DWARF – OKRUŽENJE ZA AUTORIZOVANO UPRAVLJANJE YUM/APT REPOZITORIJUMIMA
DWARF – THE FRAMEWORK FOR AUTHORIZED YUM/APT REPOSITORIES MANAGEMENT

Dušan Vudragović, Antun Balaž, Vladimir Slavnić, Aleksandar Belić, *Laboratorija za primenu računara u nauci, Institut za fiziku, Pregrevica 118, 11080 Beograd, Srbija*
Dusan Vudragovic, Antun Balaz, Vladimir Slavnic, Aleksandar Belic, *Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia*

Sadržaj – *Većina savremenih RPM kompatibilnih Linuks sistema koristi YUM ili APT alate za automatsko preuzimanje, konfiguraciju i instalaciju softverskih paketa. Upravljanje paketima se zasniva na konceptu softverskog repozitorijuma - lokacije sa koje se paketi preuzimaju i instaliraju na racunar, uz automatsko razresavanje medjuzavisnosti paketa. Kada softverski repozitorijum koristi i organizuje veliki broj saradnika iz različitih institucija, pristup repozitorijumu treba omogućiti svakom od njih. U ovom radu je opisano DWARF okruženje koje omogućava autentifikovani i autorizovani prenos RPM paketa i kreiranje APT/YUM repozitorijuma korišćenjem digitalnih sertifikata. DWARF je realizovan kao veb aplikacija koja pruža autorizovano organizovanje strukture repozitorijuma, prenos RPM paketa i nezavisnu izgradnju različitih delova repozitorijuma, odnosno repozitorijuma kao celine. DWARF se trenutno koristi u SEE-GRID-SCI Grid e-Infrastrukturi.*
Abstract – *The most of modern RPM-compatible Linux systems use YUM or APT tools for automating retrieval, configuration and installation of software packages. These package management utilities rely on the concept of software repository - storage location from which software packages may be retrieved and installed on a computer, with automatic resolution of package dependencies. When a software repository is managed and organized by many contributors from different institutions, access to the repository has to be provided to each of them. In this paper we describe the DWARF framework that allows RPM uploading and creation of APT and YUM repositories, with the authentication and authorization based on digital certificates. The DWARF is implemented as a web application that offers authorized repository structure organization, RPMs uploading, and independent building of different parts of repositories, as well as building of all repositories. The DWARF is currently deployed by the SEE-GRID-SCI Grid eInfrastructure.*

1. INTRODUCTION

Applications for most operating systems (OS) consist of multiple files that must be copied to specific locations on the computer's file system so that the application can be successfully executed. This is true for all common PC OS, such as MS-DOS [1] or Microsoft Windows [2], as well as for Linux [3] and other Unix-like OS [4].

In the typical case of Linux or some other Unix-like OS, additional issues must also be considered. Unix-like OS is always a multiple-user system, so it tracks the ownership of files precisely, using a well defined system of file permissions. Administrators can grant access to files on a per-user or per-group basis, and can control how users may access those files, for example, allowing some users the permission to read only certain files, while others may be given write access to the same set of files. Administrators can also deny access rights to the same set of files to some other users. So, installation of an application on Linux requires consideration of all these details. After files are copied into their appropriate locations, they must be configured to have correct permissions and correct ownerships.

Administrators occasionally need to upgrade or remove installed software from the computer. Usually, software upgrades are done by the removal of the application (its old version), followed by the installation of the new version of the application. However, upgrades have additional issues, since all applications must be properly configured before they can be used. The upgrade of an installed application takes its current configuration into account, preserving old configuration information and applying it to the newly installed version.

All these steps make installation of a new application onto Unix or Linux system a labor-intensive process. To further complicate matters, Unix applications have primarily been distributed as source code. Typically, the source code is provided in some sort of archive that first must be unpacked. After unpacking the source code, it should be configured to support the options and systems, and then compiled so as to produce an executable program that can run on a particular operating system.

After compiling the source code, the application should be installed by putting all of its components (executable programs, libraries, documentation, configuration files, etc.)

into the correct locations on a hard drive and setting correct permissions on all those files. Sometimes, it is needed to perform other steps to prepare the system for the software (allocate space for log files, create special user accounts associated with the application, etc.).

