A Web Interface for Nessus Network Security Scanner

Chuming Chen Manton M. Matthews
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208, USA

Abstract

A fully functional web interface (NessusWeb) for the Nessus network security scanner has been developed. NessusWeb provides public accessibility for authorized users and supports SSL communication, multiple sessions and centralized scan configurations and management of scan reports. It was created using a multi-tier distributed architecture. The client tier is a web browser. The Apache Secure Web Server including Apache HTTP server and Tomcat serves as the web tier. As a servlet engine, Tomcat is used to generate dynamic web content and monitor the Nessusd server through the SSL channel using the NTP (Nessus Transport Protocol). The Nessusd server forms the business tier, which performs the actual network security scans. User scan configurations and network vulnerability scan results generated by the Nessusd server are saved into a MySQL database, which provides the back-end storage. Two user interfaces were implemented, one for the administrator and one for all other users. This paper presents an overview of the design and implementation of the NessusWeb project.

Keywords

NessusWeb, Nessus, Network Security Scanner, Apache

1. Introduction

A network security scanner is a software tool that can remotely audit a given network and determine whether "attacker" has broken into it or whether there are vulnerabilities in the network. As a network administrator, you can use this tool to check the vulnerabilities of your network and patch them before somebody attacks you. Compared to other commercial network security scanners (Internet Security Scanner by ISS, STAT Analyzer by Harris, and Retina by eEye etc.) [1], Nessus is a free and open source network security scanner [2] for any POSIX systems. As a security-auditing tool, it is implemented in a "client – server" architecture. The server (called Nessusd) is in charge of scanning and finding the vulnerabilities of a given network. It has a vulnerability database, which is a set of vulnerability checks it can perform. The client is a front-end that is used to monitor and configure the server. It also serves as a result repository and report generation tool. SSL (Secure Socket Layer) [10], the industry standard for security, guarantees the security of the communication between the client and server. As of today, there is only one version of the Nessusd server that runs on any POSIX system, and there are multiple clients. One called Nessus, which has a command-line version and a GUI version that works with GTK [2]. Another has been written for Win32 (named NessusWX) [3]. For most of these client tools, the user interface for displaying the scan report can only be used on the machine with the client tool installed. Thus accessibility is limited compared to NessusWeb, which is available on the Internet. Those client tools can generate scan reports in different formats (for example, ASCII, html, PDF, XML, etc.), but the data in these reports are not conveniently searchable, there is no way to search for vulnerability about a particular port, or to show what type of vulnerability check has been conducted to generate this report. As an administrator, you may wish to not only check on the health of your machines, but also have centralized management of the scan settings and scan reports, so that you can do network security analysis. Furthermore, there is no user management

and network security analysis functionalities integrated into these tools.

Using a web interface is one of the solutions to provide convenience and public accessibility for allowing authorized users to configure and monitor network security scans. There are some existing web interfaces [4, 5] that provide certain public accessibility, but they are not real clients of the Nessusd server. They are constructed merely by wrapping a web interface around the Nessus command line client, a C program for POSIX systems. In these web interfaces, user inputs from the html forms were used as the arguments passed to the Nessus command line client. Since this command-line tool can only take limited arguments, these web interfaces are not fully functional client tools. There is no ways for a user to do complicated scan configurations, i.e. select plugins, set up plugin preferences, set scan options and knowledge base options etc. There is also no direct communication between those web clients and the Nessusd server.

The drawbacks of the existing Nessus client tools provide good justification for developing a new interface for Nessus Security Scanner. The objective of this project is to create a new fully functional web interface, which is publicly accessible for authorized user and supports SSL communication, multiple sessions, and centralized management of scan settings and scan reports. The authors designed and implemented multi-tier distributed application architecture [16]. In this architecture, the web browser is the client tier, the Apache Secure Web Server [6] serves as the web tier, the Nessusd [2] server is the business tier, and the MySQL[7] database server serves as a back-end tier. The project was implemented using Serlvet [8], JSP (JavaServer Page) [9], JSSE (Java Secure Socket Extension) [10], and JDBC (Java Database Connectivity) [11] technologies.

