any company or institute. As the complexity of computer applications and operating systems increases, proactively preventing networked computing systems from becoming targets of security attacks is a vitally important challenge for any IT department or home user.

A patch is an additional segment of software or OS that addresses security issues, reduces the risk of security vulnerabilities, and even provides extra functionality. Vulnerability is an imperfection of the developed software or operating system. Hackers

can exploit this to obtain root control and therefore gain access to file systems saved on the computer (Mell, Bergeron, Henning 2005).

As use of networked computing systems increases, vulnerabilities and security attacks are growing due to the advancement of Internet accessibility (Wu, Yip, Yiu, and Ray, 2005). In 2012, 4,347 vulnerabilities in software and operating systems were identified; the number has grown in the last three years (Florian, 2013). In the U.S. for Fiscal Year 2012, corporations spent $3 billion on desktop security software (McMillan, 2012). These alarming numbers and costs are key factors in determining the optimal remediation method of vulnerability management. Reduction of these threats can be achieved by enhancing network security protection management, addressing device configuration problems, and training users. There have been many attempts to develop the best Patch Management Process (PMP), which reduces security vulnerabilities, and Patch Vulnerability Management (PVM) system, which addresses software flaws, fixes critical security failures on OS, and detects malicious code.

The methodology applied in this study is based on a daily Systems Engineering (SE) framework for applying SCPMP in the academic environment. In addition, a probabilistic Maintenance Optimization Tool (MOT) using the Markov Decision Process (MDP) has been included in the proposed framework. The MOT acts as a decision-making tool for selecting the ideal method for identifying patching deployment procedures and reducing security risks and IT costs. Developing an SE framework that has been integrated into a MOT will modify the current patch management process and improve the security deployment policy. This allows for reduction of vulnerability attacks and makes operation of compromised machines possible. Based on networked computing

system requirements, environment, and infrastructure, my research on the SE framework will help security policy makers and ITDS to make appropriate decisions when deploying patches across the entire network.

1.1 Statement of the Problem

Currently, no method systematically and proactively deploys and installs Security and Critical Patches (SCP) across an entire networked computing system. The combination of the SCP and the MOT enhances deployment-process reliability and improves the efficiency and security of the computing environment. There have been numerous PMP and security vulnerability methods; however, my study offers the most efficient PMP policy based on SE principles. The proposed framework will reduce the costs of IT operations, mitigate the risk of security attacks, and avoid leakage of critical data and information.

In the IT industry, there is always a risk of managing and administering compromised machines. Possible flaws in an OS or software packages are exploited, and therefore must be corrected. When this occurs, critical data are released; this not only exposes personal information, but also damages the OS and system performance (Wu et al., 2005). I propose an SE framework for implementing a patching process in any environment or computing system. This ensures that the risk of security attacks or delays in the PMP schedule will be mitigated.

1.2 Relevance and importance

Deploying the SCP and remediating vulnerability attacks are continuing problems that every IT management department has faced for decades.

This research introduces a new optimized framework for the SCPMP to the existing knowledge base of current SCP deployment procedures and assesses its performance by running multiple tests on more than 850 machines and gathering data and information on the deployment processes. This study aims not only to evaluate whether installation of urgent security patches can mitigate the risk of attack, but also to reduce the cost of IT operating systems. Further, I investigate the strength of embedding the MDP to evaluate patching data and implement the SCPMP, which is organized within the SE framework.

In this research, I have considered two major patch types in security network engineering: OS platform patches which includes Microsoft Windows and Mac OS X patches and software application patches, which are responsible for fixing the flaws or updating to newer versions. The breakdown lists of these two patch types will be shown in the next chapter.

1.3 Contribution to the body of knowledge

The proposed SE framework includes data analysis and maintenance optimization. This contribution to the process of SCP deployment will not only aid academic computing systems, but also any homologous ITDS who run and deploy security patches and manage the PVM. As the security deployment process becomes

more complex, the embedding of probabilistic models into the PMP will aid IT decision makers to perform patch vulnerability management more efficiently (Bommannavar and Bambos, 2011). Systems engineering fundamentals underlie the conceptual design of the overall process and risk assessment tasks, and SE principles modify the PMP in the early stages by using computational methods during the testing and evaluating of SCPMP phases. This proposed SE framework presents a phenomenological decision-making tool that evaluates the task of implementing and developing the optimized policy for better design of a PVM in any networked computing system environment.

1.4 Purpose

The primary goal of this study is to create a practical framework that relies on SE principles to implement SCPMPs. An experimental and theoretical case study will be conducted to test the SE framework, which will analyze and demonstrate the necessity of implementing the SCPMP on networked computing systems in the appropriate time period. Since the PMP imposes higher costs on the IT department, system improvement and a different PVM must be considered. Toward that end, I propose a maintenance optimization procedure that has been embedded in the proposed SE framework to minimize IT operating costs and mitigate risk by developing a better PMP decision. The proposed framework is intended to be used by ITDS and IT decision makers to implement the SCPMP and reduce security and vulnerability problems.

Cyber security has become a key element in the day-to-day lives of those in the IT industry, who depend on the deployment and installation of SCPs to continuously enhance the security, performance, and stability of their OS and software packages. In

addition, deployment of the SCP decreases the cost of IT operations and mitigates the risk of losing important data and information.

1.5 Background

It is vital that the latest SCPs are installed and deployed on any networked computing systems ITDS manage to protect them from being compromised. Contemporary IT departments and technology offices are believed to have the most efficient patch management methodology to protect their machines from attack, which can cause system failures and impose higher operational costs. Thus, applying the proposed framework will keep the network environment more secure and avoid the leakage of information and data.

Systems engineering fundamentals aid engineers in developing efficient technological systems (Kosmann, Sarkani, & Mazzuchi, 2013). Moreover, SE defines a system's architecture based on the customer's requirements to develop a system and subsystem that functions properly (Nikolaidou, Alexopoulou, Tsadimas, Dias, and Anagnostopoulos, 2006). Herein, the integration of patch management and SE principles will combine system architecture, performance management, and test analysis, while also providing decision support. This is performed with the aim of developing and assisting in the implementation and deployment of security and critical patch processes in the most efficient way. In addition, applying the developed patching system to every networked computer at an early stage can decrease the cost of IT operations and mitigate the risk of a compromised networked computing system.

1.6 Significance

This study's significance arises from its consideration of five vital aspects of IT management: (1) IT security, (2) a framework for risk, (3) networked computing system management, (4) SCP deployment, and (5) IT operational costs. In the proposed framework, these five elements are responses to questions from a wide range of stakeholders. The target group includes IT engineers, who implement the SCPMP in their IT security management infrastructure. Table 1-1 presents the categories of stakeholder interests in regards to urgency and importance.

Table 1-1 Stakeholder Significance and Interest

Stakeholders	IT security	Risk framework	Networked computing systems management	SCP deployment	IT operational cost
Government departments and agencies	X	X	X	X	X
IT industry	X	X	X		X
Systems engineer community		X		X	X
Public	X			X	X

The IT division of any government department or federal agency maintains and secures its networked computing system's OS and software applications by applying an SCP. This action will prevent their system from being attacked or compromised. The top priorities of national defense and telecom agencies are cost and risk management; therefore, they endeavor to tackle cyber-attacks and unauthorized intrusions. Risks and issues that might appear in the patch deployment process will be identified by risk management process's ability to control and mitigate (Aris, Arshad, and Mohamed,

2008). One of the government department or federal agency is the Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST), which provides technical solutions and recommendations for the nation's networked computing infrastructure and IT standard models (Mell, Bergeron, and Henning, 2005). NIST aims to address security flaws, malicious activities, and system vulnerabilities by recommending and presenting practical guidelines for systematic processes. NIST also recommends the management and design of SCP deployment processes.

Automatic networked security management is increasing among central IT divisions and ITDS; IT industry personnel and technicians can monitor, deploy, and install an SCP across an entire network. By using a trusted and adequate System and Deployment Management Server (SDMS), they will save money for their companies. The proposed SE framework is for testing and executing PMP across entire networked computing systems. It will create security management based on SE principles and develop optimized patch deployment processes. This also will improve and solve potentially extreme exploitation in the event of a security vulnerability attack. The SE framework for implementing the SCPMP is to protect the computing system from malicious acts and vulnerability attacks, which have significance for all cyber security engineers who are holding significant acts on IT industry. This methodology and the planning for and design of the deployment process should be a high-priority requirement for IT security management, any networked computing infrastructure, database safety, and security system development.

Public users are considered as well. Most likely, these stakeholders are not proactive in identifying a necessary patch and implementing vulnerability management. They assume that the software provider is responsible for identifying patches and vulnerabilities and will automatically push the patches to users' systems. Therefore, they are passive about updating patches and mitigating vulnerabilities and security risks.

The groups described above have been identified as stakeholders and form the study's spectrum of PMP users; the research is designed to address the security issues that concern these stakeholders.

1.7 Scope and limitations

The scope of this research is to present an SE framework for implementing the SCPMP, which can be used with any diverse networked computing system. The goal in applying this SE framework to any networked computing system is to improve and optimize the patching processes that enhance system security and reduce IT operation costs.

Generally, OS and software packages are installed on a networked computing system that can be exploited. Their weaknesses expose codes to hackers, who attack the system and exploit its vulnerabilities.

One of the limitations of implementing the SCPMP is the unpredictable logistics of releasing new patches. When the PMP is underway, the ITDS, who are responsible for overseeing the process, are not fully apprised of software or OS vulnerability during that period. Further, a given patch is limited to certain software packages and operating

systems. Patch distribution will be more costly and requires a wide variety of knowledge about software development.

1.8 Dissertation organization

Chapter 1 interprets the SCPMP's overall structure based on SE principles. Chapter 2 reviews and discusses the relevant literature. Chapter 3 proposes a research framework for implementing the SCPMP in diverse environments and explains how each stage operates. Chapter 4 describes how to use the MOT inside the proposed framework, and presents the methodology and collected data. In Chapter 5 a case study is discussed that is focused on implementing the SCPMP in an academic department workstation by using the MDP tool. I conclude the dissertation in Chapter 6 and suggest future studies. The bibliography and appendices follow.

Chapter 2 — Literature Review

This chapter presents an overview of relevant research, beginning with an understanding of contemporary SE principles, which is important when considering the current IT industry's applications. I next review risk assessments, considering both security attacks and vulnerability events, which increase operational costs for IT departments and risk the disclosure of critical data and information. A detailed review of the security of networked computing systems is then provided, followed by a discussion of literature on the MOTs embedded in the proposed SE framework. The chapter closes with review of a patch deployment management process is provided. Figure 2-1 illustrates the sequence of topics reviewed.

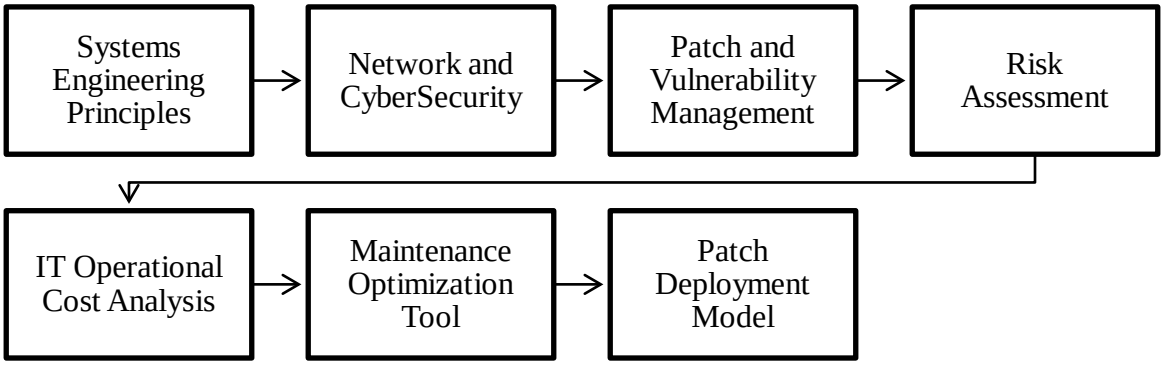

Figure 2-1 Sequence of topics reviewed

2.1 Systems Engineering

Systems engineering allows an ideal design for a process that meets essential needs. Also, SE is necessary to finalize successful outcomes and categorize the system's operational steps. Nikolaidou, Alexopoulou, Tsadimas, Dias, and Anagnostopoulos (2006) describe SE as a "*process of defining the desired architecture of a system and exploring performance requirements, ensuring that all system components are identified and properly allocated and system resources can provide the desired performance*" (pp. 492 - 496). Alternatively, the International Council of Systems Engineering (INCOSE) defines SE as a systematic method to direct the system to perform successfully and emphasizes that "s*ystems* e*ngineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation*" (INCOSE Handbook, 2007, pp. 3.3- 3.7).

The main resource I used to create the SE framework is the National Aeronautics and Space Administration ([NASA]; 2007) handbook, which is a guide for the community that provides standard SE principles and describes how SE has been applied in NASA environments. The handbook describes SE as "*a methodical, disciplined approach for the design, realization, technical management, operations, and retirement of a system*" (pp. 3). NASA divides SE into nine main tasks that are executed from beginning to end, as demonstrated in Figure 2-2:

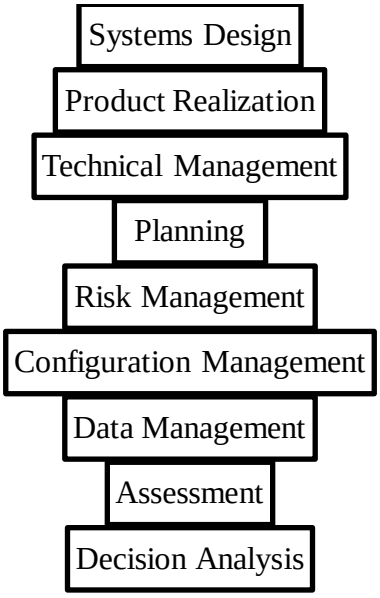

Figure 2-2 NASA Systems Engineering Processes and Requirements for executing the tasks of a project (NASA, 2007, pp. 4)

The NASA handbook explains that a system is composed of different components with a common goal and describes how to develop a process using SE fundamentals and break down an SE engine in the system design phase and mission. The SE engine uses stakeholder requirements as the baseline to construct the process, address technical needs,

and transfer the technical requirements to the system design. Figure 2-3 shows NASA systems engineering engine.

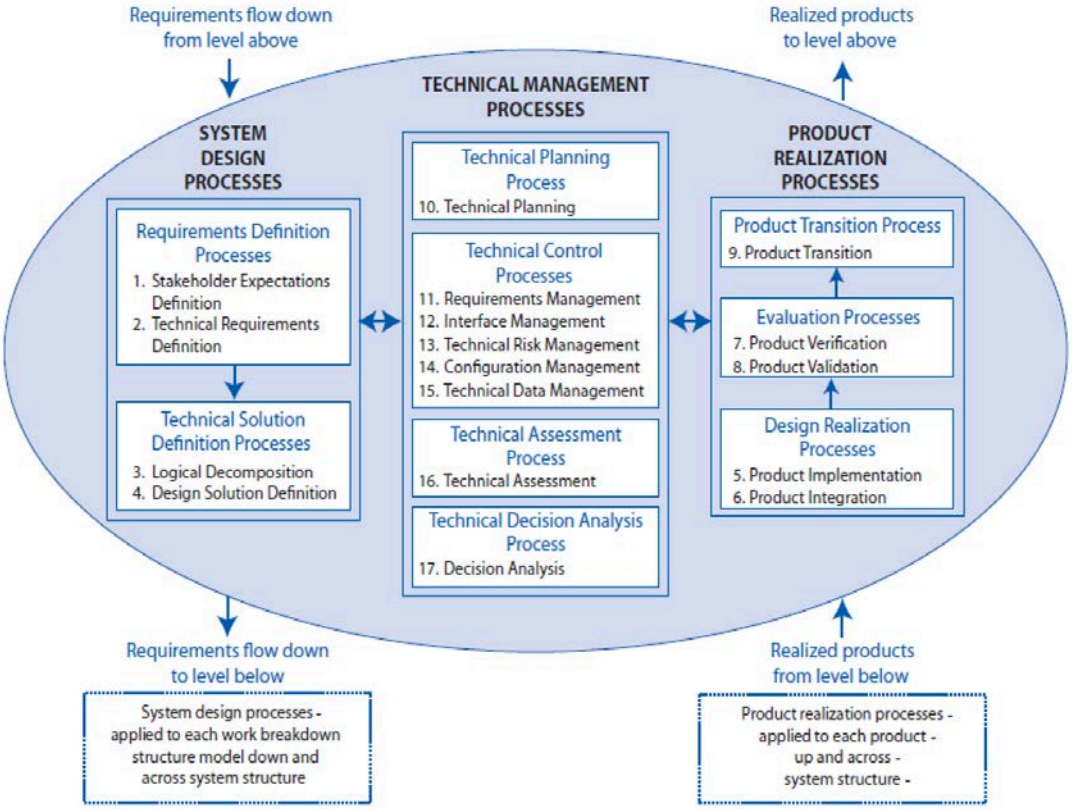

Figure 2-3 NASA Systems Engineering engine, (NASA, 2007, pp. 5)

Regardless of these definitions of SE and system design, generally SE is the realization and conversion of requirements and initial expectations into specific goals. Multiple steps in this transformation are necessary for project success. I have attempted, in this study, to use this knowledge and process of transformation to build and design an SE framework. Thus, use of SE principles is vital. Information Technology Management (ITM) enhances efficiency by focusing on system architecture (Wang, Yang, Liu, Ma, Sun, and Chen, 2007). It is crucial to design software as a reliable and resilient system in

order to predict all possible consequences (Madni and Jackson, 2009). In light of the above, we can conclude that for IT and software development, system thinking through systematic processes is essential for attaining the desired result.

2.2 Network and Cyber Security

Cyber security has become serious problems for any networked computing system (Okimoto, Ikegai, Inoue, Okada, Ribeiro, and Maruyama, 2013). Businesses, governments, and home users pay a great deal of attention to cyber security and the protection of networked computing systems to avoid becoming the target of vulnerabilities, which lead to system failures. An IT infrastructure that contains an incomplete or imperfect system design can lead to exploitations in networked computing or Internet-based systems. Mulligan and Schneider (2011) define cyber security as "the obstacle to success of the information age."

The increasing speed of IT utilization and technology enhancement ensures that there will always be some exploitable flaws in software packages and security vulnerabilities for operating systems (Tian, Huang, Zhou, and Luo, 2004). Multiple companies produce detailed lists of possible new vulnerability issues. One of the most well-known is the MITRE Corporation (mitre.org). MITRE, which is a not-for-profit organization, is attempting to generate a dictionary of Common Vulnerabilities and Exposures (CVE) to standardize the Open Vulnerability Assessment Language (OVAL) for IT experts and construe a guideline for security vulnerability for networked security mangers (Tian et al., 2004).

Incomplete lines of code or improper package design causes software package flaws and OS vulnerabilities (Chuan-Wen, Dwen-Ren, and Jui-Mi, 2005). Therefore, vulnerability management consists of the discovery of software package weaknesses, remediation testing, examination of released patch on test environment, and deployment of the tested patch on production machines. Overall, system failures and software package vulnerabilities pose threats to a networked computing system. This threat can cause compromise and the leak of critical data and security information, as well as incurring more costs for the IT department (Sihvonen, Jantti, 2010).

2.3 Patch Vulnerability Management

A patch is a section of the software or operating system that can complete or fix the system design. A vulnerability is the software hole or defect that can be exploited and raise security concerns. This corresponds to breaking the system design and permeating the system architecture to access the original code. Software or OS patches and vulnerabilities must be managed by ITDS to repair compromised machines and decrease departmental costs. As stated by Mell, Bergeron, and Henning (2005), "*Patch and vulnerability management is a security practice designed to proactively prevent the exploitation of IT vulnerabilities that exist within an organization.*" Similarly, Liu, Kuhn and Rossman emphasize that an adequate patch and vulnerability management process should be repeatable and consistent (Liu, Kuhn, and Rossman, 2009).

Prevention is more important than remediation for patch and vulnerability management. If ITDS deploy security and critical patches in before any exploitation has occurred, OS flaws will be fixed and software protected. Deployment and installation of

patches to maintain and fix errors is complex and time-consuming (Yang, Zeng, Ayachitula, and Puri, 2011). When a patch for fixing security vulnerability is released, ITDS must quickly test and install it on networked computing systems to mitigate the risk of exploitation and attack (Nunez, Gustavson, Grossman, and Tappert, 2010). Patch deployment and security vulnerability must be considered not only for protection of critical information, but also repair of software flaws and OS defects, which leads to overall better system performance. Patches and vulnerabilities are a realization of security and critical patch discovery processes to transfer and develop the remediation for the discovered errors and push the installing or scripting to the networked computing system. This is demonstrated in Figure 2-4, which shows the vulnerability life-cycle model (Okamura, Takuzane, and Dohi, 2009).

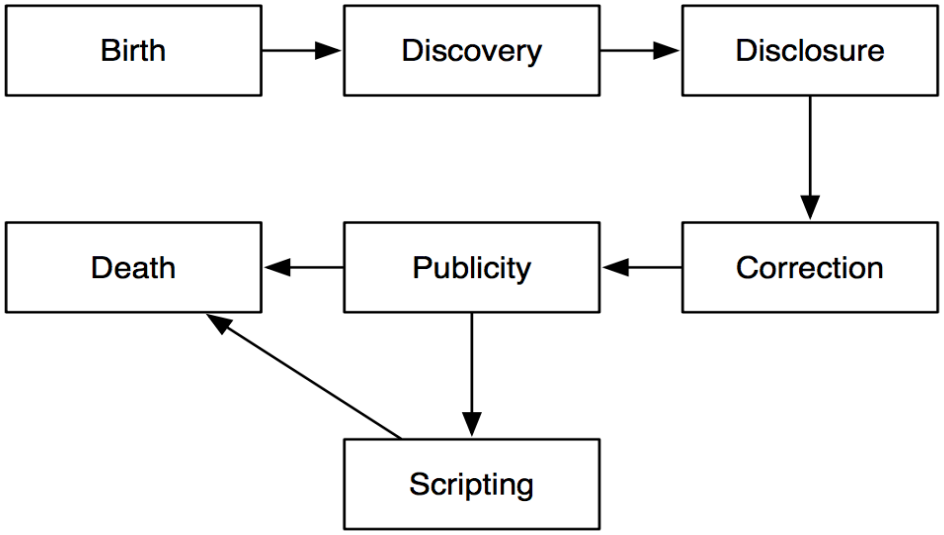

Figure 2-4 Vulnerability life-cycle model (Okamura et al., 2009)

The figure above demonstrates the steps of a generic patch and vulnerability management process. The flowchart begins with disclosure of vulnerability through patch remediation. Therefore, vulnerability and patch management is vital to the operation of a networked computing system environment to protect against cyber-attacks (Tian et al., 2004). Figure 2-5 presents the timeline of vulnerability disclosure and security management, beginning with software development and concluding with remediation and patch release (Cavusoglu, Cavusoglu, and Raghunathan, 2007).

