Building a RADIUS Protocol Based Authentication Authorization and Accounting System

Sergey Hruschev, Svetlana Cherenkova
Omsk Branch of Sobolev's Institute of Mathematics
Siberian Branch of Russian Academy of Sciences
hrushev@okno.ru, scher@omsk.net.ru

Nowadays network communications protecting becomes more and more important. This problem is complex one and consists of a lot of specific tasks. One of these tasks is to develop an authentication authorization and accounting software (for example, to manage connections to network access servers). Since non-authorized access to network resources is often a first step to various destructive actions, developing of a high quality system of remote network connections identification looks very important.

Now there are a lot of various network identification systems based on RADIUS protocol, i.e. Lucent RADIUS (formerly Livingston RADIUS), Cistron RADIUS, YARD RADIUS, KISS RADIUS, FreeRADIUS, Merit, RAM (RADIUS authentication manager), XtRadius, Ascend Radius and some others. The main feature of the systems is their orientation mainly to work under various versions of Unix OS. Their administration is usually done by Unix-like methods, i.e. by using some text configuration files. Using of this software, however, assumes a deep integration with network resources accounting systems and authorization systems in corporate networks. Since base versions of RADIUS daemons are seldom oriented on such integration, their use usually assumes installation of so named "patches". As a rule, their work is to replace standard user configuration reading and statistic saving modules. Some examples of such modifications can be found, for instance, on the Russian-language server OpenNet (http://www.opennet.ru/prog/sml/53.shtml). Such revision, naturally, does not provide increasing of reliability of the programs executing because of author's documentation on source code in most cases is absent, and modification is done by "trial-and-error" method. An additional problem for Windows oriented users is recompiling of original source code (in most cases written on language C with UNIX system libraries calls). Similar difficulties can appear in source code compatibility between different UNIX versions, for example, when a program uses specific features of a certain system.

Storing of user passwords encrypted by the DES algorithm in text files is a serious safety problem of existing systems. Due to a well-known fact of insufficient standard encryption protection of the DES algorithm, it is desirable to change the format of storing the passwords. However, this old encryption method is applied in almost every RADIUS daemon.

Most of existing problems could be solved by developing a RADIUS server, built accordingly to the base of modern programming standards providing porting to different operating systems and allowing to connect external authorization and statistic modules.

Exactly such decision was implemented in our organization. Java was chosen as the programming language because it satisfied our requirements better than others. First of all, it provides portability between several operating systems (a native support of JRE is present in Solaris, Linux and Windows, JRE is also ported on FreeBSD). Besides, Java supports various modern network features. An additional advantage of this language is a database access support. Despite of these advantages the solution has several disadvantages, for example, a bit high requirements to the RAM size of the server. However, nowadays a program using ten MB of extra RAM already does not seem overweening. Since the NAS request processing time is a critical value, making a decision about choosing Java, we examined a speed of Java-based applications. The decision was made on the base of the studying of the JSP mechanism functioning, which, being written on Java is essentially an interpreter of a certain language. By practical consideration, we discovered that an essential fall of performance occurs only at the moment of a servlet code compilation. While interpreting of a page code occurs without any visible delay. Since RADIUS server does not require any high computing resources, we decided that such results were satisfactory in our case as was subsequently confirmed by RADIUS server testing.

We created a test version of the RADIUS server which consists of authorization and statistic modules, supporting information exchange with Oracle RDBMS. These modules are written with help of the JDBC technology. This approach provided the possibility of users account records keeping in database tables, so that a potential intruder can get password information harder than from text files because of necessity to pass the Oracle RDBMS authorization procedure. By using the Java Cryptography Extensions technology it is possible to encrypt the passwords through a lot of modern efficient cryptooperation methods, for instance Blowfish.

The RADIUS server designed is a multi-threaded application. The server has the following structure:

- a main thread;
- an authorization support thread;
- a statistic support thread.

