

# A Secure Patch Management Authority

*By*

Andrew Michael Colarik

A thesis submitted in partial fulfilment  
of the requirements for the degree of  
Doctor of Philosophy,  
University of Auckland, 2003

# ABSTRACT

Throughout the Software Development Life Cycle, products are maintained, enhanced and periodically patched until they are retired or replaced. It is because of the way in which software is initially deployed that ongoing maintenance becomes a critical element in extending a software product's life cycle. It is the ongoing, incremental development process, and an interactive dependence of system platforms, interoperability requirements, and time-to-market pressures that inadvertently produce software defects.

In addition to faults and systematic improvements of product features that are required of software products, security breaches are periodically discovered. The result is the issuing of software patches. These patches have become so frequent that it has become a principle system management issue.

A systems administrator who is responsible for security measures needs to track, install said updates, and document their corrective actions. The time between the discovery of a software breach and the resulting patch creates system vulnerability. The time between the issuance of the patch and the resulting install creates system vulnerability. The possible corruption of the patch prior to the delivery phase and/or during delivery creates additional vulnerabilities. The difficulties of identifying, acquiring, and installing patches is further exacerbated by the multitude of software vendors that do not work in tandem, and operate their own patch distribution systems.

Patch management systems contain elements of software management, patch generation and accessibility, and delivery mechanisms. Secure patch management systems focus primarily on the secure delivery of the patch. This thesis provides an exploration of

patch management models, methods, and systems, and seeks to identify the underlying processes of secure patch management systems.

What is proposed and detailed is a secure patch management authority architecture as one possible solution to improving the time between patch issuance and installation. This authority will work with patch originators to provide timely notification of new patches, facilitate the distribution of said patches in a secure manner, and provide patch implementation confirmation and error reporting back to the patch originators. A critical component of this architecture is a new cryptographic algorithm that provides integrity verification and error control triangulation in the transport of the patches.

It takes a lot of courage to release the familiar and seemingly secure, to embrace the new. But there is no real security in what is no longer meaningful. There is more security in the adventurous and exciting, for in movement there is life, and in change there is power.

- Allen Cohen

# ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for it is He who gave me what talents, abilities, and insights that I may have when I was granted life. It is He who has watched over me in my best and worst moments, and as such, has always opened the way to what future may yet await me.

There are several people who have impacted my time spent working through the PhD process, and I would like to thank them here for their support and contributions.

To Professor Lech Janczewski for guiding, motivating, challenging, and mentoring me throughout my doctoral apprenticeship.

To Professor Clarke Thomborson for his friendship and also for teaching me focus, conceptual refinement, and academic scholarship.

To Professors Justo Diaz and Ananth Srinivasan for their friendship, the former for helping me start the journey, and the latter for his patience and leadership in helping me complete it.

To Dr. Jairo Gutierrez for his friendship and help in pointing to technological resources and assessment.

To Professor Brent Gallupe from Queen's University and Professor Guttorm Sindre from Norwegian University of Science and Technology for their review and comments on early drafts of my thesis.

To Dr. David Sundaram for his friendship, tremendous kindness and generosity throughout this journey.

I would also like to acknowledge all the support and friendship over the years of people too innumerable to mention here.

# TABLE OF CONTENTS

Abstract .....	ii
Acknowledgements .....	v
Table of Contents .....	vi
List of Tables .....	ix
List of Figures .....	x
Abbreviations .....	xii
1. Introduction .....	1
1.1 Concept Definitions and Distinctions .....	2
1.2 Patch Management Approaches and Systems .....	4
1.3 Germane Models .....	5
1.4 Research Opportunities .....	6
1.4.1 Strengths .....	7
1.4.2 Problems and Issues .....	8
1.4.3 Opportunities .....	9
1.4.3.1 Vulnerability Life Cycle .....	10
1.4.3.2 Patch Volume and Costs .....	11
1.4.3.3 Trusted Third Parties .....	12
1.4.3.4 Cyber Insurance .....	14
1.5 Research Objectives .....	15
1.6 Research Contributions .....	16
1.7 Organization of the dissertation .....	17
2. Review of the Literature .....	18
2.1 Germane Models .....	18
2.1.1 Life Cycle Models .....	19
2.1.2 Delivery and Distribution .....	25
2.1.3 Issues in the Chapter Literature .....	32
2.2 Patch Management Approaches .....	33
2.3 Security Environment .....	36
2.3.1 Security Services & Mechanisms .....	37
2.3.1.1 Authentication .....	37
2.3.1.2 Access Control .....	40
2.3.1.3 Data Integrity .....	41
2.3.1.4 Data Confidentiality .....	44
2.3.1.5 Non-Repudiation .....	46
2.3.1.6 Audit .....	47
2.3.2 File Transfer Protocol .....	48
2.3.3 SPMA Middleware Platform .....	50
2.3.3.1 Distributed Component Object Model .....	51
2.3.3.2 Remote Method Invocation .....	53
2.3.3.3 Common Object Request Broker Architecture .....	54
2.3.3.4 Java 2 Enterprise Edition .....	57
2.3.3.5 Web Services .....	58
2.3.3.6 Platform Comparisons .....	59
2.4 Summary .....	60