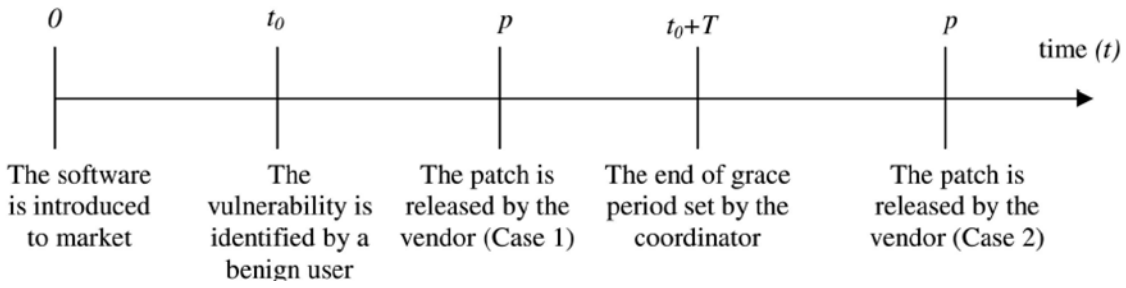

Figure 2-5 Timeline for vulnerability discovery (Cavusoglu et al., 2007)

2.4 Risk Management

MITRE and Defense Acquisition University (DOD) defines risk as "*a measure of a project's inability to achieve system life cycle objectives*" (Defense Acquisition University, MITRE Corporation, 2003). Their two objectives are to reduce (1) the impact of negative effects on system privacy and (2) the severity of consequences; identifying these objectives will affect the system infrastructure (MITRE Co. 2013).

According to SE principles, the process baseline is focused on risk management. Aris, Arshad, and Mohamed (2008) define risk management as "*the process of identifying, analyzing, controlling and managing the issue,*" which will occur during the life cycle of the system. Also, these principles emphasize that one of the key processes that must happen during system development is creation of a risk management plan. Risk management includes assessment of cost and the investigation, mitigation, and control of the vulnerability, which may threaten system security and penetrate to the database infrastructure (Lijian, Bin, and Yongjun, 2010). Therefore, risk management or risk assessment is an inevitable process that may incur costs for the company, but more likely will reduce costs in terms of remediation of security attacks or by avoiding the expense of losing security and critical data.

Since the IT department has become an invaluable sector of every business or company, operational risk is a growing concern (Hao and Yang, 2010). Aris et al. (2008) also state that risk management components must be built into IT projects to improve the impact of security. Hao and Yang (2010) define IT risk management as a key operational activity that is crucial for ITDS to track and mitigate the risk that may occur through its operation. It is essential that risk assessment occur in the early phase of the IT security project; this will help ITDS to be prepared and reduce the impact of security vulnerability on networked computing systems (Aris et al., 2008).

The framework for risk assessment presented in this study is based on key principles of risk management specifications. Particularly, it is directed to network computing system risk management and the effect of risk factors on IT security. The risk assessment that has been evaluated in this study is based on the MITRE risk assessment

toolkit, which was created by Engert and Lansdowne (1999). Table 1-1 represents the risk matrix “impact” scales that are used in this study for risk management.

Table 2-2 Risk impact categories (MITRE Corporation, 2013)

Po	Impact Category				
	Negligible	*Minor*	*Moderate*	*Serious*	*Critical*
0-10%	Low	Low	Low	Medium	Medium
11-40%	Low	Low	Medium	Medium	High
41-60%	Low	Medium	Medium	Medium	High
61-90%	Medium	Medium	Medium	Medium	High
91-100%	Medium	High	High	High	High

In this research, eight security and critical risk factors have been identified, as shown in Table 2-2 Risk impact categories (MITRE Corporation, 2013).

Table 2-3 Security and critical patch management risk matrix

Risk No.	Related Risk No.	RISK	Timeframe Start	Timeframe End	Impact	Po (%)	Borda Rank	Risk Rating	Manage/Mitigate
1	3	IF one or more urgent security and critical OS patches are released, THEN networked computing systems will be compromised.	01 Aug 2013	31 Aug 2013	C	100%	0	H	Install and deploy security and critical OS patches
2	4	IF one or more urgent security and critical software package patches are released, THEN networked computing systems will be compromised.	01 Aug 2013	31 Aug 2013	C	90%	1	H	Install and deploy security and critical software packages or application patches
3	1 and 6	IF one or more recommended OS patches are released, THEN any known security flaws are not addressed that weren't impact into the computing system's infrastructure.	01 Aug 2013	31 Aug 2013	S	60%	3	M	Test and install recommended patches
4	2 and 7	IF one or more recommended Software application patches are released, THEN any known security flaws are not addressed that weren't impact into the computing systems infrastructure.	01 Aug 2013	31 Aug 2013	S	60%	3	M	Test and install recommended patches

5	2	IF one or more version update of software package is released, THEN the software version will be changed but doesn't affect application security.	01 Aug 2013	31 Aug 2013	Mo	80%	2	M	Test and install recommended patches
6	1 and 2	IF there are no patches to be deployed, THEN ITDS follow the regular cycle of former patch deployment or install new versions of software packages.	01 Aug 2013	31 Aug 2013	N	40%	6	L	Test and install recommended patches
7	1	IF one patch that contains some minor changes on embedded software on OS is released, THEN the operating system will have the new version of embedded software, but doesn't affect application security.	01 Aug 2013	31 Aug 2013	Mi	75%	5	M	Test and install recommended patches

Security vulnerabilities are categorized as having high, medium, or low impact on networked systems (Hao and Yang, 2010). Table 2-4 presents the risk-assessment matrix based on Table 2-4's risk factors and the probability of occurrence.

Table 2-4 Risk assessment martix

Risk Matrix		IMPACT				
		Critical (C)	Severe (S)	Moderate (Mo)	Minor (Mi)	Negligible (N)
Probability or likelihood	0-10 % very unlikely the risk will occur					
	11-40 % unlikely the risk will occur					6
	41-60 % even likelihood the risk will occur		3			
	61-90 % likely the risk will occur	2	4	5	7	
	91-100 % very likely the risk will occur	1				
Risk Rating	High					
	Medium					
	Low					

A main focus of this risk assessment is to estimate the risk factors for security and critical OS and software patch types, which can have significant impact on networked computing systems. This estimation will help ITDS to identify and foresee security management issues when receiving new OS or software package patches. They will be categorized to ensure that all networked computing systems are patched.

Organizational IT risk management can improve engineering patch deployment management policies and reduce the probability of damage (Xiaocong and Ling, 2010).

In Figure 2-6, Grob, Strauch, and Buddendick (2008) define IT risk management as individual risk concentrations. Due to better Internet network connectivity and improved networked computing system performance, there is always some risk of attack by outside hackers, which necessitates management of compromised machines (Dantu, Loper, and Kolan, 2004).

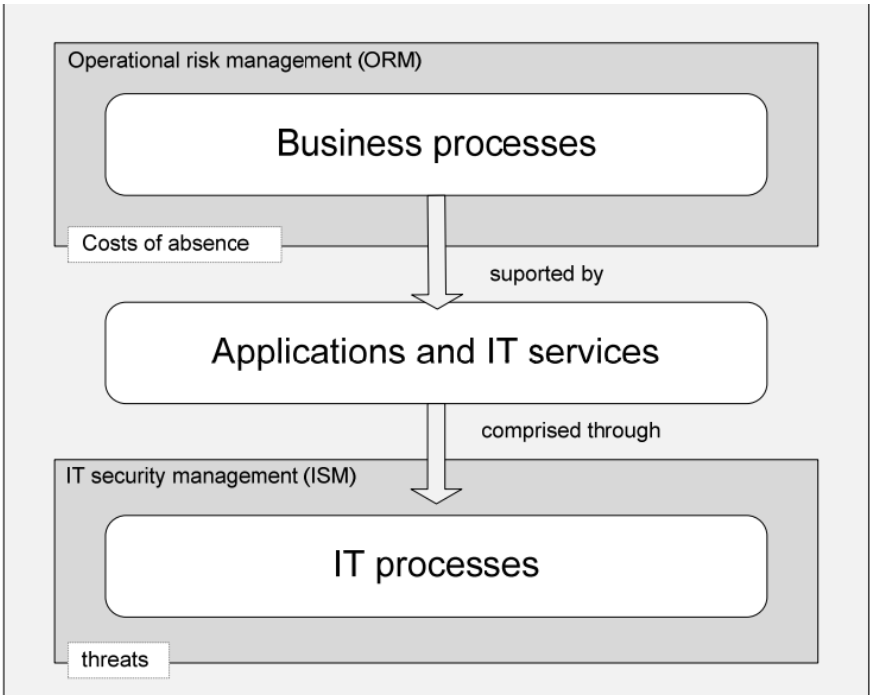

Figure 2-6 IT risk management (Grob et al., 2008)

In the IT risk-management estimation section of this study, I use the Markov Decision Process (MDP) as a risk-mitigation technique. SCP deployment is the process of identifying security vulnerabilities, assessing and installing the risks of those vulnerabilities, and fixing security issues (Dantu, Kolan, Akl, Loper 2007).

2.5 Maintenance Optimization Tool

In this section, I will mainly focus on three well-known researchers—Mazzuchi, van Noortwijk, and Kallen—who have done significant work on maintenance optimization. In this study, I integrate their MOT with networked computing patching processes. In their 2007 maintenance optimization (MO) paper, they define maintenance optimization as a "*problem of determining cost-optimal maintenance decisions for an object (system or structure or one of its components) to ensure safe and economic operation*" (pp. 1-5). The aim of MO is to design the maintenance process, ensure resolution of system failures, and minimize the total cost of maintenance activity (van Noortwijk, Dekker, Cooke, and Mazzuchi, 1992). According to Noortwijk et al., the following are requirements for MO:

- A section or unit of the system or subsystem with failure descriptions and impact models for preventive maintenance (PM) activities,
- Estimation of the costs of PM for system flaws and failures,
- Assessment of unit performance life, and
- An overall numerical methodology for the optimization activity.

The above principles of MO precede selection of the MOT, which can be fitted for the SCPMP and the development of optimized decision making. The Markov Decision Process (MDP) has been determined as a useful methodology for a PMP. The MDP determines an action that should be taken by decision makers at each stage of the project (Rajabi and Fotuhi, 2006).

The final result of applying the MOT in implementing the SCPMP is an optimal policy that manipulates and improves patch management policy in a more efficient way.

This policy is created in the context of IT costs and mitigation of network security risk. In addition, an optimized policy can be presented by transition and control between hidden states and process observations (Friston and Samothrakis, 2012).

2.6 IT Operational Cost Analysis

According to the United States Computer Emergency Readiness Team (US-CERT) trends report, the numbers of cyber-attacks have been increasing in 2010, 2011, and 2012 fiscal years (US-Cert 2012). Security Trends Report The cost of IT operations is growing for every business, institution, government agency, and individual user. Management of these costs is not only focused on high-tech and advanced network infrastructure, but also on the preservation of security. The need to make IT operations more secure is noticeably increasing; therefore, for ITDS is important to manage their operation in the most efficient way while preventing networked computing systems from outside attack. Cost estimation for operation of the IT infrastructure may be different for each department; however, overall investment in IT is increasing for cyber-security purposes (Mulligan and Schneider, 2011). Yet cyber security and vulnerability-prevention strategies can reduce the organizational budget. In addition, the organization will avoid the cost of fixing vulnerability issues and re-imaging compromised machines. Mitigation of IT cost by applying an SCP consists of:

- Time—the rebuilding or remediation time period;
- Resources such as technicians and technology
- Money for IT expenses and wages (Mell, Bergeron, and Henning, 2005).

Any organization incurs two types of PMP costs: (1) the cost of damage from exploitation and information leakage due to disclosure vulnerabilities and software package flaws, and (2) the cost of patch updates and identification, detection, testing, and deployment (Cavusoglu, Cavusoglu, and Zhang, 2008). The NIST publication on patch management (Mell et al., 2005) includes examples of the cost analysis of IT operations and the patching process; patching and network monitoring are two of the costs for patch management and prevention. If each organization has the same prevention and remediation cost, they will choose the prevention process, because it has a lower cost than security remediation. In the NIST example, Mell et al. (2005) assume that one security virus has been released, which promptly attacks the networked computing systems and compromises the organization's workstations. The authors used equations to analyze and estimate the cost of managing 800 networked computing systems. Using the NIST database and this example aids in understanding the work required, where we are now, and where we will be in the near future in terms of security vulnerability and understanding the patching process (Kuhn, Rossman, and Simon, 2009). Based on the NIST example (Mell et al., 2005), a breakdown of the cost analysis alternatives is provided:

1) The potential cost of the remediation process:

Cost not mitigated = Workstations * Fixing Time * Hourly rate

$$\boldsymbol{C = W * T * R} \qquad \text{(Mell et al., 2005)}$$

The time of the re-imaging process is about 2 hours. The staff loses machine function for 2 hours, which brings total downtime to 4 hours at the rate of $60/hour (the average wage of one IT technician plus one staff member is $30+$30=$60).

$\boldsymbol{W}$ = 800 Workstations

$\boldsymbol{T}$ = 4 hours

$\boldsymbol{R}$ = \$60/hours

$\boldsymbol{C_1}$ = 800*4*60

$\boldsymbol{C_1}$ = \$192,000

The total cost of operating the compromised machine can be as much as \$192,000 for one fiscal year.

2) Mell et al. (2005) assume the following for their prevention and monitoring case:

The average cost of one monitoring and patch central management server is \$10,000 per year. Therefore, we can derive that the cost of maintenance is \$10,000 for the IT department. The total deployment and installation patch time is about 15 minutes, which includes post-installation tasks such as rebooting; therefore, the patch installation time is a quarter of an hour:

$\boldsymbol{C_2}$ = 0.25 * 800 * \$60 + annual monitoring cost

$\boldsymbol{C_2}$ = \$12,000 + \$10,000

$\boldsymbol{C_2}$ = \$22,000 (Total cost of monitoring and installing one patch by using the management server center)

$\boldsymbol{C_2}$ is the total cost of installing one patch on 800 workstations (\$12,000) plus the additional cost for the monitoring network (\$10,000).

The comparison of ($\boldsymbol{C_1}$) and ($\boldsymbol{C_2}$), which are the total cost of remediation and managing compromised machines and the cost of prevention and maintenance, can determine the amount each organization can save in terms of IT operations. Patch management and security prevention process can mitigate the cost of IT operations and

protect networked computing systems from becoming compromised, which leads to critical system failures (Mell et al., 2005).

2.7 Patch Deployment Models and Frameworks

Recently, there have been many security and critical patch deployment models (PDM), frameworks, and research related to IT security management industries. Adams (2007) points out that the patch deployment process, like any other operation or service, requires staff, technology, and procedures. Adams also notes that the Information Technology Infrastructure Library (ITIL) and Microsoft Operation Framework (MOF) have been used by IT decision makers and security managers to develop and execute the PMP in their environments. Adams presents the typical model of patch management in his paper, which is shown in Figure 2-7:

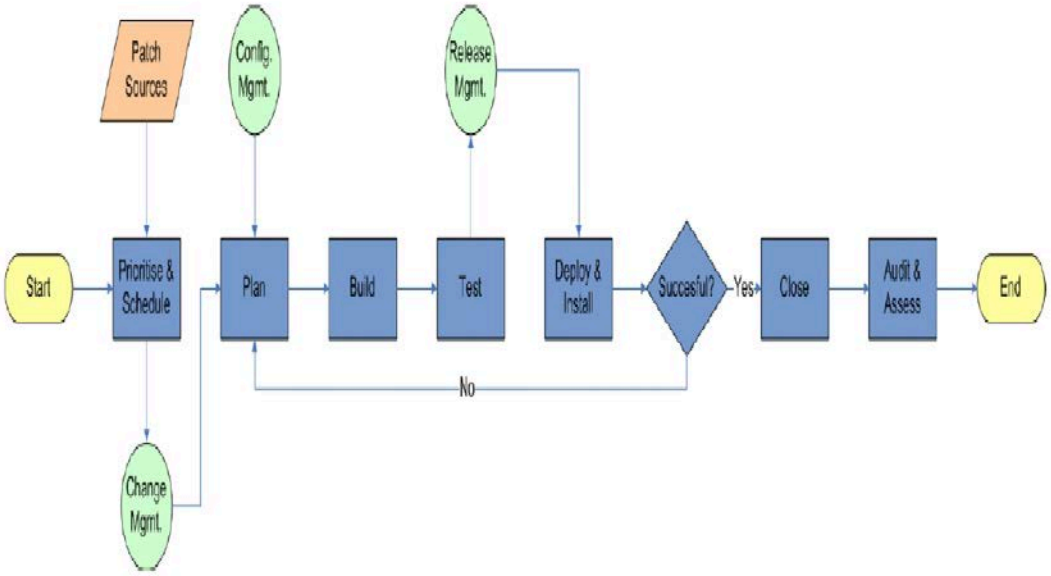

Figure 2-7 A generic model of patch management (Adams, 2007)

The anti-vulnerability SE framework is a combination of SE fundamentals with network security principles that focus on immunizing computing systems from cyber-attacks and exploitation of security vulnerabilities (Lin, Mao, and Xie, 2006). Patch management must be a consistent process. The aim of the proposed SE framework is to assist ITDS in running the patching process as a regular task. Consistency of the replicated patching process will enhance the level of network security (Lin, Hsieh, Hung, and Tsaih, 2008).

Lin He and colleagues (He, Gao, Chen, and Wang, 2011) highlight the importance of the patch management process, while exploring some aspects of the vulnerability process that should be receive particular attention:

1. The quality and authenticity of the patch, assuring that the released patch will fix the disclosure bug.

2. The remediation and fix of the vulnerability that is necessary to prevent the networked computing system from being attacked.
3. The relationship between patch installation and scheme results.
4. The rules regarding installation order and deployment protocol.
5. The factors involved in patch installation outcomes and expectations of deployment results, which may change some software package versions or OS settings.

2.7.1 Generic patch types:

As mentioned before, there are two major patch types in security network engineering: OS platform patches, which fix OS holes and vulnerabilities, and software application patches, which are responsible for fixing the flaws or updating to newer versions (increasing application performance and enhancing efficiency). The following list, demonstrating the number of patches that have been deployed in the SCPMP, is addressed in this research:

- Patches
    - Software
        - Adobe family
            - Acrobat
            - Reader
            - Flash Player
            - Air
        - Apple iOS embedded applications
            - iTunes
            - QuickTime
            - iLife
            - Safari
        - Microsoft applications
            - Office
                - Word
                - PowerPoint

        - Excel
        - Access
        - Outlook
    - Dot Net Framework
    - Silverlight
    - Internet Explorer
  - Symantec family of Norton
    - Symantec Endpoint (SEP)
  - Oracle (Microsystem's)
    - Java JRE
    - Java Environment
    - JDK
  - Trend Micro
  - Mozilla
    - Firefox
    - Thunderbird
  - McAfee Antivirus
  - Real player
- Operation Systems (OSs)
  - Windows Platform
    - Win 2K
    - Windows 2008 Server
      - Windows 2008 Server Standard R2
      - Windows 2008 Sever R2
    - Windows 2012 Server
    - Win XP
      - Service Packs (SP1 and SP2)
    - Win Vista
    - Windows 7 versions
      - Professional
      - Enterprise
      - Home edition
      - Business
  - Macintosh Platform
    - Jaguar 10.2
    - Panther 10.3
    - Tiger 10.4
    - Leopard 10.5
    - Snow Leopard 10.6
    - Lion 10.7
    - Mountain Lion 10.8

Operating system codes and software packages, which are an inherently human creation, may have coding errors; this will be fixed by patching and debugging the packages (Ramaswamy, Bratus, Smith, and Locasto, 2010). As the above list of generic patches demonstrates, for this study I use the most common software packages and OS in daily usage.

2.7.2 Scholarly patch deployment models and frameworks

I will now review three studies that are similar to mine; they were selected because of their significant impact on deployment patch processes:

2.7.2.1 Windows security patch management

In this model, Park et al. (2007) create an automatic process for deploying and installing security and critical Microsoft Windows patches on networked computing systems. This is done through automatic downloading in accordance with XML methodology. Figure 2-8 demonstrates the system architecture of their automatic Windows patch management process.

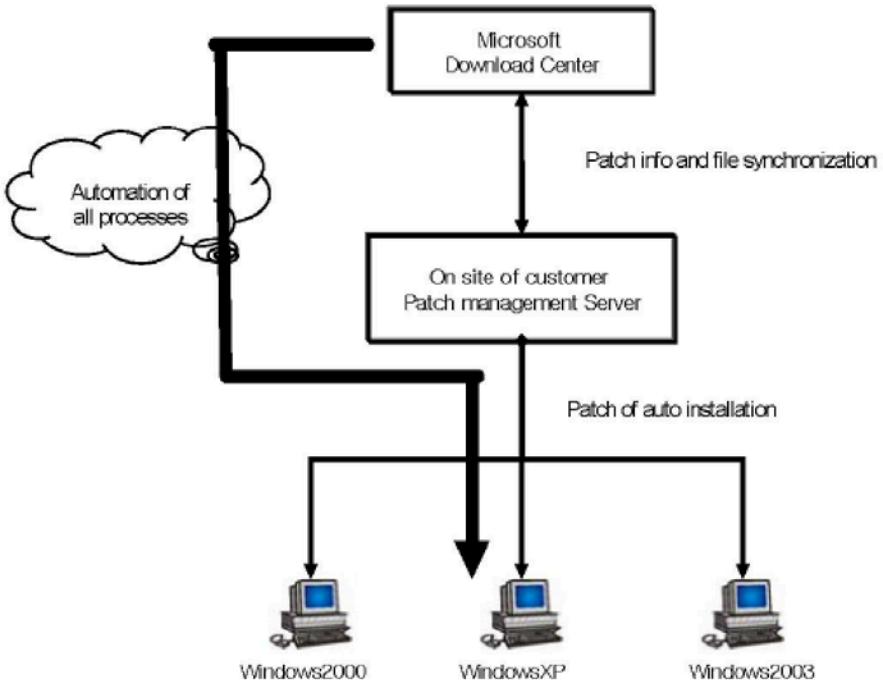

Figure 2-8 System architecture of an automatic Windows patch management process based on XML methods (Park et al., 2007)

Park et al. (2007) propose automated system management, which consists of the following three subsystems:

1. A subsystem for synchronizing with the Microsoft download center for updating patch files. This system was named the Soft Update Service (SUS).
2. A patch-management-server subsystem. This server not only manages and monitors the SUS, but also administers patch files and patch clients. This system generates an XML format patch information file by downloading patches from the SUS.

3. A patch agent is the last subsystem. Its most important functionality is installing patches on networked computing systems (clients).