2. System Design

The existing implementation of the Nessus network security scanner used "client–server" architecture. In this architecture, the Nessusd server carries out most of the business logic (scanning network and generating scan results). The Nessus client is basically used to configure and control the scanning processes. It is also in charge of receiving scan results sent by the server, formatting them into a proper format and saving them to local storage. Putting data management on the client side inevitably degrades its performance and scalability with the increase of the number of scanning processes. Adding a database management system, which serves as the back-end storage and handles all the data management, will dramatically increase the system performance, efficiency and scalability.

In this project, the authors developed a multi–tier distributed system. Figure 1 shows this multi-tier architecture divided into different tiers described in the following list:

- Client tier — Web Browser
- Web tier — Apache Secure Web Server
- Business tier — Nessusd Server
- Backend tier — MySQL Database Server

Although the NessusWeb consists of four tiers as shown in Figure 1, we can still consider it to be three-tier application. Because they are distributed over three different locations: the client machine, the NessusWeb machine, and the database server machine at the back end. This kind of three-tiered application extends the standard client and server model by placing a multi-threaded application server between the client application and back-end storage [12].

2.1 Web Browser

The client tier of the NessusWeb multi-tier distributed application is basically a web browser, which allows authorized users to logon to the NessusWeb system, set up their scan configurations, send scan configurations and scan requests to the Apache Secure Web Server. Once a scan is done, the user can view the scan report, scan history and scan configurations generated from the Apache Secure Web Server. The user can also reuse the settings from the previous scan sessions. As a super user, the administrator can not only conduct all the operations allowed for normal users but also manage user accounts (which includes creating user accounts, setting user permissions, and generating user certificates for the Nessusd server), and set up the Nessusd server and mail server, which will be used for user registration and user scan report notification etc. The Apache Secure Web Server running in the Web tier generates those dynamic web pages. Users use the web browser to render the pages received from the web server. In this case, the web client is

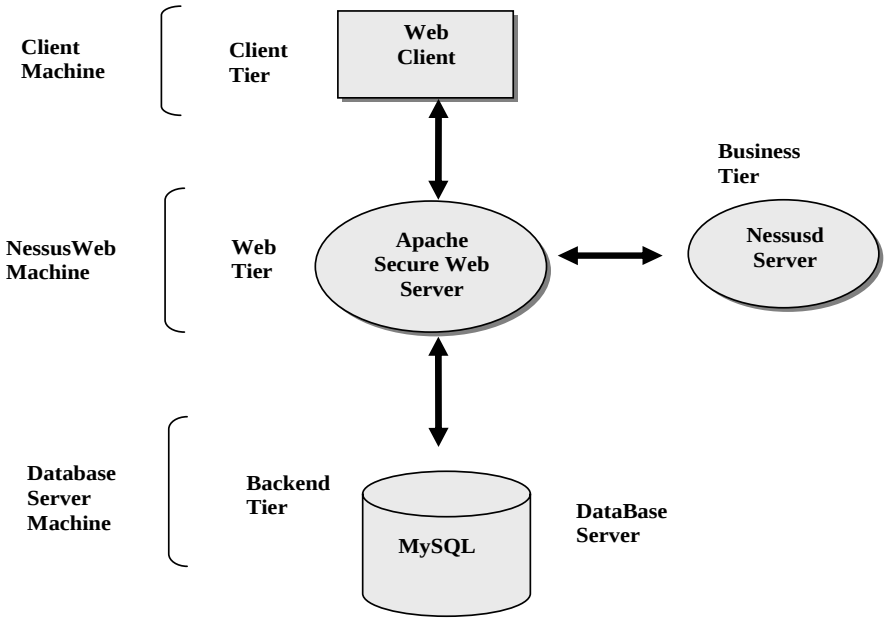

Figure 1 NessusWeb multi-tier distributed architecture

actually a thin client, which usually does not query databases or communicate with the Nessusd server. Those heavyweight operations are off-loaded to Java servlets running on the Apache Secure Web Server.

2.2 Web Server

The Apache Secure Web Server is the actual client to the Nessusd Server (in charge of scanning the networks). In addition to handle requests from the web client, it also has the ability to communicate with the Nessusd server to configure and monitor the network scan processes. Furthermore, it can save all the scan settings and scan results into the database. If the user wants to view the scan history or a particular scan report, the web server can retrieve the information from the database and send it back to the user web browser. Because scanning the network is very time-consuming, a lot of network traffic needs to be generated during this process, it really doesn't make sense to keep user hanging around for a long time. The user will be detached from the scan process when he finishes configuring a scan and sending the scan request. An email with a scan report attached will be sent out to notify the user once the scan finishes. The scan results are saved into the database, so the user can come back and view the results or query against the database to do network security analysis.