3. Methodology .....	62
3.1 Multi-methodological Approach.....	63
3.2 Summary .....	66
4. Industry Patents.....	67
4.1 Architectural Recovery .....	67
4.1.1 Search Criteria .....	69
4.1.2 Taxonomy Phases .....	70
4.1.3 Security Provisions .....	72
4.1.4 Notable Features for Inclusion in SPMA Architecture.....	75
4.1.5 Development of a Unified Model .....	76
4.1.5.1 Component Descriptions.....	81
4.2 Limitations .....	83
4.3 Summary of Implications.....	83
5. SPMA Framework Design.....	87
5.1 SPMA Model .....	87
5.2 SPMA Architecture.....	88
5.2.1 End-User Host System.....	90
5.2.2 Secure Repository .....	93
5.2.3 Patch Originator .....	96
5.2.4 Process Scenarios.....	98
5.2.4.1 Patch Issuance, Notification & Transfer from PO to SR Processes .....	99
5.2.4.2 EUHS Patch Availability Request & Patch Transfer Processes .....	109
5.2.4.3 SR Error and Reporting Information Transfer to PO Processes .....	114
5.2.5 Public Key Infrastructure / Trusted Third Party Component.....	117
5.3 Quality Characteristics and Guidelines.....	118
5.4 Summary .....	120
6. SPMA Implementation .....	121
6.1 Hash Triplet Description.....	121
6.2 Proof of concept generation .....	126
6.3 Summary .....	128
7. SPMA Evaluation .....	129
7.1 Multi-methodological Method.....	129
7.2 System Development Research Process .....	130
7.3 Quality Characteristics.....	131
7.4 Summary .....	132
8. Contributions, Limitations, Future Research, and Conclusions .....	134
8.1 Contributions.....	134
8.2 Limitations .....	135
8.3 Future Research .....	136
8.4 Conclusions.....	137
Appendix A.....	139
A.1 Client Initiated Transport Service .....	139
A.2 Client Initiated Update Service .....	140
A.3 E-mail Initiated Update Service.....	145
A.4 Client Initiated Broadcast.....	147
A.5 Client Requested Server Notification Broadcast .....	148
A.6 Client Initiated Configuration Management .....	149

A.7 Agent Based Software Distribution .....	150
Appendix B .....	151
B.1 Patent Applications .....	151
B.2 Patents .....	152
Appendix C .....	155
C.1 FileRepositoryClient Source Code.....	155
C.2 FileRepositoryServer Source Code .....	160
Bibliography .....	165

# LIST OF TABLES

Table 1: Overview of Life Cycle Literature Review .....	5
Table 2: Patch Delivery and Distribution Models .....	6
Table 3: Previously Reviewed Delivery & Distribution Items .....	25
Table 4: Typical Approaches to Patch Management .....	35
Table 5: Middleware Platform Standards Security Services .....	60
Table 6: Protocol Taxonomy (5 sub-taxonomies) .....	70
Table 7: TOGAF Security Services Guidelines (Open Group, 2003) .....	73
Table 8: Security Provisions by Taxonomy Protocol Phase .....	74
Table 9: Sub-characteristics of Software Quality (ISO, 2001) .....	119
Table 10: Hash Triplet Send Request Response .....	122
Table 11: Received Patch File Response .....	123
Table 12: Quality Sub-Characteristic Evaluation of SPMA .....	131

# LIST OF FIGURES

Figure 1: Vulnerability Exploit Cycle (CERT, 2002).....	11
Figure 2: Reported Vulnerabilities (CERT/CC, 2003) .....	11
Figure 3: Secure Patch Management Authority Model .....	16
Figure 4: Simple Staged Model (Rajlich & Bennett, 2000).....	20
Figure 5: Versioned Staged Model (Rajlich & Bennett, 2000) .....	20
Figure 6: Unified Configuration Management Approach (Dart, 2000).....	23
Figure 7: Download Service Hierarchy (Symborski, 1988) .....	26
Figure 8: Active Criteria for a Distributed Update (Segal & Frieder, 1989).....	27
Figure 9: SSDS Architecture (Bartoletti, Dobbs & Kelley, 1997) .....	28
Figure 10: Cro-Magnon Engine Architecture (Bargen & Taplin, 1999) .....	29
Figure 11: Common Rbone with Multiple Rendezvous Points (Li, 2002).....	30
Figure 12: Secure Patch Management System Framework (Janczewski & Colarik, 2003ii) ..	31
Figure 13: Distributed Component Object Model Architecture (Microsoft, 1996).....	52
Figure 14: Architecture of Remote Method Invocations (Emmerich, 2000).....	53
Figure 15: Object Request Broker Structure (Object Management Group, 1995i) .....	54
Figure 16: CORBA Layers (Lang, Gollmann & Schreiner, 2001).....	55
Figure 17: CORBA ORB Security Services (Object Management Group, 2002ii) .....	56
Figure 18: Java 2 Enterprise Edition Architectural Diagram (Shannon, 2001).....	57
Figure 19: Multi-method Approach to IS Research (Nunamaker, Chen & Purdin, 1991) .....	63
Figure 20: A Model of Knowledge Construction (Ostwald, 1996) .....	64
Figure 21: Process for System Development Research (Nunamaker, Chen & Purdin, 1991) ..	65
Figure 22: Framework Component Separation – EUHS .....	77
Figure 23: Framework Component Separation – PO.....	77
Figure 24: Functional Clustering – EUHS.....	78
Figure 25: Functional Clustering - PO.....	78
Figure 26: Refined Functional Clustering – EUHS .....	79
Figure 27: Refined Functional Clustering – PO .....	80
Figure 28: Unified Patch Management System Model.....	81
Figure 29: Secure Patch Management Authority Model .....	88
Figure 30: Research Hierarchy for Secure Patch Management Authority .....	89
Figure 31: SPMA End-User Component Model.....	91
Figure 32: SPMA Secure Repository Component Model.....	94
Figure 33: SPMA Patch Originator Component Model .....	96
Figure 34: Event-Driven Process Elements .....	98
Figure 35: Patch and Hash Presentation to the SPMA System.....	99
Figure 36: Patch Originator Inventory Evaluation.....	100
Figure 37: Patch Originator Transmit Notification.....	102
Figure 38: Secure Repository Message Received by Patch Originator .....	104
Figure 39: Secure Repository Inventory Evaluation.....	105
Figure 40: Secure Repository Patch Request from Patch Originator .....	106
Figure 41: Patch Originator Patch File Transmission to Secure Repository .....	107
Figure 42: Secure Repository Patch File Received from Patch Originator .....	108
Figure 43: End-User Host System Software Information Collection .....	109
Figure 44: End-User Host System Patch Availability Request to the Secure Repository .....	110

