SisBrAV – Brazilian Vulnerability Alert System

Robson de Oliveira Albuquerque, Daniel Silva Almendra, Leonardo Lobo Pulcineli, Rafael Timoteo de Sousa Junior, Claudia J. B. Abbas

Universidade de Brasília - Campus Universitário Darcy Ribeiro - Faculdade de Tecnologia - Depto de Engenharia Elétrica e Redes de Comunicação - Laboratório de Redes - sala B1 - CEP: 70910-900 - Brasília - DF – Brazil

Luis Javier Garcia Villalba

Universidad Complutense de Madrid (UCM) - Departamento de Sistemas Informáticos y Programación (DSIP) - Facultad de Informática, Despacho 431- C/ Profesor José García Santesmases s/n - Ciudad Universitaria - 28040 Madrid - Spain

Abstract. This paper describes the project and implementation of a vulnerability search and alert system based on free software. SisBrAV (acronym in Portuguese for Brazilian Vulnerability Alert System), will consist in a spider mechanism that explores several security-related sites for information on vulnerabilities and an intelligent interpreter responsible for analyzing and sorting the relevant data, feeding it into a database. With that information in hands, an email notifier sends out alerts, in Portuguese, about new vulnerabilities to registered clients, according to the operating systems and services run in their environment. In addition to the email notifier, a web server will also be implemented, for systems administrators to perform an on-demand custom search in the vulnerabilities database.

1 Introduction

In a daily basis, a large number of vulnerabilities, which affect a variety of systems and services, are detected. Manufacturers and developers work extensively in order to release, as fast as possible, a patch that fixes the problems found in their products. On the other hand, the hacker community is continually growing, producing malicious codes, exploits and viruses that take advantage of those vulnerabilities very rapidly. With the incredibly large quantity of information that can be found on the Internet today, as well as the increasing number of hacker sites that provide easy access to lots of malicious tools and exploit codes, it is of great importance that every enterprise's systems security team be well advised and informed about what are the threats to their environment and what they can do to avoid them, protecting their systems, services and network quickly and proactively. Even individuals with one or two PCs at home should be concerned with their system's vulnerabilities, applying the latest patches in their software, avoiding any security problem that may happen.

de Oliveira Albuquerque R., Silva Almendra D., Lobo Pulcineli L., Timoteo de Sousa Junior R., J. B. Abbas C. and Javier Garcia Villalba L. (2005).
SisBrAV – Brazilian Vulnerability Alert System.
In *Proceedings of the 3rd International Workshop on Security in Information Systems*, pages 67-76
DOI: 10.5220/0002578700670076
Copyright © SciTePress

Existing vulnerability database systems are created and maintained by human administrators, who are responsible for searching, analyzing and evaluating new vulnerabilities everyday, and then updating the database regularly with new entries, in a pretty much manual process. The initial idea of the project depicted in this paper, was that there could be an automatic process of gathering the relevant vulnerabilities information, sorting it according to predefined rules, feeding a database and generating email alerts for specific recipients whose environment could be affected. The solution should be based on free software and should also be portable to many platforms. SisBrAV project is, thus, the result of that idea.

2 Related Issues

Up to this date, in Brazil, there isn't any system such as SisBrAV, which automatically looks for new vulnerabilities and informs the users about them. In the other hand, a large number of security sites can be found in the Internet, and almost all of them have a vulnerability alert section, updated daily, enclosing vulnerabilities for many systems and programs. Thus, information regarding vulnerabilities can be easily accessed through the Internet, but it is very difficult to glimpse which vulnerabilities represent real threats among the large number encountered. So, there is plenty of information, but a lack of simplicity in the process of filtering these pieces of information, in order to keep only in the important ones.

The main challenge in the SisBrAV project is the sorting process, since the system will search for vulnerabilities in many sources, and each of them organizes the information in a particular way. The interpretation of the data collected must be very precise, as well as the sorting process, since the clients must be informed only about the threats to his specific environment. The importance score for each vulnerability must also be precisely assigned, making it possible for the client to assign different priorities when establishing security countermeasures for the vulnerabilities he has been informed about.

Two other elements are also critical for the efficiency of the SisBrAV system: the organization of the vulnerability information and the generation of customized alerts to each client according to his systems and services. The information must be sorted in an accurate but simple manner, and the alerts must be clear and succinct, as well as they must be sent only to the clients whose environment is threatened by the vulnerabilities.