2.3 Nessusd Server

As the business tier of the NessusWeb architecture, the Nessusd server conducts the actual network scan operations. The Nessusd server consists of a vulnerability database (a collection of plugins), knowledge base of the current active scan and the scan engine. The network scans performed by the Nessusd server are coded as external modules (plugins) written in either C or NASL (Nessus Attack Scripting Language) [2]. Since the Nessusd is used to conduct network scans, it should be configured correctly, and only the authorized user can use it, this is maintained by a user database associated with the Nessusd server. In the NessusWeb architecture, all communication between the Apache Secure Web Server and the Nessusd Server is encrypted. The Nessusd server uses OpenSSL (C implementation of SSL and TLS), the JSPs or Servlets running in the Apache Secure

Web Server uses JSSE (Java implementation of SSL and TLS).

2.4 Database Server

As a back-end tier, the database processes and saves user scan settings that are sent to the Apache Secure Web Server. When the Apache Secure Web Server starts communicating with the Nessusd Server, it will retrieve configuration information from the database, and use it to construct NTP (Nessus Transfer Protocol) [2] messages and pass them to the Nessusd server to monitor and control the scan process. It also saves the scan results passed back by the Nessusd server into the database. The users can retrieve all their previous scan settings and scan reports by sending requests to the Apache Secure Web Server. The web server will retrieve the relevant data from the database, dynamically generate web pages, and send back to the user web browser. By operating this fashion, we have centralized control and management of scan configurations and scan reports.

The MySQL database server is used to serve as the back-end storage tier in this project. The Servlets running in Tomcat handles all these database access operations using JDBC. Opening and closing the database connection is the most frequent and most expensive operation in this project. Unlike the standard J2EE [12] applications, which have an automatic connection pooling mechanism, the pure JSPs we are using here can't manage the connection themselves. To support connection pooling in this project, we have used the Turbine application framework [13]. It has all the components that developers need to create secure web application including database connection pooling.

3. System Implementation

Several different technologies including Java Servlets, JSP, Java Secure Socket Extension (JSSE), Java Database Connectivity (JDBC), Turbine etc. were used to implement portions of the NessusWeb project (Figure 2).

3.1 Secure Communication

The current version of the Nessusd server is designed to only allow access from the authorized

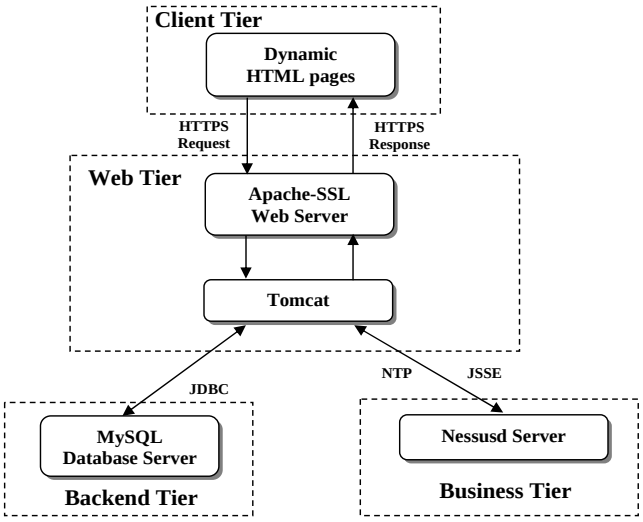

Figure 2 Implementation technologies used in the NessusWeb project

users. Users can either use a combination of login-name and password or a certificate to access the server. When the administrator initially installs the Nessusd server, by default Secure Socket Layer (SSL) and Transport Layer Security (TLS) [14] are installed. SSL and TLS are industry standards for secure communications between clients and servers. It is possible for the administrator to change this installation procedure and not install SSL and TLS. But if this is done NessusWeb will not be able to communicate with the Nessesd server in a secure manner. Because the NessusWeb is a web-interface, with its URL available for public access, its own security has been the highest priority for the entire project. Putting the NessusWeb interface pages under the Apache Secure Web Server provides the first layer of protection to guarantee the security of communication between the NessusWeb Client (web browser) and the web server.

