Adaptive Model for Authentication Optimization Process

Mihai DOINEA
Academy of Economic Studies, Bucharest, Romania
mihai.doinea@ie.ase.ro

A security architecture for web based distributed applications is proposed with accents on one of the security component, part of this framework. Optimization criteria is identified and discussed. The authentication is presented, identifying the possible directions to be addressed in the optimization process. An adaptive model of authentication is outlined and the advantages brought by it are debated. Optimization process is conducted to improve the overall security level. The model is validated, conclusions are drawn and future work is presented.
Keywords: *Security, Optimization, Distributed Applications, Dynamic Authentication*

1 Authentication adaptive model

The security of web based distributed applications is considered to be ubiquitous in the life cycle management processes. Security must be viewed as a separate layer independently treated but also in a perfect symbiosis with other levels of the application and of the entire distributed infrastructure in which the application is running and sharing resources.

The administration and access controls are implemented and an optimization process of the security level is conducted using adaptive algorithms based on probabilistic and neuronal algorithms, such as Bayesian analysis [1] and feed forward neural networks, by studying the behavior of the distributed application's users. In works like [2], the impact of authentication security level neuronal algorithms appliance was analyzed, having a favorable result.

The process of determining the behavioral legitimacy emphasized in the following picture has also the role of being a weight in the algorithm for adjusting the system and user credibility by taking into account the classification results of user's authentication request.

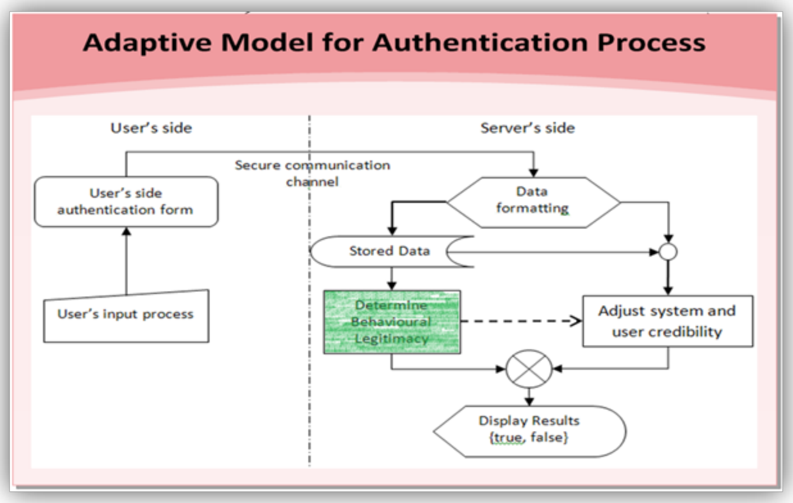

Adaptive Model for Authentication Process
User's side
Server's side
Secure communication channel
User's side authentication form
Data formatting
Stored Data
User's input process
Determine Behavioural Legitimacy
Adjust system and user credibility
Display Results (true, false)

Fig. 1. The adaptive model for authentication process

The model discussed emphasizes the access level in which an authentication adaptive algorithm is implemented in order to analyze individuals' behavior and classify its actions based on hidden patterns revealed in later analysis.

This algorithm uses classification techniques to tell whether an access attempt to a distributed system, made from an account, is in-

itiated by its original owner being implemented on to the authentication process.
The same characteristics could be used for determining a level of credibility both for user and system which are mixed resulting an individual number of failed login attempts for each account.
On the other hand, an ***A**daptive **M**odel for **A**uthentication **P**rocess*, *AMAP*, relies on user's characteristics that were recorded over a time period, and from which, patterns could be deducted in order to classify further login requests and decide whether or not give access to the application's resources.
This approach is emerging from the necessity of using the power provided by the uniqueness of users' behavior which ultimately is hard to fake by third parties that wish to gain illegitimate access. The process of giving access, besides the default implemented authentication method, is reinforced with this type of evaluation based on things that no one can simulate, not even computers with their almost infinite computing power, using different patterns that might reveal an impersonating attempt, such as:

- the number of user's repeated attempts before approval;
- the IP evaluation of user's login requests;
- the time interval, frequency used for authentication or for other events.