SisBrAV will implement a module for each function it performs. The following section will describe how all these modules work and what functions they perform.

3 SisBrAV Modules

SisBrAV will be consisted of 5 modules. The Vulnerability Search Mechanism (VSM) module will consist in a spider that accesses and indexes many vulnerability documents in several security sites. The Interpreter, Parser and Sorter (IPS) module will be a program that analyses the data provided from the spider, defining priorities and classifying the entries, according to predefined rules. The Central Database (CDB) module will store all vulnerability data, clients' info and keywords for English-Portuguese translation. The Email Notifier (E-Note) module will alert by email the registered clients about new/updated vulnerabilities specific to each client's environment. At last, the Vulnerability Web Server (VWS) module will be a server, accessible by any registered client, to perform an on-demand, customized vulnerability search in the Vulnerability Database. The details of each module are depicted in the next sections.

3.1 Vulnerability Search Mechanism

The vulnerability search and indexing process is made by a spider mechanism. A spider is a program that explores the Internet by retrieving a document and recursively retrieving some or all the documents that are referenced in it. It acts as an untiring human being who follows all links he finds in a web site, and all the links in the subsequent documents he sees. The spider indexes (fully or partially) all the documents that it accesses into a database, which can afterwards be used by a search engine.

The spider tool used in SisBrAV will be htdig, which is one of the programs that constitute the Ht://Dig package (6). Ht://Dig is a free web search engine, created in accordance to the GPU (General Public License) rules, and is consisted of many individual programs, like htdig, htdump and others.

The most recent stable version of Ht://Dig is 3.1.6, so this will probably be the version implemented in SisBrAV. A brief description of the htdig program is necessary, for there are some options which are used in the system, for its best performance and accuracy.

Htdig is a spider program (or search robot), which does what is called the "digging" process, retrieving HTML documents using the HTTP protocol, gathering information from these documents and indexing them, creating specific database files which can then be used to perform a search through these documents.

Htdig has many options, which are/will be used in the SisBrAV system, either to produce a desired result or for debugging purposes. The –c <configfile> option specifies another configuration file instead of the default. Another important option is the -h <maxhops> option, used to restrict the dig to documents that are at most maxhops links away from the starting document. This option is used every time the initial digging is run, to assure that htdig will index only the relevant documents for each site. The –i option is used to perform an initial digging. With this option, the old databases are removed, and new ones are created. There are also some options very useful for debugging, such as –s and –v, used to print statistics about the dig after completion and to set the verbose mode, respectively. For test purposes, one important option is the -t option, which tells htdig to create an ASCII version of the document database, making it easier to parse with other programs, so that information can be extracted from it for purposes other than searching. It generates the files db.docs and db.worddump, which formats will be explained later.

Finally, the url_file argument can also be passed, telling htdig to get the URLs to start indexing from the file provided, overriding the default start_url in the configuration file.

As said before, when using htdig with the –t option, it produces two ASCII files, db.docs and db.worddump. The db.docs file contains a series of lines, each of them relating to an indexed document. The lines contain information such as the document URL, title, modification time, size, meta description, excerpt, and many other useful document information. The db.wordlist file has a line for each word found in the indexed documents. Each line in this file contains a word, followed by the document ID where it was found, its location in the document, its weight, number of occurrences, etc.

The default configuration file for htdig is the file htdig.conf. That's where all configuration options for htdig (and the other tools, if they are used) are set. Since all of its parameters will probably be left with their default values, this file's content will not be copied in this paper. At first, the security sites indexed by SisBrAV's htdig will be the ones listed in the items (5), (7), (8), (9) and (10) in the References section. The number of sites can be (and will be) expanded to a much higher number, but initially only these five sites were chosen. The way htdig indexes each site will be almost the same: the only parameter that will differ from one site to another is the number of hops from the starting URL. For example, if maxhops is set to 2, htdig will index the starting URL, then it will follow all the links in that URL and index all the documents, and finally it will also follow the links in these documents, indexing the documents it finds, and then stop the digging process. Since each site has its way of displaying their documents, the number of hops necessary to gather all relevant vulnerability information will vary from site to site.

To solve this issue, a simple UNIX bash script will be used to read a file that contains lines with an URL and a number (which defines the maximum hops from the initial URL), separated by a TAB. The script will produce different htdig commands, according to the number of maximum hops defined. The number of maximum hops for each site is defined by the SisBrAV administrators, who inspect the sites and check the number of levels the spider will have to crawl down in order to obtain the maximum amount of relevant information about the vulnerabilities, and the minimum unnecessary information.

