Aspects Concerning the Optimization of Authentication Process for Distributed Applications

▪

Ion Ivan
Ph.D. Professor
Mihai Doinea
Assistant
Dragoș Pălăghiță
Student
Academy of Economic Studies, Bucharest

Abstract. *There will be presented the types of distributed applications. The quality characteristics for distributed applications will be analyzed. There will be established the ways to assign access rights. The authentication category will be analyzed. We will propose an algorithm for the optimization of authentication process.*

For the application "Evaluation of TIC projects" the algorithm proposed will be applied.

Key words: optimization; security; authentication; distributed application.

▪

JEL Codes: C10, O31.
REL Codes: 10B.

1. Types of distributed applications

In the dedicated literature there are described types of user's authentication for distributed applications, the characteristics of biometric, mutual, multifactor, using tokens, username and password, using certificate, CHAP and Kerberos authentication being specify.

Authentication using username and password, called one way authentication, consists in filling fields with the credentials associated with the client and validating those data entered in the form with:

- the username and password which are found clearly in the source code;
- the username and password found on the server, information being sent by means of methods called GET/POST to the server where the validating process will authenticate or not the credentials introduced by the user comparing them with the information stored in databases;
- information stored in databases, using cryptography algorithms for a high level of security as well as the information sent via network.

This method of authentication with username and password it's one of the first methods that validates the identity of a user and one of the most unsecure as well if this process isn't administered well enough.

Biometric authentication is the method of authentication which implements hardware devices and processes that can determine the physiological characteristics of a person called user. Some of the most used methods of authentication by means of measuring biometric data are:

- fingerprint methods;
- iris scanning methods;
- hand geometry;
- face geometry;
- voice recognition;
- keystrokes.

Beside the ones presented above, it is worth to mention: facial thermogram, AND, walk style and many others who can identify uniquely a person. They are much harder to implement and to expensive.

Mutual authentication is presented as using methods through which the both parts, the user and the authenticator, will be validated. The process has the fallowing principle: the server will try to validate the identity of any user that request a connection and after that the user will validate the server identity based on a digital certificate.

Multifactor authentication is the method in which the authentication, the process of validating the identity of the requestor, is very complex. This is due to the fact that there will be used multiple authentication methods, for that the multifactor authentication it is known as strong authentication. Usually the methods used in this type of authentication are complementary, meaning that one of the methods is used just in case the other fails, avoiding to not letting wrong users to get access to the system.

CHAP authentication defined in RFC 1994 standard is based on an authentication protocol used for validating the identity of users who are connected from distance to the servers using secret information that is not passed through a network channel. The authentication is based on a process with three steps:

- after the physical connection was set between the hardware components and the CHAP authentication method was chosen, the one who make the authentication will transmit a message to client;
- the client will respond with a value calculated through a one way function, MD5;
- the response is validated by the one who made the authentication.

At that the CHAP algorithm is just a one way authentication, when this process is used by two parties the CHAP authentication becomes a mutual process for verify the user's identity.

Authentication using tokens. By the authentication process point of view, will have the following types of tokens:

- session ones;
- access tokens;
- tokens used for authentication process.

Session tokens are a value generated, usually through a hash function, without any interest, transmitted by the server to client for easily controlling his session of work. The client will save this value and any time he tries to access any of the server services will send it through browser cookies or by GET/POST means.

Access tokens are objects that include information about the user who accessed services and has passed an authentication process. This kind of tokens together with the service accessed by the user form his set of rights.

Token as a hardware is a component used in an authentication process who stores enciphered keys like digital signature or biometric data like fingerprints.

Kerberos authentication is a method that can be used as a one way authentication but also as a mutual process of validating the identity of users. The way that systems based on Kerberos make run the authentication process is based on using session keys and securing information changed between client and server with a cryptography method. Kerberos authentication process has the following steps:

- for using a service which resides on a server which has a Kerberos authentication system, client first must obtain a ticket which grant him access for it; for that he must send a request to the server, specifying the service which will be accessed, his identity as a public key and the timestamp from the moment of sending his request;
- the authentication system will respond to the previous request with a response enciphered in which client can find the ticket which will grant him access to the service; more, the response will include a timestamp for security reasons and a temporarily session key;
- the ticket is then used by the user for connecting to the services offered by a server in a secure way thanks to the various factors which converge to the same result, meaning a secure connection between two parties.

Every method of authentication is used for a certain level of security, disaccords are followed by important negative aspects for the user if for simple applications will be solicited a high level of security and if for systems where financial transactions are made, the level of security will be relatively low.

On the development phase of a distributed application the target group is analyzed because the quality characteristics of systems developed are correlated with the target group who access the resources made available by the application and who through the level of satisfaction of users will determine the next phase in development of distributed applications.

It is defined a collectivity C having c_1, c_2, ..., c_N as elements and a distributed application A formed by M_1, M_2, ...,M_K modules.

