A CROSS-SITE PATCH MANAGEMENT MODEL AND ARCHITECTURE DESIGN FOR LARGE SCALE HETEROGENEOUS ENVIRONMENT

Chuan-Wen Chang

Department of Information Management,
Chinese Culture University,
Taipei, Taiwan

Dwen-Ren Tsai

Department of Computer Science,
Chinese Culture University,
Taipei, Taiwan

Jui-Mi Tsai

Computer Center,
Hsing Wu College,
Taipei, Taiwan

ABSTRACT

Many reports indicated that most damages caused by computer viruses and hackers' attacks due to management problems. Computing environments implementing well managed patch management processes with quick response mechanisms will survive from most of serious attacks, such as MyDoom and Sasser Warm attacks in 2004.

Medium or large enterprises usually have heterogeneous computing environments. For example, a company may use an Apache Server in a Linux-base PC as its Internet web server, use an IBM AIX running IBM DB2 database system as a database server, and equip all employees with Windows-based PCs running Microsoft Office for their daily work. Also, employees might work at many different locations. In the enterprise patch management (PM) market today, there are very few complete off-the-shelf solutions.

A systematic efficient PM process model with complete patch management activity process cycle and patching strategies was proposed. We also propose an automatic five-layer PM system application architecture supporting heterogeneous environment. The model, hopefully, will make enterprise patch process more efficient, and will reduce the risks suffer from patch management challenges.

Keywords: Information Security, Patch Management, Network Management.

1. INTRODUCTION

According to the latest statistics and surveys from Computer Emergency Response Team/Coordination Center (CERT/CC) [1] [2] there are totally 17,946 system vulnerability reports received by CERT/CC during first quarter of 2005. It exceeds the quarterly average number of previous year. There are US$140-millions losses caused by all information security problems in 2004. Over 38 percent of these losses were due to computer viruses. These statistics indicate that system vulnerabilities had become one of the most serious information security problems. System vulnerability usually is caused by a poor or improper system design or implementation. Someone might try to steal your data or damage your systems through vulnerabilities. Unfortunately, there still exist no effective solutions for this problem now. Many academic and industrial researches are on going. These researches include intrusion detection system (IDS), dynamic monitoring, source code examining, etc.

IDS (with active protecting mechanism) (see [3] [4]) is a great idea to prevent invalid system access from outside. The working model of IDS is to detect and identify all possible intrusion activities by examining system logs within a reasonable time period. Besides pure pattern matching, statistics [5] [6], data mining [7], and artificial intelligent [8] [9] were used to leverage efficiency of IDS. Dynamic monitoring technology prevents invalid system access through known system vulnerabilities using system call level monitoring [10] [11]. Dynamic monitor intercepts all possible system calls with known patterns and redirect them to rewritten secured procedures. These strategies may reduce some potential risks of access problems; however, system vulnerabilities still exist. Hackers or malicious programs are still trying to find other ways to attack your system.

Besides academic researches, software vendors and third parties are developing automatic software-dependent patch management mechanisms and standalone patch program deployment products, which trying to fix system vulnerability problems through better patch program deploy management (e.g. [12]). Even though, some problems are still unsolved for large enterprise environment, such as,

0-7803-9245-0/05/$20.00

1. **Lack of process models.**
   Most of current patch management products only focus on patch program deployment technology, but not provide corresponding patch management models.
2. **Additional prerequisites.**
   Some products need additional infrastructure, such as proprietary directory service, which could increase overall deploy cost and might violate enterprise's IT management policies.
3. **Difficult support to heterogeneous environment.**
   Current large scale enterprises usually use heterogeneous computer systems, and deploy their computer systems in different physical location. Most of current patch management products only support one of few platforms, and not support extranet management scenario [13].

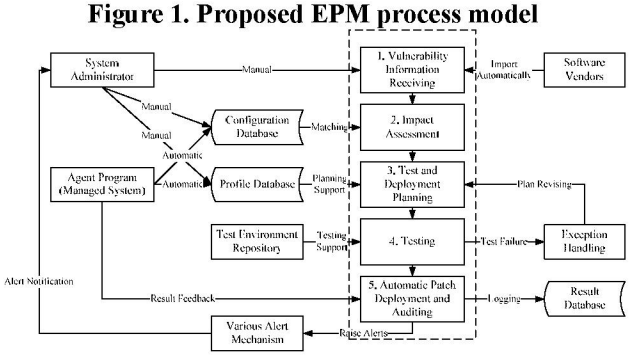
In this study, a patch management process model and automatic patch program deployment system architecture are proposed. This management process model and the architecture design provide fundamental practices to cross-site, large scale enterprises with heterogeneous environments.