The main thread is used for starting, operating, and stopping the server. Two auxiliary threads are created and started inside main one. When the server is in the working mode, the main thread is used for operating of the server. By now a simple server control with the command transmission through sockets has been realized. Both commands and server answers are text lines, so that server operating can be done either by starting its second copy in client mode, or by using TELNET client.

As follows from the context, the authorization support thread provides the NAS clients authorization. This thread gets client authorization requests and sends them to an authorization manager, which after processing request returns one of the following return packets: Accept, Challenge or Reject. Thereby, the RADIUS server itself is separated from the authorization mechanism.

Similarly, the statistic support thread receives statistic packets and dispatches response packets to NAS. In this case requests are also processed by a specialized statistic manager.

Due to separation of the processes of information processing and transmitting the server kernel becomes universal enough. This approach saves from the server kernel code changes while changing authorization or statistic processing mechanisms. Having tested and revised functions of all parts of the system, we are going to develop a standard interface for the kernel and external modules interaction.

Installing and administrating of the RADIUS server are practically independent from any operating system. It is enough simply to copy JAR-archives to a necessary directory and setup the NAS connection and database server connection parameters. All adjustments related to user connection parameters depend only on authorization and statistics modules used. In our case, external modules use information stored on the Oracle database server; a set of corresponding tables was created for that. For each user a set of RADIUS attributes is defined. They are packed in a packet and sent to the RADIUS server kernel.

The safety of the network connections management system in this case is provided by the quality of passwords storing mechanism and by the protection quality of the channel by which information is transmitted between the RADIUS server and NAS.

The safety of RADIUS protocol communications is provided by several components:

1. an installation on NAS and the RADIUS server a private key of a sufficient length;
2. a generation on NAS package-authenticators by using a "good" random number generator as well as absence of repetitions of values generated earlier;
3. the durability of the MD5 hash algorithm [4].

Authenticator generation depends completely on NAS used. A "good" random numbers generator is understood here as a one providing equidistributed values. So, an equiprobable (and unpredictable) appearance of package authenticators is provided. The absence of the repetitions makes impossible a Response-package forgery by an intruder by using the Response-authenticator once more. Without knowing the authenticator generation algorithm used by NAS it is hard to say that the algorithm is "good" or "bad". However, due to the number of possible certificates is equal to 2^128, it is possible to expect that even simple generating of the certificates giving equiprobable results will provide an acceptable quality.

Private key entering is probably the most important from the three things, since such keys choice is made by a NAS administrator. Installing an empty key weakens the software protection essentially, so that some NAS simply does not allow executing the requests on the RADIUS protocol in this case. However, some NAS continue to work with an empty key, too. One example of such NAS is Cisco 2511 with different IOS versions.

During several last months we repeatedly encountered in the Internet messages concerning finding the vulnerability in the RADIUS protocol protection mechanism. At present it is difficult to say anything about validity of this information because it is unofficial. Information provided in these messages, however, is indicative of presence of some methods allowing to analyze MD5 hash-values and to discover the private key by using the package authenticator. Authors of these messages offered to use private keys with over than 16 or even over than 32 symbols length to resist such methods.

In general case, to attack the system which uses the RADIUS protocol, an intruder must have:

1. a possibility of interception of UDP packets transmitted from NAS to the RADIUS server and back;
2. a possibility of substitution of UDP packets transmitted from RADIUS server to NAS;
3. a knowledge of the private key or an efficient algorithm of its reconstruction.

Impossibility to perform these actions will, in most cases, provide a safe network connections management.

We plan to improve the RADIUS server in several ways. First of all, it is an integration with a billing system. The second one is an IPv6 support.

References:

1. Rigney, C., Willens, S., Rubens, A and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
2. Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
3. Aboba, B., Zorn, G and D. Mitton, "RADIUS and IPv6", RFC 2869, August 2001.
4. Rivest, R and S. Dusse, "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.