The *AMAP* grants access based only on the unique combination of username and password and on the evaluation of login behavior that will also increase the efficiency of the authentication process, being much harder to break as the behavior is hard to simulate and anticipate.
In their attempt of trying to hack the system, malicious attackers must match the user's login behavior with respect to a set of characteristics that are stored in the database when the events presented in the following table are triggered.

Table 1. Registered System Events

Event code	Event Description	Recorded behavior characteristics
100001	Account accessed	Internet Address, user account, event frequency
100002	Wrong Password	
100003	Account Locked	
100004	Wrong User	
100005	User not approved	
100006	Correct User	The nature of the authentication request based on a perceptron network and a naïve Bayesian classifier
100007	False User	
100008	Account Created	Number of events recorded per user's account
100009	Password Changed	
100010	Changing Password Cancelled	
100011	Password Reset	
100012	Activation Failed	
100013	Activation Succeeded	

Although the classification techniques of the *AMAP* system are known for their false positive and false negative results, this is not a risk or a detriment, but a stronger point in revealing the hazardous nature of user's actions, sending alert messages to notify the true owners that their accounts were used along with the characteristics of those login requests. The main idea is that when users are trying to access the system, all the previous miss attempts or successful logins are used in determining the nature of the current access attempt.

2 Authentication processes optimization criteria

One of the gates of getting access to resources in a web based distributed environment is through the authentication level. If this is either, weak and easy to breach, or it's difficult to be accessed, its trusted users will loose the confidence and migrate to another system which offers same resources. The authentication system must be calibrated in order to obtain a balance between the level of security and the effort made by hackers to gain access to the system's resources. The authentication schemas [3] used in web based distributed applications are characterized by:

- one way authentication used to verify the identity of just one party, likely the users that are trying to access the distributed system's resources;
- mutual authentication represents a two way authentication method in which both the user and the system are verified mutually between each other in order to allow a secure connection;
- multi factor authentication is also called a strong factor authentication in which different methods are used to establish the right identity of the user who asks for permission, such as: username and password, biometric data and a token or smart card hardware;
- CHAP authentication is a specialized way of initializing communication, coming from challenge handshake authentication protocol used in PPP, point to point protocol servers, verifying periodically the identity of users that are accessing the application's resources;
- biometric data represents a set of methods for stopping identity theft by checking a user's unique characteristics like: eye's retina, fingerprints, vocal print, hand or facial physiognomy;
- Kerberos methods are a mutual authentication protocol developed by MIT based on a trusted third party recognized by either two of the entities who are in the process of authentication;
- token access is used as a reinforce method of authentication using a hardware device which is assigned to a user and correlated with its account; in order to access the resources of an application, the user accesses his account after which he strengthens its identity by generating a *OneTimePassword* for the final authentication step.

The concept of *Adaptive Model for Authentication Process* presented also in [4] is using a function *f*, as the one described above, that takes information about the authentication request events, as a vector of characteristics x_i, as input and classify the user's attempt, resulting a {*true*, *false*} response as output:

$$f(x) = y,$$
$$f: A \rightarrow B,$$
$$B = [0;1],$$
$$A = \left\{ x_i = \begin{pmatrix} x_{i1} \\ \dots \\ x_{in} \end{pmatrix}, \forall i = \overline{1,m} \right\}$$

where:

A – a set of vectors of unique login request characteristics;

n – the number of characteristic taken into account;

m – the number of vectors, combinations of characteristics;

B – the output classification of user's login requests.

For being able to correctly authenticate users through *AMAP*, a routine procedure on the server is conducted based on user's login characteristics, resulting in a set of indicators composed by probabilities which will tell the probable nature of the login attempt event. The procedure premises are unfolded as follows:

- the existence of previously recorded authentication characteristics request in the database;
- procedures for preprocessing the stored information;
- the existence of a set of metrics used to measure the user's behavior.

The actual classification routine is evaluating the input data, determining how the event login characteristics reveal patterns that might lead to the discover of possible attacks against user's accounts. This process is following the next steps:

- identifying a set of indicators and probabilities for the current login attempt;
- depending on the values revealed by the measures previously taken three situations can occur:
    - the classification values aren't conclusive not being in an acceptable error limit and an inform message is sent to the owner's account email along with the authentication characteristics, asking for approval of whether the login attempt was his or not;
    - a positive match is found and the user can access all the application's resources put at their disposal by the membership policy;
    - the result values are classifying the user's attempt as not being the correct owner of the account, and the event is recorded in the database.