2. PATCH MANAGEMENT PROCESS MODEL

2.1. Overview

Enterprise patch management (EPM) is not only concern with how to deploy patch program into enterprisea. Instead, IT manager should treat EPM as a completed process cycle. Reference to Sundgren's research report [14], we develops a complete model with five major stages for EPM process, which is shown in Figure 1.

Figure 1. Proposed EPM process model

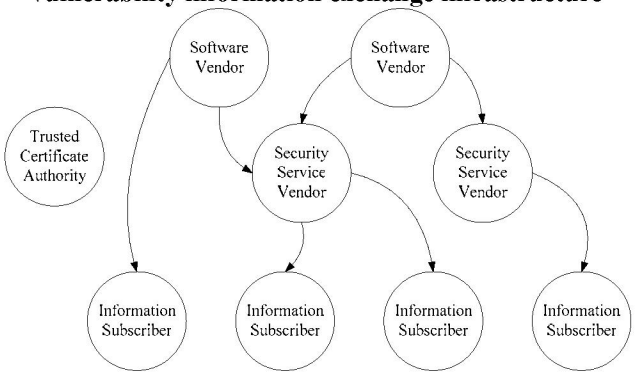


2.2. Vulnerability Information Receiving

Unlike newspaper is always deliver every morning, we can't predict when a system vulnerability or patch information will be released precisely, therefore, system administrators should keep sensitive on watching all new patch information from software vendors. When a new patch program released, system administrators should log related information, such as severity rating, affect software (or component), recommendation, and procedure on how to patch those affected systems.

Besides having human involvement to such monitoring process, it's better to have an infrastructure which allows patching and exchangimg related information automatically. The design concept is illustrated in Figure 2.

Figure 2. Conceptual diagram of automatic vulnerability information exchange infrastructure



Before this infrastructure is established, a universal metadata dictionary about patch information should be well defined. This dictionary provides a standard taxonomy for related information, and is very similar to ICAT Metabase [15] and CVE [16], with additional patch information, such as how to apply patch programs, and patch program itself. This design puts all patch required information in a single message package, and is available to any receiver.

Information exchange works through subscribe and push model. Software vendors or independent security service providers act as information publisher pushing all patch information to all subscribers (enterprises). This information subscribe model could be a free service or a contract-based service, of course should be on a trusted base. Automatic exchange model will dramatically benefit on real-time information delivery and will avoid any failure of human involvement. Independent certificate and PKI technology also be used to secure this exchange model.

2.3. Impact Assessment (Scoping)

Applying a patch program to existing systems is very risky, because most of patching processes change current stable system configurations. Sometimes systems might be brought down when or after patch programs were applied. Therefore, to precisely identify impacted systems on a

system vulnerability is very important, especially in a large enterprise with many different system configurations. Patching only to those affected systems also will save overall network bandwidth and will reduce time required for a patching process.

Configuration Database (CMDB) will be the key component for impact assessment process. CMDB should contain all important technical information such as system name, IP, name and version of installed operating system and application software, other management information such as system functionality, install location, service time may also be included. Most of large enterprises maintain their own configuration database. In scoping process, patching target selection can be performed through automatic matching between affected software provides by system vulnerability information and installed software listed in CMDB. Automatic scoping process could be completed efficiently through most existing RDBMS and programming technology. This will create a precise and limited target system list.

2.4. Testing and Deployment Plans

Test plan and deployment plan should be conducted at the end of this stage. Test plan focus on how to make sure those designate systems could keep on service after patch deployed. Software vendor should and usually made related testing, such as integration and compatibility test, before patch programs are shipped. However, enterprises should work out their own testing plans before rolling out, especially to those in house developed software.

Patch scheduling is another key factor to a successful patch deployment. Deploy patch program to all target systems is not good enough [17]. For example, patching process may stop some important current long-running tasks in application server and interrupt regular business activities. Profile Database in our model was designed for this issue.

IT department could define a matrix for patching process by patch severity and profile of managed systems in profile database. Except those patches with high severity require immediately installation, normal patching process can be scheduled at most appreciate time slot, depending on system type, functionality, workload, even on location, to minimized business impacts caused by patching process. Most appropriate time slot for patching could be estimated through administrator's experience. More precise and real information could also be collected from designate client agent installed in every managed system.