Htdig generates several Berkeley DB type files. These files will then be analyzed by the IPS Module, as explained in the next section.
by the IPS Module, as explained in the next section.

3.2 Interpreter, Parser and Sorter

The IPS Module will probably be written in Java. It will use an heuristics algorithm to perform the content analysis of the data stored in the Berkeley DB files created by htdig, in order to feed the Central Database with accurate vulnerability information. The data is parsed and the vulnerabilities are grouped between previously determined, hierarchically distributed classes.

At first, the IPS program will perform the sorting process. Initially, it analyses all the entries in the database, to find ambiguous or duplicated information for a same vulnerability. Then, it parses the content of the information, in order to group the vulnerability entries in classes, according to its main aspects: remote/local, type, low/medium/high importance score, etc. It also determines the systems/services in which that vulnerability occurs. If there is more than one entry for the same vulnerability, it correlates all the information found in the entries, to make sure the attributes are set as precisely as possible. For example, if a given vulnerability is issued in three different sites, and one of them scores the vulnerability as of medium importance and the others say its importance is high, the IPS will set this attribute to "high".

The hierarchical class tree used to group the vulnerabilities is described in figure 1.

Each document indexed by the spider in the VSM module will be related to a specific vulnerability. The IPS module performs the vulnerability sorting process for each document, by following the tree shown in the above figure. Initially, the algorithm determines if the vulnerability is local or remote, according to the information found in the document. It then classifies the vulnerability into a specific vulnerability type, among the predefined types registered in the system, such as Denial of Service, Buffer Overflow, Password Retrieval, Authentication Bypass, etc. Afterwards, an importance score is assigned to the vulnerability. At last, the IPS finds out what operating systems – and their versions – are affected by the vulnerability, and what programs/services – and their versions – are threatened by it. As well as the vulnerability types, there will also be a large list of systems and services (and their respective versions), which IPS will use in the sorting process.

After a given vulnerability is sorted, the IPS checks if there is any other vulnerability with exactly the same characteristics, affecting the same systems/services. If so, it performs a series of tests, to check if both entries refer to the same vulnerabilities. In these tests, other information is analyzed, such as the vulnerability date, the document URL (if the root site is the same, it's probably not the same vulnerability, since a security site must not have duplicated documents for the same vulnerability), and other information.

After the vulnerabilities have been classified, the IPS feeds the Central Database with that data.

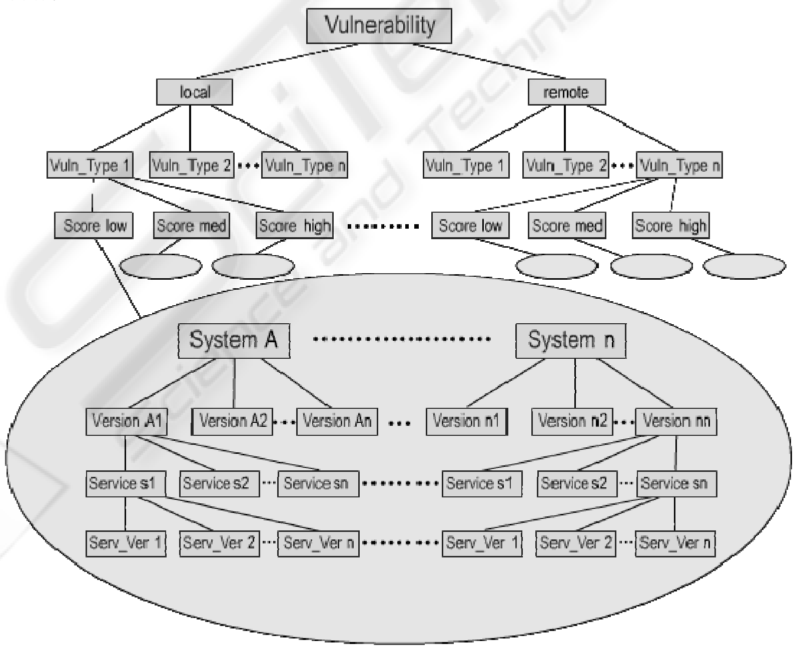

Fig. 1. IPS – Hierarchical Classes Tree: the vulnerabilities are sorted and grouped into different classes

Since the database is not hierarchical, but relational, the IPS will also have to convert the results of the sorting process before actually feeding the Central Database.