Figure 45: Secure Repository Patch Availability Request from End-User Host System .....	111
Figure 46: Secure Repository Patch Availability Evaluation for End-User Host System.....	112
Figure 47: Secure Repository Patch Transmission .....	112
Figure 48: End-User Host System Patch Package Received .....	113
Figure 49: Secure Repository Evaluate Error & Reporting Logs for Transmission.....	114
Figure 50: Secure Repository Transmission of Log Information .....	115
Figure 51: Patch Originator Receives Logs from Secure Repository .....	115
Figure 52: Error Correction Process .....	116
Figure 53: Public Key Infrastructure Entities (Nykanen, 2000).....	117
Figure 54: Six Characteristics of Software Quality (ISO, 2001).....	118
Figure 55: Conceptual Data Transfer Sequence Diagram .....	124
Figure 56: Proof-of-Concept Data Sequence Transfer Diagram .....	127
Figure 57: Client Initiated Transport Service [P5] .....	139
Figure 58: Client Initiated Update Service 1 [A2].....	140
Figure 59: Client Initiated Update Service 2 [P9, P14, P15, P16, P20] .....	140
Figure 60: Client Initiated Update Service 3 [P12] .....	141
Figure 61: Client Initiated Update Service 4 with Back-out limit [P1] .....	141
Figure 62: Client Initiated Update Service 5 [P24] .....	141
Figure 63: Client Initiated Update Service 6 with Separate Server Maintenance [P13] .....	142
Figure 64: Client Initiated Update Service 7 [P6] .....	142
Figure 65: Client Initiated Update Service 8 [P2, P3, P7].....	143
Figure 66: Client Initiated Update Service 9 [P23] .....	143
Figure 67: Client Initiated Update Service 10 [P22, A7, A8].....	144
Figure 68: Client Initiated Update Service 11 [A4].....	144
Figure 69: Client Initiated Update Service 12 [A5].....	145
Figure 70: Client Initiated Update Service 13 [A9].....	145
Figure 71: E-mail Initiated Update Service [A3].....	146
Figure 72: Client Initiated Mailbox Transmission [P8].....	146
Figure 73: Client Initiated Broadcast – Wireless 1 [P17].....	147
Figure 74: Client Initiated Broadcast – Wireless 2 [P19].....	147
Figure 75: Client Initiated Broadcast Service [P10, P18].....	148
Figure 76: Client Requested Server Notification Broadcast [A1] .....	148
Figure 77: Client Initiated Configuration Management 1 [P4].....	149
Figure 78: Client Initiated Configuration Management 2 [P21].....	149
Figure 79: Agent Based Software Distribution 1 [P11].....	150
Figure 80: Agent Based Software Distribution 2 [A6].....	150

# ABBREVIATIONS

ACL	- Access Control List
AH	- Authentication Header
API	- Application Program Interfaces
CA	- Certificate Authority
CCMC	- Common Criteria Management Committee
COM	- Component Object Model
CORBA	- Common Object Request Broker Architecture
DAP	- Directory Access Protocol
DCE	- Distributed Computing Environment
DCOM	- Distributed Component Object Model
DN	- Distinguished Name
DNS	- Domain Name Server
DSA	- Directory System Agent
DSP	- Directory System Protocol
DUA	- Directory User Agent
EIA	- Electronic Industries Association
EJB	- Enterprise JavaBeans
ESH	- Encapsulating Security Header
EUHS	- End-User Host System
FTP	- File Transfer Protocol
GUI	- Graphical User Interface
GSS	- General Security Services Application Program Interface
HTTP	- Hypertext Transfer Protocol
IDL	- Interface Definition Language
IETF	- Internet Engineering Task Force
IP	- Internet Protocol
IPSec	- Internet Protocol Security
ISO	- International Organization for Standardization
JAAS	- Java Authentication and Authorization Service
JDBC	- Java Database Connectivity
JVM	- Java Virtual Machine
J2EE	- Java 2 Enterprise Edition
LDAP	- Lightweight Directory Access Protocol
MAC	- Message Authentication Code
MD5	- Message Digest 5
MOM	- Message-Oriented Middleware
ORB	- Object Request Broker
PAM	- Pluggable Authentication Module
PCIPB	- President's Critical Infrastructure Protection Board
PCMS	- Patch Communication Management Service
PDA	- Personal Digital Assistant
PGP	- Pretty Good Privacy
PKI	- Public Key Infrastructure
PO	- Patch Originator