2.5. Testing

To ensure that target system can work properly after patching process is very difficult. Unfortunately, there is no perfect test tool or solutions which can perform automatic testing to all system existed in current enterprise environment in system level. Administrator is still responsible to conduct required testing plan and procedure to every different system. Although there is no automatic testing mechanism available, we can speed up overall testing process by reducing a huge amount of time on environment preparation.

With many software vendors' researches and developments, virtual machine had become a mature technology which is suitable and available in system simulation, and could leverage testing environment preparation. Administrators can pre-install and pre-configure each test environment with different system configurations, and store them in a mass storage. This strategy could save significantly overhead before performing non-automatic system testing.

2.6. Automatic Patch Deployment and Auditing

The last stage in the model is to deploy tested and approved patch program into those systems requiring patching automatically. This patch management system should not only deploy patch program, it also should keep track on every deploy status and result, to make sure every single patching job is executed successfully. Otherwise, it will raise alerts through pre-defined mechanisms, such as e-mail or short message through cell phone for additional exception handling. Detail design and system architecture to this automatic patch management system will be described in section 3.

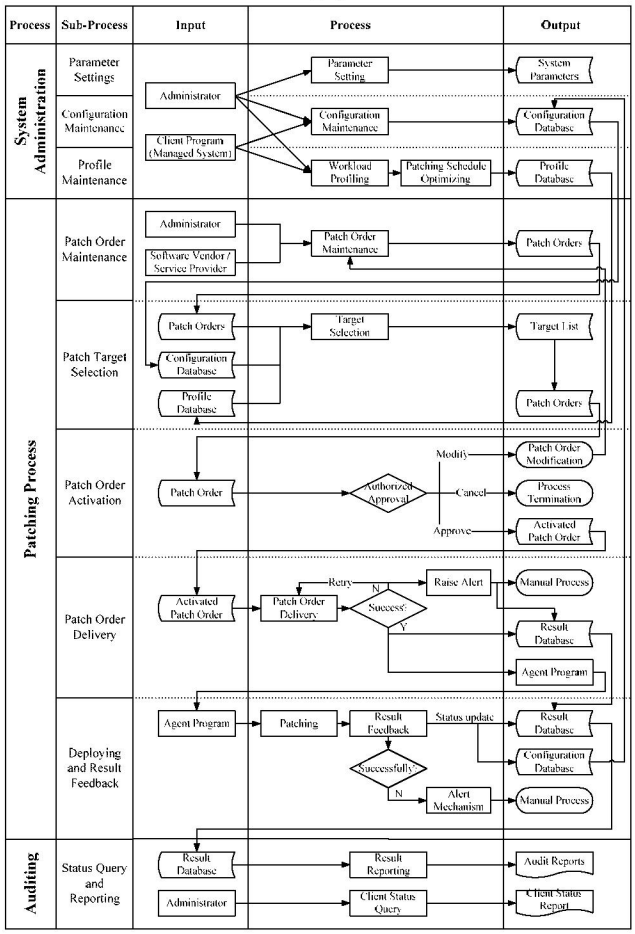
3. AUTOMATIC PATCHING SYSTEM IMPLEMENTATION

3.1. Functional Design

There are three major functional process categories designed within our automatic patching system. Some individual functional processes (sub-process) also developed to support those major process categories, shown in Figure 3.

System Administration processes maintaining system wide parameters and setting, including CMDB and Workload Profile Database to each managed systems. In our design, information of CMDB maintenance and Profile Database will be collected automatically by client agent installed in each managed system. Through this automation, our patch system will always keep and maintain most up-to-date information without human efforts, and will reduce overall system operation costs, especially in a large enterprise with many managed systems.

Figure 3.Function design and process description of Automatic Patching System



Patching process is the core of this automatic system, which provides all major patch management functionalities from patch order generating, deploying, and to result collecting. For the goal to have enterprise patching process as simplify as possible, all patch process were designed automatically, except order activation and testing. In the patch order maintenance process, we have developed a XML/Web Service interface in our implementation which allows patch information receiving from software vendor or security service provider automatically. In target selection process, this system will filter all target systems from CMDB automatically. For the deployment process, management system will send patching information in the designed schedule suggested by profile database. Finally, any process failure either fail in message transmission or client deploy fail will cause automatic retry and an alert will send to administrator for additional exception handling.

No matter a completed patching task is success or fail, task result will be logged in both patching and managed client. Administrator can query patching result status in patch order basis or individual managed system.

3.2. Distributed Deployment Model

Required time and network bandwidth are two critical resources to system patching process in a large enterprise deployment. Requiring more time to complete all patching tasks means increase overall risk of system vulnerability. And more bandwidth used also means less bandwidth could be allocated to other systems within in same network.