3.3 Central Database

In order to store all the information regarding vulnerabilities and their attributes, clients' profiles, systems and services data, as well as the English-Portuguese translation data, SisBrAV will have a Central Database. It is most likely that it will be implemented using a MySQL server, which is GPU compliant, and its architecture will follow the SQL ANSI standard, to guarantee its portability and scalability.

The CDB will be divided into three smaller databases, each one storing specific information, although the three of them relate to each other. The first database is the Vulnerability Database, which will contain all the vulnerability information already sorted into defined groups, as seen in the IPS section. The second base is the Client Database, which will keep the client-related data, such as their names, contact information and the systems and software running in their environment. At last, the third database will be the English-Portuguese Translation Database, storing a number of keywords, each one relating to keywords in the other idiom, according to certain parameters.

Mostly based on the schema designed by the Open Source Vulnerability Database Team (5), the Vulnerability Database is the most important part of the whole SisBrAV system, for it is the central repository of all vulnerability information. Its structure, which is still being developed, will probably keep the main OSVDB structure, although there will be some changes in certain tables, and other tables will be removed or added. The Vulnerability Database, when fully implemented, will be similar to figure 2.

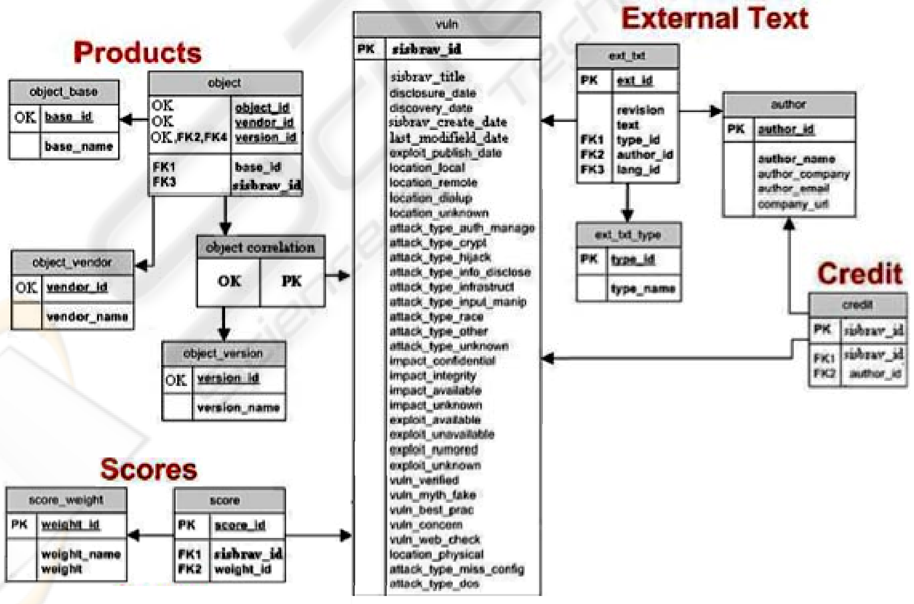

Fig. 2. Vulnerability Database Schema

The External Text section in this database consists in tables that describe certain aspects about a vulnerability. They exist inside the database, but usually describe information that one would use externally to the database. For example, a Solution Description, or a Vulnerability Description is an external text.

The tables in the above schema also deserve some explanation. The vuln table is the main table in the schema. It's where the SisBrAV IDs live. Other information stored in this table includes various dates and vulnerability classification data. The ext_txt_type table defines the types of external texts. For example, Vulnerability Description, Solution Description, Technical Description, Manual Testing Notes. The ext_txt table stores the external text blobs for any type of text that is larger than 1024 characters. Other information stored is the language, type, author, and revision. When the texts are updated/fixed/modified the new text is reinserted into this table and the revision number is incremented. The contributors for anything in the ext_txt table are identified in the author table, making it possible to have a contributor's line to any SisBrAV ID. The authors are used to track the external text authors, as well as the credited researcher of each vulnerability. In the Products section, the object correlation table performs a link between the PK of the vuln table and a key named Object Key (OK).

As a result, it is possible for other tables to link to the Products tables without using a PK. The object table binds vendor, base, version and vulnerability together, storing product information.

