Trust Evaluation for Web Applications based on Behavioral Analysis

Luiz Fernando Rust C. Carmo, Breno G. de Oliveira and Augusto C. Braga
Computer Center (NCE) – Federal University of Rio de Janeiro (UFRJ)
Caixa Postal 2324 – 20.010-974 – Rio de Janeiro – RJ – Brasil
{rust,breno,augustocesar}@nce.ufrj.br

Abstract. This paper deals with a joint use of a trust evaluation approach and access control mechanisms for improving security in Web-usage. Trust evaluation is achieved by means of both behavioral evaluation and credentials exchange, in such way that transitions among different access policies are automatically fired whenever a user behavior is validated. Behavioral analysis uses machine-learning techniques to gain knowledge about users navigation tracks, creating a user signature to be compared with a current behavior of the respective user. This mechanism is validated through experimental evaluation.

1 Introduction

Nowadays there is a huge concern for Web application security. The public key infrastructure (PKI) is definitely a great ally in solving many security issues, especially those related to authentication. The difference between authentication and authorization is discussed in [1]: (i) an authentication service proves that the identity of an object/subject is in fact the one it claims to have; while (ii) authorization means the grant of permission based on the authenticated identification. This latter definition can be altered with the introduction of the “trust” concept: usually an entity can say that it “trusts” another one whenever it assumes that the second entity will behave exactly as it expects. This way authorization can be rewritten as the “grant of permission based on the deposited trust”. The fact is that, maybe more important than knowing who you are relating to, is knowing how this person/object will behave. On the other hand, a fraudless authentication is still the best way to foresee an announced behavior profile, and the combination of authentication with continuous behavioral analysis enables a gradual verification of the coherence between both factors, generating a trust consolidation.

It has been seen recently a sensitive increase in proposals to incorporate trust based mechanisms in Web applications, Web services and ubiquitous computing. Most of those mechanisms propose the use of digital credentials for an effective management in the establishment of trust. The goal is to, after an initial authentication stage, manage the user's demands and increase the level of trust through the exchange of credentials in predetermined moments. In order to do that, trust management is linked to an access control mechanism in a granular way (i.e. RBAC [1]) so the evolution of the trust relationship implicates an alteration of the access privileges given to this user.

A classic example of this kind of approach is the relationship between a user and a shopping *website* [2]: after the authentication, the user (i) browses the *site* and selects a given object to buy; at this time there is a need of more access privileges to (ii) consolidate the payment; this transition is controlled by a credit card number (credential example) that, in case of success, implies a trust level increase and the attribution of a new “role” in the access policy, providing the necessary privileges for the (iii) continuity of the operation.

This paper basically proposes a trust evaluation strategy guided not only by the gradual disclosure of credentials, but also by a behavioral evaluation mechanism based on the continuous analysis of the current conduct of a user vis-à-vis with its past activities, generating the grounds for a possible evolution in the trust level. In the previous shopping *website* example, if the user is an identified client, it is possible to gradually establish a certain signature for his browsing habits in a way that the browsing trajectory is analyzed against this signature in each new operation, promoting an automatic increase in the trust level, without the need for an exchange of credentials.

The basic idea is to develop a continuous user behavioral evaluation system, where trust and access restrictions can be inflicted automatically in Web applications. However, the mechanism is not restricted to Web applications, and the ubiquitous computing seems to enforce even more the role of trust, not only for the inherently high decentralization, but also for the propagated non-intrusive *modus operandi*. In this kind of application, trust would completely replace traditional authentication methods, supporting the concept of free users (without certificates, logins or passwords).

Web services are also gaining increased importance as technologies enabling the development of service oriented distributed applications. And, along with the crescent number of services, specially in corporate networks, there is also a growth in the complexity required to authenticate and administrate user privileges, creating a favorable environment for the use of the trust concept [3].

2 Related Work

Approaches bringing together access control policy models and trust managers are relatively new and seek the establishment of trust in a gradual and interactive way, to dynamically update access privileges [4].

In [2] it is proposed a trust negotiation *framework* (*Trust-Serv*) for Web services. The trust relationship evolves through the exchange of credentials controlled by a state machine associated with the application. Some examples of credentials are credit cards, passports and membership documents. The credential attributes are evaluated in order to enable, or not, a state movement and the corresponding alteration in the user's access profile. [5] proposes an adaptive access control and trust negotiation *framework* that combines an access control and authorization API and a trust manager (*TrustBuilder*), in order to control when, and how, sensitive information can be revealed. This proposal is based on a reactive analysis in face of a failure event, i.e, wrong credentials, implicates a greater suspicion level on a user, which in turn implicates in access privilege restrictions.

Both previous works use credentials to inflict security: the difference is that the former is proactive — increasing trust upon success, and the latter is reactive — increasing the suspicion (lowering trust) upon failure. This article's approach does not discard the use of credentials, but suggests a joint use of behavioral analysis for trust evolution.