PR	- Patch Recipient
P2P	- Peer-to-Peer
RMI	- Remote Method Invocation
RPC	- Remote Procedure Call
SCM	- Software Configuration Management
SDLC	- Software Development Life Cycle
SDLCP	- Software Development Life Cycle Process
SAS	- Security Attribute Service
SLC	- Software Life Cycle
SOAP	- Simple Object Access Protocol
SPMA	- Secure Patch Management Authority
SPMS	- Secure Patch Management System
SR	- Secure Repository
SSDS	- Secure Software Distribution System
SSL	- Secure Socket Layer
SQL	- Structured Query Language
TAFIM	- Technical Architecture Framework for Information Management
TCP	- Transport Control Protocol
TGS	- Ticket Granting Server
TOGAF	- The Open Group Architectural Framework
TTP	- Trusted Third Party
UDDI	- Universal Description Discovery Integration
URL	- Universal Resource Locator
WSDL	- Web Services Description Language
XDS	- X/Open Directory Service
XML	- Extensible Mark-up Language
XOM	- X/Open Management

# 1. INTRODUCTION

Today's software is released with the expectation that additional service packs, patches and new versions will be introduced as part of the service support mechanism. When it is discovered that the software product has a bug, flaw or security breach, a patch is issued. Once a patch is issued, the concern then shifts to getting the patch to the end-user for installation. Thus, in order to extend the software life cycle, the development of Patch Management Systems has become an area of study and system development.

Patch Management Systems are a combination of hardware, software, people, and procedural components working together to apply incremental fixes to a set of programs. A patch is an actual piece of object code that replaces an executable program or is inserted into the program code. Specific patches are developed to update operating systems, utilities, and application software when the original code is found to be flawed or insecure.

With the continued expansion of web-based user support systems, computer users are increasingly being expected to acquire their patches through the Internet. At its foundation, the Internet is an open-architecture distributed system (Crocker, 1982). Due to modern security issues that continue to evolve, a Secure Patch Management System (SPMS) becomes increasingly necessary to ensure the authentication of the identity of the patch's originator prior to downloading the file and verifying the integrity of the patch code prior to installation. In addition, the time delay between the issuance of a patch and its installation may allow intruders to exploit a known system-wide vulnerability. As a result, timely notification becomes a key component to a SPMS (Janczewski & Colarik, 2003ii). What is needed is the design and implementation of a Secure Patch Management Authority (SPMA) that provides timely notification and secured delivery of patches.

The following sections introduce concept definitions, specific reasons for patch issuance, a brief description of patch management approaches and systems, a brief description of the germane models, and the opportunities and objectives of this research.

## 1.1 Concept Definitions and Distinctions

Before a discussion of the relevant models and systems can be conducted, a clarification of definitions and their distinctions must first be presented to provide a better understanding of the subject domain. In this section, the concepts of a patch, an update, patch management, secure patch management, and their distinctions will be presented.

A patch is an actual piece of object code that replaces an executable program or is inserted into the program code. It is “a piece of code added to software in order to fix a bug, especially as a temporary correction between two releases” (Johnson, 2000). A patch is built to correct security, stability, increased performance, and functionality in a software product in reaction to defects, systems failures, or to proactively avoid a known problem (Hewlett-Packard, 2001).

An update is defined as information that updates something, the act or an instance of bringing something up to date, and an updated version of something (Johnson, 2000). This definition, therefore, can be considered to overlap the definition of a patch. In the Versioned Staged Model proposed by Rajlich and Bennett (2000), there are service patches and there are evolution versions, and this further overstates the activity of patches. Therefore, for purposes of this research, an update shall be considered as a collection of patches (i.e. a service pack), and a new version of an existing software release shall be considered an evolution.

Management is defined as “the act, manner, or practice of managing, handling, supervision, or control” (Johnson, 2000). Therefore, for purposes of this research, patch

management shall be considered the managing, handling, supervision, and/or control of patches.

Janczewski and Colarik (2003ii) propose that secure patch management is defined as a system that contains the following:

1. A patch event notification that informs the patch recipient about the availability of a new software patch release;
2. An integrity and authentication check of the patch event notification providing assurance to the patch recipient that the notification about the patch in question is genuine and has not been modified;
3. A patch version verification to ensure that the patch is applicable;
4. A patch transport and delivery mechanism and/or procedures allowing the transport of a patch from the patch originator to the patch recipient;
5. A patch file integrity verification that checks for any intentional or unintentional modifications to the patch;
6. A trial installation of the patch to determine if the prospective patch is well suited to the application and/or system environment;
7. A patch deployment resulting in the permanent installation of the patch on the end-user host system after the trial test is validated; and
8. A recipient audit component for the verification and recording of all communications and transactions.

A SPMS is an extension of patch management in that it encompasses additional security measures and procedures to validate the steps in the entire process. With regards to this thesis, a Secure Patch Management Authority is a system comprised of a secure repository that serves as an intermediary between the patch originators and the end-user host systems, a

Public Key Infrastructure (PKI) Trusted Third Party (TTP) for securing the communications between the parties, and provides items 1-5 and item 8 of a SPMS. Therefore, a SPMA is primarily concerned with the notification and delivery of patches in a secure manner.

