Spider Package Manager

Srirama Dheeraj[1],S.Subashka[2]

[1] *Student, Computer Science,SRM Institute of Science & Technology,Tamil Nadu, India*

[2]*Professor, Computer Science, SRM Institute of Science & Technology, Tamil Nadu, India*

ABSTRACT

Package Managers are a trendy way to distribute software for modern operating systems.The main domain of this project is to automate the process of downloading and installing the Software. This can save a lot of time for the user without doing it manually. First using the provided software name as an input, Spider search for the software if the software exists, then the software will be downloaded and installed to the user's computer without any interaction. This has the capability of version control like it can download the latest version or previous version of the software. This is a command line interface in which the user can concurrently process the commands in different shells.

Keywords— package manager, web crawler, blind traversal algorithms, data mining, information retrieval.

1. INTRODUCTION

Package management is the task of determining which packages should be installed on a host and then downloading and installing those packages. Package managers are a popular way to distribute software (bundled into archives called packages) for modern operating systems. Package managers provide a privileged, central mechanism for the management of software on a computer system. As many packages are installed in the root context of the operating system, package management security is essential to the overall security of the computer system. The general purpose of the package manager are:

- Getting the software package name as an input from the user.
- Searching for the software and extracting the links and checking for the desired software with executable extension.
- Downloading the software package from the link Obtained.
- Installing the software package with security measures.
- Update the software package.
- Uninstall the software package.

2. LITERATURE SURVEY

There are a large number of package managers for Linux including Slaktool, pacman, YaST, urpmi, and Portage .These package managers have flaws similar to those in APT and YUM..To fetch information from the internet using the muti-thread crawlers[1] for scheduling the software search and download. To support package signatures,[6] in general these systems do not address the larger problem of key distribution, revocation, or trust delegation. dependency resolvers download the requested package as well as other packages needed to install the requested package Di Ruscio [7] describes strategies to overcome problems between upgrades.

3. ARCHITECTURE OF PACKAGE MANAGER

A package manager is one of the main components of the modern operating systems. The architecture of the package manager has spiders for crawling for the desired software packages and installer ,updater, uninstaller , downloader, item pipeline for scheduling the tasks of the package manager.

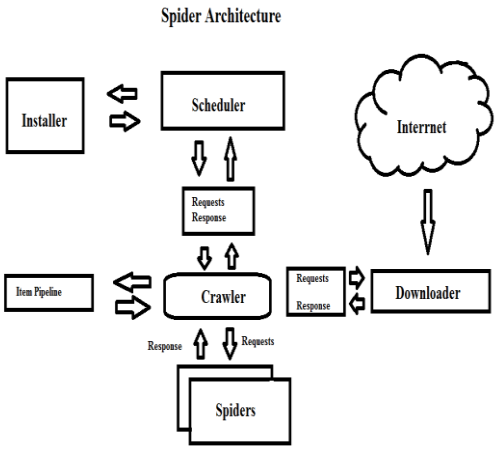

Fig -1: Spider Package Manager Architecture

4. OVERVIEW

The overview contains about the install, uninstall and update problems. It basically explains about the download process and installing through the security protocols.

4.1 The Install Problem

Consider a user with the installation profile "x" who wishes to install the package a. The install problem is to determine whether there is some set of new packages including a that can be added to the machine, such that the resulting set of packages is a valid installation profile**.** proceeds by traversing the dependency graph and building up the set of other packages that must be installed before a. To recognise the given command and validating if the package exits or doesn't exists. To be efficient it restricts the number of backtracks performed due to conflicts and thus loses completeness, in the sense that apt-get may incorrectly report that there is no suitable set of new packages even though one exists. Parsing the command and to satisfy the install problem we use SAT-based install checking.This package manager checks for the architecture of the windows operating system and installs the software according to the windows operating system architecture like 32bit or 64bit operating systems.

4.2 The Uninstall Problem

In many configurations a new package cannot be installed because of conflicting dependencies with other packages already installed on the system. In this case, we must first uninstall the packages prohibiting the installation before attempting to install the new package. We would like to find the optimal set of packages that must be removed in order to make the new package installable. To search for the package and uninstall the package, this has to be done in few seconds. To satisfy this we need enhanced SAT solver computes an over approximation of the packages that must be removed.

4.3 Security Problem

To install software with security protocols, for that it requires malware analysis and detection of the files. We have read several papers for handling security issues and processed this method. We are using exe filter for this problem and using trusted libraries in the package manager. Before installing asking user's permission for installing the software packages.

4.4 Putting it all together

In this we combine the altogether and adding a new feature called update command. In which spider package manager searches for the package and update the package. At first it searches for the package exists or not and if it exists it updates the package with the latest distribution of the package. This was tested with different ways and ran each installation attempt using the different configurations

5. EVALUATION

To evaluate the practicality of spider package manager we perform a comparative study of Debian's package installer, apt-get. The goal of this study was to quantify three measures: the running time of Opium versus apt-get, the amount of benefit provided by the completeness of spider package manager, and the amount of benefit provided by the minimization capabilities of spider package manager.