The bottleneck of patching process usually happens in patch deployment through low bandwidth WAN connection in single management server scenario. Setup additional member management server in each location with low bandwidth could be a good solution candidate for this issue. In a multi-server management scenario, central management server prepares and sends patching order message with patch programs into local member server. After required patch data distributed, client agent in local managed system can communicate with local server instead of central server with low bandwidth.

3.3. Protocol Considerations

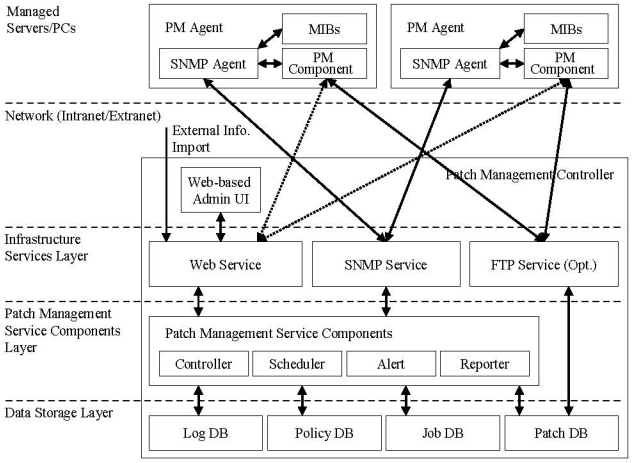
Heterogeneous environment support is another key requirement to large scale enterprises. To make sure that we can fulfill this requirement, proprietary protocol for all communication between management servers and managed systems should not be used. Our implementation leverage SNMP protocol to all control messages. There are three major benefits for this design as following.

1. **Heterogeneous environment support.**
   SNMP is an IETF standard protocol supported by most operation systems. And it's not difficult to develop SNMP client agent in most platforms existing in current enterprise.
2. **Cross-site management support.**
   SNMP protocol uses standard UDP ports for socket communications, so no any other special port required in those firewalls among different locations. It means that this manage system can control all managed systems located in both intranet and extranet.
3. **High performance.**
   SNMP is one of light weight protocol compare with other proprietors. So it will significantly reduce overall control communication bandwidth consumed, especially for those enterprises with a large number of managed systems.

3.4. System Architecture

To support our designed PM scenario, we construct a web based management five logical layer prototype for concept proofing. The architecture is shown in Figure 4.

Figure 4. Prototype system architecture



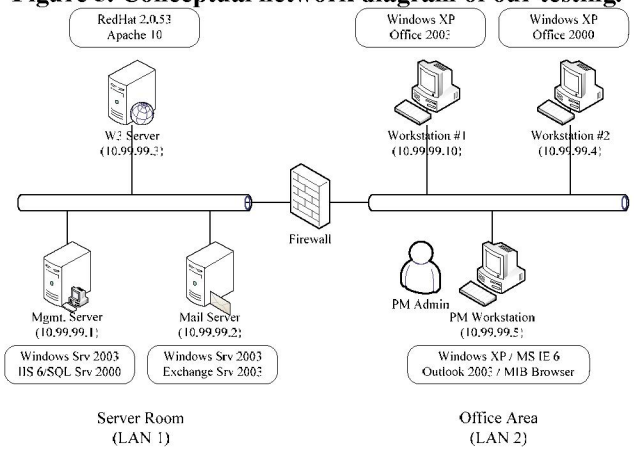
1. Infrastructure Services – A Web Service hosts administrative web pages for administrators, and a SNMP service component sends PM instructions from the management console to managed computers. FTP is an optional component for patch delivery.
2. PM Service Component Layer – Controller is the core component for application logic. Scheduler component is a service for scheduling tasks such as a planned scheduling patch deployment task. Alert component is responsible for alert generation through E-mail or SMS. Reporter component accepts administrator's status queries and generating regular management reports.
3. Data Storage Layer – A storage engine stores application data such as configuration baseline, PM order, deploy log and so on. We use a RDBMS for data storage practice.
4. Managed Systems – Managed client had installed a SNMP agent component, which receives patch orders from management server and pass those commands to client PM component for execution and feedback of deployment status. SNMP agent is a basic network service and supported by most major OS platforms, this design implicating most platforms from different venders can be managed within this PM application design architecture.
5. Network – Firewall may setup between management server and managed systems in a cross-site environment. In our solution design, only minimal managed network traffic through SNMP, HTTP (or FTP) protocol are required, and will keep firewall secure and integrity.