## 1.2 Patch Management Approaches and Systems

As an outcome of the literature review in Chapter 2, five primary categories of patch management have been identified. These are as follows:

- Advisory Materials that rely on individuals to procedurally systematize their systems through standards and policy controls in the acquisition and installation of patches.
- E-mail Subscription Services that issue E-mail notices containing details of the application of a patch and/or the patch itself to those belonging to the subscription service.
- Remote Maintenance Services that provide technical personnel that remotely access systems in order to effect patch selection and deployment. These services typically provide their own infrastructure for timely notification and acquisition of the patches through a knowledge base of the client's system versions and requirements.
- Automated Software Modules that consist of a sub-component contained within an operating system or application suite.
- Third Party Applications that are stand-alone software packages that perform the acquisition and updating process.

There are numerous products and services that provide patch management, and examples of systems in each of these categories will be identified as part of the literature review in Chapter 2.

## 1.3 Germane Models

A preliminary literature review was conducted to better identify the broad topical domain of patch management systems. Conceptually, patch management systems come into existence only after software is developed and put into production. Their primary purpose is to extend the functional life cycle of the product. Accordingly, a discussion of the Software Development Life Cycle (the steps and activities throughout a software's life), Software Development Life Cycle Process (the processes required to engineer software), and other contributing sub-categories towards the management of software needs to be presented first in order to illustrate an initial high-level of abstraction, and then narrow the scope downward with regards to applicable patch management system models. The literature has been organized as a general overview beginning with the Software Life Cycle, followed by the Software Life Cycle Process, Maintenance, Configuration Management, and Delivery and Distribution (see Table 1).

Model	Author(s)	Year	Software Life Cycle	Software Life Cycle Process	Maintenance	Configuration Management	Delivery & Distribution
Waterfall	Boehm	1981	X				
Spiral	Boehm	1988	X				
Divided Software Life Cycle	Brehm & Markus	2000	X				
Simple Staged	Rajlich & Bennet	2000	X				
Version Staged	Rajlich & Bennet	2000	X				
Process Cycle	Madhavji	1991		X			
Enterprise	Dutton	1993		X			
Relation of Primary Life Cycle	IEEE	1997		X	Sub-process	Supporting process	
Process-Centered Software Lifecycle	Gamalel-Din & Osterweil	1988			X		
Compiler-Integrated Software Management	Krauser	1991			X		
Design Maintenance System Concept	Baxter & Pidgeon	1997			X		
MANDAS Framework	Bauner et al.	1997				X	Agents
Configuration Management Tool	Dart	2000				X	Connected process

**Table 1: Overview of Life Cycle Literature Review**

While these models do not provide detailed methods and approaches to patch management, they do instead address the role that patches and software evolutions play in the life of a software product. In Chapter 2, a review and discussion of these models will be presented. In the explicit area of delivery and distribution, there are six primary models that contribute to the patch management domain (see Table 2).

<b>Model</b>	<b>Author(s)</b>	<b>Year</b>
Download Service Hierarchy	Symborski	1988
Active Criteria for a Distributed Update	Segal & Frieder	1989
Secure Software Distribution System	Bartoletti, Dobbs & Kelley	1997
Cro-Magnon	Bargen & Taplin	1999
Revere	Li	2002
Secure Patch Management System Framework	Janczewski & Colarik	2003ii

**Table 2: Patch Delivery and Distribution Models**

In general, these models propose differing approaches to identifying, acquiring, and delivering software patches and evolutions, but are lacking many functional and security features needed in a SPMA. The models in Tables 1 and 2 will be discussed in further detail in the literature review in Chapter 2.

## 1.4 Research Opportunities

With regards to patching system vulnerabilities, Bruce Schneier (2001) identified five phases in the window of exposure. These are as follows:

- Before the vulnerability is discovered
- After the vulnerability is discovered, but before it is announced
- After the vulnerability is announced
- After an automatic attack tool is written for the vulnerability
- After a vendor patch has been issued for the vulnerability but not installed

It is in the last phase that this line of research attempts to make a contribution. This thesis seeks to discover the underlying processes of patch management in order to reduce the time

between the issuance of a patch and its installation. While the remaining phases are significant to patch management, they require different lines of research, and are therefore reserved for future investigation.

The environmental factors and conditions that encompass the information technology industry will play an important role throughout this line of research, and influence the potential success of any resulting developments. The following sections present some of the environmental strengths, problems, issues, and opportunities.

### 1.4.1 Strengths

Over the last decade, computer usage continues to expand in the major economic sectors such as finance, health, manufacturing, telecommunication, and entertainment industries. This growth expansion in recent years appears to be entering a maturity phase. At the end of 2002, there were an estimated 544 million personal computers utilized by over 590 million Internet users (International Telecommunication Union, 2003) with access to over 3 billion discrete web pages (Google.com, 2003) through more than 160 million Internet domain hosts (PCIPB, 2002) on a global basis. The percentage of business computer use in New Zealand, Australia, and Canada is over 80% and continues to rise, as does Internet access of these systems (Statistics New Zealand, 2002). The Internet is facilitated and managed in part by the use of international data communication and security standards created by organizations such as the Internet Engineering Task Force (IETF) and the International Organization for Standardization (ISO). These standards organizations facilitate the collaboration of individuals, organizations, and governments to develop and establish global standards. It is these standards that allow a diverse number of manufacturers to produce systems that communicate with each other. As the number of Internet users and systems mature, there will be increased demand for global standards in software

development, interoperability capabilities, and improved security measures, such as Secure Patch Management Systems in order to further develop and refine this environment.