For all these reasons, typically a package management system is used to simplify the software installation and maintenance process. The package management system is a collection of tools used to automate the process of installing, upgrading, configuring, and removing software packages from a computer.

In such a framework, the software is distributed in packages, usually encapsulated into a single file. The package, together with the software itself, often contains metadata information that describe the package's details, including its name, checksums, and dependencies on any other packages that it needs to work. It may also include information on how to configure the package for use and how to remove the package cleanly when it is no longer required. The package manager then uses this information to install, configure, and remove packages as requested by the user/administrator. Upon installation, metadata is stored in a local package database.

2. PACKAGE MANAGEMENT SYSTEMS

Package management systems present a uniform and simple way for users to install and remove software with a single command. Different management systems usually provide many command-line and even graphical interfaces for interactive software installation. They also allow non-interactive installations, which is ideal for automated/non-attended maintenance of computers.

Each package manager system relies on the format and metadata of all the packages it is used to manage. That is, each package manager needs a group of files to be bundled for the specific package along with the appropriate metadata, such as dependencies etc. Often, a core set of utilities manages the basic installation of packages, while package managers are built on top of them, and use these utilities to provide additional functionality.

By the nature of free software, packages under similar or compatible licenses are available for the use on a number of operating systems. These packages can be combined and distributed using configurable and internally complex packaging systems to handle many possible permutations of the order different applications can be installed and to manage version-specific dependencies and conflicts. Some packaging systems of free software are also themselves released as free software. One typical difference between package management in proprietary operating systems, such as Mac OS X [5] and Windows, and those in free software OS, such as Linux, is that free software systems permit third-party packages to also be installed and upgraded through the same mechanism, whereas the package management system of Mac OS X and Windows will only upgrade software provided by Apple [6] and Microsoft [7], respectively (with the exception of some third party drivers in Windows). The ability to continuously upgrade third party software is typically added by adding the URL of the corresponding repository to the package management's configuration file.

RPM [8] (Red Hat Package Manager) is one of the frequently utilized package management systems. Originally developed by Red Hat [9] for Red Hat Linux [10], RPM is now used by many Linux distributions. It has also been ported to some other operating systems, such as Novell NetWare [11] (as of version 6.5 SP3) and IBM's AIX [12] as of version 4.

Working behind the scenes of the RPM Package Manager is the RPM database. It consists of a single database containing all the metadata information of the installed packages and multiple databases used for indexing purposes. The database is used to keep track of all files that are changed and created when a user installs a package, thus enabling the user to reverse the changes and easily remove the package later. If the database gets corrupted (which is possible if the RPM client is killed etc.), the index databases can be recreated using a command-line RPM interface.

Each RPM package has a label, which contains the following information: software name, software version (version taken from the original upstream source of the software), package release (number of times the package has been rebuilt using the same version of the software), and architecture the package is built for. Additionally, libraries are distributed in two separate packages for each version. One contains the precompiled code, while the second one contains the development files such as headers, static library files, etc. for the library in question. Those packages have "devel" appended to their name field. RPM files with the "noarch" extension refer to files that do not depend on computer's architecture, i.e. that do not to be compiled for a specific architecture (usually configuration files and/or scripts, that may rely on other packages, or programs written in interpreted programming languages).

Since RPM Package Manager is popular, one can find many places where the thousands of free application packages are made available publicly. These storage locations from which software packages may be retrieved and installed on a computer are called software repositories.

The default installer for RPM packages does not follow dependency information automatically. It requires the user to manually download additional missing RPMs potentially required by the package being installed. Moreover, circular dependencies between mutually dependent RPMs cannot be satisfied with the default RPM installer unless the user is aware that it is necessary to specify both of the RPMs together. This leads to what is known as "dependency hell", particularly for packages with many dependencies, each of which has its own large set of dependencies, and so on. For this reason, wrappers around the RPM installer tool have been created to help avoid such problems.

The most powerful and the most popular open source wrappers around the default RPM installer are: Advanced Packaging Tool [13] (APT) and Yellow dog Updater Modified [14] (YUM). Both of them are providing command-line and graphical user interfaces for retrieval, configuration

and installation of software packages, either from binary files or by compiling source code. Also, in order to find software and resolve dependencies, both of them rely on the concept of software repositories.