For the fact that the system cannot tell from the start which user is impersonating another's identity, the only way of making the difference between them is by measuring the level in which the characteristics with which they are trying to access the system are similar to previous recorded data, as presented in [5].

The adaptive authentication component's main goal is to optimize the security level of the web based distributed application taking into account the user's measured behavior. This optimization process must be conducted in such a way that the user's usability features would not be affected, in other words, the security components implemented in the application should not interfere with the capacity of the users to use freely the resources provided by the distributed architecture. Figure 2 reveals a balance which must be achieved, according to [6], between the total limitations of the distributed applications interface, which are part of the security level and contribute to the optimization process, and the usability factor perceived by users at a normal rate.

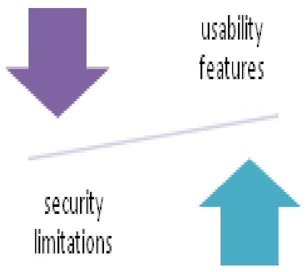


Fig. 2. Security optimization balance

The usability of a web based distributed application is negatively influenced by the security features implemented for the optimization process. The following metric measures the number of additional operations that a user must undertake in order to complete a process which now is directly affected by the security features that were implemented. It describes the degree in which a user is constrained to cope with the security level in order to achieve its objectives.

$$USM = \frac{ano}{ino}\%,$$

where:

ano – additional number of operations made by user when security is implemented;

ino – initial number of operations made by users for achieving their objectives.

If value of *ano=0* then the *USM* = 0%, meaning that the security features used for optimization are not influencing at all the operation flow which a user must do for achieving its goals.

3 Authentication optimization model

The adaptive authentication algorithm comes to add a plus of security by using existing information about the user's access behavior. The information is used to tackle the optimization of the authentication process using automatic stored data after each event triggered in the system, such as users' successful logins, failed password attempts, password reset or change operations, data and time application's events, network internet and physical addresses, operating system user of the remote device.

Let k_1, k_2, …,k_n be the characteristics taken into account for assessing the authenticity of a logging request, counting the number of total successful and unsuccessful logins and trying to identify the number of false positives and false negatives access attempts. These characteristics are used to approach the following directions:

- establishing the number of password attempts at user and system level;
- determining the true nature of user's logging request, either true, false, and false positive or false negative using a Bayesian network which has the characteristics taken into account as input and the output being given as true or false [7] and neuronal algorithms using a feed forward network.

The Figure 3 presents the information flow from the moment of originating the authentication request until its classification in either true or false login attempt.

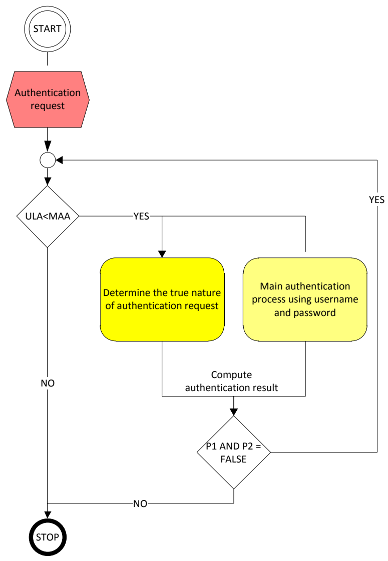

START
Authentication request
ULA<MAA
YES
YES
Determine the true nature of authentication request
Main authentication process using username and password
NO
Compute authentication result
P1 AND P2 = FALSE
NO
STOP

Fig. 3. Adaptive authentication algorithm schema

The events are stored in a database along with user's coordinates and the station from which the event was triggered, counting the number of total successful and unsuccessful logins.

Based on such events recorded the system and user's credibility are automatically adjusted based on how higher or how low is the rate of such events. The level of credibility it reflects directly upon the number of maximum failed login attempts calculated for a user before his account will be locked out.

Let $UFLA_i$ be the user's i maximum number of failed login attempts with the following possible values $UFLA_i \in \{1,2,3,4,5\}$. The $UFLA_i$ level is strictly dependent on the system credibility level, because if a user is registered to a highly vulnerable system, its safety is also affected and vice versa.