2.7.2.2 SANS Patch Management Process

The second patch management model is the SANS model, adapted from the Global Information Assurance Certification Paper (GIAC), was developed by Maik Medzich (2004). He presents a practical PMP model, which is related to my methodology for implementing SCPMP in a diverse environment. In this model, which is presented in Figure 2-9, Medzich depicts the relationship between patch management processes and IT management. Without collaboration between the IT manager and the security patching department management staff, there will be a significant gap between protecting networked computing systems and managing IT assets. Medzich's paper attempts to close this gap in the proposed SE framework.

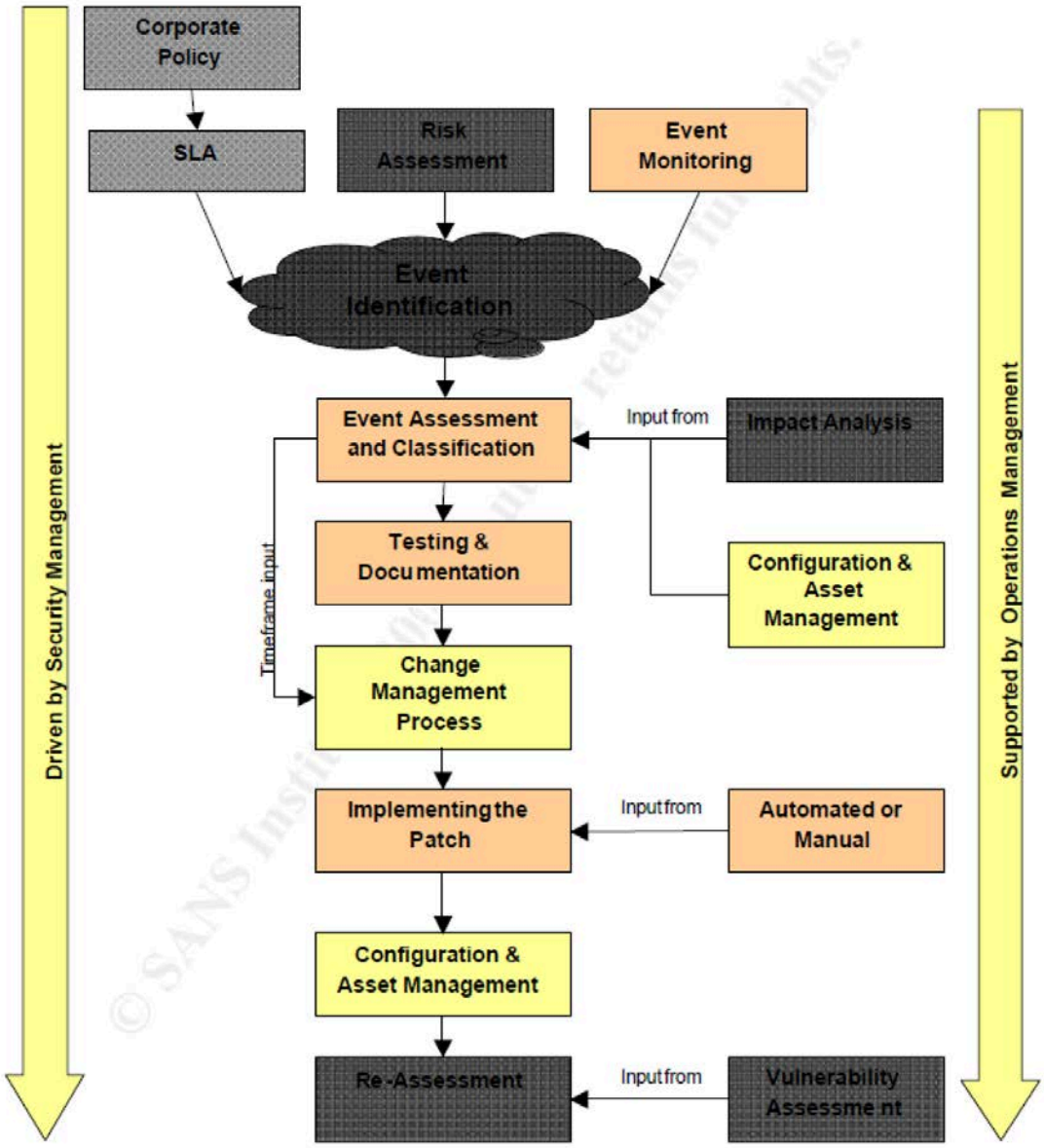

Figure 2-9 SANS's patch management process (Medzich, 2004)

As shown in Figure 2-9, the SANS patch management model is composed of the following core components, which are necessary to implement, run, and test with the patch management process (Medzich, 2004):

A. **The IT management framework** is retrieved from the ITIL framework. ITIL creates and offers practicable IT service management worldwide. It provides a comprehensive experiment for organizations and individuals. (itil-officialsite.com).

B. **Security Management**, a crucial part of every patch management process, is used to protect and control security perspectives and uses approaches similar to SANS patch management processes. Medzich (2004) states that security management is composed of important rules that must be obeyed by other parts of the administration responsible for patch rollout. Medzich identifies the core responsibilities of the security management role as follows:

a. Awareness of the latest vendors' patch security updates,
b. Conducting risk assessments,
c. Architecture and documents management,
d. Creating and establishing security policies,
e. Managing security escalation authorities,
f. Managing test, analysis, and implementation of patch deployment,
g. Evaluating testing processes, and
h. Accepting and approving overall patch deployment processes.

C. **Operations Management**, one element of patch management, has a close relationship with security management. The security staff assists operating staff in performing their core responsibilities. However, security staff have their own responsibilities, such as allocation of team members based on specialization in different platforms (Medzich, 2004).

D. **Asset Management** is responsible for managing IT asset databases. In order to solve security flaws, this management staff identifies the particular OS that needs to be patched or the specific software package that needs updating. IT inventory and asset management is the first step in IT service management. This department uses the patch management process to define and store system configurations to the database-management system (Medzich, 2004).

E. **Problem Management** and incident reports are why I chose the SANS patch management process in creating my SCPMP framework. The Urgent Security and Critical Patch (USCP) will be linked to security management components by incident and problem management. This means that all upcoming threats and malicious events will be covered, classified, defined, and recorded (Medzich, 2004). With my co-researchers, I have addressed problem and incident management within an SE framework and described it in the maintenance optimization phase.

2.7.2.3 Cooperative software maintenance model

The last model of the patch management process that has been considered in this study, which is shown in Figure 2-10, is the software patch maintenance database. This includes managing information for IT security and staff and codifying expert knowledge within ITDS (Gupta and Qureshi, 2011).

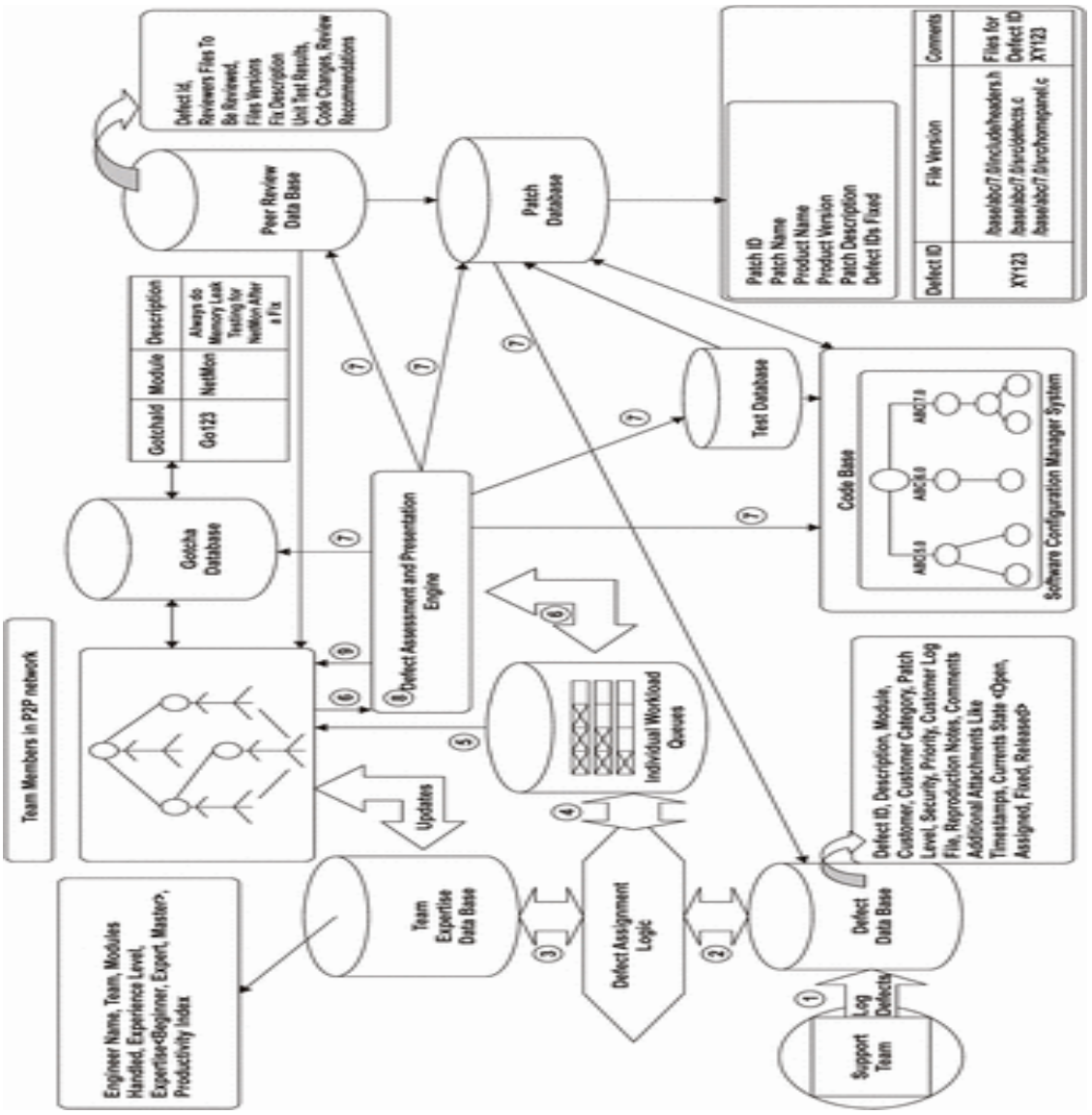

Figure 2-10 A cooperative software maintenance framework (Gupta et al., 2011)

This software maintenance model (Gupta and Qureshi, 2011), which includes the following core components for maintaining software packages, will be briefly explained.

A. **Team expertise or ITDS database**: This component is responsible for storing and collecting information based on IT staff members' and technicians' activities and performance.

B. **Defect database**: This database consists of customer experiments, as well as IT members' knowledge.

C. **Peer-to-peer communication subsystem**: The P2P is the sharing system that can be accessed by the entire IT security management department to enhance members' knowledge. The P2P communication system is an advanced communication and collaboration system that gives important opportunities to ITDS. For instance, all IT department staff will be able to post questions, which allow patch information and files to be shared with colleagues.

D. **Test database**: This database contains the test experiments necessary to understand and analyze test results.

E. **Patch database**: This database stores all patch file documents and information from maintenance teams.

This research has been considered the cooperative software maintenance model to cover industrial purpose of the PMP model and demonstrate the patching process for software package's remediation.

2.8 Literature review summary

In this chapter I covered three well-known patch management processes for implementing and maintaining patch deployment systems. All of the patch management frameworks and processes discussed above have been considered in building the proposed SE framework. Based on the SE Vee Model presented in Figure 2-11(Forsberg and Mooz, 1991; INCOSE, 2007), I have compared patch management processes based on SE principles reviewed in the literature.

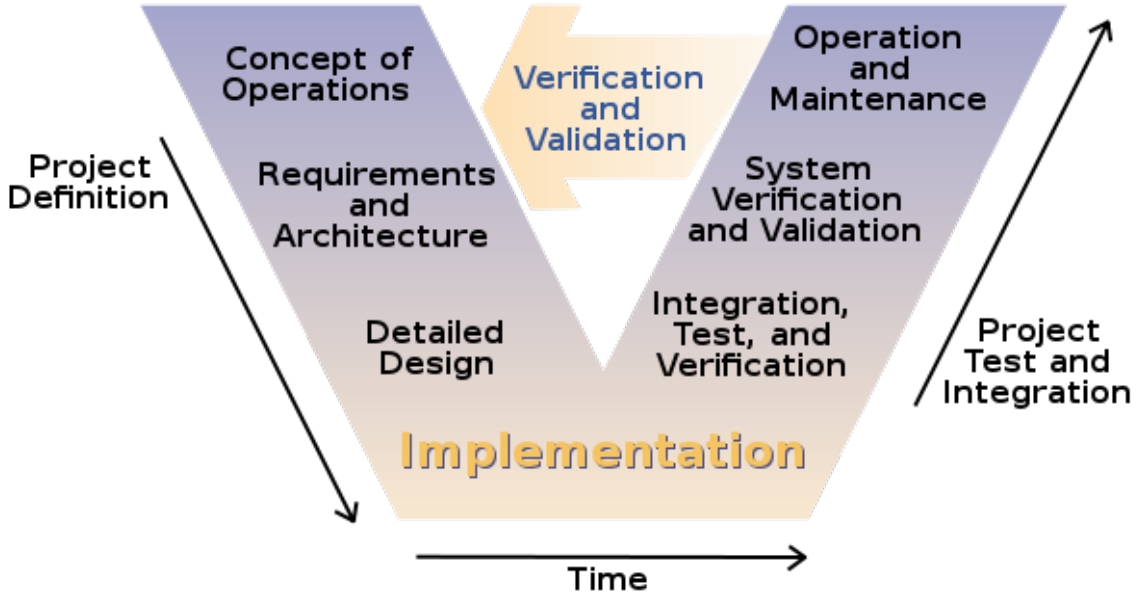

Figure 2-11 Systems engineering Vee Model (Forsberg and Mooz, 1991)

Table 2-5 Patch management process comparison based on SE principles

Literatures and scholarly research / SE fundamentals for implementing SCPMP	Windows Security patch management (Jin Park, Sub Park, Lee, Kim, Boon Lee, Cho 2007)	SANS Patch Management Process (Medzich 2004)	Coorperative Software maintenance model (Gupta and Qureshi 2011)
Requirements (Download patches and detection)	X	X	X
Design (Patch assessment)	X		X
Implementation (Test and deployment)	X	X	
Verification and Validation		X	X
Maintenance		X	
Storage and Database management	X	X	X
Risk Assessment		X	

Table 2-5 demonstrates the PMP principles highlighted in previous patch management models based on SE fundamentals. In the next chapter, I apply the principles that have been derived from the SE Vee Model to implement SCPMP and construct a comprehensive SE framework.

Chapter 3 — Research Framework

The main objectives are to present, test, and implement an experimental security and critical patch process by using SE principles on networked computing systems in academic and diverse industrial environments. This will maximize the efficiency and effectiveness of SCPMP to mitigate the risk of security attacks and not only reduce the cost of IT operations, but also avoid leakage of critical data and security information. In this chapter I propose a conceptual framework for implementing SCPMP in academic settings, such as in faculty, staff, student, and laboratory workstations. I then conduct a cost estimation of IT operational performance based on SE principles. In the literature review I demonstrated that patch deployment management is a core element of cyber security and overall IT department operations, yet this process must ideally be modified to protect more departmental workstations and increase supplemental security in the environment.

Patch deployment is a circular process and must be consistent. PMP focuses on both patch deployment design and performing the completed cycle (Chang, Tsai, and Tsai, 2005). A generic development model has been considered by the IT department, which is a circular patch management process. With this method, there are five critical steps that must be taken to implement one complete cycle of the patching process. Figure 3-12 illustrates the five steps of generic circular patch deployment.

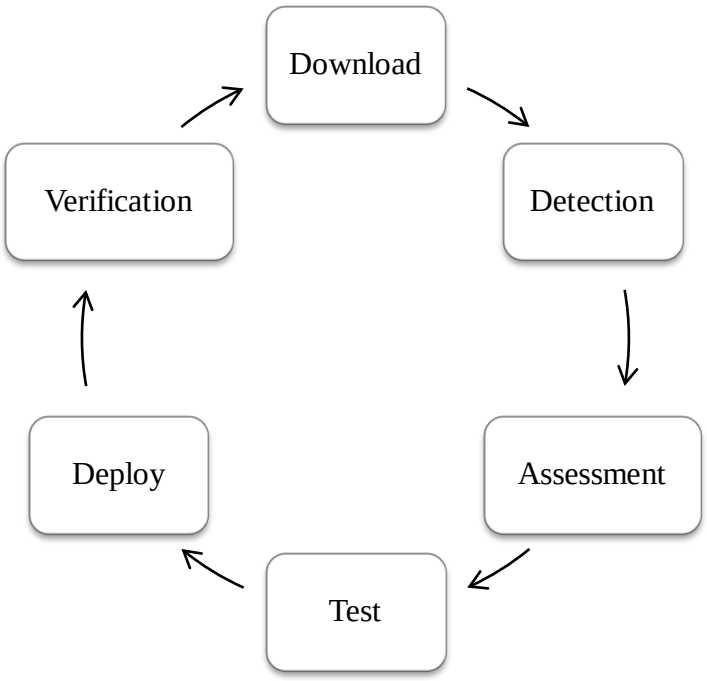

Figure 3-12 Circular patch management method

3.1 Download

The first step in the patch installation process is the downloading the vendor's patches. Based on subscriptions with different vendors, patches will be stored in a patch depository. There are several ways to download patches or use organizational server management that will cover in the next chapter. The whole process should be performed by ITDS in accordance with their policies in terms of subscription agreements. This will allow for confirmation of the integrity of patch sources and loading in patch-server management or external shared storage.

3.2 Detection

Detection is the identification and verification process that should be run across the entire network to regulate not only which patch has been installed, but also which patch needs to be installed. Detection and vulnerability scans are a staple of patch management (Lin, Chin, and Laih, 2008). In order to detect the workstations, ITDS use network server management monitoring tools to scan networked computing systems for either missing security patches or to verify the status of patches that have been installed on clients' machines. The detection process must be automated to ensure that IT security department staff covers the entire networked computing system. Thus, running and performing identification processes on patches across the network is critical for determining which machine is missing the latest security and critical patches.

3.3 Test

Ideally, the expanse and strategy of the ITDS patch management process in the testing phase will be directly related to the complexity of the environment, while also depending on critical data and information. For instance, the number of supported OS and software applications can affect the patching strategy. ITDS will use this strategy to patch machines and classify the data and level of network security in their environment. The test phase inn patch deployment management is the process of verifying patch installment results on beta test machines and comparing those results with the subsequent analysis of production workstations. This ensures that no unexpected results will occur during actual deployment. Testing is necessary for the patching process (Cavusoglu et al., 2008).

Perfect test environments or beta machines should be mirrors of production workstations, and the configuration should be as close as possible (Chan, 2004). In the area of patch management, several researchers have addressed the testing and design of test environment workstations. Testing infrastructure or test environment workstations is one of the important pieces of the patch management strategy. Therefore, the testing phase will allow ITDS not only to recognize and predict the consequences of installation, but also to ensure that installing patches (OS or software applications) will not disrupt any computational operations before the patches are widely deployed.

3.4 Risk Analysis

The proposed SE framework adds one more step to the current patch management process. The proposed step will enhance patch management efficiency and decrease the cost of IT operations. Risk analysis uses MDP tools and proposes a decision-making methodology with which IT managers will choose the most efficient method for deployment rollout. Also, the risk analysis phase, called the maintenance optimization phase (MOP), will be fully described in the next chapter. Figure 3-13 illustrates the modified model of circular patch management deployment.

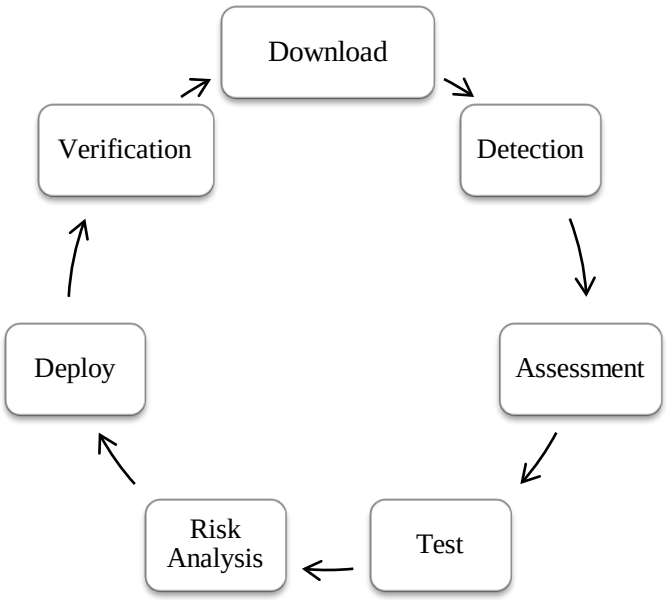

Figure 3-13 Modified model of circular deployment of the patch management process

3.5 Deployment and Installation

After MOP and the decision phase, which have been performed in accordance with the results of the MDP tool output, patches are deployed and installed on production network computing systems. This is the most important phase of PMP; it affects the organization's technical performance and determines when to deploy and install patches. In the past, many ITDS have created scripting language for a custom patching process that incorporates OS tools to apply and install in their environments (Chan, 2004). As network security has grown and ITDS have become more concerned about PMP, more management server centers are available to deploy and install patches automatically. These management servers classify OS and software application patches in different groups for validation and verification. IT security administrators then use the tools for installation.

3.6 Verification

The verification phase ensures that:

- Target systems that have been identified as the focus of security attacks have been patched to fix their vulnerabilities. Generally, the verification phase ensures that deployed patches have been installed on target workstations according to the patch deployment plan.
- Systems that have been patched are performing as before and that patching has not interrupted system performance (Chan, 2004).

3.7 Selection of Hypotheses and Sub-Hypotheses

The main goals of this study are to demonstrate that by using the proposed SCPMP framework, in academic and diverse industrial environments, will (1) mitigate the risk of managing a compromised machine, (2) reduce the cost of IT day-to-day operations, (3) and improve the efficiency of the current PMP in IT departments. To accomplish these goals, an experimental case study will be presented to analyze the cost of implementing SCPMP in an academic environment. The following hypotheses will be addressed in this work:

- First, the proposed SE framework can improve the current PMP and this framework will lower the presumptive risk. While there might still be risk of security attacks, but the “known” risk will be lower.

- Second, the proposed SE framework to implement the SCPMP in a diverse networked computing system environment will reduce the costs incurred by ITDS for running the patching process.
- Third, the patch deployment structure will efficiently help managers to allocate the PMP task to those who will implement and finish the deployment process.

3.8 Overall SCPMP structure and framework

As discussed in Chapter 1, this study shows the viability of an SE approach to maximize the protection of networked computing systems and prevent them from becoming compromised or vulnerable to security attacks. Previously my colleagues and I (Mohammadi et al., 2013) created a conceptual SCPMP framework to understand the overall structure, deployment, and installation of OS and software patches on networked computing systems by using a management server center.

