IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

COST EFFECTIVE & AUTOMATED CENTRALIZED PATCH MANAGEMENT FOR LARGE SCALE HETEROGENEOUS AND DISTRIBUTED SYSTEM

Vishal Gupta
Senior Engineer

Shilpa Singh
Deputy Manager

Varun Gupta
Manager

Design and Engineering, Network Centric Systems
Bharat Electronics Limited, Ghaziabad, India

Abstract: Administration and management of application versions, system configurations and operation policies is a challenging and costly activity for large enterprises. Currently organizations involved in building large scale distributed systems deploy multiple applications written and implemented in different technologies and operating in heterogeneous environment. Different computing nodes have unique set of applications and policies, to provide designated services. For systems built to suit specific industry operation, periodic improvement of application logic and enhancement of functionalities becomes a regular affair. However, every time a new application patch/version is released for correction of bugs or enhancement of functionalities, it becomes a cumbersome task to timely and accurately deploys the right set of application versions and policies at the intended target machine. In this paper we have envisaged a centralized patch and configuration management framework through which the distribution of application patches and policies can be centrally controlled, monitored and implemented for large scale systems with geographically distributed endpoints having heterogeneous identities and operating environment. This allows centralized visibility of deployed managed resources/device and eliminates the necessity of manual distribution and update operations at each endpoint. This caters the need of scalable, centralized, authoritative repository, through independent and secure framework.

Keywords: patch management, version management, policy management, managed devices etc

1.Introduction

Keeping software systems up to date by applying security patches and other applications patches is critical security hygiene, and failing to timely patch software systems may lead to devastating consequences. A significant number of cyber security breaches have been a result of exploitation of vulnerability for which patches have not been updated centrally [1][2][3].

Security patch management in large and complex systems is a hugely challenging task that involves different types of stake holders making several interdependent technological and socio-technical decisions. The high rate of security patch releases, service disruptions caused by faulty patches and reboots following patch installation make the process even more challenging. The Patch updates are usually done by manual intervenes. Because of manual intervenes and manual processing, chances of happening of fatal error is more [4][5]. Such evidence from a growing number of security incidents indicate that a significant amount of research is needed to help understand the prevailing challenges in patch management that may cause delays in applying security patches, and the available solutions to overcome those challenges.

Manual Application patch management for large scale heterogeneous and distributed systems is a costly affair for organizations. The traditional methods range from plugging of external media to individual systems and copying of files to taking manual remote access of each endpoint in sequence and copying files over the network[6]. Even with provisioning of large number of correctly prepared similar media and dedicating requisite amount of personnel to carry a series of defined procedures, the task involves considerable labor costs, errors and system downtime [7-8].

There may be different methods to manage the patches even to cater the redundancy scenario but also in such methods, though the patch may be available at multiple locations, in a crisis situation the user realizes that there is no central repository where a previously released patch can be referred to for disaster recovery.

Patch management refers to the process of applying software patches. It is defined as the process for identifying, acquiring, installing, and verifying patches for products and systems [9-11]. The focus of patch management from a software vulnerability life cycle point of view lies on the time frame between when a patch is available until it's installed, as highlighted with dash lines in Figure 1[12].

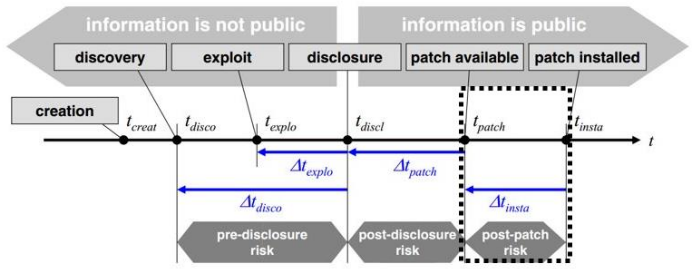

Figure1: General Layout of Patch Deployment [11]

The other sections of the paper are organized as follows. Section 2, presents the related Literature survey, while section 3 describes the proposed approach. Section 4 describes the implemented results and its explanation and finally the conclusion in Section 5.