From problem to problem, from the user's requests and from owner's financial resources there are defined:

- the period of development the application;
- level of complexity;
- quality level;
- the variety of used resources;
- the degree of solving the problem;
- the communication with other applications.

If there are severe restrictions regarding the nature of the problem, solution structure and especially user behavior it is preferred to define a linear distributed application structure (figure 1).

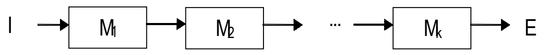

Figure 1. *Linear application structure*

The input data i is only made from compulsory fields and the ultimate output is structure enforced. When launching in execution, inline processing is made with modules M_1, M_2, ..., M_k and they will be stopped only if all the modules are correctly executed. The intermediate result R_i, output of module M_i is intermediate input data D_{i+1} for M_{i+1} module (figure 2).

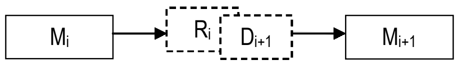

Figure 2. *Data transfer between the linear structure's modules*

The linear structure is used when the distributed informatics application makes one processing, has a very large coverage area, the number of entities in the data bases is also very large so there is a need for a guaranteed security level such that every final result fundaments a decision in terms of YES/NO or accepted/rejected, GOOD/BAD, accompanied by actions with immediate effects. Linearity is in regard with the connections between modules. Within a module all the control structures are used, without enforcing linearity inside the module as well.

When a problem P decomposes in sub-problems SP_1, SP_2, ..., SP_H, and during the process one of the sub-problem's corresponding processing is activated they are associated with a two level tree structure (figure 3).

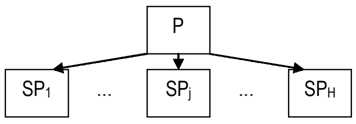

Figure 3. *The decomposition of problem P in sub-problems*

This approach has corresponding module access options belonging to user, which in turn correspond to a sub-problem and in the frame of the sub-problem to the modules required to solve particular problems. For the distributed application a tree structure is defined in which module M_0 is the main program and the other modules are placed on 1, 2, ..., r levels (figure 4).

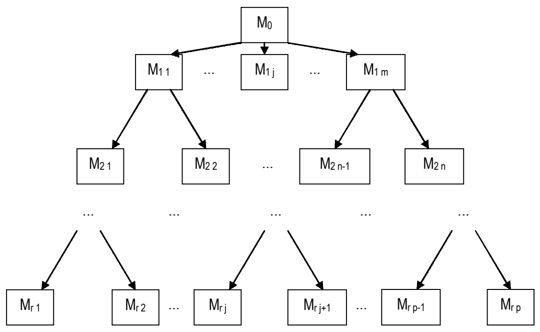

Figure 4. *Tree structure associated to the distributed application*

The modules on the last level, r, are terminal modules which close the successful processing and which are finalized with result display and saving the data.

Starting from the main module and finishing with one of the last modules systematically, a set of modules is referred, by user decision based selection. If the number of terminal modules is S, it results that the users have at their disposal an application with the help of which they select resources to solve S personal problems. The tree structure assumes the existence at each branch level of the calling module until one of the terminal distinct modules for the same processing and the distinct modules for printing the same results. This means that a very high level of redundancy is obtained because of code duplication.

The initial versions of un-optimized distributed applications have tree structures. This structure permits simultaneously the development of modules corresponding to sub-trees, partial testing of the application at sub-tree level. The foundation of the whole application is made by assembling sub-trees from level g, by calls from modules at level g-1. The existence of the tree structure creates favorable premises to define the problem which is to be solved with the data from each user, gradually.

If the tree structure is overtaking an optimization process which assumes that the modules with similar functions be referred at the level of a single module and the referring of modules on level g of modules on superior levels g-1, g-2, …, 1, a network type structure will be obtained (figure 5).

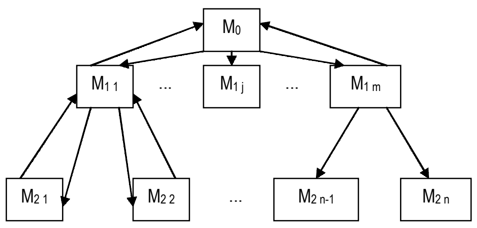

Figure 5. *Network structure for a distributed application*

The definition of a new network structure allows comebacks to modules placed in upper levels to make data completions, for making new selections to facilitate the user to obtain favorable versions.

The network type structure is compatible with the use of library type components provided that the referred modules from a set of locations that are taken from libraries have a high level of generality and the parameter list will be easy to manage.

If in a tree structure three terminal modules exist out of which one displays an elementary variable, the second one, elements of a vector, and the third one the elements of a matrix, when referring a single module to display in a network type structure, the module will have a high degree of generality and will be realized in one of the forms:

- the three modules are reunited and the selection is made in the framework of an alternative multiple structure for each of the three situations;
- the matrix display module is used if the number of columns and lines is 1, the elementary variable is displayed; if the number of columns is 1 and the number of lines is random or the number of columns is random and the number of lines is 1 an array is displayed, if both the number of columns and lines is random then a matrix is displayed.