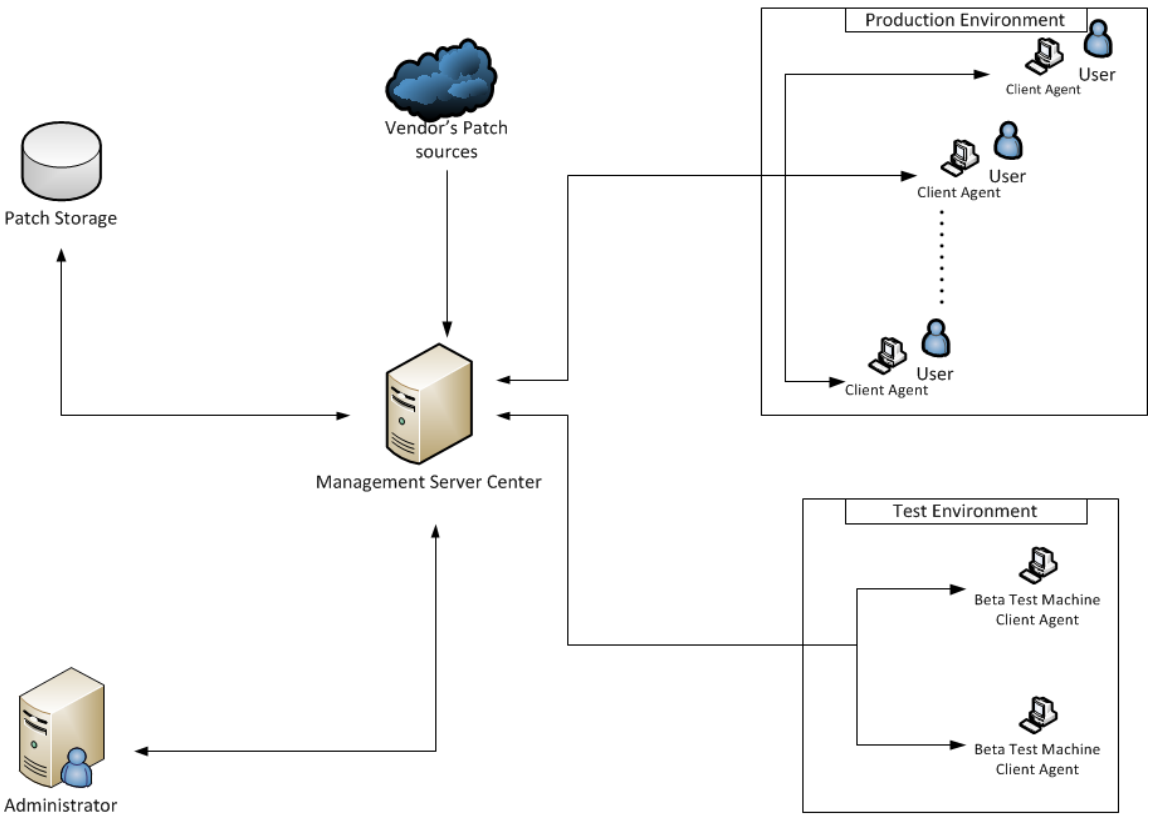

Figure 3-14 SCPMP overall structure (Mohammadi et al., 2013)

We specify three initial requirements for the SCPMP:

3.8.1 Management server center

The core part of this system, as seen in Figure 3-14, is the management server center (MSC), which performs multiple tasks. It generates reports about detection and deployment tasks that have been completed.

3.8.2 Patch storage

Patch storage can be an external or internal depository and is a necessary component for storing and verifying vendors' patches and utilization by the MSC. The majority of MSC servers have their own hard drive storage. However, based on environmental security, it may be separate from the MSC. There are two types of patches:

new versions of software installation patches and security patches that repair software or OS vulnerabilities.

3.8.3 Client agent

One of the important functionalities of the client agent is to install patches that have been deployed by the MSC on clients' workstations (Park, Park, Lee, Kim, Lee, and Cho, 2007). The client agent, which is a developed software base with communication skills (synchronization) to interact with the MSC, has the following capabilities:

- Scanning internal system configurations and components,
- Generating custom or completed hardware and software reports,
- Detecting installed patches and unpatched software, and
- Installing patches or scripts sent by MSC.

This software can be installed on supported platforms such Mac OS and Microsoft Windows. The agent will install as an administrator privilege in the top level of the OS file system and run as the root account to execute tasks that originate from the MSC. In the PMP, the client agent must be installed on the following networked computing systems:

3.8.3.1 Beta tester

It is recommended that some beta test machines have the same configuration as the production machine to ensure that the deployment and installation of patches will be manageable and has been executed previously (Mohammdi et al., 2013). Moreover, in order to verify installation levels and deployment results—such as the number of booting processes required for the patch package to be deployed and the time required to install

those patches—a test environment is mandatory. This will avoid further issues or predict the consequences. The perfect test networked computing systems should mirror the production computing system environment as closely as conceivable (Liu, Kuhn, and Rossman, 2009).

3.8.3.2 Production environment

Production machines or environment includes the staff, faculty, laboratory, and student machines that contain critical information and secure data that are in daily use. These production machines are the target of patch deployment and installation processes and must be protected from security vulnerabilities and being compromised.

3.9 Research Framework Summary

The research framework is based on general patch management procedures developed by several researchers and prior quantitative observations of SCPMP. The earlier example of the patch deployment model led to the launch of the SCPMP framework by adding the advantages of SE principles. I will use MDP equations to analyze and calculate the cost of making decisions in accordance with the patch deployment strategy. To achieve the goals of my research, I will use the proposed SE framework to implement SCPMP in an academic environment based on Markov Decision Process tools. Research methodology and its application will be described in the next chapter. The core steps in this research framework are:

- Propose an SE framework to implement security and critical patch management processes

- Use the Markov Decision Process to compute and calculate the maintenance optimization phase, which is embedded in the proposed SE framework
- Validate and verify the MDP model
- Assess an experimental case study,

Chapter 4 — Research Methodology

This chapter focuses on the proposed methodology and classifies the steps of PMP to execute the proposed SE framework. I will use experimental systems and security vulnerability engineering to explore an effective way to implement the SCPMP, mitigate the risk of administering compromised machines, and decrease the cost of IT operations. The proposed methodology will examine the PMP to create a security policy and explore an efficient design for the deployment process.

The proposed SCPMP framework's concepts and objectives, which are categorized below, demonstrate that SE fundamentals are embedded in the framework and detail the practical reasons for implementing the SCPMP on networked computing systems:

1. To create an optimal PVM policy for preventing workstations from vulnerability and security attacks,
2. To fix software package flaws that are at risk of being hacked and leave networked computing systems open to being compromised,
3. To repair security holes in OSs or software applications,
4. To make IT operational activities more efficient,
5. To reduce not only the cost of SCPMP, which is part of IT operational costs, but also mitigate the risk of security attacks, and
6. To protect critical information and security data, which are saved on any networked computing system (Mohammadi et al., 2013).

In previous work (Mohammadi et al., 2013), we detailed the importance of implementing SCPMP in any networked computing system. Figure 4-15 presents our research framework.

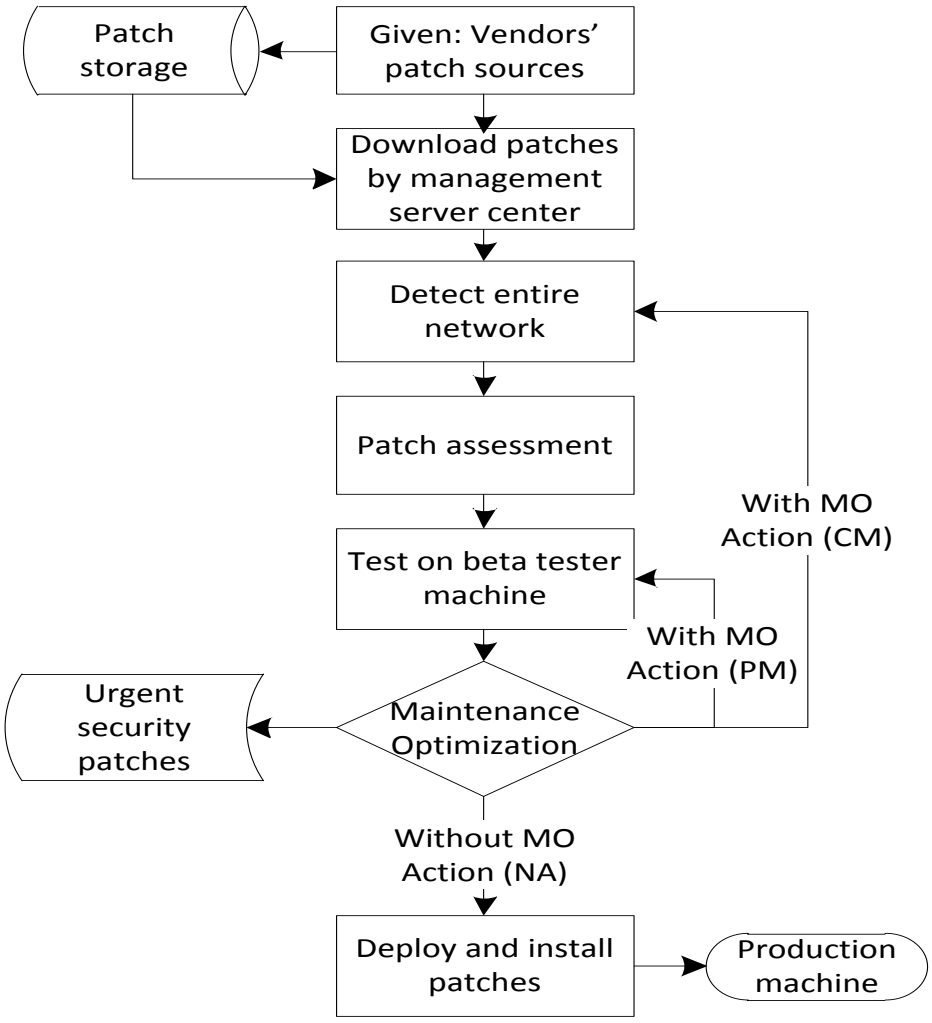

Figure 4-15 SCPMP framework (Mohammadi et al., 2013)

Implementing SCPMP in diverse environments should follow the steps depicted in Figure 4-15. One decision-making phase will occur if an Urgent Security Patch (USP) is released to fix OS or software package vulnerabilities. This decision-making station,

which I will call the maintenance optimization phase (MOP), will help IT decision makers and security and critical patch deployment department technicians to determine what steps must be taken to protect their environment from being compromised or hacked.

4.1 Maintenance optimization stage

Pandey, van Noortwijk, and Klatter (2006) define maintenance as "a combination of actions carried out to restore a structure to a specified condition in which the structure can perform its required functions." Hence, the MOP aims to reach a cost-optimal decision for an economic operation by ensuring that the system's performance is reliable and secure (Mazzuchi et al., 2007). Embedding a maintenance optimization tool in implementing the SCPMP will (1) decrease the cost of security patch deployment processes, and (2) make network computing systems more secure. It is recommended that a MOT be integrated into the implementation of SCPMP, since it can help make confidential decisions in terms of conducting more efficient patch deployment processes.

Embedding a MOT is a probabilistic process for making decisions regarding receiving urgent security patches. This study's methodology for implementing patch management processes is to follow the proposed SE framework and use the MOT to make a decision when the USP has been released. The subsequent MOT phase will aid IT decision makers in deciding whether to continue with the patch deployment process step that has already been started and tested; they may choose to block the process's cycle and jump to the detection stage so as to include urgent patches for verification and validation.

4.2 Modeling the Markov Decision Process

Ross (1970) studied an MDP as an application for modeling probability activities and defined the MDP as a broad tool that includes many optimization cases. As mentioned above, I use the MDP and the Markov formula in this research in the maintenance decision-making phase. To model the MDP for the proposed SE framework, I drew from four prior studies to construct a proper formula and equation based on IT department requirements: (1) Mazzuchi et al. (2007); (2) Amari, McLaughlin, and Pham (2006); (3) Ross (1970); and (4) Mohammadi et al. (2013). The main goal for modeling the MDP based on implementing an SCPMP in diverse environments is to create the best policy for IT decision makers to use when they are deciding which patch deployment process better fits their networked computing system environment and meets their security requirements.

4.2.1 Inputs

The following steps for constructing the MO stage principles are based on Mohammadi et al. (2013):

4.2.1.1 Patch Deterioration Stages

The Markov Chain has been used to model the "multi-stage" of deterioration. In the patch management process, the type of patch (OS or software applications) determines the level of deterioration. The reason that I use the Markov Chain as a deterioration model is because all patches in the model go in one-direction, where the first stage is no action and last is a fail stage (Amari et al., 2006). In this research, there are four types of patches or deterioration stages:

$$S = \{Sn, Sv, Sr, Su\}$$

These are characterized as the type of patch that can be released, as shown in Figure 4-16:

1. S_n, where no urgent security patches are released;
2. S_v, where a new version of a software application that adds new features to the current version or optional patches for the installed platform is released;
3. S_r, where recommended and important patches are released; and
4. S_u where urgent security and critical patches have been released to address three critical vulnerabilities (Mohammadi et al., 2013).

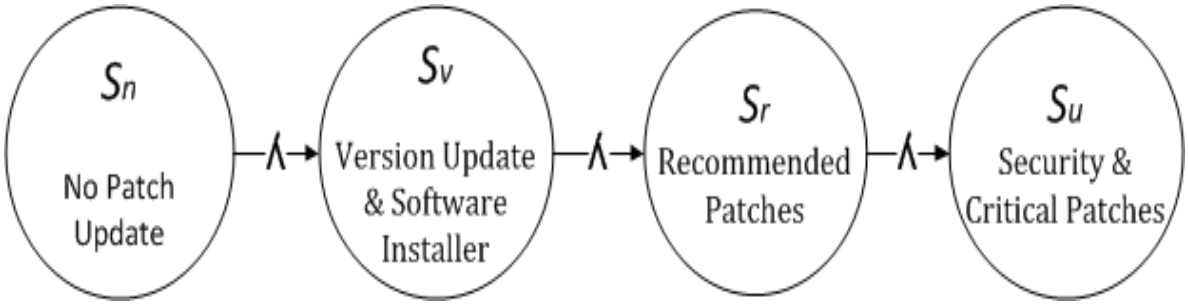

Figure 4-16 Patch type deterioration stages (Mohammadi et al., 2013)

Figure 4-16 illustrates all states of existing patch or deterioration types that can be found in the midst of implementing the SCPMP cycle. A stationary process has been chosen when a decision whether to take action depends on the patch type and the stationary process and may not rely on maintenance decision time (Mazzuchi et al., 2007).

4.2.1.2 Maintenance Actions (MA)

IT decision makers should make decisions based on the circumstances they face. Therefore, there are four Maintenance Actions (MA) that must be taken based on patch deterioration types:

$$\boldsymbol{A = \{a = NA, MM, PM, CM\}}$$

As shown in Figure 4-17 and defined by the above equation, the existing actions for the proposed SE framework are:

- Where no decision has to be made and the ITDS should proceed with the patch deployment cycle without any MO action, the No Action (NA) is performed.
- Where ITDS adds the Urgent Security and Critical Patches (USCP) to the current patch deployment packages and no detection phase will be taken, the Minimal Maintenance (MM) must be performed.
- Where ITDS rolls back to the test stage to deploy and install the USCP on beta test machines and then proceeds with the process, the Preventive Maintenance (PM) action is performed.

- Where ITDS adds the USCP and rolls back to the detection stage on the proposed PMP methodology and includes it in the previous patch package, the Corrective Maintenance (CM) action must be performed as the cycle continues.

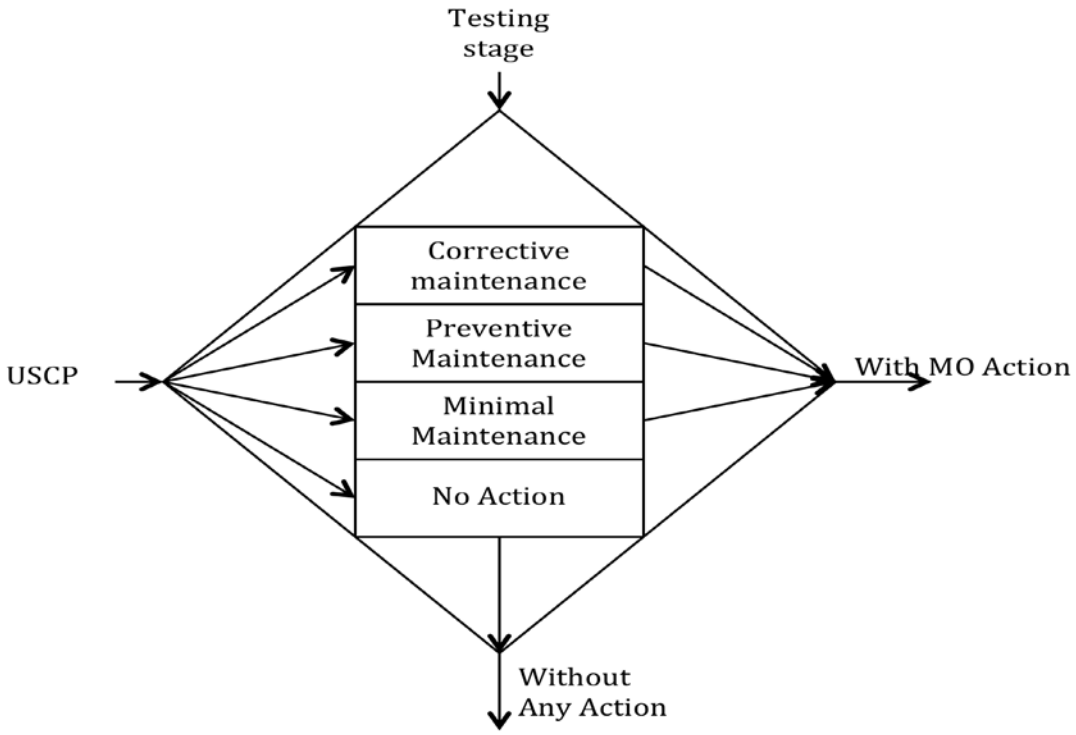

Figure 4-17 Maintenance decision actions (Mohammadi et al., 2013)

4.2.1.3 Cost Estimation

As discussed in Chapter 2, there are two important cost estimations for IT operation departments, based on prior research and the literature. A hypothetical patch or deterioration process, ***P(t)***, is a stochastic process that will perform in overtime and go through each decision stage of the maintenance optimization phase on the proposed

SCPMP framework, as is shown in Figure 4-18. Here, ***P(t)*** begins at approximately zero and proceeds toward the CM stage, which is the failure threshold. The maximum damages for networked computing systems and the IT environment will occur if ***P(t)*** passes the CM stage and reaches the failure threshold.

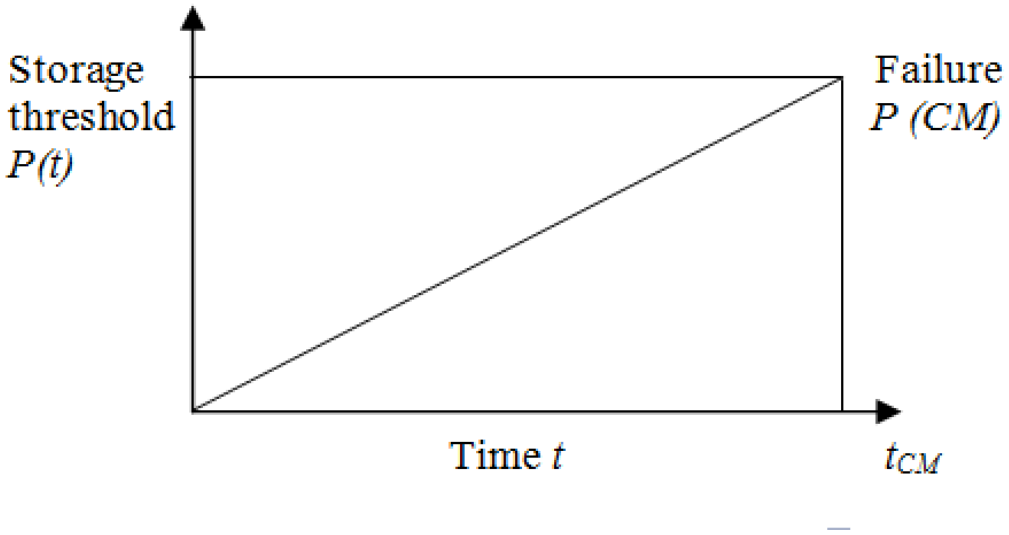

Figure 4-18 Deterioration process (Mohammadi et al., 2013)

Therefore, the total costs of implementing the SCPMP in any environment by ITDS will rely on the MO stage process, which means that if there is no USCP release and ITDS follow the regular the PMP cycle, a change could occur once the USCP is released. The logical concepts regarding cost formation can be denoted by:

- C_K: The fixed and regular cost of the SCPMP, which includes management server maintenance cost—i.e., the day-to-day cost of IT department operations without any USCP;
- $C_A(a)$: The cost for deployment of patches and maintenance actions;

- $C_F(s)$: The cost of compromised networked computing systems and security attacks per unit time, which lead to security system failures or leakage of critical data; and
- $C_S(s)$: the cost per unit time for each state S.

The main challenge in implementing the SCPMP is to determine which patching policy should be used to enhance process efficiency, mitigate the risk of attack, decrease the cost of operations, and avoid leaking critical data and information.

4.2.1.4 Equations and optimizing

The main goal of this subsection is to create a formula based on prior research and literature to optimize IT operations and determine the best policy for the patching process. The iterative policy has been developed by using MATLAB© software simulations in this research (Cros, 2012) to determine and establish an efficient patch deployment policy. As in previous research (Mohammadi et al., 2013), I assume that each state, time, and action is discrete when building the formulation, which is denoted by the transition from state S to state S' is Λ and A. A set of possible MA has occurred if:

(4-1) $$a \in A, size(A_a) = 1$$ (Ross, 1970)

When one of the patch types in released in state S at time t=0,1,2… MO $a \in A$ should be made. This occurs after one unit of time in the patch deployment process in stage S' with probability:

(4-2) $P(s'|s,a) = \mathbf{Pr}[X_{t+1} = s'|X_t = s, a_t = a]$ (Ross, 1970; Mazzuchi, et al., 2007)

The equation (4-3) estimates the time that will be taken in state S' when the prior state is S with a action taken,

(4-3) $T(s'|s,a) = E[T_{t+1}|X_t = a, a_t = a, X_{t+1} = s']$ (Amari et al., 2006).

To estimate the expected cost $C(s,a)$ of implementing a security and critical patch management process based on previous work (Mohammadi et al., 2013), when the patch deployment process is in state S and a action has been taken:

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

The probability of migration from one patch state to another state in MDP relies on the type(s) of USCP that has been released and does not depend on the time performance and maintenance action taken (Mazzuchi et al., 2007). Therefore, the value function of "*expected objective value obtained from the discounted cost incurred*" is denoted by:

(4-5) $V(s) = C(s,a) + \sum_{s,s' \in S} P(s'|s,a) \,.\, á. V(s')$ (Ross, 1970; Mazzuchi et al., 2007).

