UDC 004.056:004.77

K. LOBACHOVA

National Tavrida University, Ukraine

A MULTIFUNCTIONAL SYSTEM FOR ASSESSING AND COMPARING SOFTWARE DEPENDABILITY

The new software dependability assessment and bug tracking system is presented. The key features like visual reporting, charting and dependability comparison, vulnerabilities and update notifications, task management, NVD data import, etc. as well as the main purpose and functionality of the system are described. The system is universal and can work with wide range of custom, OTS and combined software products.

software dependability, vulnerability reports, issue management

Introduction

It is common knowledge that nowadays it is virtually impossible to produce a flawless bug-free piece of software. Software systems are becoming very sophisticated involving various technologies, different programming techniques and aspects, under such circumstances the errors are inevitable.

That is why software dependability is often a significant consideration, high level of dependability is quite an important part of the modern software product.

But unfortunately a certain difficulty exists in assessing the correctness of software especially for the complex systems. The amount of vulnerability information in different products is virtually unmanageable. There are number of resources specially designed to help identify and solve the known security problems before a hacker takes advantage of them. CVE [1], NVD [2], Secunia [3], SecurityFocus [4], OVAL [5], CERT [6] – it is not a complete list of such vulnerability channels. We reviewed many of them, some reviews, related information and assessment results were provided in our previous works [7], [8], [9]. So we admit that such resources can be of good help to the developers concerned about security, however they all are far from being universal as they can cover only a limited part of all the developer needs related to dependability assessment. Being software developers ourselves, we decided to create a special dedicated system for software developers where they can register their product, track its vulnerabilities and report about them to the community, subscribe to vulnerability reports from other software vendors, assess the dependability of the complex software products and present the results in a compact, comprehensible, visual form using graphs and diagrams.

The structure of this document is as follows. Introduction Section describes relevant background information and provides a brief overview of our previous experience in the subject. It helps to understand the area in which the new dependability assessment system can be useful for the software developers and thereby the reason we started working on this issue.

The rest of the paper deals with the system itself. For the sake of clarity we split the system description into several sections so Section 1 outlines only the general idea and the main functionality of the system, whereas the more detailed descriptions of some parts of the system are provided later in sections 2 and 3. Section 2 describes the task management and failure/bug recording functionality and Section 3 describes visual reporting, charting and dependability comparison features. Section 4 mentions some world popular systems and resources, raising a point of compatibility and cooperation between them and our system. In the final section some concluding remarks, the outlook for the future and the focus of our further work are indicated.

1. System overview

As dependability assessment often becomes a problem, we decided to create a system that should help companies and individual software developers in this field. The aim of the system is to assist in the bug tracking and management processes, analyze software dependability during the life cycle and present the results in a visual easily accessible form. The main idea is to provide the infrastructure, the place where developers can add, store, change and manage their product vulnerability information, receive the notifications about the software they are interested in and create such notifications about their own software. All the information should be recorded in the database and be able to be retrieved at any moment to review dependability characteristics, draw graphs and diagrams, compare different products in terms of reliable functionality etc. using a special handy tool provided by our system.

The essential part of the system is database. The structure of the database is presented on figure 1.