5.1 Runtime

The runtime of spider manager normalized to the runtime of apt-get. To get a sense of the scale, the average runtime of apt-get was 9.14 seconds. The dominant components of the Spider manager runtime are reading the distribution, performing the slicing optimization, and performing conflict resolution. The actual time to run Pueblo or GLPK accounts for only a very small proportion of the total runtime of Spider manager. The runtimes of install attempts that optimize size are very similar to the runtimes for attempts that optimize popularity, which is an indicator that runtimes are unlikely to depend significantly on the objective function.

5.2 Completeness

To quantify the benefit provided by spider manager's completeness we look at the number of times that the other package mangers fails to find a way of installing a package when in fact there is a solution (which spider is guaranteed to find because it is complete). Out of the several install attempts, other packager managers were not able to find a solution and out of these cases, Spider was able to find a solution with great efficiency. The completeness of Spider manager has the potential to improve the end-user experience for a large fraction of users.

5.3 Minimization

We first evaluate the impact of Spider manager's ability to minimize the number of packages that are removed from the system. For our traces, spider removed less packages than the other package manager in 209 cases out of the 52,311 install attempts where apt-get succeeded. This is a small percentage of all install attempts but the impact in those cases can be significant. In 9 cases apt-get removed 10 or more packages than was necessary, with the worst of these cases being the example mentioned in the introduction where apt-get removed 61 packages, including the kernel, whereas Spider only removed 21 packages, none of which was the kernel.

6. FUTURE SCOPE OF PACKAGE MANAGER

Already a lot of research is going on in the field of package manager extraction techniques and package manager security. In future work can be done to improve the efficiency of algorithms. Also, the accuracy and timeliness of the package manger can also be improved. The work of the different crawling algorithms can be extended further in order to increase the speed and accuracy of web crawling [9]. A major open issue for future work about the scalability of the system and the behaviour of its components. This can be implemented further to encrypt the packages for the security purposes.

7. CONCLUSION

Package managers are an important aspect of all the modern operating systems. Downloading and upgrading the packages without any user interaction requires a lot of security protocols. Building an effective package manager to solve different purposes is not a difficult task, but choosing the right strategies and building an effective architecture will lead to implementation of highly intelligent package manager. A number of crawling algorithms are used by the search engines. A good crawling algorithm should be implemented for better results and high performance. A good malware analysis algorithm should be used to solve the security issues.

8.REFERENCES

[1] Mutual Exclusion Principle for Multithreaded Web Crawlers,International Journal of Advanced Computer Science and Applications, Vol. 3, No. 9, 201a2.

[2]. J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "A look in the mirror: Attacks on package managers," Conference on Computerand Communications Security, 2008.

[3]. M. Eriksson, "An example of a man-in-the-middle attack againstserver authenticated ssl-sessions," 2009.

[4]. Justin Cappos and John Hartman. Why It Is Hard to Build a Long Running Service on Planetlab. In *Proc. of the 2nd Workshop onReal,Large Distributed Systems*, San Francisco, CA, Dec 2005.

[5]. Siddhartha Annapureddy, Michael J. Freedman, and David Mazi`eres.Shark: Scaling File Servers via Cooperative Caching. In *Proc. 2nd*

[6]. Package Manager Security, Anish Athalye, Rumen Hristov, Tran NguyenConference on Computerand Communications Security, 2011.

[7]. D. D. Ruscio, P. Pelliccione, A. Pierantonio, and S. Zacchiroli.Towards maintainer script modernization in foss distributions In International Workshop on Open Component Ecosystems 2009 (IWOCE 2009), . ACM, 2009.

[8].J. Samuel, N. Mathewson, J. Cappos, and and R. Dingledine,Survivable key compromise in software update systems.In Proceedings of the 17th ACM conference on Computer and communications security, pages 61– 72. ACM, 2010.

[9]. J. Aldrich, C. Chambers, and D. Notkin. Archjava: connecting software

architecture to implementation. In ICSE, pages 187–197, 2002.

[10]. F. Mancinelli, J. Boender, R. di Cosmo, J. Vouillon, B. Durak, X. Leroy, and R. Treinen. Managing the complexity of large free and open source package-based software distributions. In Proceedings of the International Conference on Automated Software Engineering (ASE 06), 2006.

[11]. D. L. Berre and A. Parrain. On SAT technologies for dependency management and beyond. In S. Thiel and K. Pohl, editors, 12thInternational Conference on Software Product Lines, Second Volume (Workshops), Limerick, Ireland, Sept. 2008. Lero Int. Science Centre, University of Limerick, Ireland.