` The assumption of a discrete and optimal policy is equally important, and the proposed formulations address maintenance optimization by applying MDP to each patch deployment transformation. Where $\alpha \in (0,1)$ is a discounted factor for one unit of time and equal to

(4-6) $$\alpha = \frac{1}{1+\lambda}$$ (Mazzuchi et al., 2007),

this will help to minimize the expected discounted cost for one unit of time. $\lambda_{s,s'}$ is the rate of transition time and migration from state S to state S', and $V(s)$ is the value function using α. Equation (4-5) is calculated to result in a better maintenance policy and optimal solution (Mazzuchi et al., 2007). Since the ITDS is performing PMP, once the USCP is released, a maintenance action must be taken in state S and the maintenance cost

of ***C(S,a)***. The total probability of transition from state ***S*** to state ***S′*** with proper maintenance action equals $\boldsymbol{P}(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})$. In addition, the transition probability will multiply to a discounted factor and value function of state $\boldsymbol{V}(\boldsymbol{s'})$.

As mentioned before, this study minimizes the cost value of IT operations and maximizes the performance of SCPMP in a diverse environment. An optimized policy, which will be derived from the equation (4-5) of ***V***, ***C***, ***P***, and ***T*** functions, can be written on a matrix-vector. Equation (4-7) is derived from the Bellman formula for minimization of the expected cost (Bellman, 1952),

(4-7) $\boldsymbol{\pi}(\boldsymbol{s}) = \arg \boldsymbol{min}_{\boldsymbol{a}}\{\boldsymbol{C}(\boldsymbol{s},\boldsymbol{a}) + \sum_{\boldsymbol{s},\boldsymbol{s'}\in\boldsymbol{S}} \boldsymbol{P}(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a}) . \acute{a} . \boldsymbol{V}(\boldsymbol{s'})\}$ (Bellman, 1952; Amari et al., 2006).

I define the iterative policy as setting an optimal policy $\boldsymbol{\pi}(\boldsymbol{s})$ by re-evaluating the value function ***V*** and minimizing the expected cost ***C(s,a)***. Policy iteration proves that the extracted policy reduces the IT operation cost for implementing SCPMP (Amari et al., 2006). By way of explanation, an extracted optimal policy$\boldsymbol{\pi}(\boldsymbol{s}) = \boldsymbol{a}$ has been chosen based on the decision that was made during the MO stage. As mentioned before, in the MO stage, a decision has been made on the maintenance action and the estimation of the patching cost process.

4.3 Research Methodology Summary

The case study in Chapter 5 will use an experimental approach and data analysis to execute the SE framework and test implementing the SCPMP in a diverse environment to measure the direction in which the proposed SE framework aims ITDS. It will create

an optimal policy to mitigate PMP risk and decrease IT operation costs. The MDP and its related formulas will be computed to test security data, which will be drawn from a target test environment, to prove that the offering policy will optimize the target test environment of the SCPMP.

Chapter 5 — Case Study

The proposed SE framework for implementing the SCPMP will be demonstrated through a case study in an academic networked computer system and its effectiveness tested. One reason I chose an academic environment is because of its similarity to a diverse network computing system's configurations and varied uses. For instance, in the School of Engineering and Applied Science there are eight departments that make numerous demands on software applications and OSs. In the literature review and conceptual exploration stage of implementing the SCPMP process, I considered NIST's practical example of patch and determined that patch management is critical step for any IT environment. In the chapter on the research framework chapter I proposed a practical SE framework to implement a security and critical patch management process, which I will use in this case study.

One of the main goals of this research is to illustrate how ITDS can mitigate the risk of being hacked and reduce IT costs. In addition, the study demonstrates that implementing the SCPMP and mitigating security vulnerabilities are not straightforward and simple procedures that follow a standard process for decision making (Mell et al., 2005). Instead, this process requires consideration of all possible circumstances that could be involved when deploying and applying patches to fix security holes. On the other hand, patch management is a systematic process in which a well-defined procedure based on the environment must be applied. In this example, based on a networked computing system configuration, one must determine a specific patch management process to enhance an academic department's network security and reduce operational

interruptions to protect the research, laboratory, faculty, and staff workstations from vulnerability and security attacks.

5.1 Target Environment and Networked Computing Systems

The intent of this sub-section is to describe the study's target environment and networked computing systems and explain how data will be analyzed. As mentioned previously, the workstations of academic networked computing systems are located in different departments. These computing systems are connected by 100mbps and 1000mbps. Networked connectivity is important for this case study, because all workstations, such as laptops, desktops, and severs, will be managed by a management server center. The management server center agent is installed on all networked computing systems and monitors their processes and recognizes their inventory of system configurations. The client agent and test environments were described in the research framework chapter. This illustrates the importance of the test computing system configuration, which should be matched with production workstations in order to anticipate the same consequences of both testing and actual deployment processes.

School of Engineering and Applied Science (SEAS) workstations are the focus of this case study. SEAS's different departments have their own required software applications and various OSs. However, all departments are connected to the core server, which manages their connectivity and applies patches through the management server center.

5.2 Data collection

This is divided into three major subsections. The first subsection introduces the management server center, OS, and a list of different OSs that will be defined in the second subsection. In the last subsection, I will list the software packages that are the targets of the patching process and the patches that must be applied to implement the SCPMP.

5.2.1 Management Server Center

The management sever center not only used an applied SCPMP performance, but also monitors the network and downloads patches from Lumension (lumension.com). This is according to the subscriptions and download options that ITDS chose based on the networked computing system's target environment. This management center is the Kbox 1000 series (Dell KACE K1000 System Management Appliance Version 5.4, kace.com), which is supported "on machines running Windows and Macintosh operating systems only" (Dell KACE K1000 System Management Appliance Version 5.4, kace.com). Figure 5-19 illustrates the patch management workflow based on the assigned schedule for detecting and deploying patches (Dell KACE K1000 System Management Appliance Version 5.4, kace.com). The PMP based on the Dell Kace methodology, which is shown in Figure 5-19, should take the following steps:

- Download patches based on the relevant patch subscription
- Apply labels to categorize the patches
- Apply labels to make groups for target networked computing systems
- Detect the scheduled process

- Deploy the process (Dell KACE K1000 System Management Appliance Version 5.4, kace.com).

As mentioned before, there is one standard patch management process that has been considered by the Kace company for its management center.

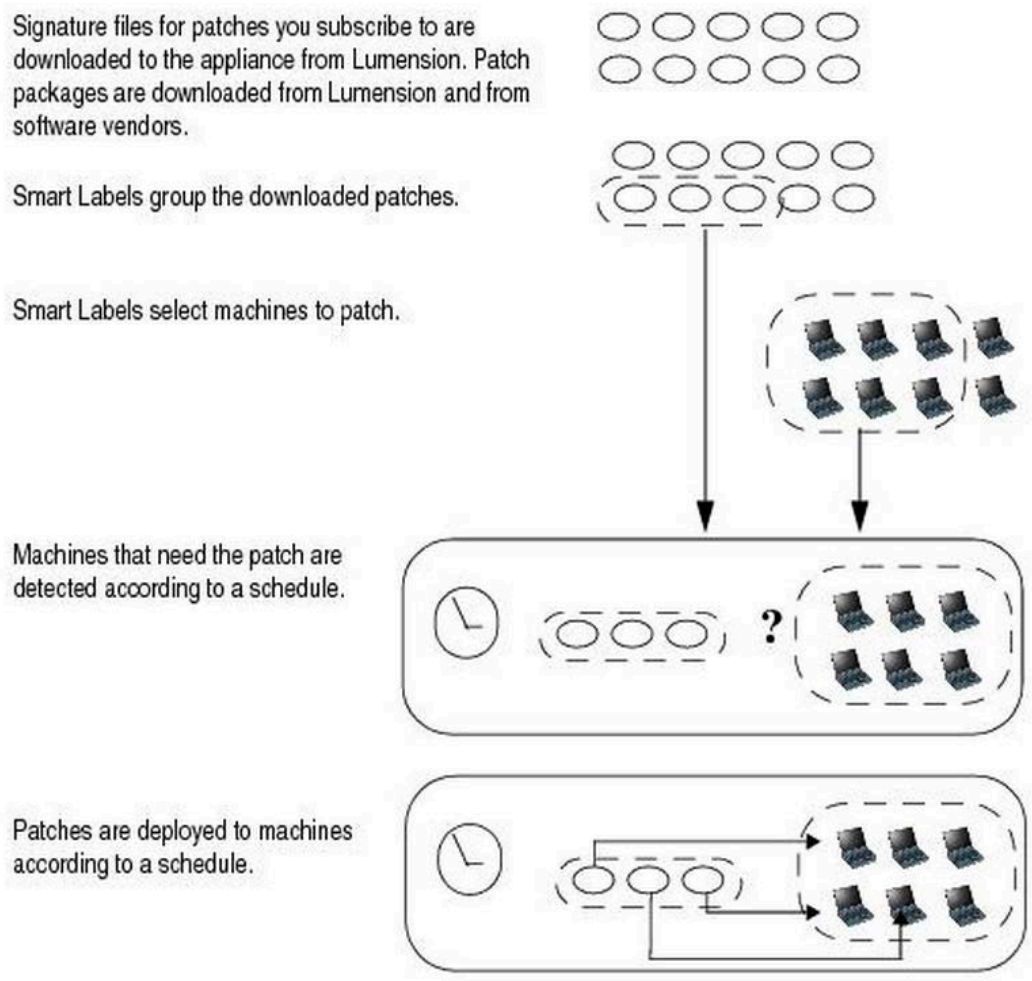

Figure 5-19 Patching workflow (Dell KACE K1000, kace.com)

This Kbox 1000 management server instructs ITDS to perform the following tasks to implement SCPMP:

1. Download patches based on the subscription settings and save them on internal or external storage,
2. Detect, monitor, and scan networked computing systems
3. Categorize workstations by applying labels,
4. Deploy patches and install on computing systems,
5. Create schedules for detection and deployment tasks, and
6. Generate practical reports of each performance and task (Dell KACE K1000 System Management Appliance Version 5.4, kace.com).

5.2.2 Operating Systems

Table 5-6 lists the managed OSs installed on workstations on each department at SEAS. The OS list for applying the patch management process will be considered.

Table 5-6 List of managed OSs and computer statistics for SEAS (Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu)

Operation Systems and Computer Statistics	Number of workstations
Mac OS X 10.4.11 (x86)	1
Mac OS X 10.5.8 (x86)	3
Mac OS X 10.6.6 (x86)	1
Mac OS X 10.6.7 (x86)	2
Mac OS X 10.6.8 (x86)	40
Mac OS X 10.7 (x86)	2
Mac OS X 10.7.1 (x86)	1
Mac OS X 10.7.3 (x86)	1
Mac OS X 10.7.4 (x86)	7
Mac OS X 10.7.5 (x86)	29
Mac OS X 10.8 (x86)	1
Mac OS X 10.8.2 (x86)	3
Mac OS X 10.8.3 (x86)	8
Mac OS X 10.8.4 (x86)	42
Mac OS X 10.8.5 (x86)	9

Mac OS X Server 10.5.8 (x86)	1
Microsoft Windows 7 Enterprise	8
Microsoft Windows 7 Enterprise x64	18
Microsoft Windows 7 Home Premium	1
Microsoft Windows 7 Home Premium x64	5
Microsoft Windows 7 Professional	98
Microsoft Windows 7 Professional N x64	2
Microsoft Windows 7 Professional x64	445
Microsoft Windows 7 Ultimate	2
Microsoft Windows 7 Ultimate x64	3
Microsoft Windows 8 Pro x64	2
Microsoft Windows 8 x64	1
Microsoft Windows Server 2008 R2 Standard x64	2
Microsoft Windows XP Professional	100
Microsoft(R) Windows(R) Server 2003, Standard Edition	4
Microsoft(R) Windows(R) XP Professional x64 Edition	4
Microsoft® Windows Server® 2008 Standard	2
Microsoft® Windows® Storage Server 2008 Standard x64	2
Total Computers in Inventory	850

As the above table demonstrates, different OSs need to be evaluated for patch deployment and security purposes. The breakdown of each departmental networked computing system is depicted in Table 5-7; there are 850 networked computing systems to be patched and managed.

Table 5-7 SEAS departmental networked computing systems (seascf.seas.gwu.edu)

Departments and environments	Windows	Macintosh (Mac OS X)
Computer Science (CS)	42	31
Engineering Management and Systems Engineering (EMSE)	102	23
Civil Engineering (CEE)	44	6

Electrical and Computer Engineering (ECE)	84	24
Mechanical Engineering (MAE)	119	19
Administration and staff (SEAS Admin)	52	5
Computing Facility (CF)	201	40
VA Campos (Science of technology)	32	26
Total	676	174
All managed networked computing systems	850	

5.2.3 Software packages and patch counts

Based on software packages limitation for security patches, Table 5-8 presents the list of software packages that are candidates for SCPMP and are supported by the Kbox 1000 management sever center (Dell KACE K1000 System Management Appliance Version 5.4, kace.com).

Table 5-8 Software packages for applying patches (Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu)

#	Software Package
1	Adobe Acrobat and Reader software
2	The Symantec family of Norton antivirus software
3	The McAfee family of antivirus software
4	Mozilla Firefox
5	The Machine Associates eTrust family of antivirus software
6	Microsoft Office applications
7	Apple applications, such as QuickTime, iTunes, and iLife software
8	Applications in Java environments

The total downloaded patches that need to be applied to fix bugs or vulnerability issues are shown in Table 5-9. There are 23,445 patches, which include OSs and software

patches that are downloaded and stored in the patch storage. The complete list of 2,013 patches that have been considered in this case study is in Appendix B.

Table 5-9 Patch bulletin information (SEAS; Dell KACE K1000 System Management Appliance Version 5.4, kace.com; seascf.seas.gwu.edu)

Patch Bulletin Information	Numbers
Total Patches	6511
Enabled Patches	7583
Total Packages	23445

The above tables demonstrate the extent of and differences in the objective environment. For data collection purposes, a basic level of data integration through using the Dell Kace Kbox 1000 Management Server is assumed due to the configuration of different networked computing systems located in the target environment. This is considered a standard patch management operation. For this SCPMP case study, four steps must be taken: (1) describe the problem description and define test experiment terms, (2) present alternatives to and inputs of the selected data and information based on the cost analysis table, (3) evaluate the problem formulation and maintenance actions, and 4) determine the optimal solution and obtain results. Case study stages are extracted from the data collection tables, figures, and MDP, which were discussed in the literature review chapter. In terms of the technical quality of PMP, it is important to determine the patch types and the impacts of departmental operations. With regard to the significance of applying patches and protecting networked computing systems in academic environments, this case study is manipulated.

5.3 Problem Description

This case study demonstrates a simple PMP in academic departmental workstations. The detailed analysis of this case study is based on MATLAB© code, which is provided in Appendix A. The proposed PMP was implemented over one work week (Monday to Friday work hours). Patch types are classified into four stages:

1. New versions of software update patches,
2. No patch update is released,
3. Recommended patches, and
4. Critical patches.

Patching consistency will help in choosing an optimal policy, which, in turn, will allow ITDS to decrease the cost of the IT department SCPMP.

5.4 Inputs

As mentioned before, five business days are assumed to be sufficient for the entire patching process. Therefore, the mean time between two patching cycles is 168 (7/24) hours, which will be denoted by:

$$\lambda = 0.168$$

The cost breakdown—the average wage for one network administrator, who handles and manages the Dell Kace Kbox1000 server, plus the technician's hourly pay for rebuilding compromised machines (swz.salary.com)—is shown in Table 5-10. Based on this table, the average cost of implementing SCPMP to make maintenance

optimization decisions will be calculated while the IT managers determine the patch management process.

Table 5-10 costs breakdown list (Mohammadi et al., 2013)

Cost of IT	Engineer Kbox admin	One	Annual Salary		$65,000
	Dell Kace Kbox 1000	1000 Machine	$10 Per license		$10,000
	Total				$75,000
	Total Per Week				$1,442
Cost of Failure cF(s)	Technician	$21.64/hours	21.64*40*52		$45,001
	Loaner Machine	5 Laptop	Average $800 Per unit	$4000 for three years	$1,333
	Total				$46,334
	Total Per Week				$891
Cost of Inspection cI(s)	Two hours per each inspection				$109
Cost of staying in state cS(s)	Dell Kace Kbox1000 cost per day				$27
Cost of NA action cM(NA)					$0
Cost of MM action cM(MM)	Total cost of IT peer hour*one shift of engineer work				$26
Cost of PM action cM(PM)	Total cost of IT peer hour*one full day				$206
Cost of CM action cM(CM)	Total cost of IT peer hour*three full day				$618

The table presents the breakdown of the overall costs of the IT department. It includes one network engineer to manage the Management Server Center and one

technician to rebuild the compromised networked computing systems. The state transition probabilities $\sum_{s,s' \in S} P(s'|s,a)$ based on this case study are presented in Table 5-11.

Table 5-11Transition probabilities and time functions (Mohammadi et al., 2013)

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

5.5 Problem Formulation and Maintenance Actions

The total domination of the maintenance actions space for patch deterioration types is $A = \{a = NA, MM, PM, CM\}$, which is assumed to be finite and discrete (Amari et al., 2006). Further, the size of this is denoted by:

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

Then the maintenance action will be as follows:

(5-8) $$A_1 = \{NA\}, A_4 = \{CM\}$$

(5-9) $$A_2 = A_3 = \{NA, MM, PM\}$$

In the case of no action of the maintenance optimization phase, the IT decision maker determines the patching circumstances. ITDS select no action for the patching cycle and continue with the tested patch package. This is described by:

(5-10) $$P_{s,s'}(NA) = \begin{cases} 1 & for \quad s = s' \\ 0 & for \quad s \neq s' \end{cases}$$ (Amari et al., 2006)

For the purpose of taking minimal maintenance (MM) and preventive maintenance (PM) actions, the following applies:

(5-11) $$P_{s,s'}(MM) = P_{s,s'}(PM) = \begin{cases} 1 & for \quad s = s' + 1 \\ 0 & for \quad s \neq s' + 1 \end{cases}$$ (Amari et al., 2006)

When the corrective maintenance (CM) action taken by IT security managers and the patching cycle reaches the perfect stage, based on the released urgent patches and deterioration circumstances, the CM formulation will be:

(5-12) $$P_{s,s'}(CM) = \begin{cases} 1 & for \quad s' = 1 \\ 0 & for \quad s' \neq 1 \end{cases}$$ (Amari et al., 2006)

In this case study, the diversity of configuration of networked computer systems has been covered. I have also considered the different kinds of usage throughout the academic environment, such as faculty, staff, and laboratory workstations and researchers' computers.

5.6 Optimal Policy Solution and results

The goal of this subsection is to determine one optimal policy for networked computing systems in an academic environment with respect to any maintenance actions required for implementation of SCPMP. The optimal policy will allow IT decision makers and ITDS to minimize patch management processes and reduce the cost of

maintenance. The optimal policy will also help reduce PMP and IT department costs; it will keep the equation (4-5) as low as possible for any maintenance action (MA) that has to be taken (Amari et al., 2006; Mazzuchi et al., 2007). Table 5-12 demonstrates an optimal policy based on the cost analyses presented in Table 5-10.

Table 5-12 Optimal Policy based on cost analysis (Mohammadi et al., 2013)

State	Maintenance Action	Value
$\mathbf{S_n}$	NA	5492
$\mathbf{S_v}$	MM	7004
$\mathbf{S_r}$	PM	4493
$\mathbf{S_u}$	CM	4277

To calculate the optimal policy, the MATLAB© Markov Decision Process (MDP) toolbox was used with respect to the costs of IT operations in the academic environment. Appendix A contains all MATLAB codes used to compute this case study's optimal policy (Cros, 2012).

```
Cost =
   1.0e+03 *
    5.4920    5.5180    5.4920    5.4920
    7.0040    7.0300    7.0040    7.0040
    4.4930    4.5190    4.6990    4.6990
    4.2770    4.2770    4.2770    4.8950
V =
   1.0e+03 *
    5.4920
    7.0040
    4.4930
    4.2770
```

```
policy =
     3
     3
     1
     1
iter =
    2
cpu_time =
    0.0200
```

The above iterative policy allows IT decision makers to decrease the cost of PMP to protect networked computing systems and avoid the need to hire a technician to re-image and rebuild compromised machines.

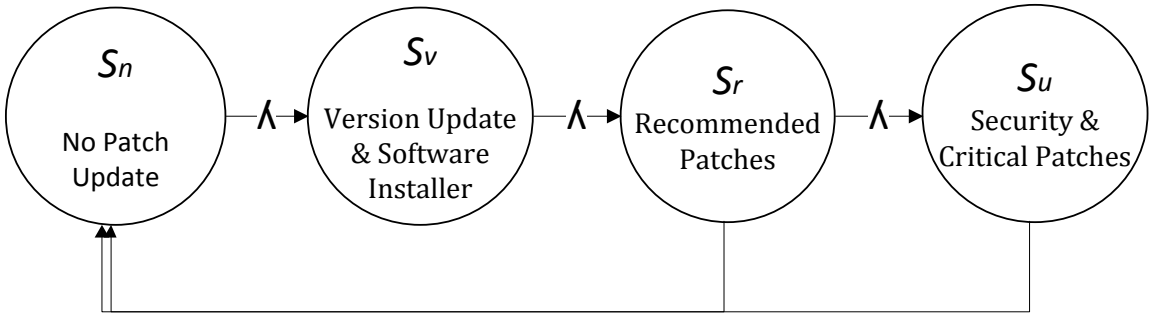

Figure 5-20 Optimal policy for deploying patches based on MDP calculation

Figure 5-20 shows the optimal policy, determined by MDP and maintenance optimization processes based on the cost estimation and PMP that should be taken by ITDS and the IT decision maker.

Chapter 6 — Discussion and Conclusions

In this chapter I discuss the case study results. The intent of this research is to develop a practical framework based on SE fundamentals for implementing the security and critical patch management process in a diverse environment. To that end, an academic environment with 850 networked computing systems was tested to ensure that the framework and research methodology would decrease IT departmental operation costs and mitigate the risk of being attacked or administering compromised machines.

6.1 Conceptual Model

This research has demonstrated that the SE framework can make the SCPMP, which is a critical IT operation, more secure and reliable. Software package security vulnerabilities and OS flaws have been highlighted in terms of reducing the cost of IT operations. Costs rise due to the necessity to hire more technicians to re-image compromised machines or buy loaner workstations to temporarily replace attacked computing systems while they are being rebuilt. The overall conceptual model and research roadmap is presented in Figure 6-21, which identifies research goals for the deployment of patches throughout a managed networked environment. Figure 6-21depicts the planning phase of the research created by work experiments that are seen in its patching process. Next, the proposed research goals and recommendations regarding research methodologies are identified. The case study concluded with one practical process.