The $UFLA_i$is calculated based on the following formula:

$$UFLA_i = \left\lceil \frac{SFLA + FAU_i}{2} \right\rceil, i = \overline{1, nu}$$

where:

SFLA – is the system maximum number of failed login attempts also with the following possible values$SFLA \in \{1,2,3,4,5\}$;

FAU – represents the level of failed attempts established for the user's behavior with values between$FAU \in \{1,2,3,4,5\}$;

nu – the number of total users registered to the system.

The system credibility, SFLA is determined easily based on the level of probability of the two components which are part of it as presented in the following table:

Table 2. System's failed password attempts

SFLA	$\frac{A_{11} + A_{12}}{2}$
1	0-20%
2	21-40%
3	41-60%
4	61-80%
5	81-100%

where:

A_{11} – is the probability of correct login attempts calculated based on the category 1 events;

A_{12} – is the probability of correct login attempts calculated based on the category 2 events.

The A_{1j} probability is given by:

$$A_{1j} = \frac{\sum_{k=1}^{nsl_j}(EVENT_{kj} == SEV_j)}{ntl},$$

where:
nsl$_j$ –the total number of successful login events from category j;
ntl –the total number of login events;
SEV$_j$ –the successful event code with$SEV_j \in \{100001,100006\},\ j = 1,2$;
EVENT$_{kj}$ –the *k* event code recorded for the *j* category.
The level of failed password attempts established for the user's behavior is also composed of two parts based on the aforementioned category events, as shown in Table 3:

Table 3. User's failed password attempts

FAU$_i$	$\frac{A_{i21} + A_{i22}}{2}$
1	0-20%
2	21-40%
3	41-60%
4	61-80%
5	81-100%

where:
A$_{i21}$ – is the probability of correct login attempts calculated based on the category 1 of events for the user *i*;
A$_{i22}$ – is the probability of correct login attempts calculated based on the category 2 of events for the user *i*.
The *A*$_{i2j}$ probability is given by:

$$A_{i2j} = \frac{\sum_{k=1}^{unsl_{ij}}(EVENT_{ikj} == SEV_j)}{untl_i},$$

where:
unsl$_{ij}$ – the total number of successful login events from category *j* for user *i;*
untl$_i$ – the total number of login events for user *i*;
SEV$_j$ – the successful event code with$SEV_j \in \{100001,100006\}, j = 1,2$;
EVENT$_{ikj}$ – the *k* event code recorded for the *j* category for user *i*.
A security optimization is pursued in adjusting the previously defined indicators based on the credibility determined by the algorithm, for system and user level.

After this stage is completed than the actual classification algorithm is run obtaining another result which will alter the degree of trust assigned to a user, meaning that the whole system will adjust itself based on new triggered events determined by the user's actions.
The problem of classification can be seen as a decision theory, where the methods of classification are decision instruments used for object recognition. Starting from the initial population of *k* objects, let it be called$\Omega = \{\omega_1, \omega_2, \dots, \omega_k\}$, that are each partitioned into one of the classes, $\mathcal{C} = \{c_1, c_2 \dots., c_{Card\mathcal{C}}\}$, a function is designed in order to recognize each new added object by transposing it into one of the available classes. For that, a classification method is considered a function $f\colon \Omega \to \mathcal{C}$, verifying the following relation:

$$\forall \omega_i \in \Omega, with i = \overline{1,k},\ \exists!\ c_j \in \mathcal{C}, with j = \overline{1, Card\mathcal{C}}, so that f(\omega_i) = c_j.$$

Going deeper into the data analysis, the objects that are part of the population are n-dimensional elements, described by *n* characteristics, resulting in an input vector $\omega_i = \{\omega_{i1}, \omega_{i2}, \dots, \omega_{in}\}$, where ω_{ij} is the value that object *i* has for the *j* characteristic, also called feature.
The manipulation of objects and affiliation to classes is done through plurality of classification methods available in the literature. For the analysis of user's behavior two methods of classification are used and the compared results are presented:

- the naïve Bayesian classifier used for analyzing the discrete values recorded for user's behavior, outputting a probability with which the event is classified in one of the elements of $\mathcal{C}$;
- the neural network, a single layer perceptron with n nodes, used for describing the weighted relation between the vector of characteristics $\omega_i = \{\omega_{i1}, \omega_{i2}, \dots, \omega_{in}\}$and the output result $c_j \in \mathcal{C}$.