The second layer of protection is the secure communication between the JSPs or Servlets running in the Tomcat JSP container of the Apache Secure Web Server and the Nessusd server. The Nessusd server is implemented in C and utilizes OpenSSL. The NessusWeb is implemented in Java and uses the Java Secure Socket Extension (JSSE), which implements a Java version of SSL (Secure Sockets Layer) v3 and TLS (Transport Layer Security) v1 protocols and includes functionalities

for data encryption, server authentication, message integrity [10].

The "nessus-mkcert" script is used to generate server RSA private key, CA (Certificate Authority) RSA private key, server Certificate and CA certificate during the installation of the Nessus software packages. A "nessus- mkcert- client" script is used to generate keys and certificate for clients implemented using OpenSSL. In the NessusWeb project, the client is comprised of a collection of Java classes running in the Tomcat server, which is a pure Java implementation, so the authors used the JDK 1.4 keytool and the methods suggested by Angell [15] to generate keys and certificates for each new user. These client keys and certificates are used to initialize the information exchange between the client and the server as part of the SSL handshake [10].

3.2 Access Control

In the NessusWeb system, there are basically two types of users, the administrator and other users, which will refer to as "normal users". Normal user can perform a network scans, view scan history and perform network analysis. Administrators have all the capabilities that normal users have and in addition, the administrator is also in charge of setting up the whole system and managing user accounts. The user interfaces and functionalities exposed to the normal user is thus a subset of those available to the administrator. In fact, both user interfaces are implemented using the same JSP pages. When a user signs on the system, the system will know the user's access privileges and the JSP pages will dynamically generate appropriate content to present to the different categories of users. The access control was implemented using a session variable "login", which is set after a user successfully logs onto the system. For each of the succeeding page requests, this session variable is checked to determine what user interface and functionality should be presented to the login user. If the user tries to bypass the login process or refresh the page after the session timeout, the user will be redirected back to the login page. This process guarantees that only authorized users can access the appropriate pages.

3.3 User Management

The Nessusd server keeps a user database used for authentication and access control when users try to connect to the Nessusd server using client tools. From the Linux or Unix console, an administrator can run "nessus-adduser" command to add a user to this user database. The NessusWeb project completely automates this process. The administrator can add a new user to the system by entering all the required user registration information through user management web pages. The administrator can also remove a user from the system or reset a user's password.

3.4 Automatic Plugin Update

The Nessus project web site maintains an up-to-date database of vulnerability checks (plugins), but the Nessusd server can only use the local copy of this vulnerability database. On the system set up page of NessusWeb, which includes setting up the Nessusd server host and port, the Mail server host and port, the administrator can check a checkbox named "Update Plugins" to update the local copy of this vulnerability database. The "nessus-update-plugins" script is wrapped by a Java class to retrieve the newest plugins from the Nessus project web site and update the PLUGIN table in the NessusWeb database. The administrator is encouraged to regularly go to this system set up page and keep updating the plugins for the whole system.

3.5 Network Scan User Interface

Both the Nessus X-window client and the NessusWX windows client have a very nice user interface for setting up scan preference options, including the plugins, the targets and the knowledge base used in each scan session. In the NessusWeb project, JavaScript is used to create the network scan user interface with seven tabbed pages: Plugins, Preferences, Scan Options, Knowledge Base, Target Selection and Start the Scan in an HTML inner frame.

For each scan session, the user can select the plugins he wants to use, set preference values for those plugins that need scan preferences, set the scan options, setup the knowledge base and enter the names of the targets. The changes made by the user on each page will be saved in the session variables before the scan starts, so the user can go back and

forth to change them. Once the user clicks the "Start the Scan" tab, the secure connection and communication with the Nessusd server will be established. The scan setting information is saved in the database, formatted into the corresponding NTP (Nessus Transfer Protocol) messages and sent to the Nessusd server. A new thread running as daemon is initiated after the above steps. Since the scan thread is now running in the background, the "Start the Scan" page will return immediately, and the user can start a new scan session without waiting for the results of this perhaps time-consuming scan. Because some lengthy vulnerability checks with multiple machines and multiple plugins could take several hours to complete. The scan thread daemon communicates with the Nessusd server to control and monitor the scan process. The information on the vulnerabilities found by the Nessud server and their corresponding estimate of the severity of the vulnerability will be sent back to the thread that initiated the scan. This thread saves this information into the NessusWeb database SCAN_REPORT table for each scan session. The Nessusd server informs the scan thread whenever the scan process terminates, then the scan thread will retrieve the scan results from the database, format a HTML network scan report and send it as an email attachment to inform the user that the scan is completed. By running the scan process as a daemon, the user doesn't need to hanging around for the completion of a long scan process, which improves the system performance and user satisfaction.