The comebacks from inferior levels towards upper ones are used when the data validation processes require only complete and correct data. The limitation of the number of comebacks is obtained by parameterization, such that the enforced number of admitted operating errors is not exceeded.

The network type structure must be regarded as the result of the optimization process and not in a different fashion.

2. Access right attributes

For the online informatics application there are NT types of users. In each TU_i user type of type *i* a collectivity is formed having NU_i elements.

The total number of users NTU of the application is given by:

$$NTU = \sum_{i=1}^{NT} NU_i$$

When the specifications are elaborated, it is important to correctly define the user types, such that their job descriptions are orthogonal, meaning that there will not be two users that have interrelating jobs. In this context for each user type strict access is associated with a certain group of application resources. No other user of type TU_i will have access to the resources belonging to a user of type TU_j, $i \neq j$, $i,j = 1,2,3, \ldots, NT$.

In this type of conception, the access rights matrix is a square matrix in which:

- alongside the lines there are NT users;
- the columns hold NT processing functions.

It is concluded that the user of type TU_i has access to the processing functions grouped under FP_i.

The MDA access matrix is:

TU/ FP	FP_1	FP_2	FP_3	...	FP_i	...	FP_{NT}
TU_1	*			...		...	
TU_2		*		...		...	
TU_3			*	...		...	
...	...	...	...	...		...	...
TU_i				...	*	...	
...				...		...	
TU_{NT}				...		...	*

An online informatics application enforces the existence of administrating component which has the role of supervising the way processing unfolds, which needs the user TU_{NT+1}, with special access rights regarding the access to all processing functions for operative interventions when needed, when a operating error was signaled. The modified access rights matrix is:

TU/ FP	FP_1	FP_2	FP_3	...	FP_i	...	FP_{NT}
TU_1	*			...		...	
TU_2		*		...		...	
TU_3			*	...		...	
...	...	...	...	...	...	...	...
TU_i				...	*	...	
...	...	...	...	...	...	...	...
TU_{NT}				...		...	*
TU_{NT+1}	*	*	*	...	*	...	*

The resource access functions of the online application must be treated distinctly for the users that have maintenance roles. This group includes the users that bring the data base up to date, execute operations in conformity with the transactions made by the end users of the application.

In an e-commerce application, the client represents the one that:

- gets informed, this meaning no authentication because the access rights to the application's resources imply only consulting a list of components stored in a data base, obtained by applying a selection request;
- order products with payment upon delivery;
- order products with online payment;
- become regulars by repeated purchasing;
- enter comments regarding purchased goods and rate them.

For the current processing there is staff which accesses the application for:

- procurement of supplies;
- delivery to clients;
- bringing data bases up to date;
- defining new processing functions for development and implementation;
- creating access rights for clients;
- processing the requests in the queue;
- strictly financial information validation.

In organizational tree structures, adopted in conformity with the management schema, the processing functions access orthogonallity restriction is retrenched.

The individuals on level *h+1* from the tree structure have access rights to certain resources, and the individual on level *h*, figure 6, which coordinates them, besides his specific rights, has rights to check the aggregated results of all the individuals that report to him.

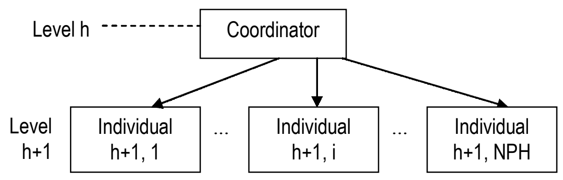

Figure 6. *The coordination report from level h towards level h+1*

Adjacently, the access rights matrix will include levels from the tree structure associated to the organization and processing functions, resulting double access for coordinators from level NIV_h for their specific functions but also their access to the functions of individuals in their suborder from level *h+1*.

The access matrix is:

NIV/ FP	FP_1	FP_2	FP_3	...	FP_{i-1}	FP_i	FP_{i+1}	...	FP_{NT-1}	FP_{NT}
NIV_1	*			...				...		
NIV_2	#	*		...				...		
NIV_3		#	*	...				...		
NIV_4			#							
...	...	...	...	...				...		...
NIV_{i-1}					*					
NIV_i				...	#	*		...		
NIV_{i+1}						#	*			
...				...				...		
$NIV_{NRNIV-1}$									*	
NIV_{NRNIV}				...				...	#	*

The „*” symbol represents the personal access rights of individuals on level *h*, and the „#” symbol represents the right to check the way in which the subordinates from level *h+1* are working.