The Naïve Bayesian [8] is a classification method that uses statistics and mathematics principles for evaluating a flexible function used in the classification of a set of characteristics d into different classes h, depending on the a priori data recorded. It is based on the Bayes rule [9], namely:

$$P\left({}^{h}/_{d}\right) = \frac{P\left({}^{d}/_{h}\right) \cdot P(h)}{P(d)}$$

where:

$P\left({}^{h}/_{d}\right)$ –the probability of h to be met, given data d and knowing information regarding the a priori appearance of the hypothesis;

$P\left({}^{d}/_{h}\right)$ – the conditioned probability, given the data d and the h hypothesis met;

$P(h)$ – is the a priori probability of the hypothesis h, the probability that h is true, before knowing d;

$P(d)$ – the appearance probability of d.

The function associated to this rule of probability, called the Naïve Bayesian function, $\mathcal{B}: \Omega \to \mathcal{C}, \mathcal{C} = \{c_1, c_2\}$, is calculated as:

$$\mathcal{B}(\omega_i) = \begin{cases} c_1, if \prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_1}\right) \geq \prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_2}\right) \\ c_2, if \prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_1}\right) < \prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_2}\right) \end{cases}$$

where:

$\prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_1}\right)$– is the computation of the probabilities that the features of the vector$\omega_i$are belonging to an event of classc_1;

$\prod_{k=1}^{n} P\left({}^{\omega_{ik}}/_{c_2}\right)$ – is the computation of the probabilities that the features of the vector$\omega_i$are belonging to an event of classc_2.

Another implementation of a classification technique is done using the artificial neural networks that simulates the networks of the neural cells of the central nervous system and are formed out of a perceptron [10], the elementary level of such a network, and the bindings between them. In figure 4, the main components of a neural network are presented:

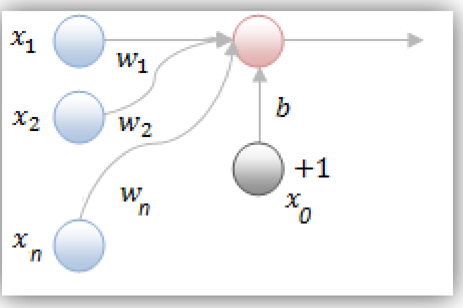


Fig. 4. Artificial neural network representation

where:

x_i – the value of the input data, with $i \in \{1,2, \dots, n\}$;

x_0 – the bias, the default input of value 1;

w_i – the weight associated to the x_i input, with $i \in \{1,2, \dots, n\}$;

b – the weight for the bias.

The total input of the perceptron is calculated as a weighted sum of the inputs, as in the following relation having n characteristics analyzed:

$$input_{total} = \sum_{k=1}^{n} w_k * x_k \,.$$

The output of the perceptron is the result of using the threshold function:

$$output = f(input_{total}, \theta) = \begin{cases} 1, & input_{total} \geq \theta \\ 0, & otherwise \end{cases}$$

whereθ is the threshold value.

Function f determines the level of activation of the perceptron based on the values of the input data and the threshold. Generally, the output of the perceptron is expressed with the net input, using the formula:

$$output = f(net) = \begin{cases} 1, if net \geq 0 \\ 0, otherwise \end{cases}$$

where$net = input_{total} - \theta$.

For assimilating the $(-\theta)$ value, the input values receive another perceptron of $(-\theta)$ intensity having the default value of 1, called bias, also represented in figure 5.2. So:

$$net = input_{total} + (-\theta) * 1 = \sum_{k=1}^{n} w_k * x_k + (-\theta) * 1 = \sum_{k=0}^{n} w_k * x_k,$$

where$w_0 = -\theta$ and $x_0 = 1$.

The training process of the neural network is composed out of the following steps:

- *weights' initialization*, the n+1 weights of the n input features along with the bias with random values;
- *defining the number of epochs and the size of the total error*, the training is done in a controlled framework, depending on the number of epochs and the total error;
- *perceptron activation* by calculation of each feature, using *input$_{total}$;*
- *interpretation of the activation value*, assigning output value based on the *input$_{total}$* and the function *f* expression;
- *weights and total error adjustment*, the values of the weights and total error are adjusted during the increase of the current epochs ;
- *final results interpretation*, if the total error is grater then the error set by the threshold, means that the values used for the training are not enough, and the network cannot be trained; by contrary, the network is trained, and the weights of each feature can be used for further classifications.