### 1.4.2 Problems and Issues

There are some serious weaknesses that can potentially impact the direction and successful development of a SPMA. Perhaps the most important of these is the lack of interoperability standards for software applications (Yang & Papazoglou, 2000), and the lack of software compatibility (PricewaterhouseCoopers, 1999i). This is, in part, due to the operating systems' architectures and design capabilities as well as the proprietary nature of core operating system processes. As a result, any patches applied to application programs must also consider the operating system platform and version. There is also an increased likelihood that multiple applications simultaneously accessing system resources may be brought into conflict when an application program and/or the operating system itself are updated. The methods of creating patches and their resultant installation may conflict with existing software interacting and competing for system resources. In addition, sometimes the patch designed to fix a flaw will introduce new flaws into the system due to prior design decisions or customization. This may result in not applying the fix because other system functions which are deemed critical would be disrupted (Landwehr et al., 1994).

Another area of weakness to be considered is the timely notification and safe dissemination of patches. There are many associated issues connected to the gap between becoming aware of a patch and acquiring it. The source providing the patch may not be secure or reputable, allowing for the contamination of the needed patch by Trojan Horses and/or viruses. Once contaminated, the downloaded and executed patch can be used to affect system integrity, execute code, create a covert communication channel, etc. (Olovsson, 1992). In a survey of over 2,500 security practitioners in 2000, 80% of the respondents

reported viruses, Trojans, and worms as the cause of external security breaches. This figure rose to 89% in 2001 (Briney, 2001). More recently, similar findings were reported by AusCERT et al. in the 2003 Australian Computer Crime and Security Survey despite the high usage of anti-virus software and policies to control the introduction of malicious software. Also, companies continue to report that the Internet connection is a frequent point of attack (78%), and this trend has increased steadily over the past five years (CSI/FBI, 2003). Therefore, provisions for patch integrity and timely delivery for deployment must be considered.

The previous section contained a brief discussion of the benefits in using international standards organizations to collaborate and develop new approaches and systems. A significant disadvantage of those organizations is that the process is filled with partisanship, political manipulation, and compromises (Fox, 2002). As a result, features and capabilities that have significant advantages may never be incorporated into a standard if the significant parties cannot make the required changes to promote their particular agenda, limit liability, and/or operate under governmental regulations (PricewaterhouseCoopers, 1999i). As a result, new developments and ideas may take years to implement. Lastly, the nature of the “Request For Comments” process that leads eventually to an agreed standard contains few specific system level details. These details are left to the manufacturers to implement. It is usually the first significant organization to implement a new class of product that establishes the de-facto standard, and as such, is a major decision-maker in any innovations proposed.

### 1.4.3 Opportunities

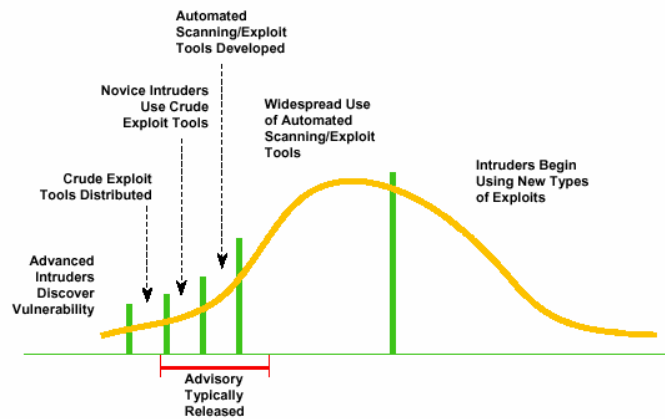
There is an abundant number of opportunities available to this line of research, both theoretical and practical. The extension of the functional life cycle of a software product brings with it considerations of the vulnerability life cycle, the volume of patches produced

annually and the means of distribution and deployment, the financial effects of not enforcing security procedures and policies with regards to insurability, and the utilization of trusted third parties. These opportunities will be discussed with regards to a SPMA in the following sections.

#### 1.4.3.1 Vulnerability Life Cycle

Once a software operating system or application program is distributed, a host of individuals and researchers globally begin the task of discovering flaws and weaknesses, either through on-going use and reporting, or by purposeful means (Fisk, 2002) (Janczewski & Colarik, 2003i). It is the destructive testing nature of the information infrastructure that leads to the discovery of vulnerabilities. A vulnerability can be defined as “a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy” (SANS, 2003). Though developers try to minimize vulnerabilities, time-to-market pressures prioritize functional requirements over security issues (Bowen & Segal, 2000). Organizations recognize the importance of system security but are unable to keep up to date and prevent vulnerabilities because they increasingly must rely on manufacturers’ software as a basic building block for underlying network and system processes (Martin, 2001) (PricewaterhouseCoopers, 2002).

Shipley (2001) states that the vulnerability life cycle has three phases: research and discovery, disclosure, and exploitation. This is further detailed by the Vulnerability Exploit Cycle, and is composed of six activities that encompass the vulnerability discovery, distribution and use of exploit tools, development and use of scanning/exploit tools, and finally the use of new exploits involving Internet connected systems (see Figure 1).