If when defining the problem for which the informatics application is developed the definition of user classes, the definition of processing functions and the access types have another approach the following need to be clearly defined:

- the interferences generated by multiple access;
- the effects of multiple access;
- the risks implied by resource multiple access;
- the responsibility distribution for multiple access;
- the inclusion of supplementary information for the individualization of operations which take place when using resource multiple access.

For systems with a generalized character against a criteria, like the subsidiary system of a bank, the criteria being the territory dispersion or an educational structure, based on a level organization, with components dispersed in the territory, the access matrices are clearly defined, enforcing in the job description enough restrictions in order to:

- reduce the interferences generated by multiple access;
- minimize the effects of multiple access;
- manage the risks implied by resource multiple access;
- identify precisely who made a modification, from what office was the modification made, at what time, such that the distribution of responsibilities when dealing with multiple access is managed at the highest level.

The access matrix is a dynamical construction in two ways:

- if the access is very clear for a user a resource, but the evolution of processing proves that it is a very restrictive decision making typology, from the matrix in figure 6 it shall be moved on to another matrix in which some resources become multiple access, with the condition that this is reflected in the management schema, and in the data base structure identification fields must appear of the ones that accessed the resource;
- if the initial application includes a lot of resource multiple access elements along with risks and ambiguities, the following step is to refine the access,

eliminating rights, such that in a matrix column a lot less „*” or „#” operators will be found.

The access matrix construction process is iterative.

The refinement of the access matrix depends on experience and must lead to a balanced construction.

For online informatics applications belonging to the same class, the access matrix must not differ significantly.

3. User authentication

Authentication is a process necessary for selecting users in order to hand in access rights to the resources of the distributed application.

Any other distributed application, against to user access, disposes of components that are accessed without restrictions by them, ensuring transparency in the organization – individual relation, of components for which authentication is necessary and rights belonging to specialists needed to operate on the application modules.

Authentication assumes:

- creating a data base for authentication, DBA, which contains user identification elements, a string of characters representing user names, a string of characters representing the access passwords;
- the procedure of taking the input from the users;
- a mechanism of checking the existence of user defining coordinates in the DBA data base;
- the implementing of access rights by traversing the MDA matrix.

Authentication has implemented versions such as:

- procedures in which the application administrator defines access rights and creates a record in the database DBA for each user; users use the character strings announced by the administrator to gain access to application resources; there are situations in which the procedures allow each user to generate own character strings for user name and password, respecting rules enforced regarding uniqueness, length and structure of the character strings;
- procedures in which the user directly defines the necessary character strings for the authentication process and inputs identification data which are used to execute transactions; a distributed application built in this manner must include user classes, and the user belonging selection automatically creates initializations in the access rights matrix;
- authentication procedures which use peripheral equipment through which vocal, digital and ocular prints are taken from the user; there are cases in which the users receive cards and authentication is realized by running the card through a device; in this case the card is also a support for storing information, which allows independent transfers of data towards autonomous applications.

There are standard procedures for implementing specific authentication functions and it is extremely important to add new software elements to these procedures in order to reduce risks like:

- the risk of user access in an area given to another user because of reduced accepted orthogonallity level through the structure of the distributed application structure regarding the content of the data base, DBA, to user name fields and password fields;
- the risk of triggering irreversible operations on files and in the personal data bases of each user which is authenticated, by hiding destructive operations, making content copies which are used to restore files and data bases affected by destructive processes, if the user realized in a reasonable time interval of the produced accidents; a balance is made between logical erasing and physical ones based upon the users statistical behavior;
- the risk of deploying processing characterized by operations which suppose irreversible modifications of files and databases, processing made for the first time by the user; procedures are meant to assist processing, either putting the user in a situation to decide, either archiving previous information before running the destructive operations; the user must know that indifferently which processing he makes there are complete archives with values of operands from the last *NKA* sessions reported to the current session of the informatics application.

Informatics applications record user behavior related to the following:

- time required for authentication;
- number authentication errors;
- types of authentication errors;
- number of opened sessions by authentication in a time interval;
- the moments of modifying the strings that define the user name and password;
- the structure of the strings that define the user name and password in report to the users' personal data;
- the moments and operations in which the user proceeds to making irreversible modifications to the content of the data bases;
- the available resources in the application reported to the access time established in the contracts, subscriptions and direct payments;
- warning messages sent to the users.

All this information is stored awaiting processing to contribute to perfecting internal management of the distributed application.

If the user makes modifications on the distributed informatics application is necessary, the authentications through biometrical elements or character strings is enough.