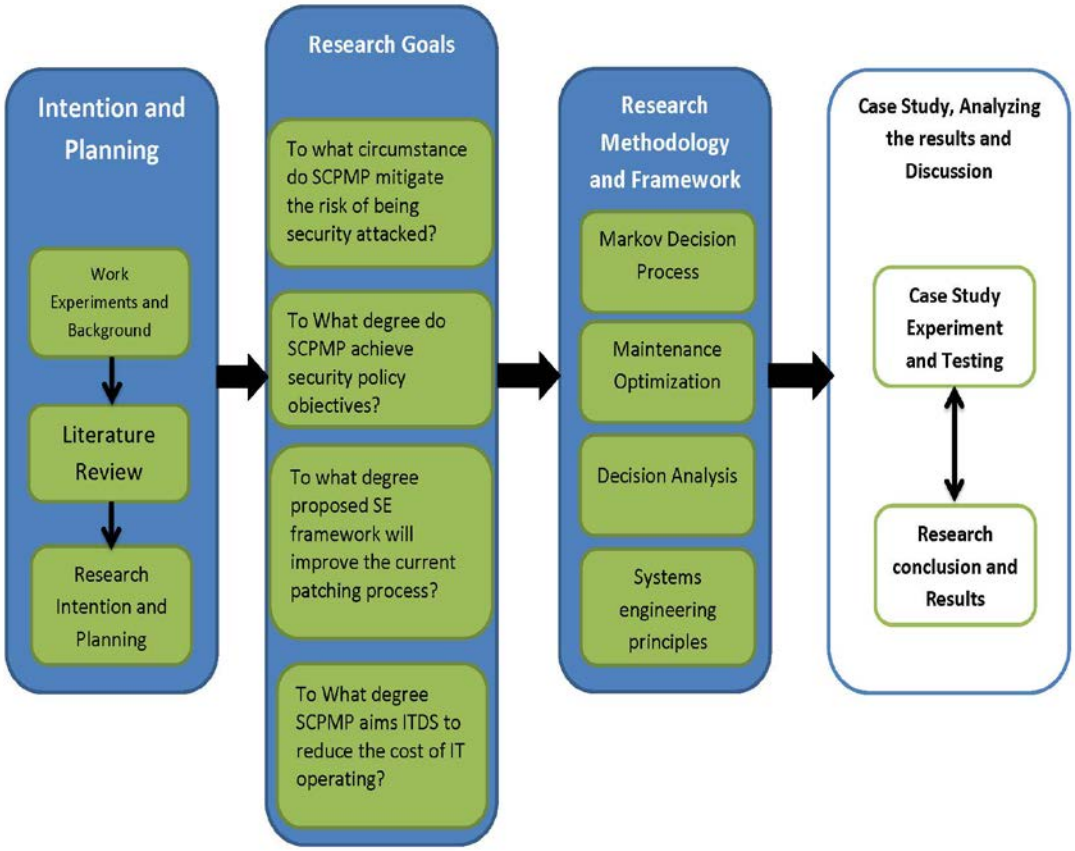

Figure 6-21 Conceptual model and research roadmap

Information from the literature review, research methodology output, Markov decision process toolbox, and proposed SE framework has been combined to present a comprehensive view of the patch management process that complies with security vulnerability policies and standards while enabling engineering activities. Thus, the process fulfills the requirement to enhance system quality and achieve program performance goals.

6.2 Hypotheses Results

The first hypothesis focused on the importance of employing the proposed SE framework, which will improve the current patch management process and mitigate the risk of becoming the target of security attacks. Accordingly, in chapter 2, three different patch management processes: 1) Windows Security Patch management (Park et al. 2007), 2) SANS Patch Management Process (Medzich 2004), and 3) Corporative software maintenance model (Gupta and Qureshi, 2011) were reviewed. Based on these models, there is the lack of systems engineering principles in any current patching protocol as shown in Table 2-5 Patch management process comparison based on SE principles). The proposed SCPMP shows that patch management requirements are necessary for every IT department to address security issues in the SCPMP strategy. In addition, section 2.4 (risk management) demonstrated that once the SCPMP applies, the risk of becoming the target of security attacks is mitigated.

The second hypothesis presented the effects of using the proposed SE framework to implement the SCPMP in a diverse networked computing system environment. These effects can reduce the costs incurred by ITDS to perform the patching process. Findings from the case study experiment, as illustrated in Table 5-10, and IT Operational cost analysis in section 2.6, determined that the proposed SCPMP can save at least 10% of IT operational costs for running patching processes in an academic environment while any urgent security and critical patches release.

The last hypothesis involved the impact of the patch deployment structure, which helps IT managers allocate tasks to those who will implement and complete the deployment process. Currently, the patch management process is an ad-hoc process and

there are no procedures and distinguished steps for applying PMP. However, this research demonstrated the categorized patch management processes in a well-defined list of tasks. These tasks are shown in SE framework in the Figure 4-15.

6.3 Future study and work

In this subsection I will recommend future work to expand the PMP knowledge base. Future studies can be divided into two important areas of patch management. One concerns the types of patches and concentrates on vulnerability disclosure. The other focuses on methods for evaluating and assessing patch performance.

6.3.1 Future study of vulnerability disclosure

I have addressed Microsoft Windows, Apple OS X, and related software patches, including an academic networked computing system's various configurations. However, one important and common OS that was not included is Linux OS. Although Linux may require a different approach than that used with OS X or Windows, it should be patched and included in ITDS security protocols.

6.3.2 Evaluation and assessment performance method

As mentioned before, patch management is a probabilistic process and stochastic analysis is involved in evaluating SCPMP. Therefore, using other probability formulas might be useful for a computational environment. One of the proper theorems for implementing the SCPMP is the Bayesian stochastic theorem. Here I have used the Markov decision process for the maintenance optimization phase in the proposed SE

framework. However, different decision tools can be used to guide IT managers in making decisions and generating optimization policies.

6.4 Conclusion

The security and critical patch management process, as it relates to an IT operations system, requires optimization and monitoring techniques. As implementation of the security vulnerability management and patching process becomes more complex, application of the proposed SCPMP and patching framework will become a critical task. More importantly, the proposed framework addresses shortcomings related to risk management assessment and IT cost evaluation. The proposed SE framework and the approach to implementing SCPMP presented here were designed to address the applicability of maintenance optimization tools for the patching process. This will aid IT department staff and managers to make the most optimal decisions in applying the patching cycle in their environment and networked computing system. The proposed systems engineering framework was analyzed and tested by a practical case study, which was implemented in an academic environment and operations system. Not only was the proposed SE framework used to monitor and control the PMP, but also to improve on and offer optimized patching polices to reduce IT costs and mitigate security risks. Once any urgent security and critical patch is released, ITDS identify the patch types and current circumstances to choose the best method for applying the patching cycle. The proposed SE framework is applicable to any automated patching system and human-involved network security management process. However, the number of computing systems and

size of the network environment should be taken into account when an entire system is being evaluated and costs estimated.

Finally, the SE framework for implementing the SCPMP aims to make the patching process more efficient at a lower cost. A statistical analysis using the MATLAB software to evaluate the process obtained better results for maintenance optimization and provided an outline for an interdisciplinary approach to automating the patch management process.

Bibliography

Ahmed Patel, Wei Qi, Mona Taghavi, (2012),"An evaluation of a secure and trustworthy mobile agent-based e-marketplace system", Information Management & Computer Security, Vol. 20 Iss: 2 pp. 123 – 146 ,doi: 10.1108/09685221211235634

Amari, S.V.; McLaughlin, L.; Pham, H., "Cost-effective condition-based maintenance using markov decision processes," Reliability and Maintainability Symposium, 2006. RAMS '06. Annual , vol., no., pp.464,469, 23-26 Jan. 2006, doi: 10.1109/RAMS.2006.1677417

Aris, S.R.S.; Arshad, N.H.; Mohamed, A., "Risk management practices in IT outsourcing projects," Information Technology, 2008. ITSim 2008. International Symposium on , vol.4, no., pp.1,8, 26-28 Aug. 2008, doi: 10.1109/ITSIM.2008.4631922

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.

Bo Yang; Sai Zeng; Ayachitula, N.; Puri, R., "SLA-driven applicability analysis for patch management," Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on , vol., no., pp.438,445, 23-27 May 2011, doi: 10.1109/INM.2011.5990544

Bommannavar, P.; Bambos, N., "Security Risk Management in Computing Systems with Constraints on Service Disruption," Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on , vol., no., pp.1,6, July 31 2011-Aug. 4 2011, doi: 10.1109/ICCCN.2011.6005875

Cavusoglu Hasan, Cavusoglu Husyin, Zhang Jun, 2008, 'Security Patch Management: Share the Burden or Share the Damage?', Management Science, 54, 4, pp. 657-670, Business Source Premier, EBSCOhost, viewed 6 October 2012.

Cavusoglu, Cavusoglu, and Zhang: Security Patch Management: Share the Burden or Share the Damage? 658 Management Science 54(4), pp. 657–670, 2008 INFORMS, http://dblp.uni-trier.de/db/journals/mansci/mansci54.html#CavusogluCZ08

Cavusoglu, H.; Cavusoglu, H.; Raghunathan, S., "Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge," Software Engineering, IEEE Transactions on , vol.33, no.3, pp.171,185, March 2007, doi: 10.1109/TSE.2007.26

Ching-Huang Lin; Chih-Hao Chen; Chi-Sung Laih, "A Study and Implementation of Vulnerability Assessment and Misconfiguration Detection," Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE , vol., no., pp.1252,1257, 9-12 Dec. 2008, doi: 10.1109/APSCC.2008.212

Chuan-Wen Chang; Dwen-Ren Tsai; Jui-Mi Tsai, "A cross-site patch management model and architecture design for large scale heterogeneous environment," Security Technology, 2005. CCST '05. 39th Annual 2005 International Carnahan Conference on , vol., no., pp.41,46, 11-14 Oct. 2005, doi: 10.1109/CCST.2005.1594837

Dantu, R.; Kolan, P.; Akl, R.; Loper, K., "Classification of Attributes and Behavior in Risk Management Using Bayesian Networks," Intelligence and Security Informatics, 2007 IEEE , vol., no., pp.71,74, 23-24 May 2007, doi: 10.1109/ISI.2007.379536

Dantu, R.; Loper, K.; Kolan, P., "Risk management using behavior based attack graphs," Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on , vol.1, no., pp.445,449 Vol.1, 5-7 April 2004, doi: 10.1109/ITCC.2004.1286496

Dell KACE K1000 System Management Appliance Version 5.4, Online Help, October 2012, © 2004-2012 Dell Inc. All rights reserved, www.kace.com

Duanyang Zhao; Furnell, S.M.; Al-Ayed, A., "The Research on a Patch Management System for Enterprise Vulnerability Update," Information Engineering, 2009. ICIE '09. WASE International Conference on , vol.2, no., pp.250,253, 10-11 July 2009, doi: 10.1109/ICIE.2009.233

Florian Cristina, “report: the most vulnerable operating systems and applications in 2012” February 2013, http://www.gfi.com/blog/report-the-most-vulnerable-operating-systems-and-applications-in-2012/

Forsberg, K. and Mooz, H., The Relationship of System Engineering to the Project Cycle. Proceedings of the National Council for Systems Engineering (NCOSE) Conference, 1991, Chattanooga, Tennessee, pp 57-65.

Grant Adams, Patch Management: Change, Configurations and Release or something more?, FOX IT the authority in service management, 2007, http://www.itilnews.com/uploaded_files/Patch_Management_-_Article_for_itSMF_Conference.pdf

Grob, H.L.; Strauch, G.; Buddendick, C., "Applications for IT-Risk Management – Requirements and Practical Evaluation," Availability, Reliability and Security, 2008. ARES 08. Third International Conference on , vol., no., pp.758,764, 4-7 March 2008, doi: 10.1109/ARES.2008.168

Gupta, A.; Qureshi, S.R., "Collabra: A framework for cooperative Software Maintenance," Information Society (i-Society), 2011 International Conference on , vol., no., pp.133,138, 27-29 June 2011

Hao Wang; Bo Yang; Liang Liu; Qian Ma; Ke Wei Sun; Ying Chen, "Knowledge Enhanced IT Service Management," e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on , vol., no., pp.173,180, 24-26 Oct. 2007, doi: 10.1109/ICEBE.2007.20

He Xiaocong; Kang Ling, "A risk management decision support system for project management based on Bayesian network," Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on , vol., no., pp.308,312, 16-18 April 2010, doi: 10.1109/ICIME.2010.5478061

http://seascf.seas.gwu.edu/

Hui-Ling Lin; Ching-Shun Hsieh; Shao-Shin Hung; Derchian Tsaih, "An Efficient Framework for Distributed Enterprise Portal for Maintaining Global Patch Consistency," Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on , vol.1, no., pp.415,418, 13-15 Dec. 2008, doi: 10.1109/FGCN.2008.223

International Council On Systems Engineering (INCOSE), Systems Engineering Handbook Version 3.1, August 2007, pp 2.3 to 2.6

J.A.M. van der Weide, M.D. Pandey, J.M. van Noortwijk, Discounted cost model for condition-based maintenance optimization, Reliability Engineering & System Safety, Volume 95, Issue 3, March 2010, Pages 236-246, ISSN 0951-8320, http://dx.doi.org/10.1016/j.ress.2009.10.004. (http://www.sciencedirect.com/science/article/pii/S0951832009002415)

Jason Chan, Essential of patch management and policy and practice, PatchManagement.org website hosted by Shavlik Technologies, LLC, January 2004, http://www.patchmanagement.org/pmessentials.asp

Jiangping Wan; Shiqing Zhu; Yunfeng Wang, "Empirical Analysis on Risk Factors of IT Service Management Project Implementation," Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on , vol., no., pp.1,4, 12-14 Oct. 2008, doi: 10.1109/WiCom.2008.2813

Jung-jin Park; Jin-sub Park; Jeong-gi Lee; Bong-hoi Kim; Geum-boon Lee; Beom-joon Cho, "Windows Security Patch Auto-Management System Based on XML," Advanced Communication Technology, The 9th International Conference on , vol.1, no., pp.407,411, 12-14 Feb. 2007, doi: 10.1109/ICACT.2007.358382

Karl Friston, spyridon Samthrakis, Active inference and agency: optimal control without cost functions, Biological Cybernetics Journal, 2012, pp 523-541, dpi:10.1007/s00422-012-0512-8, URL:http://dx.doi.org/10.1007/s00422-012-0512-8

Kosmann, W., Sarkani, S., & Mazzuchi, T., (2013). "Optimization of Space System Development Resources", Acta Astronautica, Vol. 87, June-July 2013, 48-63

Kuhn, R.; Rossman, H.; Liu, Simon, "Introducing "Insecure IT"," IT Professional , vol.11, no.1, pp.24,26, Jan.-Feb. 2009, doi: 10.1109/MITP.2009.10

Lin He; Ling Gao; Dongqi Chen; Ruyi Wang, "Research of Vulnerability-Patch Associated Repair Model Based on SVM," Multimedia Information Networking and Security (MINES), 2011 Third International Conference on , vol., no., pp.356,360, 4-6 Nov. 2011, doi: 10.1109/MINES.2011.29

Liu, Simon; Kuhn, R.; Rossman, H., "Surviving Insecure IT: Effective Patch Management," IT Professional , vol.11, no.2, pp.49,51, March-April 2009, doi: 10.1109/MITP.2009.38

Liu, Simon; Kuhn, R.; Rossman, H., "Surviving Insecure IT: Effective Patch Management," IT Professional , vol.11, no.2, pp.49,51, March-April 2009, doi: 10.1109/MITP.2009.38

Maarten-jan Kallen , Jan M. Van Noortwijk, Optimal Inspection and Replacement Decisions for Multiple Failure Modes, Published in: Probabilistic Safety Assessment and Management (PSAM7-ESREL'04): Proceedings of the 7th International Conference on Probabilistic Safety Assessment and Management, June 14-18, 2004, Berlin, Germany. Spitzer, C., Schmocker, U. and Dang, V.N. (eds.) Vol.4, pp.2435–2440, London: Springer, 2004.

Madni, A.M.; Jackson, S., "Towards a Conceptual Framework for Resilience Engineering," Systems Journal, IEEE , vol.3, no.2, pp.181,191, June 2009, doi: 10.1109/JSYST.2009.2017397

Maik Medzich, Deploying a process of Patch Management, SANS institutes, Global Information Assurance Certification Paper, 2004, http://www.giac.org/paper/gsec/3876/deploying-process-patch-management-relation-risk-management/106152

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.

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

McMillan Robert, "Is antivirus software a waste money?" March 2012, http://www.wired.com/wiredenterprise/2012/03/antivirus/

Mell Peter, Bergeron Tiffany, Henning David, "Creating a Patch and Vulnerability Management Program", National Institute of Standards and Technology Special Publication 800-40 Version 2.0, Natl. Inst. Stand. Technol. Spec. Publ. 800-40 Ver. 2.0, 75 pages (November 2005), http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf

MITRE Company, Risk definition, December 1999, modified 2013, http://www.mitre.org/work/sepo/toolkits/risk/StandardProcess/definitions.html

MITRE Company, Risk Management Toolkit, December 1999, last modified 2013, http://www.mitre.org/work/sepo/toolkits/risk/ToolsTechniques/RiskMatrix.html

Mohammadi, H., Mazzuchi, T.A., & Sarkani, S., A systems engineering framework for implementing a security and critical patch management process in diverse environments (Academic departments workstations), Journal of Information Technology Management (JITM), December 2013, Accepted for Publication in Volume 24, number 4

Mounzer, J.; Alpcan, Tansu; Bambos, N., "Dynamic Control and Mitigation of Interdependent IT Security Risks," Communications (ICC), 2010 IEEE International Conference on , vol., no., pp.1,6, 23-27 May 2010 doi: 10.1109/ICC.2010.5502671

National Aeronautics and Space Administration (NASA), Systems Engineering Handbook, NASA Headquarters Washington, D.C, December 2007, pp. 3-5

Nikolaidou, M.; Alexopoulou, N.; Tsadimas, A.; Dais, A.; Anagnostopoulos, D., "A Consistent Framework for Enterprise Information System Engineering, "Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International , vol., no., pp.492,496, Oct. 2006, doi: 10.1109/EDOC.2006.6

Nunez, Y.; Gustavson, F.; Grossman, F.; Tappert, C., "Designing a distributed patch management security system," Information Society (i-Society), 2010 International Conference on , vol., no., pp.162,167, 28-30 June 2010

Okamura, H.; Tokuzane, M.; Dohi, T., "Optimal Security Patch Release Timing under Non-homogeneous Vulnerability-Discovery Processes," Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on , vol., no., pp.120,128, 16-19 Nov. 2009, doi: 10.1109/ISSRE.2009.19

Okimoto, T.; Ikegai, N.; Inoue, K.; Okada, H.; Ribeiro, T.; Maruyama, H., "Cyber security problem based on Multi-Objective Distributed Constraint Optimization technique," Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on , vol., no., pp.1,7, 24-27 June 2013, doi: 10.1109/DSNW.2013.6615540

Pamela A. Engert and Zachary F. Lansdowne, Risk Matrix Toolkit guide, version 2.2, December 1999, http://www.mitre.org/work/sepo/toolkits/risk/ToolsTechniques/files/UserGuide220.pdf

Pandey M. D., Noortwijik J.M., Klatter H.E., The potential applicability of the Life-Quality Index to maintenance optimization problems, 2006, pp. 1-8. Proc. IABMAS Conference, Lisbon Portugal

Rajabi-Ghahnavie, A.; Fotuhi-Firuzabad, M., "Application of Markov Decision Process in Generating Units Maintenance Scheduling," Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on , vol., no., pp.1,6, 11-15 June 2006, doi: 10.1109/PMAPS.2006.360308

Ramaswamy, A.; Bratus, S.; Smith, S.W.; Locasto, M.E., "Katana: A Hot Patching Framework for ELF Executables," Availability, Reliability, and Security, 2010. ARES '10 International Conference on , vol., no., pp.507,512, 15-18 Feb. 2010, doi: 10.1109/ARES.2010.112

Risk Management Guide For DOD Acquisition, Department of Defense Defense Acquisition University, Fifth Edition (Version 2.0), Published by the Defense Acquistion University Press Fort Belvior, Virginia, June 2003, http://www.risk-services.com/RMG20June2003.pdf

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

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.

Schneider, Fred B.; Mulligan, Deirdre K., "A Doctrinal Thesis," Security & Privacy, IEEE , vol.9, no.4, pp.3,4, July-Aug. 2011, doi: 10.1109/MSP.2011.76

Seo, Jung-Taek, Yun-ju Kim, Eung-Ki Park, Sangwon 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.

Sihvonen, H.; Jantti, M., "Improving Release and Patch Management Processes: An Empirical Case Study on Process Challenges," Software Engineering Advances (ICSEA), 2010 Fifth International Conference on , vol., no., pp.232,237, 22-27 Aug. 2010, doi: 10.1109/ICSEA.2010.42

Tian, H. T.; Huang, L.S.; Zhou, Z.; Luo, Y.L., "Arm up administrators: automated vulnerability management," Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on , vol., no., pp.587,593, 10-12 May 2004, doi: 10.1109/ISPAN.2004.1300542

Van Noortwijk, J.M.; Dekker, R.; Cooke, R.M.; Mazzuchi, T.A., "Expert judgment in maintenance optimization," Reliability, IEEE Transactions on , vol.41, no.3, pp.427,432, Sep 1992, doi: 10.1109/24.159813

Wang Lijian; Wang Bin; Peng Yongjun, "Research the information security risk assessment technique based on Bayesian network," Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on , vol.3, no., pp.V3-600,V3-604, 20-22 Aug. 2010, doi: 10.1109/ICACTE.2010.5579740

Wu, W.; Yip, F.; Yiu, E.; Ray, P., "Integrated vulnerability management system for enterprise networks," e-Technology, e-Commerce and e-Service, 2005. EEE '05. Proceedings. The 2005 IEEE International Conference on , vol., no., pp.698,703, 29 March-1 April 2005, doi: 10.1109/EEE.2005.83

www.itil-officialsite.com

www.kace.com

www.Lumension.com

Xiaoling Hao; Nan Yang, "IT operational risk assessment and control model based on Bayesian Network," Natural Computation (ICNC), 2010 Sixth International Conference on , vol.3, no., pp.1105,1109, 10-12 Aug. 2010 doi: 10.1109/ICNC.2010.5583696

Zhiqiang Lin; Bing Mao; Li Xie, "A practical framework for dynamically immunizing software security vulnerabilities," Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on , vol., no., pp.8 pp., 20-22 April 2006, doi: 10.1109/ARES.2006.11

Appendix A

Matlab codes used to calculating the optimal policy include six MATLAB© dot m files (Cros, 2012).