2.Literature Survey

Different Authors have proposed different approaches for the patch management and security vulnerability. There is large diversity in the study of their research. We have summarized the research work of researchers in Table 1.

Table1:- Different approaches for the patch management and security vulnerability

analysis, patch type and severity, patch reliability and

Technique Proposed	Performance matrices	Platform used	Results	Related work
HEFT: Heterogeneous Earliest Finish Time algorithm	Computation Cost	Java based simulator	In the results it was found that there is a huge difference between the performances of Heterogeneous Earliest Finish Time (HEFT) based upon DAG for calculating weights.	[3]
Improved cost based algorithm for task scheduling in cloud computing	Task computation time, cost	Cloudsim 1.0b	The results indicate that the proposed algorithm is more efficient than activity based Cost- Effective algorithm.	[5]
A cost based resource scheduling cost paradigm in cloud computing.	Cost	Java Cloudware, Pure java based platform in cloud	This algorithm evaluates cost and is first such algorithm used in cloud environment.	[8]
Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud	Execution time, cost	Java and randomly generated workflows	The proposed algorithm reduces the cost and execution time and manages deadline and overall budget simultaneously.	[11]
Customer-facilitated cost-based scheduling (CFCSC) in cloud.	Makespan, monetary cost	Cloudsim	From the results it is recorded that the proposed algorithm is helpful in load balancing and minimizing the total monetary cost when compared with HEFT.	[15]
Resource scheduling base on fuzzy clustering in cloud computing.	Response time, waiting time, running cost	Java based simulator	The proposed algorithm is Cost – Based Clustering (CBFCMP) is a fuzzy algorithm and is more efficient and gives more accurate results when run in different iterations.	[1]
Cost-effective task scheduling using hybrid approach in cloud.	Make span, energy consumption, cost	Matlab 7.1	This paper proposed a novel heuristics called Nearest Neighbor (NN) and a variant of PSO called NNCA_PSO. NN heuristic achieved aligned allocation of tasks to VMs by reducing the difference between execution time requirements of tasks and execution time characteristics of VMs.	[16]
Optimization of task scheduling and cloudlets cost scheduling algorithms on cloud using cloud simulator.	Execution cost, task completion	Cloudsim2.1.2	Both the cost and time of completion are lessened by the proposed algorithm, thus proving advantageous over the sequential logic.	[17]

3. Proposed Approach
3.1 General Illustration

As illustrated in Figure 2, at the first stage of the patch management, process is patch information retrieval, where practitioners learn about new patches, download and distribute them to the relevant end points to be installed. In the second stage, namely vulnerability scanning, assessment and prioritization, the organizational systems are scanned to identify existing vulnerabilities residing in the software, which can include software bugs, missing patches, insecure configurations, and vulnerable ports and services [13]. It is followed by risk assessment of the identified system vulnerabilities to quantify the vulnerability risks, in order to prioritize patches to decide the order of patch installation. It was found that several socio-technical factors such as the applicability of patches and their impact to managed systems, positive cost benefit compliance to organization policies impact practitioners' decisions to apply the patches during this stage [14, 18]. It also requires managing coordination issues with multiple stakeholders inside and outside organization to reach a consensus on the time to install patches as well as receiving organizational approval for patch installation. The last stage is post-deployment patch verification, which includes several activities such as patch monitoring to verify successful installation of patches, handling post-deployment issues in compliance to organization policies, and patch auditing to verify and remedy exploitation of the newly patched vulnerabilities. The post-deployment issues are usually handled through uninstalling patches, rolling back to snapshot or backup, reverting to the previous software version, troubleshooting and finding workarounds [20-23].