If the interaction between several distributed applications is necessary, each of which posses its own authentication system, to ensure continuity to the processing stream, it is required that the user is authenticated each time. On each authentication, the user must respect the rules set for the application which resources he desires to access from the base application. In the case of on-line shopping, there is an access to the virtual shop's resources and one to the e-banking application from the bank where the user has an account. The first authentication respects the rues set for the virtual shop and the second one enforces the rules set by the electronic bank payments. There are situations in which the statistical studies on informatics applications A_i and A_j regarding:

- the orthogonallity of strings associated with user names, SNU_i;
- the orthogonallity of strings associated with passwords, SPU_i;
- the orthogonallity between SNU_i and SNU_j;
- the orthogonallity between SPU_i andSPU_j;
- the orthogonallity between SNU_i and SPU_i;
- the orthogonallity between $(SNU_i \cup SNU_j) \cap (SPU_i \cup SPU_j)$

determines the elaboration of procedures which restructure character strings such that by increasing the level of orthogonallity inside the application and inter-application will reduce the risks when the interaction user – application is unfolding.

4. Optimization algorithms

Currently to optimize means to have:

- a set of restrictions in which a process evolves;
- an objective function which must be maximized;
- a list of variables which influences the evolution of a process;
- a field for defining the evolution of each variable.

Currently to optimize means:

- finding a set of values for which the objective function has the highest value;
- obtaining the set of values by a finite number of steps of an algorithm;
- to demonstrate that the unique set of values also called solution is unique;
- to highlight that the optimum solution maximizes the objective function with no consideration to other values of variables in their definition domain.

Correspondingly, the minimizing objective function is defined.

In software production several versions of informatics applications are developed. The informatics application A_i is designed in the $A_i^0, A_i^1, A_i^2, ..., A_i^S$ versions. For each version the level of the objective function is computed $f_{ob}()$, of maximum. A string with $S+1$ values results, $\{f_{ob}(A_i^0), f_{ob}(A_i^1), f_{ob}(A_i^2), ..., f_{ob}(A_i^S)\}$.

To optimize the development process of the informatics application A_i means:

- to choose from the string of $S+1$ values $\{f_{ob}(), f_{ob}(), f_{ob}(), ..., f_{ob}()\}$, the element with the maximum value, this results in the A_i^{optim} application which will be developed;
- to cross the development cycle for the in the A_i^{optim} application;
- to implement the finished product.

It is observed that in the general optimization problem case the optimum solution regards any of the values in the defining the domain of the variables, in the area of software optimization, the optimum solution endorses a finite number of choices. In the current case there are $S+1$ choices and to optimize means selecting that choice from the $S+1$ set which maximizes the objective function.

The choices have a dynamical character. It means, that at a given time, the software development team imagines the A_i^{S+1} application version. To the initial array of versions this one is added as well, the array becomes { $A_i^0, A_i^1, A_i^2, ..., A_i^S, A_i^{S+1}$ }. The S+2 values array is computed { $f_{ob}(), f_{ob}(), f_{ob}()$, ..., $f_{ob}(), f_{ob}()$ } from which the maximum value element will be selected, element which indicates the informatics application version for which the stages of the development cycle will be traversed.

According to the development cycle of a software application the stages which have to be covered in order to obtain a final form in the development process are:

- analysis;
- design;
- implementation;
- testing.

In order to optimize an application one needs to optimize each stage of the development cycle of the product. Multiple different readings are created for each stage,

$$VERS_{i,j}, i = \overline{1,4}, j = \overline{1,r} ,,$$

where:

i – stage *i* in the development process of the application;

j – different reading created for a stage in the development process.

The optimization of each stage leads to a final result, a distributed informatics application, which because of this iterative process (figure 7), unfolded in the frame of analysis, design, and implementation and testing according to dynamical programming represents an optimized informatics application.

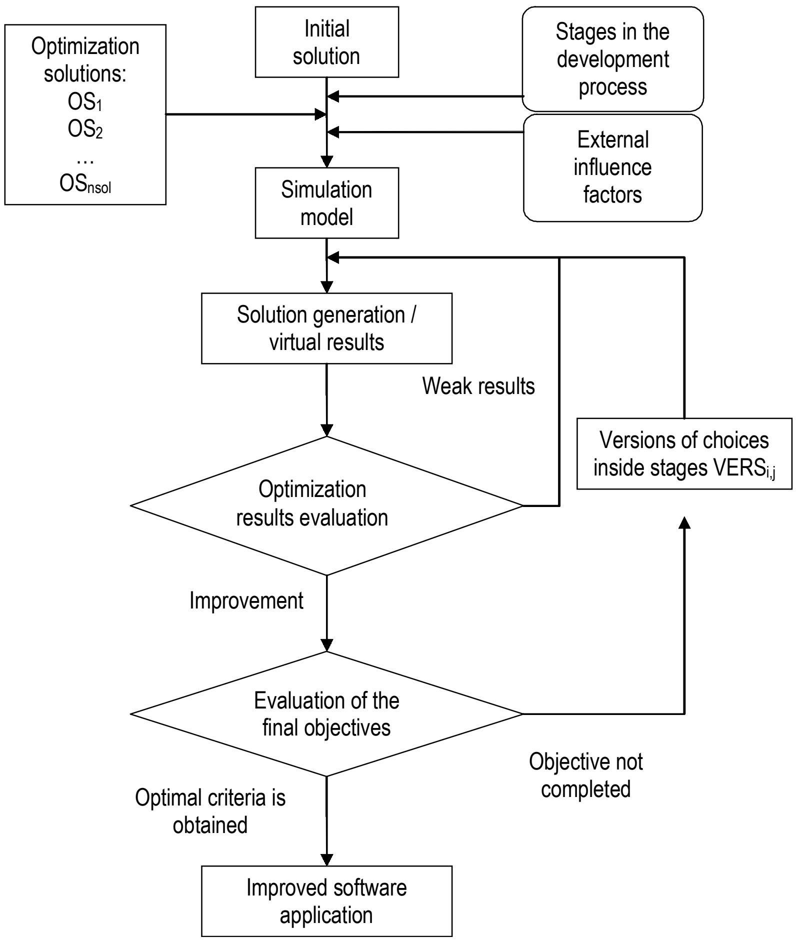
Optimization solutions:
OS1
OS2
...
OSnsol
Initial solution
Stages in the development process
External influence factors
Simulation model
Solution generation / virtual results
Weak results
Versions of choices inside stages VERSi,j
Optimization results evaluation
Improvement
Evaluation of the final objectives
Objective not completed
Optimal criteria is obtained
Improved software application