The name object might seem sort of vague, but it means the object that the vulnerability exists within. The object_base table contains product names. For example, Windows, Exchange, Apache, and MySQL are all examples of product names. The object_vendor table contains the vendor names. For example, Microsoft, Sun Microsystems, and Apache Software Foundation are all examples of vendor names. The object_version table contains the version names. For example, 1.0, 2.0, 0.1, XP, 2000, or 95 are all examples of version names. Another crucial table is the score table, used to bind a scoring weight to a vulnerability. It is intended to allow every vulnerability in the database to be associated with one scoring weight. The score_weight table is used to store any type of scoring information needed for scoring calculations. Finally, the credit table adds support for identifying credit for discovering a vulnerability. Instead of storing author like information, a reference to the author table is made, as the data is extremely similar.

The second part of the CDB is the Client Database, responsible for storing all client-related data, involving personal/enterprise identification information, contact emails, products (systems and services) running, etc. Its structure is shown in figure 3.

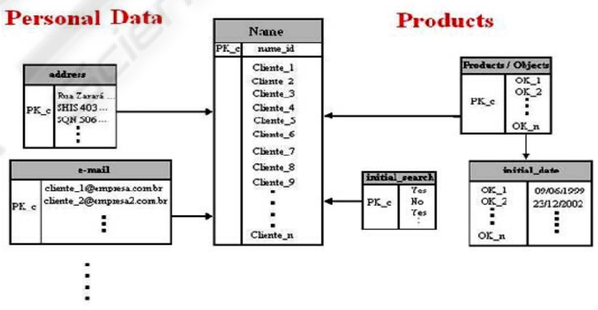

Fig. 3. Client Database Schema

In the above schema, the Products section refers to the products that each client registers, in order to receive only vulnerability alerts related to the systems running in his/her environment. The initial date to look in for vulnerabilities is also stored for each client. Specific product characteristics data is kept in the Vulnerability Database, as it was shown in the previously. All data related to a client himself is found in the Personal Data section. Contact emails, telephone numbers, addresses and personal/enterprise information, as well as the clients' passwords are entered in this part of the database.

The Name table consists in the main table of the Client Database, containing each client's account ID. All the clients are bound to their products through the Products/Objects table. The initial_date table stores the initial date to search for vulnerabilities, for each product a client registers in the database, while the Initial_search table contains entries that specify if a client is a new registered client in the system (represented by a "Yes" entry) or not ("No"). These entries are used by E-Note, to define if an initial search must be executed or not. At last, the Personal Data tables, such as address, e-mail and others, store client specific information, such as email, telephone numbers, address, personal/enterprise information and login username and password.

The last subpart of the CDB is the English-Portuguese Translation Database, which is still being designed. It will contain a large number of keywords in English and in Portuguese, in addition to semantics and syntax rules, making it possible for the E-Note module to translate the main description of a vulnerability entry to compose a mainly Portuguese email alert.

3.4 Email Notifier (E-Note)

This program will look for updated vulnerability information in the database. After retrieving the information, the program checks, for each registered client, if there are any new/updated vulnerabilities which affect the client's environment. If so, an email message – in Portuguese – is formatted, to inform the client about the new vulnerabilities discovered in his systems and services.

This message consists in a brief explanation of the vulnerability, in Portuguese, and one or more links for further information on that issue.

When a client registers in the SisBrAV system, he will have to inform what systems he has and what programs he runs, thus defining the scope of vulnerabilities SisBrAV should be concerned with, when generating alerts to that specific client. Besides that, the client also defines the start date, determining the initial point from which the system should begin the search in the vulnerability database. With that data in hands, E-Note will search in the database only the information that is really necessary for that client, generating a customized email message to him.

The E-Note module will also be written in Java, to guarantee its portability. E-Note is divided in two programs: one program performs the search in the database and the other sends the email alert.

For each new client added in the system, all the data about his systems and services is stored in the clients table, in the SisBrAV database, and a flag is set for this client, with a logical value that represents "NEW". The start date from which he wants to be informed about existing vulnerabilities is also stored in the database clients table.

Every time E-Note is run, it checks if there are any new clients in order to search for all the vulnerability entries that occur specifically in their systems and are newer than the start date defined by the client. It then generates the email alert to those clients, notifying about all vulnerabilities found. Afterwards, the “NEW” flag in the clients’ entry in the database is set to a value that stands for “OLD”.