The Bayesian networks along with neural networks are methods used for classification, [11]. Regarding the aspect of similarities and differences, the Bayesian network operates upon discreet values, while the neural network has as input continuous, as long as discreet, values. For the similarities, both neural and Bayesian networks has as output discrete value, indicating the class from which the classified object is part of.

4 Optimization process validation

The method used for validation of the model implemented is called cross-validation [12], and is defined as being a statistical method for evaluation of a model through its own results. Besides being an estimative method of how accurate the predictive model is, cross-validation is used for removing testing hypotheses suggested by the current data, also called a type three error in data, when the first two are the false positives and false negatives. Also called rotation estimation, it uses a data set, partitioning it into complementary subsets, training and testing subsets. In the particular case of k-fold cross-validation [13], the number of subsets formed is equal to k, commonly using the value of 10, as follows:

$$\Omega = \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_k, with \forall i,j \in \overline{1,k} and i \neq j, \Omega_i \cap \Omega_j = \emptyset$$

After forming the k subsets, by rotation, one subset is put aside and, with the other k-1 subsets, the training is done on the predictive model tested, afterwards, the testing being done with the last subset. This process is done for *k* times, so that each subset is once used as a testing set. For each round, the results of the testing set are withheld and averaged. The objective is to compare the results of the trained model at each round with the real results that each object from the specific testing subset has. Summarizing the process, the outcome total percentage of correct classification is:

$$AP = \frac{\sum_{i=1}^{k} \frac{\sum_{j=1}^{\mathrm{Card}\,\Omega_i} (\omega_{ij}{}^{e} \wedge \omega_{ij})}{\mathrm{Card}\,\Omega_i}}{k},$$

where:

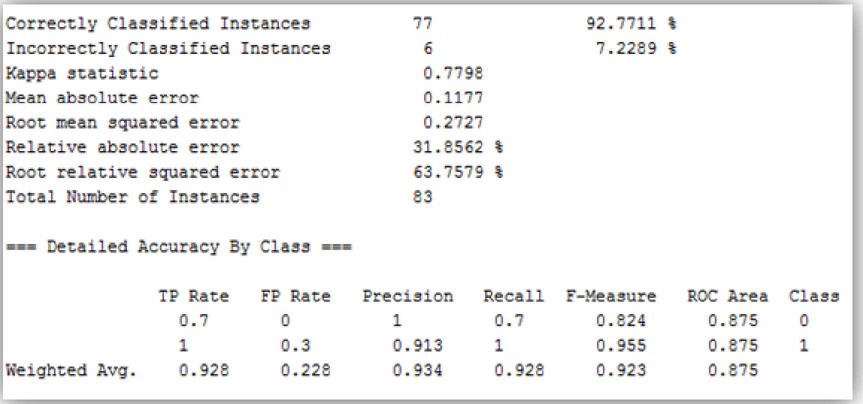
AP – the average percentage resulted from the cross-validation procedure;

Ω_i – the complementary subsets formed;

ω_{ij}– the real output value for the ω_j object, part of the Ω_i subset;

$\omega_{ij}{}^{e}$ – the estimated output value for the ω_j object, part of the Ω_i subset, using the trained model with the $\Omega - \Omega_i$ remaining set.

According this type of validation the naïve Bayesian model implemented obtained the following results presented above in the analysis made for a user account registered in the system.