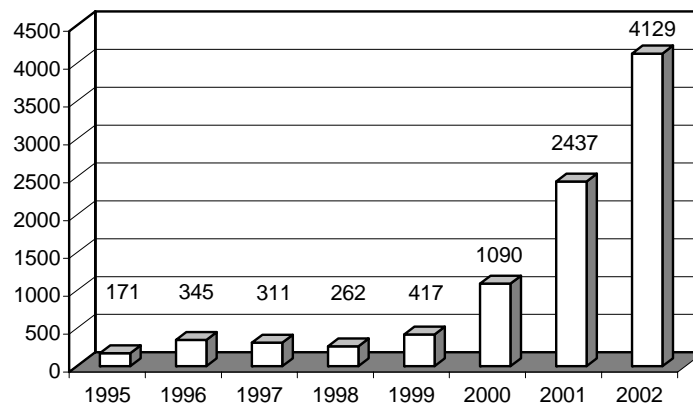


**Figure 1: Vulnerability Exploit Cycle (CERT, 2002)**

CERT (2002) states that 95% of system intrusions are a result of known vulnerabilities or configuration errors. The Code Red's (I & II), NIMBA, and Slammer worms that affected hundreds of thousands of systems are examples of known vulnerability exploits (SANS 2002). Implementing and utilizing a SPMA can significantly reduce intrusions such as these by eliminating much of the exploit cycle through the timely distribution and application of patches.

#### 1.4.3.2 Patch Volume and Costs

For the past five years, the number of reported vulnerabilities has nearly doubled from each succeeding year (see Figure 2).



**Figure 2: Reported Vulnerabilities (CERT/CC, 2003)**

Because of the volume of reported vulnerabilities, systems administrators and individual users may potentially be required to identify, acquire, and deploy a dozen patches per day in order to secure their systems. This is further compounded when dealing with open source code. When developers are free to modify source code, “the theoretical number of modifications equals the number of developers” (Doherty, 2001). Thus, the potential number of vulnerabilities (unreported) and time of exposure in open-source software can be quite high. For security and corporate image considerations, vendors tend not to report a vulnerability until a patch is made available (Mell & Tracy, 2002). This further emphasizes the need for prompt notification and timely delivery of a newly issued patch.

The overhead and labor costs of individuals and organizations to patching systems are not as substantial when compared to the costs of not patching systems. Security is only as strong as its weakest link (Ellison & Schneier, 2000). Whether the systems were not patched due to workload, complexity, or ignorance, the consequences can be considerable. It has been estimated that the financial impact to the US economy resulting from malicious code orientated cyber attacks to be US\$13 billion (PCIPB, 2002), and over UK£2 billion in the United Kingdom for security breaches (PricewaterhouseCoopers, 2002) in 2001. As more of the economic infrastructure is dependent on network connectivity, the economic consequences will only increase.

#### 1.4.3.3 Trusted Third Parties

Schmid, Stanoevska-Slabeva and Tschammer (2001) define trust as a perceived lack of vulnerability. A decision to trust implies a judgement value about the vulnerability in carrying out an activity. Johnson (2000) defines trust as a “firm reliance on the integrity, ability, or character of a person or object”. With regards to computing, it is ultimately a human that enables the initiation of an electronic-based activity (Schmid, Stanoevska-Slabeva

& Tschammer, 2001). It is this linkage that is causing trust to become a central issue in networked information systems, and therefore requires that new techniques be applied to the design process (Yu & Lui, 2001).

Kramer (2001) describes a trust dilemma as an interdependence between social decision-makers when the party's dependence exposes them to the prospect that their trust will be exploited and betrayed. It is this convergence between opportunity and vulnerability that makes trust dilemmas a part of social and organizational life (Kramer, 2001). There are two fundamental rule sets that help decision-makers harvest the benefits of trust when dealing with a trustworthy other in order to minimize the costs of misplaced trust. These are interpretation rules, which help categorize a given trust dilemma for prescribing the type of evidence needed in assessing trustworthiness, and action rules, which prescribe what behaviours to engage in when responding to those interpretations. It is the action rules that represent a decision maker's beliefs about codes of conduct in dealing with trust dilemma situations (Kramer, 2001). It is the disposition of an individual or organization to trust or distrust that permits a consistent willingness or unwillingness to a broad spectrum of situations and interactions (McKnight & Chervany, 2001). Thus, trusted parties must present an accepted code of conduct that permit a consistent willingness to trust by participating parties.

A trusted third party (TTP) is an organization that two or more parties utilize as an intermediary to enforce a set of rules of conduct. It is an impartial organization that delivers business confidence through the use of technical security features. Its mission is to provide a technically and legally feasible means of producing independent evidence about an electronic transaction and/or act as an arbitrator (ISTEV, 1998). The belief that a party can trust the TTP's protective structures is known as institution-based trust (McKnight & Chervany,

2001). Breach of those beliefs can have profound consequences as to the TTP's longevity, and must be considered in its functional design requirements and assurance methodologies (Pekka & Karvonen, 2001).

In a report by the TrustWeb Consortium (1998), it was suggested that trust services "could possibly provide security functionality independently of the large software manufacturers". Additional TTP services could be used to assist in securing the Internet, such as a SPMA. The use of TTPs in this line of research may reduce actual and perceived vulnerabilities, and open the possibility of research and development exploitation, as well as commercial implications.