For existing clients, the E-Note will simply check if there are new/updated vulnerabilities regarding their systems/services. If so, it generates the email alert for the specific clients whose systems are affected.

Due to the fact that the vulnerability information stored in the database is mainly in English, the vulnerabilities selected by E-Note are also in English. To make it possible for E-Note to generate Portuguese messages, an English-Portuguese translation database will bind English keywords to previously defined Portuguese sentences. E-Note performs, thus, a simple translation in the main vulnerability description. The main aspects – remote/local, high/low importance, etc – of the vulnerability are also translated. For example, if the main description of a vulnerability is “HP-UX DCE Remote Denial of Service Vulnerability”, and its importance is critical, the Portuguese message would be “HP-UX DCE: Vulnerabilidade Remota de Negação de Serviço. Importância: Crítica”. The translation database is in the format described in the previous section.

Along with the main description of the vulnerability, the email also contains links to the sites where that vulnerability is described and discussed.

3.5 Vulnerability Web Server

The idea of SisBrAV is not only to inform its users, emailing them alerts about vulnerability issues. The registered clients will also be able to perform a custom search in the Vulnerability Database through the web. With that functionality in mind, the fifth module of SisBrAV will be a Web Server that will handle these web requests.

The users will access an authentication site, where they provide their username and password (which are created and informed to him/her during the registering process). If successfully authenticated, they will be redirected to a customized database search page.

The site interface is being designed to be friendly and simple, although its security will be fundamental. The web site will probably be based in PHP, due to the fact that this language is very portable, and through its use, the database access can be implemented in a secure and simple manner. The web server chosen for the SisBrAV system was Apache, mainly because it is a multi-platform server, and also because fully supports the web publishing technology which will probably be used (PHP).

There are also other technologies which utilization is currently in discussion, such as Java servlets or JSP, because through using it would be easier to integrate the VWS module to the other modules in SisBrAV. XML is also in discussion, since it is another efficient way of implementing the database access from web. If JSP ends up being implemented, Tomcat (which is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies, fully integrated with Apache) will also be used.

4 Conclusions

In the current scenario, it is really important for anyone connected to the World Wide Web to protect his/her systems and data against the threats that continually arise. Besides having a nice antivirus tool, a firewall efficiently configured and other security technologies implemented in their network, users and enterprises must keep all of their Operating Systems, services and other software up-to-date, by applying all their latest patches and fixes. With that in mind, it's of great importance that systems and network administrators be informed quickly about any vulnerability that may be encountered in their systems, so that they can act proactively to build up defense countermeasures to guarantee the security of their environment.

SisBrAV will be an important security innovation, since it implements an idea of an automatic vulnerability searching and alerting mechanism, with very little human administration needed. Since it will have many trustable security sites as sources where it will look for vulnerabilities information, SisBrAV will be a very reliable system, extending the horizons of systems and network security. In addition to that features, it is also important to remember SisBrAV, being a Brazilian project, will implement a translation feature in order to produce Portuguese email alerts, so that Brazilian clients will feel comfortable with it. In the future, the language support can be expanded to other idioms.

Nowadays, where free software gradually gains space in the software business, a program must support many platforms, so that it can be installed in a variety of systems and interact with different technologies without incurring into stability loss or performance troubles. SisBrAV is being designed using only free software products and platform independent languages, resulting in a solution with great portability and scalability.

References

1. Deitel, H. M. – *Java, Como Programar* / H. M. Deitel e P. J. Deitel; trad. Carlos Arthur. Lang Lisboa. – 4.ed. – Porto Alegre: Bookman, 2003.
2. *SQL Tutorial*. Available from: http://www.w3schools.com/sql.
3. *PHP/MySQL* Tutorial. Available from: http://www.freewebmasterhelp.com/tutorials/phpmy
4. *Portal Java Home Page*. Available from: http://www.portaljava.com/home/index.php.
5. *Open Source Vulnerability Database*. Available from: http://www.osvdb.org.
6. *http://Dig Project Home Page*. Available from: http://www.htdig.org.
7. *Internet Security Systems X-force Home Page*. Available from: http://xforce.iss.net.
8. *Cert Knowledge Base*. Available from: http://www.cert.org/kb.
9. *SANS Newsletters*. Available from: http://www.sans.org/newsletters.
10. *Security Focus Home Page*. Available from: http://www.securityfocus.com.