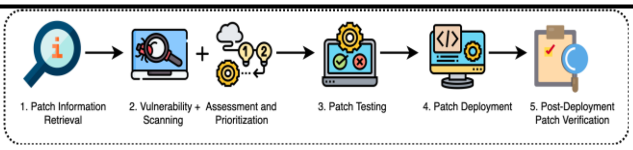
1. Patch Information Retrieval
2. Vulnerability + Scanning
Assessment and Prioritization
3. Patch Testing
4. Patch Deployment
5. Post-Deployment Patch Verification

Figure 2: An overview of Patch Management system using General Approach [17]

3.2 Proposed Technology

In our Environment, the Patch and Management is taken care through Micro Focus tool **ZENworks**. This is the centralized solution for provisioning patch management for large enterprises which enhances the ease of implementation, proper configuration, system availability and improving end user expectations. The proposed architecture is agent based centralized and automated patch management framework which caters all aspects of patch management from centralized control, efficient delivery, proper configuration, timely implementation, centralized repository and centralized reporting. We have implemented the solution considering HA (High Availability) and DC-DR (Data Center- Disaster Recovery) aspect to obtain the 24*7 operations redundancy.
An automated and centralized patch management procedure and solution caters to all these issues. Right from the detection of a need to deploy a software patch, the entire patch management process should be automated. The entire process should be streamlined through a centralized patch management server, allowing the operator deploy heterogeneous patches from a central point of control.
The proposed patch management solution is agent server based. We have deployed the servers in HA (High Availability mode) that is there are two Active – Active VMs to achieve redundancy. The VMs are further configured through load balancer as shown in Figure 4. If the work of load balancer to route the end points to the second server if first server is down.

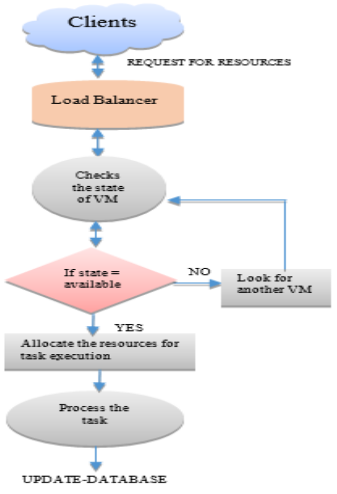

Figure 3: Access of VMs through Load Balancer [26]

3.2.1 Patch Management Solution Architecture

The patch management solution consists of various components including Servers, Agents, Database, Web Console and different services as shown in Figure 5 below. All the sub components are discussed in brief.

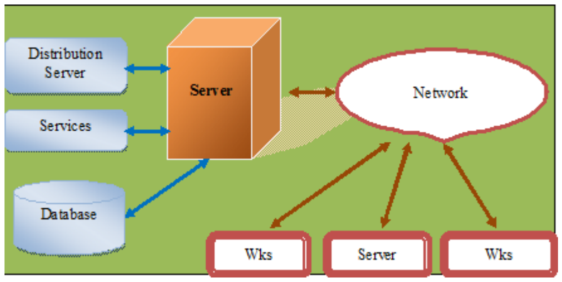

Figure 4: Patch Management Solution Architecture

Solution Components
An automated and centralized Patch Management Software should comprise of the following components.
✓Server
✓Agent
✓Database
✓Web Console

Server
The Patch Management server is required to facilitate various patch-management tasks to help administrators patch computers in the organization's network effectively. Some of the tasks include the following:
✓Installing the agent in computers in the customer's network
✓Creation of patch tasks
✓Deploying patch tasks
✓Generating reports. For example, reports related to patch status or patch compliance

Distribution Server
The Distribution server facilitates:
✓Management of patch content to facilitate its distribution to the endpoints.
✓Compression of patch files for optimized bandwidth usage.
✓Encryption of patch files to provide greater security during transmission and storage on the distribution server.

Agents
The Patch Management agent is a lightweight software application that is installed in computers which are to be managed using Patch Manager Solution.
✓It facilitates the completion of various tasks that are initiated in the Patch Management server.
✓Automatic scheduling of patch installation procedure.
✓Facilitate protection and automatic restoration of configuration files and operating environment.
✓Updates the Patch Management server with the status of patches that are deployed.
✓It checks the Patch Management server periodically for instructions related to tasks and completes the same.

Database
✓All data related to managed devices and patches is updated to the patch database.

Web Console

- ✓The Web console of Patch management Solution provides a central point from where an administrator can access all the tasks that are related to patch management.
- ✓Separate client installations are not required to access the Web console.