3.6 Automatic Email Service

One basic functionality of web sites is the automatic email service. This service is widely used in the NessusWeb project. Whenever a new user is created, updated, removed, or a user password is changed, the corresponding user registration information is emailed to the user as a reminder/security notification. When a scan terminates, an email is sent out to notify the owner of this particular scan session with the HTML scan report attached. All the email functionality was implemented using the javax.mail and sun.net.smtp.SmtpClient packages.

3.7 Online Assistance

If you are the first time user of the Nessus tools, there are many new terms, buttons and checkboxes all over the place that might lead to confusion. To address this complexity it is definitely a necessity for the system to provide a good online assistance. In the NessusWeb project, JavaScript has been used to create "ToolTips" for the links, buttons, textboxes, checkboxes and radio buttons. The system also utilizes JavaScript to implement pop up windows showing scan reports and scan settings. This online assistance makes NessusWeb web interface dynamic and user friendly.

4. Conclusions and Future work

We have created NessusWeb, a fully functional web interface for the Nessus network security scanner using a multi-tier distributed architecture. NessusWeb is publicly accessible for authorized users and supports secure communication, multiple sessions and centralized management of scan configurations and scan reports. Saving scan configuration and scan results into the database makes it easy and possible to do sophisticated network security analysis. The project was implemented using Java Servlets, JSPs, Java Beans, JSSE and JDBC. The details of this project and the page for downloading the software package can be found at http://www.cse.sc.edu/~chen7NessusWeb/.

The current version of the NessusWeb only supports HTML version of scan report and scan setting report. One of the areas for future work is to provide the user options to export scan report into different formats, for example, HTML, XML, PDF, NSR (used by X-window Nessus client) etc. Another area of future work is to increase the support for higher-level security analysis

References

[1] Astalavista, "Network Security Scanner", see http://www.astalavista.com/tools/auditing/network/securityscanner/

[2] Deraison, R., "Nessus Project", see http://www.nessus.org/

[3] Victor, "NessusWX - Nessus Client for Win32", see http://www.securityprojects.org/nessuswx/

[4] Inprotect.com, "Inprotect Web Interface for Nessus", see http://www.inprotect.com/

[5] Karas, Kristofer T., “Nessus Network Vulnerability Assessment Scanner”, see http://enterprise.bidmc.harvard.edu/pub/nessus-php/

[6] Apache Software Fondation, “Apache HTTP server project”, see http://httpd.apache.org/

[7] MySQL, “MySQL Reference Manual for version 4.0.3”,see http://www.mysql.com/documentation/mysql/bychapter/

[8] Sun Microsystems, “Java Servlet Technology”, see http://java.sun.com/products/servlet/

[9] Sun Microsystems, “JavaServer Pages White Paper”, see http://java.sun.com/products/jsp/whitepaper.html

[10] Sun Microsystems, Java™ Secure Socket Extension (JSSE) 1.0.3, API User's Guide, see http://java.sun.com/products/jsse/doc/guide/API_users_guide.html#SSLOverview

[11] Sun Microsystems, “JDBC™ Data Access API”, see http://java.sun.com/products/jdbc/

[12] Sun Microsystems, “The J2EE tutorial”, see http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html

[13] Apache Software Foundation, “Apache Turbine Project”, see http://jakarta.apache.org/turbine/turbine-22/

[14] OpenSSL, “OpenSSL Project”, see http://www.openssl.org/

[15] Angell, K. W., “Java Programming features the Java Secure Socket Extensions”, see http://www.ddj.com/articles/2001/0102/

[16] Chuming Chen, Web Interface for Nessus Network Security Scanner, Department of Computer Science and Engineering, University of South Carolina, M.S.Thesis, May, 2003