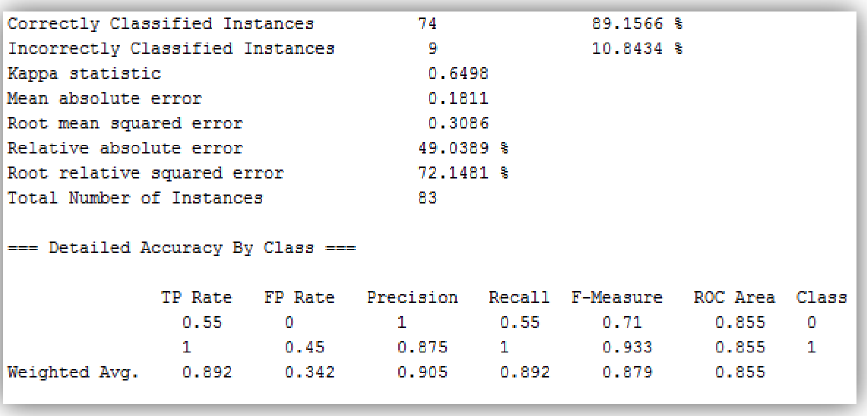


```
Correctly Classified Instances          77               92.7711 %
Incorrectly Classified Instances         6                7.2289 %
Kappa statistic                          0.7798
Mean absolute error                      0.1177
Root mean squared error                  0.2727
Relative absolute error                 31.8562 %
Root relative squared error             63.7579 %
Total Number of Instances               83

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.7       0          1         0.7       0.824      0.875    0
                 1         0.3        0.913     1         0.955      0.875    1
Weighted Avg.    0.928     0.228      0.934     0.928     0.923      0.875
```

Fig. 5. The Naive Bayesian Classifier

A 92.77% correct validation with its complementary incorrect results was obtained and only 6 instances from a total number of 83 were wrong classified.

The neural network compared with the previous method of classification obtained lower results as presented in the following figure.



```
Correctly Classified Instances          74               89.1566 %
Incorrectly Classified Instances         9               10.8434 %
Kappa statistic                          0.6498
Mean absolute error                      0.1811
Root mean squared error                  0.3086
Relative absolute error                 49.0389 %
Root relative squared error             72.1481 %
Total Number of Instances               83

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.55      0          1         0.55      0.71       0.855    0
                 1         0.45       0.875     1         0.933      0.855    1
Weighted Avg.    0.892     0.342      0.905     0.892     0.879      0.855
```

Fig. 6. The neuronal algorithm of a perceptron network

The perceptron technique was wrong in 9 cases from 83 instances meaning a percentage of 10.84% from the total amount of login events.

Based on the two methods of classification, the following table presents the confusion matrix which shows how many attempts of a specific class were classified as belonging to the other.

Table 5. The confusion matrix for the two methods of classification

	Naïve Bayesian Classifier		**Neuronal Network Classifier**	
	a	**b**	**a**	**b**
a = 0	14	6	11	9
b = 1	0	63	0	63

The confusion matrix reveals that for the Naïve Bayesian classifier, the number of incorrect events of class ***a***, classified as class ***b*** was 6 and no event of class ***b*** was classified incorrectly. In the neuronal network classifier, 9 events of class ***a*** were classified incorrectly as being of class ***b***.

In the table presented below we can find the correlation matrix between the analyzed characteristics, identifying a correlation between the login time and the IP feature.

Table 6. Characteristics correlation matrix

	LOGIN TIME	**IP**	**FREQUENCY**
LOGIN TIME	1	0.47	0.35
IP	0.47	1	0.32
FREQUENCY	0.35	0.32	1

The following figure depicts the classification of events using the tree decision technique based on the same data sample.

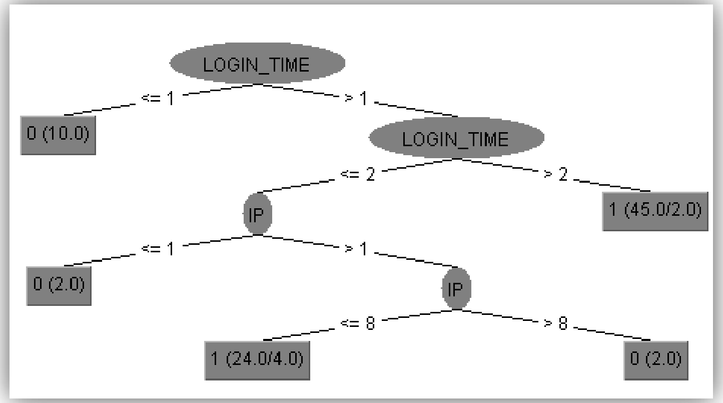


Fig. 7. The classification using tree decision technique

The tree decision technique shows how each authentication event triggered in the system can be classified based on the characteristics that describe it. It can also be observed that the frequency characteristic having the poorer correlation between it and the other two, is not taken into account at the classification moment using this technique.