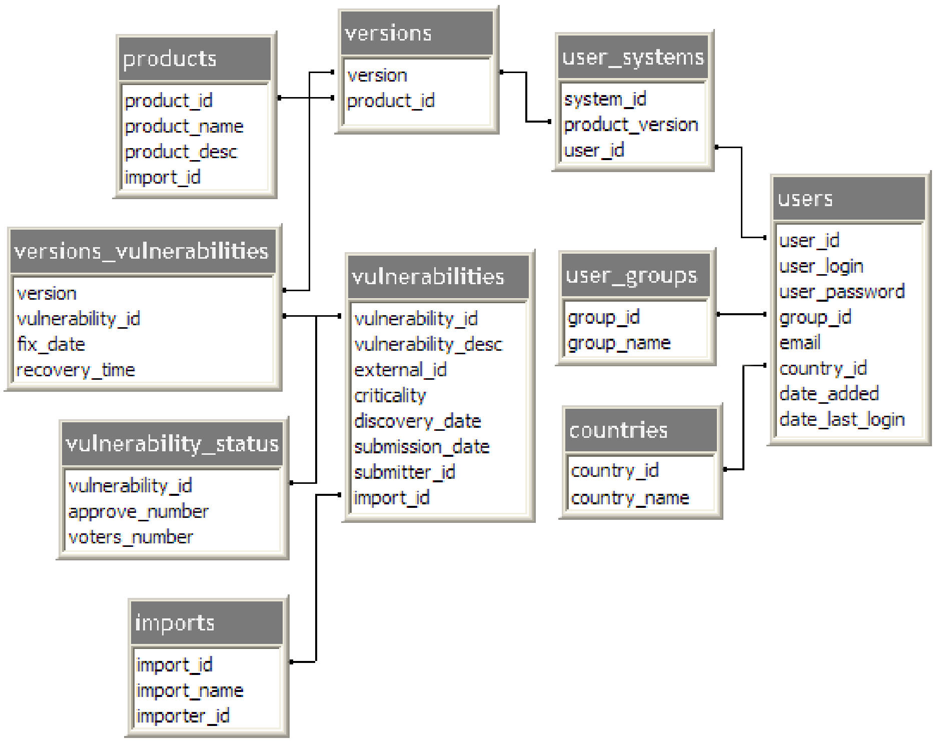

Fig. 1. Database relationship diagram

As can be seen, there is a 'vulnerability' table which collects, groups, and indexes vulnerability information. It can contain the following data: vulnerability id, vulnerability description, external id, criticality, discovery date, the date of submission, the id of the user who submitted the information and the id of the import. This import id corresponds to the 'import_id' value from the 'imports' table used to let users import their own sets of vulnerability data into our database.

'Vulnerability' table is related to the 'product_version' and successively to the 'product' table via the cross reference 'versions_vulnerability' table. Besides the vulnerability and product id-s this cross reference table stores the date of fix missing in many vulnerability systems. This date is quite important as it allows us to check the amount of time it takes for a vendor to prepare a patch and return back to stable operation after a vulnerability report.

With a reasonable degree of accuracy this characteristic can be called “recovery time” as vulnerability disclosure is very similar in its essence to system failure. So this information will be of good help in dependability assessment and product comparison. To allow users to create their own systems composed of several products the 'user_systems' table is added.

This table is designed to store the information about custom system components and the id of the user who composed such system. Non-registered users are also allowed to create and test the systems however their configuration will not be available long and will automatically be removed after a certain period of time.

As the registered and non-registered users were mentioned, it should be noted that there is 'users' table which is responsible for storing personal information and keeping track of the system users. 'User_group' is apparently used to define the group each user belongs to. It helps to control permission in our system.

2. Task management system

Task Management System is often used to help people coordinate and share task details and are designed to help the project manager and developers control a particular project or series of projects. The system should take care of many aspects of a task construction and its operation. In this chapter the structure and the functions of the Task Management System will be discussed.

The internal structure with the table relationships is shown on figure 2.

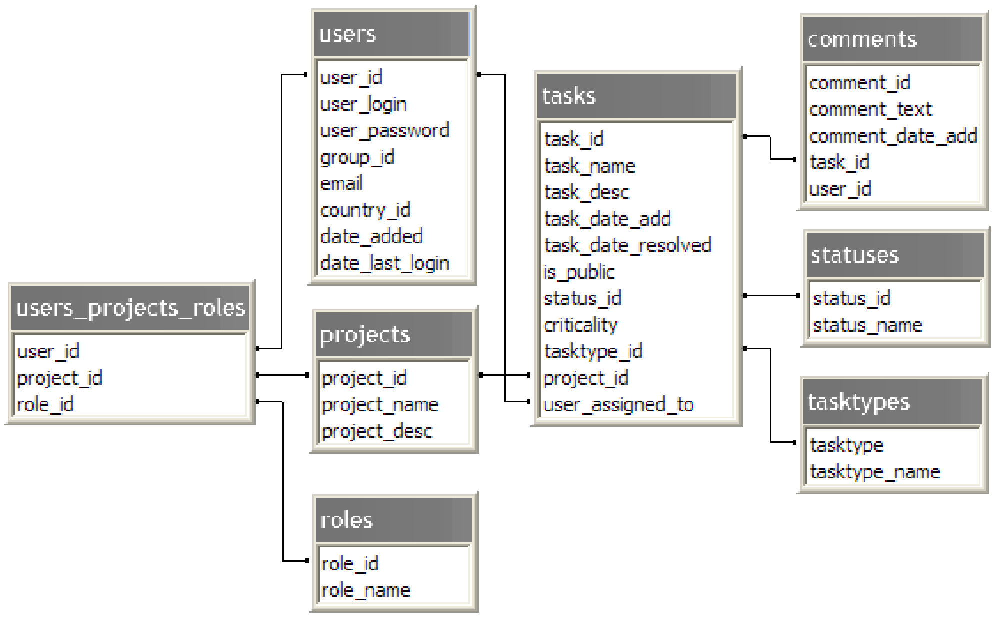

Fig. 2. Task management system: table relationship diagram

The diagram not only demonstrates the connectivity between tables, it helps to see the key functionality of the system. There can be different projects with multiple tasks investigated in each of them. 'Tasks' table is linked to the 'comments' table in a one-to-many relationship as each task is discussed by the developers and many comments can be posted before it gets resolved. This feature of keeping track of communication and actions performed to complete tasks is extremely useful for the project managers and the developers involved. Of course, each developer can be involved in several projects as well as each project can shared by multiple developers, that's why there is a many-to-many relationship between 'users' and 'projects' tables. However each project should have a manager who plan tasks, identify task personnel who will be directly involved in carrying out each task, assign tasks to the employees and control their execution checking the information about working state at any stage and allocating responsibility to the developers. Developers should not be able to re-assign their tasks or change the status of the tasks other than the ones assigned to them. So the 'roles' table which allows to controls permissions within the system is quite necessary. The main roles for now are project managers, developers and a global administrator who coordinates the process as a whole. Of course, unauthorized people are prevented from viewing and changing any details. Other tables like 'task_types' and 'criticality' allow to differentiate the issues by their nature, assign criticality rating and therefore priority level to each task.