3.2.2 Final Proposed Implemented Architecture and Patch Deployment Flow process

The Patch and Configuration Framework are implemented using an automated Patch Management Solution. Two numbers of Patch Management Servers are installed in Active-Active configuration and a management zone for defining the devices to be managed is created. The Patch Management Agent is installed on all devices to be managed through the server and each agent is registered to the Patch Management Server. The TLS compliant web console is integrated with an LDAP server for centralized authentication and grant of privileges for secure access to the patch management solution by an Administrator/operator. The Distribution Servers are also configured through the web console, for packaging, compression and encryption of the patch files for efficient and secure distribution to the managed devices. Figure 6 shows the Final Proposed Architecture.

A logical view of managed devices is created in the server web console for segregating the different devices as per their heterogenic nature and ease of patch deployment. On availability of a released patch, the patch deployment rules are defined through the web console. These rules include copying of files to relevant directories, configuration of execution environment, assignment of file permissions, and other required configurations. The patch files and rules configuration is stored by the server in the database and also at the Distribution server in a proprietary encrypted format which is not readable or modifiable by any user. Once the patch file is imported and configured, a patch schedule may be defined for installation at the managed devices or the installation can be manually invoked by the authorized user. Once the patch schedule is defined or invoked, the proprietary patch package is automatically distributed to the agents installed at the managed devices and independently installed at the managed devices. The status report of installation is synchronized between the agents and the patch management server and available at the web console for analysis.

The Patch Deployment Flow Process [Figure 7] highlights the broad areas for release and implementation of a software patch. All the involved processes such as Patch release, patch review, testing and verification of patch in development environment, and final verification after deployment have their own cycle of completion and the time taken to conduct these phases is more or less constant. The phase of Patch Deployment holds the potential for optimization and improvement impacting the overall cost of Patch Management Process.

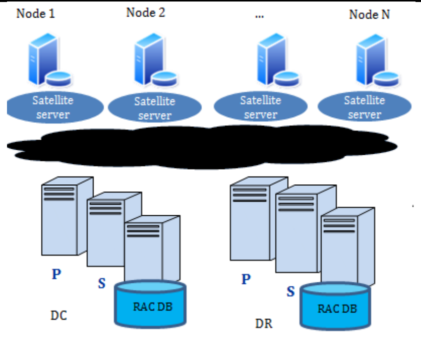

Figure 5: Patch Management Solution

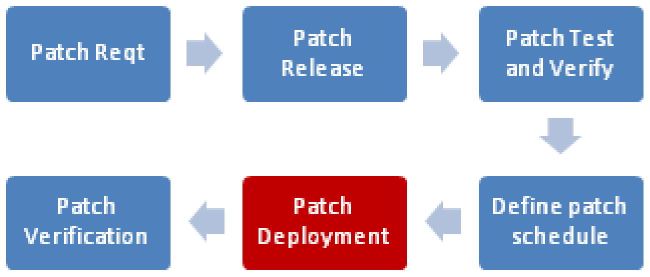

Figure 6: Patch Deployment Flow Process

4.Experimental Results and Discussion

In our Environment we have tested various patches of different categories for the deployment on the endpoints from the centralized location. The proposed flow of deployment is shown below in Figure 8.

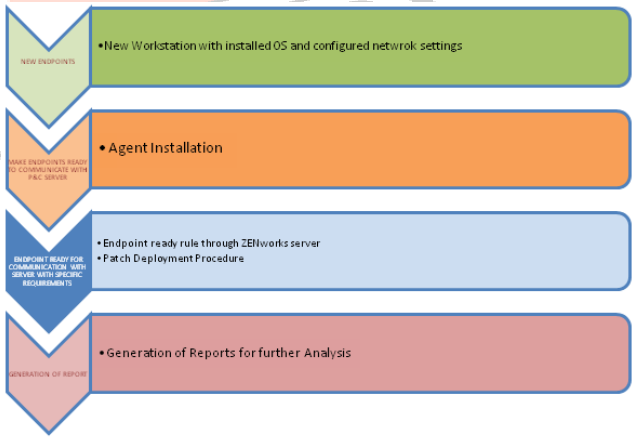
NEW ENDPOINTS
•New Workstation with installed OS and configured netwrok settings
MAKE ENDPOINTS READY TO COMMUNICATE WITH P&C SERVER
• Agent Installation
ENDPOINT READY FOR COMMUNICATION WITH SERVER WITH SPECIFIC REQUIREMENTS
•Endpoint ready rule through ZENworks server
•Patch Deployment Procedure
GENERATION OF REPORT
•Generation of Reports for further Analysis