A. Dissertation.m (Marie-Josee Cros, 2012)

```
clear all;
close all;
clc

landa=0.168;
discount=1/(1+landa);
policy0=[1; 2; 3; 4;];
max_iter=1000;
eval_type=0;
epsilon=0.001;
V0=zeros(4,1);

P(:,:,1)=[1 0 0 0;0 1/2 1/2 0;0 0 0 0;0 0 0 0];
P(:,:,2)=[0 0 0 0;0 1/2 1/2 0;0 0 1/3 1/3;0 0 0 0];
P(:,:,3)=[0 0 0 0;0 0 0 0;0 0 1/3 2/3;0 0 0 0];
P(:,:,4)=[0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 1];

T= [24 0 0 0;
    0 24 0 0;
    0 24 24 48;
    0 0 0 72];

cK=[1442 1442 1442 1442;1442 1442 1442 1442;1442 1442 1442
1442;1442 1442 1442 1442;];
cF=[891 891 891 891;891 891 891 891;891 891 891 891;891 891 891
891;];
cM=[0 26 0 0;0 26 0 0;0 26 206 206;0 0 0 618;];
cS=[27 27 27 27;27 27 27 27;27 27 27 27;27 27 27 27;];
Psum = [1 0.5 0.5 0;0 0.5 0.8333 0.6667;0 0 0.3333 0.6667;0 0 0
1];
[Q] = Q_Cost(Psum, T, cK, cM, cF, cS);
R=Q;
[V, policy, iter, cpu_time] = mdp_policy_iteration(P, R,
discount, policy0, max_iter, eval_type)
```

B. Mdp_bellman_operation.m (Marie-Josee Cros, 2012)

```
function [V, policy] = mdp_bellman_operator(P, PR, discount,
Vprev)


% mdp_bellman_operator Applies the Bellman operator on the value
function Vprev
%                      Returns a new value function and a Vprev-
improving policy
% Arguments ----------------------------------------------------
----------
% Let S = number of states, A = number of actions
%   P(SxSxA) = transition matrix
%              P could be an array with 3 dimensions or
%              a cell array (1xA), each cell containing a matrix
(SxS) possibly sparse
%   PR(SxA) = reward matrix
%              PR could be an array with 2 dimensions or
%              a sparse matrix
%   discount = discount rate, in ]0, 1]
%   Vprev(S) = value function
% Evaluation ---------------------------------------------------
----------
%   V(S)   = new value function
%   policy(S) = Vprev-improving policy

% MDPtoolbox: Markov Decision Processes Toolbox
% Copyright (C) 2009  INRA
% Redistribution and use in source and binary forms, with or
without modification,
% are permitted provided that the following conditions are met:
%    * Redistributions of source code must retain the above
copyright notice,
%      this list of conditions and the following disclaimer.
%    * Redistributions in binary form must reproduce the above
copyright notice,
%      this list of conditions and the following disclaimer in
the documentation
%      and/or other materials provided with the distribution.
%    * Neither the name of the <ORGANIZATION> nor the names of
its contributors
%      may be used to endorse or promote products derived from
this software
%      without specific prior written permission.
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
% ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED
```

```
% WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.
% IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT,
% INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING,
% BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
% DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE
% OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED
% OF THE POSSIBILITY OF SUCH DAMAGE.


if iscell(P); S = size(P{1},1); else S = size(P,1); end;
if discount <= 0 || discount > 1
    disp('-----------------------------------------------------------
--')
    disp('MDP Toolbox ERROR: Discount rate must be in ]0; 1]')
    disp('-----------------------------------------------------------
--')
elseif size(Vprev,1) ~= S
    disp('-----------------------------------------------------------
-')
    disp('MDP Toolbox ERROR: Vprev must have the same dimension
as P')
    disp('-----------------------------------------------------------
-')
else

    if iscell(P)
        A = length(P);
        for a=1:A
            Q(:,a) = PR(:,a) + discount*P{a}*Vprev;
        end
    else
        A = size(P,3);
        for a=1:A
            Q(:,a) = PR(:,a) + discount*P(:,:,a)*Vprev;
        end
    end
    [V, policy] = min(Q,[],2);

end;
```

C. mdp_computePpolicyPRpolicy.m (Marie-Josee Cros, 2012)

```
function [Ppolicy, PRpolicy] = mdp_computePpolicyPRpolicy(P, R,
policy)


% mdp_computePpolicyPRpolicy  Computes the transition matrix and
the reward matrix for a policy
% Arguments ---------------------------------------------------
---------
% Let S = number of states, A = number of actions
%   P(SxSxA)  = transition matrix
%               P could be an array with 3 dimensions or
%               a cell array (1xA), each cell containing a matrix
(SxS) possibly sparse
%   R(SxSxA) or (SxA) = reward matrix
%               R could be an array with 3 dimensions (SxSxA) or
%               a cell array (1xA), each cell containing a sparse
matrix (SxS) or
%               a 2D array(SxA) possibly sparse
%   policy(S) = a policy
% Evaluation --------------------------------------------------
---------
%   Ppolicy(SxS)  = transition matrix for policy
%   PRpolicy(S)   = reward matrix for policy

% MDPtoolbox: Markov Decision Processes Toolbox
% Copyright (C) 2009  INRA
% Redistribution and use in source and binary forms, with or
without modification,
% are permitted provided that the following conditions are met:
%    * Redistributions of source code must retain the above
copyright notice,
%      this list of conditions and the following disclaimer.
%    * Redistributions in binary form must reproduce the above
copyright notice,
%      this list of conditions and the following disclaimer in
the documentation
%      and/or other materials provided with the distribution.
%    * Neither the name of the <ORGANIZATION> nor the names of
its contributors
%      may be used to endorse or promote products derived from
this software
%      without specific prior written permission.
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
% ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED
% WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.
```

```
% IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT,
% INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING,
% BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
% DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE
% OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED
% OF THE POSSIBILITY OF SUCH DAMAGE.


if iscell(P); A = length(P); else A = size(P,3); end

for a=1:A % avoid looping over S

    ind = find(policy == a); % the rows that use action a
    if ~isempty(ind)
        if iscell(P)
            Ppolicy(ind,:) = P{a}(ind,:);
        else
            Ppolicy(ind,:) = P(ind,:,a);
        end
        PR = mdp_computePR(P,R);
        PRpolicy(ind,1) = PR(ind,a);
    end
end

if issparse(PR)
    PRpolicy = sparse(PRpolicy);
end
```

D. mdp_computePR.m (Marie-Josee Cros, 2012)

```
function PR = mdp_computePR(P,R)


% mdp_computePR  Computes the reward for the system in one state
%                chosing an action
% Arguments ----------------------------------------------------
---------
% Let S = number of states, A = number of actions
%   P(SxSxA)  = transition matrix
%              P could be an array with 3 dimensions or
```

```
%              a cell array (1xA), each cell containing a matrix
(SxS) possibly sparse
%   R(SxSxA) or (SxA) = reward matrix
%              R could be an array with 3 dimensions (SxSxA) or
%              a cell array (1xA), each cell containing a sparse
matrix (SxS) or
%              a 2D array(SxA) possibly sparse
% Evaluation ------------------------------------------------------
---------
%   PR(SxA)   = reward matrix

% MDPtoolbox: Markov Decision Processes Toolbox
% Copyright (C) 2009  INRA
% Redistribution and use in source and binary forms, with or
without modification,
% are permitted provided that the following conditions are met:
%    * Redistributions of source code must retain the above
copyright notice,
%      this list of conditions and the following disclaimer.
%    * Redistributions in binary form must reproduce the above
copyright notice,
%      this list of conditions and the following disclaimer in
the documentation
%      and/or other materials provided with the distribution.
%    * Neither the name of the <ORGANIZATION> nor the names of
its contributors
%      may be used to endorse or promote products derived from
this software
%      without specific prior written permission.
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
% ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED
% WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.
% IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT,
% INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING,
% BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
% DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE
% OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED
% OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
if ndims(R)==2 && ~iscell(R)  % R has form R(SxA)
    PR = R;
else % R has form R(SxSxA)
    PR = [];
    if iscell(P)
        A = length(P);
        if iscell(R)
            for a=1:A; PR(:,a) = sum(P{a}.*R{a},2); end;
        else
            for a=1:A; PR(:,a) = sum(P{a}.*R(:,:,a),2); end;
        end
    else
        A = size(P,3);
        if iscell(R)
            for a=1:A; PR(:,a) = sum(P(:,:,a).*R{a},2); end;
        else
            for a=1:A; PR(:,a) = sum(P(:,:,a).*R(:,:,a),2); end;
        end
    end
end;
```

E. mdp_eval_policy_matrix.m (Marie-Josee Cros, 2012)

```
function Vpolicy = mdp_eval_policy_matrix(P, R, discount, policy)


% mdp_eval_policy_matrix    Evaluation of the value function of a
policy
% Arguments -----------------------------------------------------
--------
% Let S = number of states, A = number of actions
%   P(SxSxA) = transition matrix
%              P could be an array with 3 dimensions or
%              a cell array (1xA), each cell containing a matrix
(SxS) possibly sparse
%   R(SxSxA) or (SxA) = reward matrix
%              R could be an array with 3 dimensions (SxSxA) or
%              a cell array (1xA), each cell containing a sparse
matrix (SxS) or
%              a 2D array(SxA) possibly sparse
%   discount = discount rate in ]0; 1[
%   policy(S) = a policy
% Evaluation ----------------------------------------------------
---------
%   Vpolicy(S) = value function of the policy

% MDPtoolbox: Markov Decision Processes Toolbox
% Copyright (C) 2009  INRA
```

```
% Redistribution and use in source and binary forms, with or
without modification,
% are permitted provided that the following conditions are met:
%    * Redistributions of source code must retain the above
copyright notice,
%      this list of conditions and the following disclaimer.
%    * Redistributions in binary form must reproduce the above
copyright notice,
%      this list of conditions and the following disclaimer in
the documentation
%      and/or other materials provided with the distribution.
%    * Neither the name of the <ORGANIZATION> nor the names of
its contributors
%      may be used to endorse or promote products derived from
this software
%      without specific prior written permission.
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
% ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED
% WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.
% IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT,
% INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING,
% BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
% DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE
% OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED
% OF THE POSSIBILITY OF SUCH DAMAGE.


% check of arguments
if iscell(P); S = size(P{1},1); else S = size(P,1); end;
if discount <= 0 || discount >= 1
    disp('-----------------------------------------------------
-')
    disp('MDP Toolbox ERROR: Discount rate must be in ]0; 1[')
    disp('-----------------------------------------------------
-')
elseif size(policy,1)~=S || any(mod(policy,1)) || any(policy<1)
|| any(policy>S)
    disp('-----------------------------------------------------
-')
```

```
    disp('MDP Toolbox ERROR: policy must be a (1xS) vector with
integer from 1 to S')
    disp('------------------------------------------------------
-')
else

    [Ppolicy, PRpolicy] = mdp_computePpolicyPRpolicy(P, R,
policy);

    % V = PR + gPV  => (I-gP)V = PR  => V = inv(I-gP)*PR
    Vpolicy = (speye(S,S) - discount*Ppolicy) \ PRpolicy;

end
```

F. mdp_policy_iteration.m (Marie-Josee Cros, 2012)

```
function [V, policy, iter, cpu_time] = mdp_policy_iteration(P, R,
discount, policy0, max_iter, eval_type)


% mdp_policy_iteration  Resolution of discounted MDP
%                       with policy iteration algorithm
% Arguments -----------------------------------------------------
----------
% Let S = number of states, A = number of actions
%   P(SxSxA) = transition matrix
%              P could be an array with 3 dimensions or
%              a cell array (1xA), each cell containing a matrix
(SxS) possibly sparse
%   R(SxSxA) or (SxA) = reward matrix
%              R could be an array with 3 dimensions (SxSxA) or
%              a cell array (1xA), each cell containing a sparse
matrix (SxS) or
%              a 2D array(SxA) possibly sparse
%   discount = discount rate, in ]0, 1[
%   policy0(S) = starting policy, optional
%   max_iter = maximum number of iteration to be done, upper than
0,
%              optional (default 1000)
%   eval_type = type of function used to evaluate policy:
%              0 for mdp_eval_policy_matrix, else
mdp_eval_policy_iterative
%              optional (default 0)
% Evaluation ----------------------------------------------------
----------
```

```
%   V(S)   = value function
%   policy(S) = optimal policy
%   iter      = number of done iterations
%   cpu_time = used CPU time
%---------------------------------------------------------------
----------
% In verbose mode, at each iteration, displays the number
% of differents actions between policy n-1 and n

% MDPtoolbox: Markov Decision Processes Toolbox
% Copyright (C) 2009  INRA
% Redistribution and use in source and binary forms, with or
without modification,
% are permitted provided that the following conditions are met:
%    * Redistributions of source code must retain the above
copyright notice,
%      this list of conditions and the following disclaimer.
%    * Redistributions in binary form must reproduce the above
copyright notice,
%      this list of conditions and the following disclaimer in
the documentation
%      and/or other materials provided with the distribution.
%    * Neither the name of the <ORGANIZATION> nor the names of
its contributors
%      may be used to endorse or promote products derived from
this software
%      without specific prior written permission.
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
% ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED
% WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.
% IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT,
% INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING,
% BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE,
% DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE
% OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED
% OF THE POSSIBILITY OF SUCH DAMAGE.


cpu_time = cputime;
```

```
global mdp_VERBOSE;

% check of arguments
if iscell(P); S = size(P{1},1); else S = size(P,1); end;
if discount <= 0 || discount >= 1
    disp('-------------------------------------------------------
-')
    disp('MDP Toolbox ERROR: Discount rate must be in ]0; 1[')
    disp('-------------------------------------------------------
-')
elseif nargin > 3 && (size(policy0,1)~=S || any(mod(policy0,1))
|| any(policy0<1) || any(policy0>S))
    disp('-------------------------------------------------------
-')
    disp('MDP Toolbox ERROR: policy0 must a (1xS) vector with
integer from 1 to S')
    disp('-------------------------------------------------------
-')
elseif nargin > 4 && max_iter <= 0
    disp('-------------------------------------------------------
-')
    disp('MDP Toolbox ERROR: The maximum number of iteration must
be upper than 0')
    disp('-------------------------------------------------------
-')
else

    PR = mdp_computePR(P,R);

    % initialization of optional arguments
    if nargin < 6; eval_type = 0; end;
    if nargin < 5; max_iter = 1000; end;
    if nargin < 4;
        % initialization of policy:
        % the one wich maximizes the expected immediate reward
        [nil, policy0] =
mdp_bellman_operator(P,PR,discount,zeros(S,1));
    end;

    if mdp_VERBOSE; disp('  Iteration
Number_of_different_actions'); end;

    iter = 0;
    policy = policy0;
    is_done = false;
    while ~is_done
        iter = iter + 1;
        if  (eval_type==0)
            V = mdp_eval_policy_matrix(P,PR,discount,policy);
        else
```

```
            V = mdp_eval_policy_iterative(P,PR,discount,policy);
        end;
        [nil, policy_next] =
mdp_bellman_operator(P,PR,discount,V);

     n_different = sum(policy_next ~= policy);
        if mdp_VERBOSE; disp(['       ' num2str(iter) '
'  num2str(n_different)]); end;

        if all(policy_next==policy) || iter == max_iter
            is_done = true;
        else
            policy = policy_next;
        end;
    end;

end;

cpu_time = cputime - cpu_time;
```

G. Q_Cost.m (Marie-Josee Cros, 2012)

```
function [Cost]= Q_Cost(P, T, cK, cM, cF, cS)


  Cost = cK + cM + P (:,:)*cF+P(:,:)*T(:,:)*cS
end
```

Appendix B

Appendix B contain the 2013 vulnerability reports of networked computing systems at SEAS as reported by Dell KACE K1000 System Management Appliance Version 5.4 (kace.com; seascf.seas.gwu.edu). The list of 2013 patches (From 1/1/2013 to 9/1/2013) available to be deployed and installed on workstations extracted by Dell KACE K1000 System Management Appliance Version 5.4 (kace.com; seascf.seas.gwu.edu) have been considered as well.

OVALIID	Description	Computers
16142	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16126	Use-after-free vulnerability in Microsoft Internet Explorer 8 and 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer pasteHTML Use After Free Vulnerability."	6
16122	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16176	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16175	The Vector Markup Language (VML) implementation in Microsoft Internet Explorer 6 through 10 does not properly allocate buffers, which allows remote attackers to execute arbitrary code via a crafted web site, aka "VML Memory Corruption Vulnerability."	6

16069	Use-after-free vulnerability in Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer LsGetTrailInfo Use After Free Vulnerability."	1
16067	The OpenType Font (OTF) driver in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows remote attackers to execute arbitrary code via a crafted OpenType font file, aka "OpenType Font Parsing Vulnerability."	3
16086	Heap-based buffer overflow in DirectPlay in DirectX 9.0 through 11.1 in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, and Windows Server 2012 allows remote attackers to execute arbitrary code via a crafted Office document, aka "DirectPlay Heap Overflow Vulnerability."	1
16095	Use-after-free vulnerability in Microsoft Internet Explorer 8 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CTreeNode Use After Free Vulnerability."	5
16091	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Window Handle Vulnerability."	7
16273	The SSL provider component in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle encrypted packets, which allows man-in-the-middle attackers to conduct SSLv2 downgrade attacks against (1) SSLv3 sessions or (2) TLS sessions by intercepting handshakes and injecting content, aka "Microsoft SSL Version 3 and TLS Protocol Security Feature Bypass Vulnerability."	3
16257	Race condition in the kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Kernel Race Condition Vulnerability."	7
16256	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16282	The WCF Replace function in the Open Data (aka OData) protocol implementation in Microsoft .NET Framework 3.5, 3.5 SP1, 3.5.1, and 4, and the Management OData IIS Extension on Windows Server 2012, allows remote attackers to cause a denial of service (resource consumption and daemon restart) via crafted values in HTTP requests, aka "Replace Denial of Service Vulnerability."	3
16284	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3

16313	Race condition in the kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages incorrect handling of objects in memory, aka "Kernel Race Condition Vulnerability," a different vulnerability than CVE-2013-1279.	3
16301	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16294	Use-after-free vulnerability in Microsoft Internet Explorer 8 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer SLayoutRun Use After Free Vulnerability."	5
16224	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16204	Writer in Microsoft Windows Essentials 2011 and 2012 allows remote attackers to bypass proxy settings and overwrite arbitrary files via crafted URL parameters, aka "Windows Essentials Improper URI Handling Vulnerability."	72
16249	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CObjectElement Use After Free Vulnerability."	6
16245	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CHTML Use After Free Vulnerability."	6
16244	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16239	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer onBeforeCopy Use After Free Vulnerability."	7
15875	Use-after-free vulnerability in Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CMarkup Use After Free Vulnerability."	1
15867	Integer overflow in the kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Windows Kernel Integer Overflow Vulnerability."	1
15901	Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allow remote attackers to execute arbitrary code via a crafted (1) file name or (2) subfolder name that triggers use of unallocated memory as the destination of a copy operation, aka "Windows Filename Parsing Vulnerability."	2

15845	The kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allow remote attackers to execute arbitrary code via a crafted TrueType Font (TTF) file, aka "TrueType Font Parsing Vulnerability."	3
15810	The Web Proxy Auto-Discovery (WPAD) functionality in Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not validate configuration data that is returned during acquisition of proxy settings, which allows remote attackers to execute arbitrary JavaScript code by providing crafted data during execution of (1) an XAML browser application (aka XBAP) or (2) a .NET Framework application, aka "Web Proxy Auto-Discovery Vulnerability."	1
15830	** RESERVED ** This candidate has been reserved by an organization or individual that will use it when announcing a new security problem. When the candidate has been publicized, the details for this candidate will be provided.	3
15924	The reflection implementation in Microsoft .NET Framework 1.0 SP3, 1.1 SP1, 2.0 SP2, 3.5.1, and 4 does not properly enforce object permissions, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP) or (2) a crafted .NET Framework application, aka "Reflection Bypass Vulnerability."	1
15999	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16049	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CCaret Use After Free Vulnerability."	7
15981	Microsoft Internet Explorer 8 and 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3110.	8
15960	The code-optimization feature in the reflection implementation in Microsoft .NET Framework 4 and 4.5 does not properly enforce object permissions, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP) or (2) a crafted .NET Framework application, aka "WPF Reflection Optimization Vulnerability."	1
15979	Use-after-free vulnerability in Microsoft Internet Explorer 8 and 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1307.	8
15967	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16317	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, as demonstrated by VUPEN during a Pwn2Own competition at CanSecWest 2013, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1308 and CVE-2013-1309.	8

16501	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16498	The USB kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, and Windows Server 2012 do not properly handle objects in memory, which allows physically proximate attackers to execute arbitrary code by connecting a crafted USB device, aka "Windows USB Descriptor Vulnerability," a different vulnerability than CVE-2013-1285 and CVE-2013-1286.	6
16490	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16504	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, Windows 7 Gold and SP1, and Windows 8 allows local users to cause a denial of service (reboot) via a crafted OpenType font, aka "OpenType Font Parsing Vulnerability" or "Win32k Font Parsing Vulnerability."	7
16507	Microsoft Internet Explorer 8 and 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3141.	8
16517	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3112, CVE-2013-3113, CVE-2013-3121, and CVE-2013-3142.	8
16515	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1303 and CVE-2013-1338.	7
16510	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16485	The NTFS kernel-mode driver in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges or cause a denial of service (NULL pointer dereference and system crash) via a crafted application that leverages improper handling of objects in memory, aka "NTFS NULL Pointer Dereference Vulnerability."	7
16472	Use-after-free vulnerability in Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer InsertElement Use After Free Vulnerability."	1
16470	Use-after-free vulnerability in Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CDispNode Use After Free Vulnerability."	1

16465	Use-after-free vulnerability in Microsoft Internet Explorer 7 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer COmWindowProxy Use After Free Vulnerability."	6
16474	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16475	The Windows Forms (aka WinForms) component in Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly restrict the privileges of a callback function during object creation, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (XBAP) or (2) a crafted .NET Framework application, aka "WinForms Callback Elevation Vulnerability."	7
16483	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer vtable Use After Free Vulnerability."	6
16480	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16478	The Client/Server Run-time Subsystem (CSRSS) in Microsoft Windows Server 2008 R2 and R2 SP1 and Windows 7 Gold and SP1 does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Reference Count Vulnerability."	6
16477	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3113, CVE-2013-3121, CVE-2013-3139, and CVE-2013-3142.	8
16518	Microsoft Internet Explorer 6 through 8 does not properly restrict data access by VBScript, which allows remote attackers to perform cross-domain reading of JSON files via a crafted web site, aka "JSON Array Information Disclosure Vulnerability."	5
16526	Use-after-free vulnerability in Microsoft Internet Explorer 8 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CElement Use After Free Vulnerability."	5
16634	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer removeChild Use After Free Vulnerability."	7
16621	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1303 and CVE-2013-1304.	7
16650	Use-after-free vulnerability in Microsoft Internet Explorer 8 and 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-0811.	8