Our Task Management System has complete browser-based interface and allows to administer projects/tasks remotely operating in an Internet or Intranet environment.

3. Visual reporting feature

This chapter describes one of the key features of our system that lets users draw their own custom graphs and diagrams based on software vulnerability information. It is a very useful option for managing the project in a visual way so it can be of good help to software developers, project managers and any other people who are interested in tracking software dependability characteristics.

The working area of the program is conventionally divided into two parts: area that allows selecting different options and parameters for the graph, and the section where the actual graph will appear.

The system provides the ability to create and present various kinds of diagrams such as:

1. Total severity rates of the discovered bugs for each month calculated as a sum of total severity rates of vulnerabilities found within each month.
2. Graphs of cumulative number of vulnerabilities found during some specified period of time.
3. Severity pies demonstrating the percentage of critical and non-critical vulnerabilities found.
4. Graphs taking into account recovery time information.
5. Graphs and diagrams:
   – based on the separate product information,
   – comparing different pieces of software,
   – presenting the information of the composite system consisting of several software products,
   – etc.

The system is quite flexible so users can combine different options and settings to obtain different resulting graphs.

Being a good way of visualizing information this feature should be of great help in analyzing and comparing software dependability characteristics.

4. Compatibility with other systems

It is clear that our system will have a better chance of success if we provide compatibility with other modern bug tracking and task management systems as well as with popular vulnerability resources.

The starting point was National Vulnerability Database - one of the most sophisticated vulnerability channels that integrates all publicly available US government vulnerability resources. The information from NVD is imported in our system and can be used for software analysis and product comparison. All the new advisories are automatically added to our database whenever they become available.

Support of other vulnerability resources and popular bug tracking systems will be implemented in the future system updates.

Conclusion

In this paper we gave a brief overview of the difficulties developers experience when trying to track bugs and vulnerabilities in software products. To make their lives easier and help them in this area we proposed a new system which provides a set of useful features for working with OTS and custom products, managing tasks, tracking bugs, receiving vulnerability and update notifications, assessing and comparing dependability characteristics. The main functionality of the system is already implemented and available for use. However the further work is needed to add even more dependability-related features and provide compatibility with other world-popular systems and resources. Moreover, we plan to greatly enhance dependability assessment feature so that our system finally maturate to the advanced multi-purpose and feature-rich environment with decision-making functionality aimed to help people improve software dependability characteristics.

References

1. Mitre Corp, Common Vulnerabilities and Exposures February 2008 [Электронный ресурс]. – Режим доступа: http://www.cve.mitre.org.
2. National Vulnerability Database, February, 2008 [Электронный ресурс]. – Режим доступа: http://nvd.nist.gov.
3. Vulnerability and Virus Information, February, 2008 [Электронный ресурс]. – Режим доступа: http://secunia.com.
4. Community of Security Professionals, February, 2008 [Электронный ресурс]. – Режим доступа: http://www.securityfocus.com.
5. Open Vulnerability and Assessment Language, February, 2008 [Электронный ресурс]. – Режим доступа: oval.mitre.org.
6. Computer Emergency Response Team, , 2008 [Электронный ресурс]. – Режим доступа: www.cert.org.
7. Lobachova K.I., Kharchenko V.S. Assessing Software Vulnerabilities and Recovery Time: Elements Of Technique And Results // Radioelectronic and Computer Systems, 2007, №8, P.61-65.
8. Lobachova K.I., Kharchenko V.S. Researching dependability of Apache and IIS web-servers // Proceedings of International Conference INFOTECH, Sebastopol, Ukraine, September, 10-16, 2007.
9. Lobachova K.I., Kharchenko V.S. A Conceptual Approach to Assessing Composite Component-Based Software System Reliability // Proceedings of International Conference ACSN2007, Lviv, Ukraine, September 20-22, 2007.
10. Reliability Software & Services, February , 2008 [Электронный ресурс]. – Режим доступа: http://www.relex.com.
11. CASRE Computer Aided Software Reliability Estimation February, 2008 [Электрон. ресурс]. – Режим доступа: http://www.openchannelfoundation.org.

Надійшла до редакції 22.02.2008

Рецензент: д-р техн.наук, проф. В.С. Харченко, Національний аерокосмічний університет ім. М.Е. Жуковського «ХАІ», Харків.