Figure 7: Experiment Flow of Patch Deployment

The proposed solution is implemented on **Zenworks 2017**. The Configuration Management is selected as the Patch Management Solution and installed while the servers are installed in a virtualized environment; the Agents are involved on 100 physical machines of heterogeneous hardware (Servers/ Workstations) and operating systems (Microsoft Windows Server/ RHEL 7.1). For the prototype of our proposed solution, we have tested 10 different patches of different category for the deployment of various applications ranging in size from 1kb to 1 GB. The technological framework includes C++, JAVA, Web Development Framework, COTS software, MATLAB

2018R for the development purpose. The patches in the form of bundles were imported to the Zenworks server and their rule files were created. Content of rule files is given below:

- ✓ Copying of files
- ✓ Execution of environment parameter configuration scripts
- ✓ Setting of permissions
- ✓ Configuration of policies

In previous methods the patches used to be deployed using manual access of the end point which used to consume lot of man power, time, efforts and some may lead to the error. Since the manual deployment of patches requires root/administrative credentials which may be vulnerable to security. In our work, we have recorded the deployment time of these applications on 100 managed devices and compared with the previous used technology. We have also analyzed our work in terms of accuracy which is calculated using pass/fail status of the deployed patches.

Table 2: The workstation configuration for our Environment

Processor	Dual core i7
RAM	16 GB
Hard Disk	1TB
Software	MATLAB 2018
Languages	C++, java
OS Platform	Windows, Linux, Suse

We have considered different cases in our study for the experimental results and analysis. The previous and manual process takes much time in deploying different applications and patches. The repeated process may be mitigated by using the automated proposed approach that too with the increased accuracy.

Case 1: New Application received
Automated procedure:

a)Import of executables and configuration files in Patch Management Server
b)Creation of package (Patch + deployment rules)
c)Automatic deployment at managed devices

Manual procedure (through release media):

a)Manual copying of executables and configuration files in 100 systems
b)Manual configuration of operating environment
c)Manual assignment of file permissions

Case 2: New Application patch received
Automated procedure:

a)Import of executables and configuration files in Patch Management Server in existing package
b)Automatic deployment at managed devices

Manual procedure (through release media):

a)Manual copying of executables and configuration files in 100 systems
b)Manual assignment of file permissions

In real scenario, we have tested our prototype on more than 100 endpoints with different 100 applications/patches. Here we have listed out only few mathematical results as shown in table 3 and Table 4. Table 3 shows the time taken for a patch to be successfully deployed at end machine. From our mathematical expression, we can find out that the average time taken using proposed method come out to be 4 min which is far better from the previous method which gives average time approximately equal to 22 min. Table 4 shows the deployment status of few bundles which is helpful in finding the proposed algorithm's accuracy. The proposed method provides approximate 90 percent of accuracy. Figure 3 and Figure 4 shows the graphical representation of the results.

Table 3

Types of Patches	**Previous Approach(Time taken in min)**	**Proposed Approach(Time taken in min)**
Type 1	25	5
Type2	20	4
Type 3	27	6
Type 4	25	3
Type 5	15	5