#### 1.4.3.4 Cyber Insurance

From a business perspective, security breaches are more than annoyances that may disrupt operations. They pose a significant threat to the loss or theft of intellectual property, personal and corporate reputation due to loss of privacy (Jerry & Mekel, 2002), and to electronic commerce revenues. To offset some of the potential losses, insurance companies are issuing cyber insurance. These policies are designed to reduce and mitigate the costs and liabilities associated with the disruption of services, inadvertent breach of contracts, and failure to adhere to privacy laws. While this type of insurance is useful in reducing financial uncertainty, it can be cost restrictive to small businesses and individuals. The required procedures and audits to insured systems add to the initial policy cost in labour (i.e. manpower) and overhead (i.e. records, etc).

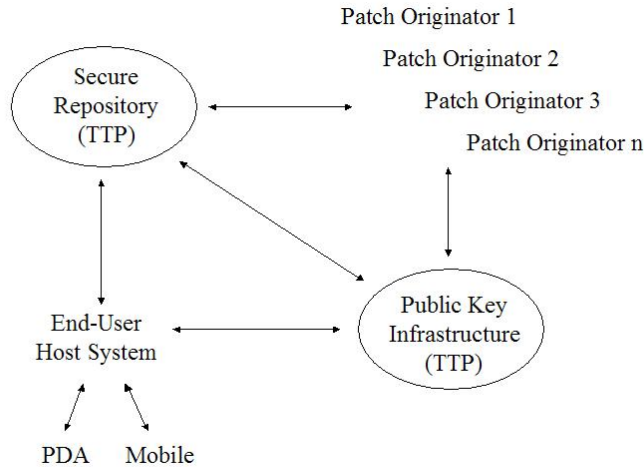
The insurance industry has historically encouraged regulators to create safety standards and promote reduced damage claims through the use of certification programs (Janczewski & Colarik, 2003i). When an individual or organization is in compliance, the customer may be given a preferred insurance rate. Some cyber insurance policies require that

a security audit be conducted prior to its issuance, and at the policyholder's expense. Moreover, there are some insurance underwriters at Lloyds of London that feel electronic commerce will emerge as the biggest insurance risk of the 21<sup>st</sup> century (Jerry & Mekel, 2002). As organizational connectivity and electronic commerce continues to grow, so may the insurable losses. The impact and influence of insurance companies needs to be considered and potentially enlisted in order to bring together the disparate software manufacturers in jointly participating in a SPMA to reduce claims caused by non-patched systems.

## 1.5 Research Objectives

The strengths, problems and issues, and opportunities discussed above open up a large domain where there is immense scope for research. The opportunities discussed (vulnerability life cycle, patch volume and costs, trusted third parties, and cyber insurance) provide an impetus for the design and implementation of a Secure Patch Management Authority (SPMA) that will take advantage of the strengths provided (volume usage and standardization efforts) while reducing many of the problems and issues (lack of interoperability, multiple platforms, timely patch notification, and establishing a de-facto standard). The primary objective of this research is to propose a Secure Patch Management Authority that attempts to reduce the time between the issuance of a patch and its corresponding notification, transport, and installation by the patch recipient, which will be presented in Chapter 5. The secondary objective of this research is to provide an exploration of the patch management domain including any relevant models and approaches, as well as any underlying technologies that may contribute to a Secure Patch Management Authority's design. The SPMA model is illustrated in Figure 3. The critical sub-component of the architecture is an integrated file integrity mechanism known as a hash triplet, and shall serve as a proof-of-concept for the viability of the proposed

SPMA architecture (see Chapter 6). It shall also serve as an artifact that may become the focus of future research (Nunamaker, Chen, & Purdin, 1991).



**Figure 3: Secure Patch Management Authority Model**

## 1.6 Research Contributions

There are multiple contributions of this research. First, an exploratory review, synthesis, and organization of the patch management domain are presented. Second, a framework and underlying architecture for the secure distribution of patches is proposed as both a system development project and to provide future research opportunities. Third, a new integrity algorithm / mechanism is introduced for patch file notification and server-to-server transfers that provides error triangulation. Lastly, through providing the above, this research facilitates an extension of the software development life cycle through the secured, timely delivery of patches from patches originators to end-users. This in turn contributes additional efficiency to the maintenance phase of the software development life cycle.

## 1.7 Organization of the dissertation

Chapter 2 is a review of the phases of the software development life cycle with an emphasis on the specific domain of patch management systems. The chapter provides the specific models and approaches available for the distribution of patch files, and the various security services and mechanisms for their secure transmission. The chapter also examines the dominant middleware platform standards with respect to security provisions. Chapter 3 is the multi-methodology approach that governs this research. Chapter 4 presents a decomposition of patch and update system patents, and organizes this into a taxonomy with an evaluation of security services design specifications. The chapter also introduces a unified model of patch management systems based on the patent frameworks identified in Appendix A. Chapter 5 describes the proposed Secure Patch Management Authority framework and architecture. Chapter 6 presents the proof-of-concept implementation, and introduces the hash triplet integrity algorithm. Chapter 7 evaluates the proof-of-concept with respect to the design criteria presented in Chapter 5 and the security and mechanisms discussed in Chapter 2. Chapter 8 summarises the research and the contributions made, and suggests future research directions.