16662	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1304 and CVE-2013-1338.	7
16655	Microsoft Internet Explorer 8 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3111.	8
16598	The Remote Desktop ActiveX control in mstscax.dll in Microsoft Remote Desktop Connection Client 6.1 and 7.0 does not properly handle objects in memory, which allows remote attackers to execute arbitrary code via a web page that triggers access to a deleted object, and allows remote RDP servers to execute arbitrary code via unspecified vectors that trigger access to a deleted object, aka "RDP ActiveX Control Remote Code Execution Vulnerability."	7
16563	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Win32k Race Condition Vulnerability."	7
16559	The Common Language Runtime (CLR) in Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly check signatures, which allows remote attackers to make undetected changes to signed XML documents via unspecified vectors that preserve signature validity, aka "XML Digital Signature Spoofing Vulnerability."	9
16575	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Win32k Race Condition Vulnerability."	7
16591	The USB kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, and Windows Server 2012 do not properly handle objects in memory, which allows physically proximate attackers to execute arbitrary code by connecting a crafted USB device, aka "Windows USB Descriptor Vulnerability," a different vulnerability than CVE-2013-1285 and CVE-2013-1287.	6
16587	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CMarkupBehaviorContext Use After Free Vulnerability."	7
16583	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer OnResize Use After Free Vulnerability."	7
16460	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3

16374	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16373	The TCP/IP implementation in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows remote attackers to cause a denial of service (reboot) via a crafted packet that terminates a TCP connection, aka "TCP FIN WAIT Vulnerability."	3
16371	Microsoft Internet Explorer 6 through 9 does not properly perform auto-selection of the Shift JIS encoding, which allows remote attackers to read content from a different (1) domain or (2) zone via a crafted web site that triggers cross-domain scrolling events, aka "Shift JIS Character Encoding Vulnerability."	6
16361	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 8 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to an object that (1) was not properly allocated or (2) is deleted, as demonstrated by a CDwnBindInfo object, and exploited in the wild in December 2012.	5
16379	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16386	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer saveHistory Use After Free Vulnerability."	7
16360	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer CPasteCommand Use After Free Vulnerability."	6
16357	The Print Spooler in Microsoft Windows Server 2008 R2 and R2 SP1 and Windows 7 Gold and SP1 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted print job, aka "Windows Print Spooler Components Vulnerability."	3
16328	Use-after-free vulnerability in Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability."	3
16326	win32k.sys in the kernel-mode drivers in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle window broadcast messages, which allows local users to gain privileges via a crafted application, aka "Win32k Improper Message Handling Vulnerability."	3
16324	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer GetMarkupPtr Use After Free Vulnerability."	7
16320	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3

16332	dxgkrnl.sys (aka the DirectX graphics kernel subsystem) in the kernel-mode drivers in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "DirectX Graphics Kernel Subsystem Double Fetch Vulnerability."	7
16342	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16352	Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3117 and CVE-2013-3124.	3
16349	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16344	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16393	Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3122 and CVE-2013-3124.	3
16396	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1308 and CVE-2013-2551.	8
16443	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16441	The USB kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, and Windows Server 2012 do not properly handle objects in memory, which allows physically proximate attackers to execute arbitrary code by connecting a crafted USB device, aka "Windows USB Descriptor Vulnerability," a different vulnerability than CVE-2013-1286 and CVE-2013-1287.	6
16438	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer SetCapture Use After Free Vulnerability."	6

16448	The kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Windows Kernel Reference Count Vulnerability."	3
16458	Race condition in the kernel in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages incorrect handling of objects in memory, aka "Kernel Race Condition Vulnerability," a different vulnerability than CVE-2013-1278.	3
16436	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16432	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16408	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16399	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16398	Use-after-free vulnerability in Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1313.	3
16412	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
16431	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, Windows 7 Gold and SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3

16415	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1309 and CVE-2013-2551.	8
14717	Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly handle function pointers, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP) or (2) a crafted .NET Framework application, aka ".NET Framework Memory Access Vulnerability."	3
14655	Microsoft .NET Framework 4 does not properly allocate buffers, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP) or (2) a crafted .NET Framework application, aka ".NET Framework Buffer Allocation Vulnerability."	1
15731	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "InjectHTMLStream Use After Free Vulnerability."	1
15580	Microsoft .NET Framework 4 does not properly compare index values, which allows remote attackers to cause a denial of service (application hang) via crafted requests to a Windows Presentation Foundation (WPF) application, aka ".NET Framework Index Comparison Vulnerability."	1
15495	Microsoft .NET Framework 1.0 SP3, 1.1 SP1, 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly validate function parameters, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, or (3) a crafted .NET Framework application, aka ".NET Framework Parameter Validation Vulnerability."	3
15524	Race condition in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 allows local users to gain privileges, and consequently read the contents of arbitrary kernel memory locations, via a crafted application, a different vulnerability than other CVEs listed in MS13-016.	3
15520	Untrusted search path vulnerability in Entity Framework in ADO.NET in Microsoft .NET Framework 1.0 SP3, 1.1 SP1, 2.0 SP2, 3.5, 3.5.1, and 4 allows local users to gain privileges via a Trojan horse DLL in the current working directory, as demonstrated by a directory that contains a .NET application, aka ".NET Framework Insecure Library Loading Vulnerability."	1
16687	Microsoft Internet Explorer 9 and 10, when script debugging is enabled, does not properly handle objects in memory during the processing of script, which allows remote attackers to execute arbitrary code via a crafted web site, aka "Internet Explorer Script Debug Vulnerability."	3
17148	Buffer overflow in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Win32k Buffer Overwrite Vulnerability."	7
17205	Microsoft Internet Explorer 7 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3115.	9

17190	Microsoft Internet Explorer 7 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3162.	9
17188	Buffer overflow in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, and Windows 7 SP1 allows local users to cause a denial of service (system hang) via a crafted application that leverages improper handling of objects in memory, aka "Win32k Buffer Overflow Vulnerability."	7
17088	Microsoft Internet Explorer 8 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3144 and CVE-2013-3163.	9
17071	The Common Language Runtime (CLR) in Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 on 64-bit platforms does not properly allocate arrays of structures, which allows remote attackers to execute arbitrary code via a crafted .NET Framework application that changes array data, aka "Array Allocation Vulnerability."	10
17012	Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3145.	3
16998	The Microsoft WMV video codec in wmv9vcm.dll, wmvdmod.dll in Windows Media Format Runtime 9 and 9.5, and wmvdecod.dll in Windows Media Format Runtime 11 and Windows Media Player 11 and 12 allows remote attackers to execute arbitrary code via a crafted media file, aka "WMV Video Decoder Remote Code Execution Vulnerability."	8
17024	Microsoft Internet Explorer 8 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3151 and CVE-2013-3163.	9
17034	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3153.	9
17376	Microsoft Internet Explorer 8 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability."	5
17363	Microsoft Internet Explorer 8 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3144 and CVE-2013-3151.	9
17360	The EPATHOBJ::pprFlattenRec function in win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, and Windows Server 2012 does not properly initialize a pointer for the next object in a certain list, which allows local users to obtain write access to the PATHRECORD chain, and consequently gain privileges, by triggering excessive consumption of paged memory and then making many FlattenPath function calls, aka "Win32k Read AV Vulnerability."	7

17379	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Vulnerability."	7
17421	Microsoft .NET Framework 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly check the permissions of objects that use reflection, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (XBAP) or (2) a crafted .NET Framework application, aka "Anonymous Method Injection Vulnerability."	10
17353	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Memory Allocation Vulnerability."	7
17273	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Dereference Vulnerability."	7
17259	Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3161.	4
17306	Cross-site scripting (XSS) vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to inject arbitrary web script or HTML via vectors involving incorrect auto-selection of the Shift JIS encoding, leading to cross-domain scrolling events, aka "Shift JIS Character Encoding Vulnerability," a different vulnerability than CVE-2013-0015.	9
17301	Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3143.	4
17293	win32k.sys in the kernel-mode drivers in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, and Windows 7 SP1 does not properly handle objects in memory, which allows local users to gain privileges via a crafted application, aka "Win32k Information Disclosure Vulnerability."	7
16975	Microsoft Internet Explorer 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3146.	1
16763	Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3119.	3
16769	Buffer overflow in win32k.sys in the kernel-mode drivers in Microsoft Windows 7 SP1 allows local users to gain privileges via a crafted application that leverages improper handling of objects in memory, aka "Win32k Buffer Overflow Vulnerability."	7

16815	Microsoft Internet Explorer 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3152.	1
16738	Use-after-free vulnerability in Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code via a crafted web site that triggers access to a deleted object, aka "Internet Explorer Use After Free Vulnerability," a different vulnerability than CVE-2013-1308 and CVE-2013-2551.	5
16704	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3112, CVE-2013-3113, CVE-2013-3121, and CVE-2013-3139.	8
1670	Unspecified vulnerability in the Cryptographic API Component Object Model Certificates ActiveX control (CAPICOM.dll) in Microsoft CAPICOM and BizTalk Server 2004 SP1 and SP2 allows remote attackers to execute arbitrary code via unspecified vectors, aka the "CAPICOM.Certificates Vulnerability."	1
16708	Microsoft Internet Explorer 8 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3123.	8
16727	Microsoft Internet Explorer 8 does not properly handle objects in memory, which allows remote attackers to execute arbitrary code by accessing an object that (1) was not properly allocated or (2) is deleted, as exploited in the wild in May 2013.	8
16720	Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3117 and CVE-2013-3122.	3
16821	Microsoft Internet Explorer 7 and 8 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability."	5
16914	Microsoft Internet Explorer 7 through 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability."	8
16883	DirectShow in Microsoft Windows XP SP2 and SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, and Windows Server 2012 allows remote attackers to execute arbitrary code via a crafted GIF file, aka "DirectShow Arbitrary Memory Overwrite Vulnerability."	8
16927	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3148.	9
16966	Microsoft Internet Explorer 6 through 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability."	8
16943	Integer overflow in the TCP/IP kernel-mode driver in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT allows remote attackers to cause a denial of service (system hang) via crafted TCP packets, aka "TCP/IP Integer Overflow Vulnerability."	7

16875	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3112, CVE-2013-3113, CVE-2013-3139, and CVE-2013-3142.	8
16847	The kernel in Microsoft Windows XP SP3, Windows Server 2003 SP2, Windows Vista SP2, Windows Server 2008 SP2, Windows 7 SP1, and Windows 8 on 32-bit platforms does not properly handle unspecified page-fault system calls, which allows local users to obtain sensitive information from kernel memory via a crafted application, aka "Kernel Information Disclosure Vulnerability."	7
16837	The Print Spooler in Microsoft Windows Vista SP2, Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8, Windows Server 2012, and Windows RT does not properly manage memory during deletion of printer connections, which allows remote authenticated users to execute arbitrary code via a crafted request, aka "Print Spooler Vulnerability."	8
16824	Microsoft Internet Explorer 6 through 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3112, CVE-2013-3121, CVE-2013-3139, and CVE-2013-3142.	8
16872	Microsoft Internet Explorer 9 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3150.	3
16867	The serialization functionality in Microsoft .NET Framework 2.0 SP2, 3.5, 3.5 SP1, 3.5.1, 4, and 4.5 does not properly check the permissions of delegate objects, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (XBAP) or (2) a crafted .NET Framework application that leverages a partial-trust relationship, aka "Delegate Serialization Vulnerability."	10
16860	Microsoft Internet Explorer 9 and 10 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site, aka "Internet Explorer Memory Corruption Vulnerability," a different vulnerability than CVE-2013-3114.	3
17430	Microsoft .NET Framework 1.0 SP3, 1.1 SP1, 2.0 SP2, 3.5, 3.5.1, 4, and 4.5 does not properly check the permissions of objects that use reflection, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (XBAP) or (2) a crafted .NET Framework application, aka "Delegate Reflection Bypass Vulnerability."	10
11815	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1
11798	Unspecified vulnerability in the Swing component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is related to unsafe reflection involving the UIDefault.ProxyLazyValue class.	1
11871	Unspecified vulnerability in the Java Web Start component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1

11880	Unspecified vulnerability in the Sound component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable researcher that this involves an incorrect sign extension in the HeadspaceSoundbank.nGetName function, which allows attackers to execute arbitrary code via a crafted BANK record that leads to a buffer overflow.	1
11893	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is a double free vulnerability in IndexColorModel that allows remote attackers to cause a denial of service (crash) and possibly execute arbitrary code.	1
11574	Use-after-free vulnerability in Microsoft Internet Explorer 6, 7, and 8 allows remote attackers to execute arbitrary code via vectors related to Cascading Style Sheets (CSS) token sequences and the clip attribute, aka an "invalid flag reference" issue or "Uninitialized Memory Corruption Vulnerability," as exploited in the wild in November 2010.	5
11619	Unspecified vulnerability in the Java Web Start component in Oracle Java SE and Java for Business 6 Update 21 and 5.0 Update 25 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1
11548	Use-after-free vulnerability in an unspecified compatibility component in Adobe Shockwave Player before 11.5.9.620 allows user-assisted remote attackers to execute arbitrary code via a crafted web site, related to the Shockwave Settings window and an unloaded library. NOTE: some of these details are obtained from third party information.	6
11714	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, and 5.0 Update 25 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is related to a calculation error in right-to-left text character counts for the ICU OpenType font rendering implementation, which triggers an out-of-bounds memory access.	1
12078	An unspecified function in TextXtra.x32 in Adobe Shockwave Player before 11.5.9.615 does not properly reallocate a buffer when processing a DEMX chunk in a Director file, which allows remote attackers to trigger a heap-based buffer overflow and execute arbitrary code.	6
12035	Untrusted search path vulnerability in Adobe Dreamweaver CS5 11.0 build 4916, build 4909, and probably other versions, allows local users, and possibly remote attackers, to execute arbitrary code and conduct DLL hijacking attacks via a Trojan horse (1) mfc90loc.dll or (2) dwmapi.dll that is located in the same folder as a CSS, PHP, ASP, or other file that automatically launches Dreamweaver.	4
12173	Unspecified vulnerability in the Deployment Toolkit component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1
12105	Microsoft .NET Framework 2.0 SP1 and SP2, 3.5 Gold and SP1, 3.5.1, and 4.0, and Silverlight 4 before 4.0.60531.0, does not properly validate arguments to unspecified networking API functions, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, (3) a crafted .NET Framework application, or (4) a crafted Silverlight application, aka ".NET Framework Array Offset Vulnerability."	1

12111	The asyncore module in Python before 3.2 does not properly handle unsuccessful calls to the accept function, and does not have accompanying documentation describing how daemon applications should handle unsuccessful calls to the accept function, which makes it easier for remote attackers to conduct denial of service attacks that terminate these applications via network connections.	27
12029	Unspecified vulnerability in the Java Runtime Environment component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, and 1.4.2_27 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is a race condition related to deserialization.	1
11990	Unspecified vulnerability in the Networking component in Oracle Java SE and Java for Business 6 Update 21 and 5.0 Update 25 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is related to missing validation of request headers in the HttpURLConnection class when they are set by applets, which allows remote attackers to bypass the intended security policy.	1
12007	** RESERVED ** This candidate has been reserved by an organization or individual that will use it when announcing a new security problem. When the candidate has been publicized, the details for this candidate will be provided.	1
12005	Unspecified vulnerability in the Networking component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality via unknown vectors.	1
12004	Unspecified vulnerability in the New Java Plug-in component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1
11268	Unspecified vulnerability in the Swing component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is related to the modification of "behavior and state of certain JDK classes" and "mutable static."	1
11330	Unspecified vulnerability in the Networking component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, and 1.4.2_27 allows remote attackers to affect confidentiality via unknown vectors.	1
11320	Unspecified vulnerability in the Deployment component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the January 2011 CPU. Oracle has not commented on claims from a reliable third party coordinator that the ActiveX Plugin does not properly initialize an object field that is used as a window handle, which allows attackers to execute arbitrary code.	1
1039	Buffer overflow in a component of SQL-DMO for Microsoft Data Access Components (MDAC) 2.5 through 2.7 allows remote attackers to execute arbitrary code via a long response to a broadcast request to UDP port 1434.	184
12177	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable researcher that this is an integer overflow in the color profile parser that allows remote attackers to execute arbitrary code via a crafted Tag	1

	structure in a color profile.	
12180	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, and 1.4.2_27 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable researcher that this is an integer overflow that triggers memory corruption via large values in a subsample of a JPEG image, related to JPEGImageWriter.writeImage in the imageio API.	1
13069	Microsoft .NET Framework 1.0 SP3, 1.1 SP1, 2.0 SP2, 3.5.1, and 4, and Silverlight 4 before 4.0.60831, does not properly restrict inheritance, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, (3) a crafted .NET Framework application, or (4) a crafted Silverlight application, aka ".NET Framework Class Inheritance Vulnerability."	1
12817	Microsoft Internet Explorer 6 through 8 allows remote attackers to spoof the address bar, via window.open with a relative URI, to show an arbitrary URL on the web site visited by the victim, as demonstrated by a visit to an attacker-controlled web page, which triggers a spoofed login form for the site containing that page.	5
12829	Stack-based buffer overflow in the AddFavorite method in Microsoft Internet Explorer allows remote attackers to cause a denial of service (application crash) and possibly have unspecified other impact via a long URL in the first argument.	5
12901	Microsoft .NET Framework 2.0 SP2, 3.5.1, and 4 does not properly validate the System.Net.Sockets trust level, which allows remote attackers to obtain sensitive information or trigger arbitrary outbound network traffic via (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, or (3) a crafted .NET Framework application, aka "Socket Restriction Bypass Vulnerability."	1
12355	The printing functionality in Microsoft Internet Explorer 8 allows remote attackers to discover a local pathname, and possibly a local username, by reading the dc:title element of a PDF document that was generated from a local web page.	5
12406	The x86 JIT compiler in Microsoft .NET Framework 2.0 SP2, 3.5 SP1, 3.5.1, and 4.0 does not properly compile function calls, which allows remote attackers to execute arbitrary code via (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, or (3) a crafted .NET Framework application, aka ".NET Framework Stack Corruption Vulnerability."	1
12225	Unspecified vulnerability in the 2D component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update and 25 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable researcher that this is an integer overflow that leads to a buffer overflow via a crafted devs (device information) tag structure in a color profile.	1
12200	Unspecified vulnerability in the CORBA component in Oracle Java SE and Java for Business 6 Update 21 and 5.0 Update 25 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this involves the use of the privileged accept method in the ServerSocket class, which does not limit which hosts can connect and allows remote attackers to bypass intended network access restrictions.	1

12181	Unspecified vulnerability in the Deployment component in Oracle Java SE and Java for Business 6 Update 21 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable researcher that this is related to "how Web Start retrieves security policies," BasicServiceImpl, and forged policies that bypass sandbox restrictions.	1
12189	Unspecified vulnerability in the CORBA component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this is related to "permissions granted to certain system objects."	1
12199	Adobe Shockwave Player before 11.5.9.615 allows attackers to execute arbitrary code or cause a denial of service (memory corruption) via unspecified vectors.	6
12226	Unspecified vulnerability in the Java Runtime Environment component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, and 1.4.2_27 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that this allows remote attackers to execute arbitrary code by causing the defaultReadObject method in the Serialization API to set a volatile field multiple times.	1
12229	Unspecified vulnerability in the Networking component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors. NOTE: the previous information was obtained from the October 2010 CPU. Oracle has not commented on claims from a reliable downstream vendor that HttpURLConnection does not properly check for the allowHttpTrace permission, which allows untrusted code to perform HTTP TRACE requests.	1
12240	Unspecified vulnerability in the Sound component in Oracle Java SE and Java for Business 6 Update 21, 5.0 Update 25, 1.4.2_27, and 1.3.1_28 allows remote attackers to affect confidentiality, integrity, and availability via unknown vectors.	1
12638	The XSS Filter in Microsoft Internet Explorer 8 does not properly perform neutering for the SCRIPT tag, which allows remote attackers to conduct cross-site scripting (XSS) attacks against web sites that have no inherent XSS vulnerabilities, a different issue than CVE-2009-4074.	5
12700	mshtml.dll in Microsoft Internet Explorer 7 and 8 on Windows XP SP3 allows remote attackers to cause a denial of service (application crash) by calling the JavaScript findText method with a crafted Unicode string in the first argument, and only one additional argument, as demonstrated by a second argument of -1.	5
12686	The JIT compiler in Microsoft .NET Framework 3.5 Gold and SP1, 3.5.1, and 4.0, when IsJITOptimizerDisabled is false, does not properly handle expressions related to null strings, which allows context-dependent attackers to bypass intended access restrictions, and consequently execute arbitrary code, in opportunistic circumstances by leveraging a crafted application, as demonstrated by (1) a crafted XAML browser application (aka XBAP), (2) a crafted ASP.NET application, or (3) a crafted .NET Framework application, aka ".NET Framework JIT Optimization Vulnerability."	1
12693	Microsoft msxml.dll, as used in Internet Explorer 8 on Windows 7, allows remote attackers to obtain potentially sensitive information about heap memory addresses via an XML document containing a call to the XSLT generate-id XPath function. NOTE: this might overlap CVE-2011-1202.	5

12566	Microsoft Windows does not properly warn the user before enabling additional Human Interface Device (HID) functionality over USB, which allows user-assisted attackers to execute arbitrary programs via crafted USB data, as demonstrated by keyboard and mouse data sent by malware on a smartphone that the user connected to the computer.	184
7378	Untrusted search path vulnerability in ATL MFC Trace Tool (AtlTraceTool8.exe), as used in Microsoft Visual Studio, allows local users, and possibly remote attackers, to execute arbitrary code and conduct DLL hijacking attacks via a Trojan horse dwmapi.dll that is located in the same folder as a TRC, cur, rs, rct, or res file.	48
3556	The Microsoft .NET forms authentication capability for ASP.NET allows remote attackers to bypass authentication for .aspx files in restricted directories via a request containing a (1) "" (backslash) or (2) "%5C" (encoded backslash), aka "Path Validation Vulnerability."	9
6778	Untrusted search path vulnerability in Adobe PhotoShop CS2 through CS5 allows local users, and possibly remote attackers, to execute arbitrary code and conduct DLL hijacking attacks via a Trojan horse dwmapi.dll or Wintab32.dll that is located in the same folder as a PSD or other file that is processed by PhotoShop. NOTE: some of these details are obtained from third party information.	4
6824	The JIT compiler in Microsoft .NET Framework 4.0 on 64-bit platforms does not properly perform optimizations, which allows remote attackers to execute arbitrary code via a crafted .NET application that triggers memory corruption, aka ".NET Framework x64 JIT Compiler Vulnerability."	1
694	Buffer overflow in Microsoft Visual Basic for Applications (VBA) SDK 6.0 through 6.4, as used by Microsoft Office 2000 SP3, Office XP SP3, Project 2000 SR1, Project 2002 SP1, Access 2000 Runtime SP3, Visio 2002 SP2, and Works Suite 2004 through 2006, allows user-assisted attackers to execute arbitrary code via unspecified document properties that are not verified when VBA is invoked to open documents.	37
5847	Cross-domain vulnerability in Microsoft XML Core Services 3.0 through 6.0, as used in Microsoft Expression Web, Office, Internet Explorer, and other products, allows remote attackers to obtain sensitive information from another domain and corrupt the session state via HTTP request header fields, as demonstrated by the Transfer-Encoding field, aka "MSXML Header Request Vulnerability."	6