These results are reflecting the status of one single account of which behavior was not fully mapped to a frequently and daily use. When valuable information will be recorded about a significant and representative set of individuals the results are going to improve, as they will simulate much better a homogenous behavior, revealing patterns which in this circumstances were neglected.

5 Conclusions

Assessing security for web based distributed applications implies the evaluation of each security control in relation with the advantages and disadvantages brought to the distributed system and as well to its users. The adaptive authentication control is meant to improve the overall system's security and not influence the normal flow of events which a user must take, for him to achieve its goals. Due to its dynamic contribution to the classic authentication process of a web based distributed application using username and password, the AMAP control optimizes security with no negative side effects on the usability level of the application.

Acknowledgements

This article is a result of the project „Doctoral Programme and PhD Students in the education research and innovation triangle". This project is co funded by European Social Fund through The Sectorial Operational Program for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies (project no. 7832, "Doctoral Programme and PhD Students in

the education research and innovation triangle, DOCECI”).

References

[1] P. A. Flach, and N. Lachiche, “Naïve Bayesian Classification of Structured Data”, *Machine Learning*, Vol. 57, No. 3, 2004, pp. 233-269.

[2] A. Joseph, D.B.L. Bong, and D.A.A. Mat, “Application of Neural Network in User Authentication for Smart Home System”, *International Journal of Intelligent Systems and Technologies*, Vol. 4, No. 1, 2009, pp. 10-16.

[3] I. Ivan, M. Doinea, and D. Palaghiţă, “Aspects Concerning the Optimization of Authentication Process for Distributed Applications”, *Theoretical and Applied Economics*, No. 6, 2008, pp. 39 - 56.

[4] M. Doinea, “Authentication process optimization for web based distributed applications” in *Proceedings of the Tenth International Conference on Informatics in Economy*, IE 2011, May 5-7, 2011, Bucharest, Romania, ASE Printing House.

[5] Y. Chen, and D. Liginlal, „*Bayesian Networks for Knowledge-Based Authentication*”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 5, 2007, pp. 695-710.

[6] M. Doinea, “Security optimization of a distributed application with free access to resources”, *Journal of Applied Business Information Systems*, Vol. 1, No. 1, 2010, pp. 55-66.

[7] I. Smeureanu, and M. Zurini, „Spam Filtering for Optimization in Internet Promotions using Bayesian Analysis”, *Journal of Applied Quantitative Methods*, Vol. 5, Nr. 2, 2010, pp. 199-211 .

[8] J. N. Sulzmann1, J. Furnkranz1, and E. Hullermeier, “On Pairwise Naive Bayes Classifiers” in *Proceeding ECML '07 Proceedings of the 18th European conference on Machine Learning*, Berlin, 2007.

[9] I. Pop, “An approach of the Naive Bayes classifier for the document classification”, *General Mathematics*, Vol. 14, No. 14, pp. 135-138.

[10] H. H. Orkcu, and H. Bal, „Comparing performances of backpropagation and genetic algorithms in the data classification”, *Journal Expert Systems with Applications: An International Journal*, Vol. 38, No. 4, 2011, pp. 3703-3709.

[11] M. Doinea, and M. Zurini, “Bluespam Filtering”, *Economy Informatics,* Vol. 10, No. 1, 2010.

[12] S. Arlot, and A. Celisse, “A survey of cross-validation procedures for model selection”, *Statistics Surveys*, Vol. 4, 2010, pp. 40-79.

[13] J. D. Rodriquez, A. Perez, and J.A. Lozano, „Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation”, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 3, 2010, pp. 569-575.

Mihai DOINEA is a PhD candidate in the field of Economic Informatics, within Academy of Economic Studies, Bucharest, Romania. His PhD thesis approaches the field of Informatics Security, with clear objectives about finding security optimization methods in distributed applications. His research is also backed up by a master diploma in Informatics Security (2006). He is a lecturer assistant, teaching Data Structures and Advanced Programming Languages at the Academy of Economic Studies. He published more than 30 articles in collaboration or as single author and his research interests are directed to areas such as security, distributed applications, artificial intelligence and optimization algorithms.