APT was originally designed as a front-end for Debian [15] package management system, but it has since been modified to also work with the RPM Package Manager system via apt-rpm [16]. The Fink project [17] has ported APT to Mac OS X for some of its own package management tasks, and APT is also available in OpenSolaris [18] (included in Nexenta [19] OS distribution). The Telesphoreo [20] is a project dedicated to porting APT to smartphone devices - currently to the iPhone [21].

YUM is a full rewrite of its predecessor tool, Yellow dog Updater (YUP), and was developed primarily in order to update and manage Red Hat Linux. Since then, it has been adopted by Red Hat Enterprise Linux [22], Fedora [23], CentOS [24], and many other RPM-based Linux distributions, including Yellow Dog Linux [25] itself, where it has replaced the original YUP utility.

When using each of two mentioned tools, an install directive is followed by the name of one or more packages desired for installation. Each package is usually specified just by the name of the package, and not by the full filename that contains information on the version and architecture. The APT or YUM will consult the configured software repositories and offer possible choices to the user. Alternatively, a specific distribution can be selected by specifying the package name with the version of the distribution. Because all packages contain the list of dependencies specified for installation, they will also be automatically retrieved and installed from software repository, if available. This was an original distinguishing characteristic of APT and YUM package management systems, preventing software installation failure due to missing dependencies, by automatically trying to satisfy all specified dependencies.

Another feature APT and YUM tool has is a remote repository retrieval of packages. A configuration file with a list of software repositories is used to locate the desired packages and retrieve them, and also to obtain information about all available (but possibly uninstalled) packages.

3. DWARF ACHITECTURE AND IMPLEMENTATION

In user/development communities where large number of partners/collaborators from different institutions jointly contribute to applications and RPMs built from applications' sources, it is useful to create a unique software repository that collects all such RPMs. In this case, in order to upload new version of a package, access to the repository should be granted to all contributors. This can be solved via a creation of a shared file system, but for security, scalability and reliability reasons, this is not a good approach. In this paper we describe DWARF framework, which provides such a collaborative environment for application developers/users.

The widely-used Public Key Infrastructure [26] (PKI) includes policies and procedures needed to create, manage, store, distribute, and revoke digital certificates. In such approach, each user has its own personal digital certificate, an electronic document which incorporates a digital signature which binds together a personal public key with an identity – information such as the name of a person or an organization, their address, etc. In a typical PKI scheme, the signature will be provided by a certificate authority (CA), an entity which issues digital certificates for use by other parties, and which is trusted by all involved entities.

Using PKI infrastructure, the newly developed DWARF framework provides possibility for RPM uploading and creation of APT and YUM repositories, with the authentication and authorization of users based on digital certificates. The DWARF architecture is shown in Fig. 1.

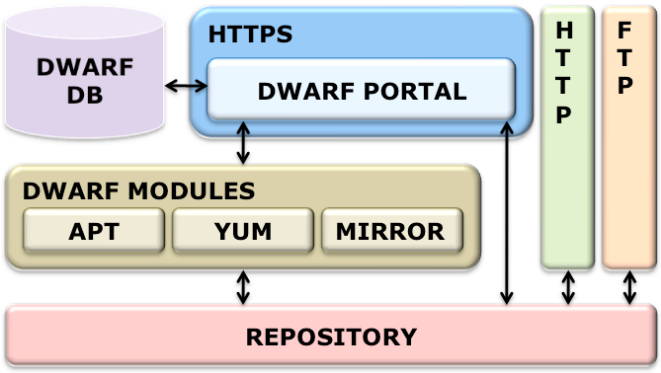

Fig. 1. Overview of the DWARF architecture.

DWARF is implemented as a web application and it is composed of the DWARF web portal, DWARF modules and DWARF database.

DWARF web portal [27], the frontend of the DWARF framework is implemented as a PHP [28] script under Apache HTTP server [29] on top of Secure Sockets Layer [30] (SSL). SSL protocol is a stronger authentication system that provides confidentiality, integrity, and authentication at the transport level. It is standardized as the Transport Layer Security protocol [31]. HTTP runs on top of SSL, which provides all the needed cryptographic strength. Integration at the server level allows the server to retrieve the authentication parameters negotiated by SSL, and SSL achieves authentication via public-key cryptography in digital certificates. There are a couple of different options for adding SSL to Apache. We use the Apache mod_ssl implementation [32]. This module provides the necessary SSL information as additional environment variables to the SSI [33] and CGI [34] namespace.