Figure 7. *The optimization stage based process*

Informatics security is a requirement for all distributed informatics applications because:

- user access is controlled;
- the software and hardware components are protected;
- the data bases are protected;
- the risks regarding data base deterioration and software modules are controlled.

To optimize the security components in a distributed application means:

- to define the optimal state criteria: maximizing the generality degree, minimizing the duration of a transaction, minimizing the accommodation duration of individuals with new applications, maximizing the level of flexibility, maximizing the diversity of solved problems in a work session, minimizing the volume of user input data as an effect of reusing data that already exists in the personal data base and in accessed data bases, coming from applications belonging to other owners;
- choosing one of the criteria in order to reach it according to the way in which the security components are designed;
- to build restriction sets for security models: a list of operations is made in the application and users are put in correspondence with elements from the list; when no resource is accessed for a defined interval of time, the user resource access session automatically ends;
- to find practical modalities which maximize/minimize the closed criteria;
- to test the software sub-system developed for the distributed application security;
- to implement the informatics application that incorporates the security sub-system.

For authentication, optimization does not include:

- the building manner of SNU and SPU arrays;
- the implementing of procedures which ensure a adequate level of orthogonallity inside each array and between arrays;
- establishing the moment in time to restructure SNU and SPU;
- the way private information is accessed if authentication credentials are lost.

Optimization in authentication process assumes:

- recording user behavior;
- setting costs for access and negative effects that come from wrong decisions;
- building an optimization model;
- choosing the most convenient alternative of authentication procedure from very small set of different procedures existing in specialized literature.

An optimization problem is considered: maximizing the degree of client satisfaction.

The relation gives the indicator used to compute the degree of satisfaction GSC:

$$GSC(x,u,v,w) = \frac{ax+bu+cv+dw}{x+u+v+w},$$

where:

x - individuals that log in order to solve *n* problems;

u – individuals that log in to solve all problems;

v – individuals that log in and solve *k* problems from *m* suggested ones;

w – individuals that log in and are satisfied with the presented information;

a – importance coefficient associated to the individuals which log in to solve *n* problems;

b – importance coefficient associated to the individuals which log in to solve all the problems;

c – importance coefficient associated to the individuals who log in and solve *k* problems out of *m* suggested ones;

d – importance coefficient associated to the individuals which declare that are satisfied with the information found.

First it is proceeded to obtain the importance coefficients a, b, c, d using one of the existing methods, out of which a method is proposed through which:

- the direct distributed informatics applications users set is built, formed out of NRS specialists $S_{1,} S_2, \dots, S_{NRS}$;
- rules are established based on which the qualifiers or points are given for each of the categories of individuals which access the application; the marks will be between 0 and 10, without grading two characteristics;
- a table is obtained, it contains NRS lines and a number of columns equal to the number of variables, 4; the specialist S_i gives a mark x_i for the *individuals that log in order to solve n problems* variable; the specialist S_i gives the mark u_i for the *individuals that log in to solve all problems* variable; the specialist S_i gives the mark v_i for the *individuals that log in and solve k problems from m suggested ones* variable; the specialist S_i gives the mark w_i for *the individuals that log in and are satisfied with the presented information* variable;
- the sums S_x, S_u, S_v, S_w, S_{TOTAL} are computed as:

$$S_x = \sum_{i=1}^{NRS} x_i, S_u = \sum_{i=1}^{NRS} u_i, S_v = \sum_{i=1}^{NRS} v_i, S_w = \sum_{i=1}^{NRS} w_i$$

$$S_{TOTAL} = S_x + S_u + S_v + S_w$$

- the importance coefficients for the four individuals categories are computed like this:

$$a = \frac{S_x}{S_{TOTAL}}, b = \frac{S_u}{S_{TOTAL}}, c = \frac{S_v}{S_{TOTAL}}, d = \frac{S_w}{S_{TOTAL}}$$