4. PROOF OF CONCEPT

4.1. Testing Scenario Design

After system prototyping, we built a testing scenario to verify that our system design can get expected design goal and results. The simulated testing environment contains management server, some managed servers and workstations in 2 connected network segments, through virtual machine technology. Figure 5 contains this.

Figure 5. Conceptual network diagram of our testing.



4.2. Testing Scenario and Results

Finally, we create a testing scenario cover a completed enterprise patching scenario. This scenario started with an alert received from a trusted security service vendor, after some required patch management activities, patch program were deployed to each selected managed system successfully. Our testing scenario contains the following major patch management activities.

1. System setup and create management baseline.
2. Alert receiving and patch order management
3. Scoping and patch program delivery
4. Emergency patch rollback
5. Deploy status audit and review

5. CONCLUSIONS

This paper builds up a systematic patch management process model and related strategies from baseline establishment, vulnerability information acquires, scoping, planning, testing and deployment. In automatic patch deployment practice, we had designed an five-layered system application architecture to support our design by

leveraging SNMP protocol for cross-site, large scale enterprises with heterogeneous environment. And enable enterprise patch management process more reliable, efficient, and reducing the risks from patch management challenges.

6. REFERENCES

[1] CERT/CC; "Vulnerabilities reported", *CERT/CC Statistics 1988-2005*, CERT Coordination Center (CERT/CC), available at http://www.cert.org/stats/cert_stats.html.

[2] CERT/CC; *2004 CSI/FBI Computer Crime and Security Survey*, CERT Coordination Center (CERT/CC), available at http://www.gocsi.com/, June 2004.

[3] Kemmerer, R.A. and Vigna, G., "Intrusion detection: a brief history and overview", *Computer*, vol. 35, no. 4, pp. 27-30, April 2002.

[4] Shieh, S. W. and Gligor, V. D., "A pattern-oriented intrusion-detection model and its applications", *Proceedings of 1991 IEEE Computer Society Symposium*, pp. 327-342, May 1991.

[5] Li, Z., Das, A. and Nandi, S., "Utilizing statistical characteristics of N-grams for intrusion detection", *Proceedings of the 2003 International Conference on Cyberworlds*, pp. 486-493, December 2003.

[6] Ye, N., Li, X., Chen, Q., Emran, S. M. and Xu, M., "Probabilistic techniques for intrusion detection based on computer audit data", *IEEE Transactions on Systems, Man and Cybernetics*, Part A, vol. 31, no. 4, pp. 266 -274, July 2001.

[7] Han, H., Lu, X. L. and Ren, L. Y., "Using data mining to discover signatures in network-based intrusion detection", *Proceedings of the 2002 International Conference*, vol. 1, pp. 13-17, November 2002.

[8] Tsai, D. R., Tai, W. P. and Chang, C. F., "A hybrid intelligent intrusion detection system to recognize novel attacks", *Proceedings of the IEEE 37th Annual 2003 International Carnahan Conference*, pp. 428-434, October 2003.

[9] Deng, H., Zeng, Q.A. and Agrawal, D.P., "SVM-based intrusion detection system for wireless ad hoc networks", *Proceedings of the IEEE 58th Vehicular Technology Conference*, vol. 3, pp. 2147-2151, October 2003.

[10] Condit, J., Harren, M., McPeak, S., Necula, G. C. and Weimer, W., "CCured in the Real World", *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, vol. 38, no. 5. pp. 232-244, May 2003.

[11] Hunt, G. and Brubacher, D., "Detours: Binary Interception of Win32 Functions", *Proceedings of the 3rd USENIX Windows NT Symposium*, pp: 135-143, July 1999.

[12] Burnett, M, "Enterprise Patch Management for Windows", InstantDoc #40710, Lab Feature, *WindowsITPro Magazine*, available at http://www.winnetmag.com/Articles/Print.cfm ?ArticleID=40710.

[13] Sundgren, J., "Developing an Effective Patch Management Process", *IDEABYTE*, Forrester Research, Inc, August 2003.

[14] Sundgren, J., "Patch Management Products Proliferate, But Absorption In Larger Solutions Is Likely", *Market Overview*, Forrester Research, Inc. March 2004.

[15] National Institute of Standard and Technology, ICAT Metabase, available at http://icat.nist.gov/.

[16] US-CERT, Common Vulnerability and Exposures (CVE), Sponsored by the at the U.S. Department of Homeland Security, available at http://cve.mitre.org/.

[17] Gerace, T. and Mouton, J., "The challenges and successes of implementing an enterprise patch management solution", *Proceedings of the 32nd Annual ACM SIGUCCS conference on User services*, pp. 30-33, October 2003.