The super-user, whose digital certificate's distinguish name is defined in the configuration file of the portal, is allowed to perform the authorization and other tasks in all sections of the repository from the DWARF web portal. In this way, each connection to the DWARF portal is secured, authenticated and authorized.

The DWARF web portal home page, shown in Fig. 2, gives an overview of repository structure together with information on the context of each repository, and latest build's timestamp.

From the DWARF web portal, an authenticated and authorized user can perform following operations on the repository:

- *Create and change repository structure* – Users are free to create paths to new distributions and components, by specifying chosen names. In the current implementation of the DWARF framework, the users are able to create APT and YUM repositories, as well as to create a MIRROR to an existing remote repository.
- *Package uploading* – Users can upload different software packages, but only to sections of the repository to which they are authorized as contributors.
- *Build repository* – After each RPM upload, a user should build the repository structure. If not, a system will do it automatically, through a cron job.

DWARF modules are implemented as bash scripts that handle build action on repositories.

After an appropriate APT repository structure is created from the DWARF portal, the RPMs must be indexed to create the APT database. This is done by the APT DWARF module, which uses the genbasedir tool [35] for this purpose. It analyzes the RPM packages in a directory tree and builds information files so that that directory tree can be used as an APT repository.

The YUM DWARF module does the similar action when appropriate YUM repository structure is created. For this purpose yum-arch [36] and createrepo [37] tools are used. The yum-arch tool creates a headers directory, which supports older versions of YUM. It searches recursively through a repository structure for RPM packages, and includes all of them in the header data. The createrepo tool creates repository information to support newer versions of YUM (and possibly other repository client programs), and it also searches recursively for RPM packages to include in the repository data. To minimize problems with different YUM clients, both kinds of YUM repository data for each repository are created. The extra repository information is relatively small and does not affect proper function of the software repository.

The MIRROR DWARF module is responsible for mirroring some existing software repository locally. To synchronize remote software repository with the local, lftp tool [38] is used. Through the DWARF web portal, a user can specify a set of command-line switches that should be used to control the repository synchronization process. One example is enabling of the delete switch, which will ensure that the files in the local repository that are not present in the remote directory are deleted. Second example is the use of only-newer option, which forces lftp tool to download only newer files than the existing ones. Another very useful option is the exclude switch, which allows specification of files and directories that should be skipped during the synchronization. The DWARF log file will contain more information about the syncing process, if verbose switch is turned on. By default, each mirror repository will be synchronized six times a day via a cron job.

The DWARF database contains information on security, repositories type, repositories metadata, mirror repositories, and logging information. All information about users and users' permissions on different repository sections is stored here. Also, DWARF database contains metadata repository information on build's timestamps, contexts, and descriptions of the repositories, as well as repository types. The rules on how to create mirror repositories and information on when and with which options repositories are built are kept in DWARF database. In addition, for security reasons, the database contains a table with all users' actions recorded. The DWARF database is realized using MySQL database technology [39].

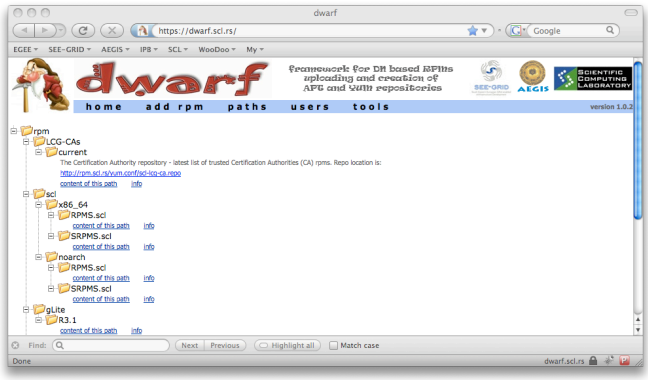

Fig. 2. Screen shot of the DWARF web portal.

Once the repository is constructed, it must be made available by HTTP and FTP servers configured and working on the DWARF web portal. The DWARF framework provides configurations that must be included in the local HTTP and FTP servers' configuration files in order to provide the context of repositories.