- the VA_1, VA_2, VA_3, …, VA_{NV} application alternatives are elaborated based on NV different types of authentication; each of the alternatives contain in their modules counters to keep track of resources accessed by users; after the users which form the set of NPRS individuals test the VA_1, VA_2, VA_3, …, VA_{NV} applications, these counters are stored in a data base with NV records, each record containing X_j, U_j, $V_{j,}$ W_j, where:

X_j - number of individuals that log in order to solve *n* problems in VA_j application;

U_j – number of individuals that log in order to solve all problems in VA_j application;

V_j – number of individuals that log in order to solve *k* problems out of *m* suggested ones in VA_j application;

W_j – number of individuals that log in and are satisfied with the information presented by the VA_j application.

- Information is extracted from the data base and the indicators are computed

$$GSC(X_j, U_j, V_j, W_j) = \frac{aX_j + bU_j + cV_j + dW_j}{X_j + U_j + V_j + W_j}, j=1,2, \ldots, NV$$

and the array is constructed$\{GSC_1, GSC_2, GSC_3, \ldots, GSC_{NV}\}$.

- The expression is evaluated:

$$GSC_{max} = max(GSC_1, GSC_2, GSC_3, \ldots, GSC_{NV})$$

If $GSC_{max} = GSC(X_h, U_h, V_h, W_h)$, it means that alternative VA_h is the best because it has the highest level of satisfaction of individuals which interact with it.

If a series of modifications are made for an application VA_s in order to improve its characteristics, another version of the application will be obtained, which is added to the existing ones, becoming VA_{NV+1}. For this application the measuring process is rerun using the same set and the indicator is computed:

$$GSC(X_{NV+1}, U_{NV+1}, V_{NV+1}, W_{NV+1}) = \frac{aX_{NV+1} + bU_{NV+1} + cV_{NV+1} + dW_{NV+1}}{X_{NV+1} + U_{NV+1} + V_{NV+1} + W_{NV+1}},$$

The expression is evaluated:

$$GSC'_{max} = max(GSC_1, GSC_2, GSC_3, \ldots, GSC_{NV}, GSC_{NV+1})$$

The high effort needed to ensure a correct base for continuing the optimization process to maintain the stability of the set and keeping the database up to date and the computing of the aggregated indicator, which measures the degree of satisfaction for new application versions, is noticed.

Before launching the versions in use by the members of the set, it is necessary to proceed with application version testing using the same sets of test data. The test data must activate all the modules of the application, and the counter variables included in the modules must supply sufficient information such that the testing process is very efficient.

The versions are justified when the user number is very high and a favorable difference of 1% in the GSC_i indicator, compared to the GSC_j indicator determines important resource saving.

The situations in which the test data sets are irrelevant and the user set is unrepresentative must be avoided. If this kind of situation occurs, the versions that will be selected as the best prove during effective exploitation that it generates a waste of resources and leads to a low degree of individual satisfaction.

5. Experimental verification process unfolding

For no matter which distributed informatics application or type of implemented authentication system, it is needed to verify experimentally:

- application behavior;
- user behavior;
- establishing the authentication process;
- the types of errors that appear;
- the effects generated by blocking resource access;
- the efficiency level regarding resource use allocated to each user;

The distributed application ADV and a model MA implemented for authentication are considered.

For checking, a representative sample of users with a NUE dimension is constructed, which ensures statistical error.

The ADV application is launched in execution.

The user behavior gathered in the user sample is registered automatically using functions built and activated from the application as follows:

- the moment when the user UE_i referred the application;
- the moment when user UE_i exited the application;
- the string inputted by user UE_i for its associated user name;
- the string inputted for UE_i's associated password;
- the number of re-input attempts $NRUE_i$ in case of unsuccessful inputs;
- the increment of the variables associated to the processing procedures activated by the user in order to solve the problem for which the application was accessed;
- the number of estimations of type *j* given by the user to the application, according to the obtained solution, NQ_j.

All this information is stored in the database of the application behavior, ADV.

Simultaneously with the elaboration of the ADV application a software product is developed destined to process the rows in the application behavior database.

The processing allows the obtaining of indicators like:

- average duration of resource accessing from individuals that make up the representative sample;
- the orthogonallity level between the character strings defined as user names in the application and the character strings inputted in the process of accessing the application;
- the level of orthogonallity between the character strings defined as passwords of the application users and the password strings inputted by the users in the application accessing process;
- the user degree of satisfaction from solving problems using the ADV application;
- the average number of unsuccessful application accessing attempts;
- the average length of the path, defined as the number of activated procedures, traveled by the users in order to solve the problems;
- the list of procedures which generated unsuccessful processing, ordered by decreasing number of failures.