Table 4

Bundles	**Deployment Status**
1	Pass
2	Pass
3	Fail
4	Pass
5	Pass

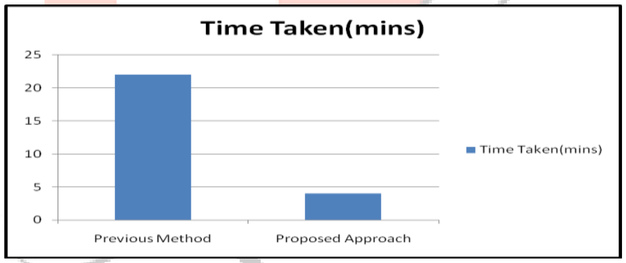

Figure 8: Experimental Results showing Time Taken

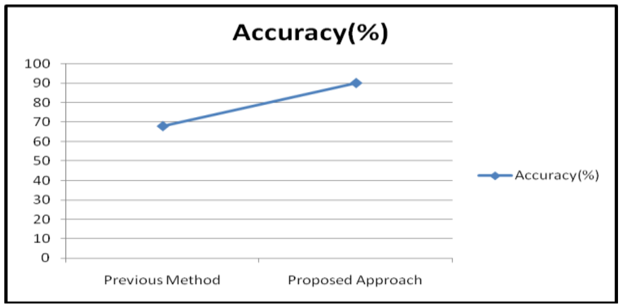

Figure 9: Experimental Results showing Accuracy

CONCLUSION

The implemented patch management framework solution with the agent and server modules has shown great improvement in patch distribution and installation time and efficiency. For large organizations, the initial cost of setting up and provisioning such framework shall prove far less as compared to dependency on manual procedures and provide substantial business returns in the longer run.

REFERENCES

[1] Cohen, Fred. (January 2004) Enterprise Patch Management: Strategies, Tools, and Limitations, SANS Institute http://www.burtongroup.com

[2] N. A. O_ce, Investigation: Wannacry cyber attack and the nhs (2017).

[3] Evaluation of Patch Management Strategies by HamedOkhravi and David M. icol,http://web.mit.edu

[4] Islam, M. A. Babar, S. Nepal, A multi-vocal review of security orchestration, ACM Computing Surveys (CSUR) 52 (2) (2019) 1-45.

[5] Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering{a systematic literature review, Information and software technology 51 (1) (2009) 7-15.

[6] A. Kitchenham, T. Dyba, M. Jorgensen, Evidence-based software engineering, in: Proceedings. 26th International Conference on Software Engineering, IEEE, 2004, pp. 273-281.

[7] M. Souppaya, K. Scarfone, Guide to enterprise patch management technologies, NIST Special Publication 800 (2013) 40.

[8] F. Li, V. Paxson, A large-scale empirical study of security patches, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 2201-2215.

[9] F. Li, L. Rogers, A. Mathur, N. Malkin, M. Chetty, Keepers of the machines: Examining how system administrators manage software updates for multiple machines, in: Fifteenth Symposium on Usable Privacy and Security (fSOUPSg 2019), 2019.

[10] Tiefenau, M. H • aring, K. Krombholz, E. von Zezschwitz, Security, availability, and multiple information sources: Exploring update behavior of system administrators, in: Sixteenth Symposium on Usable Privacy and Security (fSOUPSg 2020), 2020, pp. 239-258.

[11] S. Frei, D. Schatzmann, B. Plattner, B. Trammell, Modeling the security ecosystem-the dynamics of (in) security, in: Economics of Information Security and Privacy, Springer, 2010, pp. 79-106.

[12] Chalvatzis, D. A. Karras, R. C. Papademetriou, Evaluation of security vulnerability scanners for small and medium enterprises business networks resilience towards risk assessment, in: 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), IEEE, 2019, pp. 52-58.

[13] G. Stoneburner, A. Goguen, A. Feringa, Risk management guide for information technology systems, Nist special publication 800 (30) (2002) 800-30.

[14] U. Lakhina, N.Singh,I. Elamvazuthi, F. Meriaudeau, P. Nallagownden, G. Ramasamy and Ajay Jangra "Threshold based Load Handling Mechanism for Multi-Agent Micro grid using Cloud Computing," in *International Conference on Intelligent and Advanced System*, kuala lumpur, 2018.

[15] D. S. Linthicum, "Connecting Fog and Cloud Computing," IEEE Cloud Computing, pp. 18 - 20, 2017.

[16] P.-J. Maenhaut, H. Moens, B. Volckaert, V. Ongenae and F. D. Turck, "Resource Allocation in the Cloud: From Simulation to Experimental Validation," in *IEEE 10th International Conference on Cloud Computing (CLOUD)*, california, 2017.

[17] P. Geetha and C. R. Robin, "A comparative-study of load-cloud balancing algorithms in cloud environments," in *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, chennai, 2017.