4. CONCLUSION

We have presented the DWARF framework that provides a collaborative environment for RPM uploading and creation of APT and YUM repositories, with the authentication and authorization based on digital certificates. The DWARF is implemented as a web application that offers authorized repository structure organization, RPMs uploading, and independent building of different sections of repositories, as well as rebuilding of all repositories. The DWARF framework consists of the DWARF web portal, DWARF modules and DWARF database. It is currently deployed by the SEE-GRID-SCI Grid eInfrastructure [40].

ACKNOWLEDGEMENTS

This work is supported in part by the Ministry of Science and Technological Development of the Republic of Serbia through research grant No. OI141035, and by the European Commission through projects CX-CMCS (FP6), SEE-GRID-SCI (FP7) and EGEE-III (FP7).

REFERENCES

[1] MS-DOS 6 Technical Reference, http://technet.microsoft.com/en-us/library/cc743176.aspx

[2] Official Microsoft Windows Website, http://www.microsoft.com/Windows/

[3] The Linux Foundation, http://www.linux-foundation.org/

[4] Unix-like Definition by The Linux Information Project, http://www.linfo.org/unix-like.html

[5] Official Mac OS X website, http://www.apple.com/macosx/

[6] Apple Inc., http://www.apple.com/

[7] Microsoft Corporation, http://www.microsoft.com/

[8] Red Hat RPM Guide, http://docs.fedoraproject.org/drafts/rpm-guide-en/index.html

[9] Red Hat official web page, http://www.redhat.com/

[10] History of Red Hat Linux, http://fedoraproject.org/wiki/History

[11] NetWare Cool Solutions, http://www.novell.com/coolsolutions/netware/

[12] IBM AIX page, http://www-03.ibm.com/systems/p/os/aix/

[13] Wikipedia: Advanced Packaging Tool, http://en.wikipedia.org/wiki/Advanced_Packaging_Tool

[14] Yum website, http://yum.baseurl.org/

[15] Official Debian website, http://www.debian.org/

[16] Apt-rpm home page, http://apt-rpm.org/

[17] Fink project homepage, http://www.finkproject.org/

[18] The OpenSolaris developer community website, http://www.opensolaris.org/

[19] Nexenta OS website, http://www.nexenta.org/

[20] Bringing Debian APT to the iPhone, http://www.saurik.com/id/1

[21] Apple's iPhone website, http://www.apple.com/iphone/

[22] Red Hat Enterprise Linux homepage, http://www.redhat.com/rhel/

[23] Fedora Project homepage, http://www.fedoraproject.org/

[24] Official CentOS site, http://www.centos.org/

[25] Yellow Dog Linux home page, http://en.wikipedia.org/wiki/Yellow_Dog_Linux

[26] Ed Gerck, *Overview of Certification Systems: x.509, CA, PGP and SKIP*, The Black Hat Briefings '99, http://www.securitytechnet.com/resource/rsc-center/presentation/black/vegas99/certover.pdf

[27] Scientific Computing Laboratory of the Institute of Physics Belgrade, DWARF web portal, https://dwarf.scl.rs/

[28] PHP: Hypertext Preprocessor, http://www.php.net/

[29] The Apache HTTP Server Project, http://httpd.apache.org/

[30] The SSL Protocol: Version 3.0, Netscape's final SSL 3.0 draft (November 18, 1996), http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

[31] SSL/TLS in Detail, Microsoft TechNet., http://technet.microsoft.com/en-us/library/cc785811.aspx

[32] The Apache Interface to OpenSSL, http://www.modssl.org/

[33] Apache Tutorial: Introduction to Server Side Includes, http://httpd.apache.org/docs/1.3/howto/ssi.html

[34] RFC3875: The Common Gateway Interface (CGI) Version 1.1, http://www.ietf.org/rfc/rfc3875.txt

[35] Jorge Godoy, Alfredo Kojima, Claudio Matsuoka, *APT+RPM HOWTO*, http://www.ccl.net/cca/software/UNIX/updating-redhat/apt-howto/

[36] Yum-arch - Linux man page, http://linux.die.net/man/8/yum-arch

[37] Createrepo - Linux man page, http://linux.die.net/man/8/createrepo

[38] Lftp - Linux man page, http://linux.die.net/man/1/lftp

[39] MySQL database website, http://www.mysql.com/products/database/

[40] SEE-GRID eInfrastructure for regional eScience, http://www.see-grid-sci.eu/