All these elements and others that are completed systematically when a more complete defining of the user environment is needed, but for the application behavior as well, must be used constructively to perfect the application. Even more, the indicators that relate to the authentication components must be analyzed in detail to modify the authentication procedure structure such that the following are reduced on the server:

- the number of successful access attempts to the application resources from valid users;
- the number of successful application access attempts from invalid users;
- number of access to unallocated resources.

The unfolding of the verification process is a permanent activity that is undertaken at regular time intervals. The results of the processing related to user behavior between two moments of processing fundament decisions relating to:

- authentication procedures;
- processing streams;
- the vocabularies in the interfaces;
- the final results structures.

Only in case in which the application in improved dynamically a growth in the user satisfaction degree is obtained, the only criteria which has to be the base of optimizing processes in distributed applications, because these are developed in order to solve problems of individuals and only after to optimize the informatics streams in organizations.

6. Conclusions

Informatics security, by definition represents all the procedures and methods through which an informatics system is protected by deliberate or non-deliberate attacks which originate from the exterior of the application. This definition is applied at a smaller scale, so that to talk about the security of memory zones in an application.

Authentication represents only a component through which access to application resources is defined, managed, and protected.

It is important to move systematically in order to include all the components of the securing process for the informatics distributed application, by iterating through the following steps:

- ensuring the *availability* of applications at any given time, by implementing securing methods of informatics systems: protection in secure chambers, restricted access to these chambers;
- protecting the distributed application systems by adding intermediate password security levels to ensure the application's integrity;
- preventing unauthorized access to distributed applications to avoid any prejudice because of braking the *confidentiality* of the information in the application;
- implementing encryption systems to sign the information in the distributed application such that the *non-repudiation* principle is not broken.

For the above steps one proceeds to the fields definition in the application such that when events linked to them are produced a variable increment takes place and the data base application behavior related content is altered.

Indicators are constructed to evaluate the security level reported to the added components, to see the measure in which they are efficient or not. Even in such case the permanent character of analysis and modification of procedures is considered, so that the level of security is improved all the time. Parallel with these activities the effects lack of security causes is measured, such that efficiency computations are made to establish the cost of obtaining a maximum level of security reported to the hypothetical defined system.

References

Doinea, M., „Indicator agregat pentru securitatea aplicațiilor distribuite", lucrare de disertație la cursul de master Securitate Informatica, ASE, București, martie 2008

Patriciu, V.V., Bica, I., Ene- Pietroseanu Monica (1998). *Securitatea Informatică în UNIX și Internet*, Editura Tehnică, București, 1998

Patriciu, V.V., Bica, I. (2001). *Securitatea comerțului electronic*, Editura All, București

Boian, F.M. (1999). *Programarea distribuită*, Editura Albastră, Cluj-Napoca

Coulouris, G., Dollimore, J., Kindberg, T. (2001). *Distributed systems concepts and desing*, Addison – Wesley

Jaquith, A. (2007). *Security metrics Replising Fear, Uncertainty and Doubt,* Addison – Wesley, New York, ISBN 0-321-34998-9

www.web.mit.edu/kerberos - despre autentificare

www.wireshark.org – despre autentificare

Annex 1

Variable and operator list

C	The collectivity that makes up the target group
c_i	The ith element in the collectivity
N	The number of elements that make up the C collectivity
M_j	The jth module in the structure of the A distributed application
K	The number of modules which make up the distributed application
I	Input data
E	Results obtained after processing finished
P	The initial problem for solving
SP_h	Sub-problem h
NT	Number of user types
TU_i	The type of user *i*
NU_i	The number of users of type *i*
NTU	The total number of users in the application
FP_i	Processing functions of on-line distributed informatics applications
NIV_h	Level *h* of the tree structure associated to an organization
NRNIV	Then number of levels of the tree structure
NKA	Number of accesses
SNU_i	The string of user names in application A_i
A_i	Distributed application i
SPU_i	The string of passwords in application A_i
reuniune	The reunion operator on sets
intersectie	The intersection operator on sets
S	The number of versions in which application A_i is elaborated
$f_{ob}()$	Objective function
A_i^{optim}	The optimal application version reported to the maximum criteria $f_{ob}()$
MA	Authentication model
ADV	Distributed application undergoing the checking process
NUE	Number of users in the representative sample
UE_i	User i in the representative sample
$NRUE_i$	The number of re-inputs of the user name and password in case an error occurs
NQ_j	The number of *j* type appreciations; ; NQ_0 – the number of unsatisfied users, NQ_1 – the number of users that give the application a satisfactory qualification; NQ_2 – number of users who are satisfied and give the well qualification; NQ_3 – the number of users that appreciate the application as very good;
R	Number of choices in each development stage
$VERS_{ij}$	The solution version *j* realized in the *i* development process
Nsol	The number of application optimization solutions inside the proscess