[3] proposes a trust evaluation mechanism in Web services based in behavioral analysis through continuous tracking of a user. The goal is to provide a self-manageable mechanism for access control in an environment of Web services federations. The trust level is exponentially modeled as a function of the services required by the user: increases for certain classes of expected services and decreases for others.

One of the main differences between this approach and the proposed work is in the way that the user's behavior is evaluated. Basically, previous proposal start from an initial mapping of services/functions available in the provider in predetermined paths that, if followed, enables one to increase the trust on the user. Our proposal evaluates the user's behavior in function of his past (usage *logs*) with the purpose of increasing the trust on a user's identity (if he is really who he claims to be), using a learning based behavioral analysis technique.

Security systems based on behavioral analysis by learning can be classified in two categories according to the kind of "behavior" studied: (i) *Physical behavior* – attempts to learn some personal characteristic of a user, i.e: patterns in keyboard typing or mouse use; and (ii) *contextual behavior* – attempts to learn a user's service utilization profile, i.e: UNIX commands, Web navigation, etc. The first category is strongly related to complementary authentication mechanisms, while the latter is disseminated in the Intrusion Detection domain.

A heavily explored approach in literature, in the "physical behavior" category, consists of generating a signature from the individual dynamics of keyboard use [6, 7, 8]. Basically, this method does not use the information being typed, but the rhythm in which it is typed — time gap between two key strokes and duration of a key stroke. Collected data are modeled in fixed length arrays and, for each new authentication, a new array is created and compared to the initial one, generating a similarity index. This method offers as its main disadvantage the current tendency for the obsolescence of text-based interfaces over mouse-driven ones.

[9] proposes a re-authentication mechanism based on mouse movements. This mechanism captures mouse information (instant position, click, double click, etc) and, after creating a regular behavior model, uses a decision tree classifier to validate the current behavior and re-authenticate the user.

An example of contextual behavioral analysis approach is the work of [10] for anomaly detection, comprising both intrusion detection and the identification of hostile behavior of authenticated users. The focus of this work is the analysis of command lines against the history of commands issued by the user, through *Instance-based Learning* (IBL) [11]. Unlike the proposal of this article, there is no concern in characterizing a user's individual behavior, but only in classify it as “normal”.

Another kind of user's behavioral analysis approach, directly related to Web applications, is directed to the customization of a website's navigability. [12] propose the use of data mining techniques on *Web logs*, in order to inflict upon different user's access profiles and, automatically, adapt a *website*'s navigation options.

In conclusion, the innovative character of the approach presented in this work is supported by two main factors:

1. the use of a learning based behavioral evaluation mechanism to offer grounds for the evolution of trust;
2. proposal of a behavioral evaluation mechanism based on Web navigation path analysis, superimposed to a historical contextual signature.

3 Trust Evaluation

The concept of trust used in this work can be informally defined as a measurement of how sure the application provider is about the identity of a user and, consequently, of the way in which the user will behave.

In [2] the evolution of trust is controlled by a finite state machine (*Trust-Serv* model) previously specified, where the states represent the current level of trust in a relationship. Each state is associated to a specific access policy (i.e.: a role in a RBAC model), while the state transition is controlled by exchange of credentials, predictions/obligations (services that need to be executed first) and *timeouts*. Figure 1 describes a small excerpt of a state machine example used in the trust negotiation in *Trust-Serv*. This machine has two states, Client and Reviewer, each with its own required level of trust and controlled by a specific access policy (A & B). The transition between these two states is safeguarded by the exchange of credentials address and credit card.

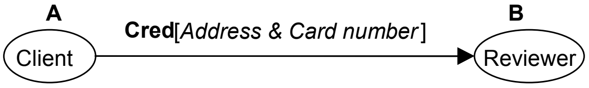

Fig. 1: Trust negotiation model

The concept of trust level/access policy is captured by the definition of a macro-state, while states represent the pages of a web application. A change from one macro-state to another (implying a change in the access policy) is performed automatically by the result of a behavioral evaluation or, in case of insuccess, by an explicit negotiation via exchange of credentials (figure 2).

Definition 3.1. A trust evaluation scenery C is defined by the 6-tuple:

$(MacroStates^C, States^C, Transitions^C, Profiles^C, \varphi^C, \omega^C)$

- *$MacroStates^C$* is the set of macro-states of *C*, where each *macro-state* M is a subset of *$States^C$*
- *$States^C$* is the set of states of *C* (pages)
- *$Transitions^C$* is the set of transitions of *C*
- *Profiles* is the set of access profiles (roles) associated to *C*
- φ^C transition attribution function, that associates every transition to a state of origin and to a states of destination

  φ^C*: Transitions* → *$States^C$ x $States^C$*
- ω^C is the access profile attribution function, that associates each profile to a set of macro states

  ω^C*: $Profiles^C$ → Set of the parts of $Macro\text{-}States^C$*

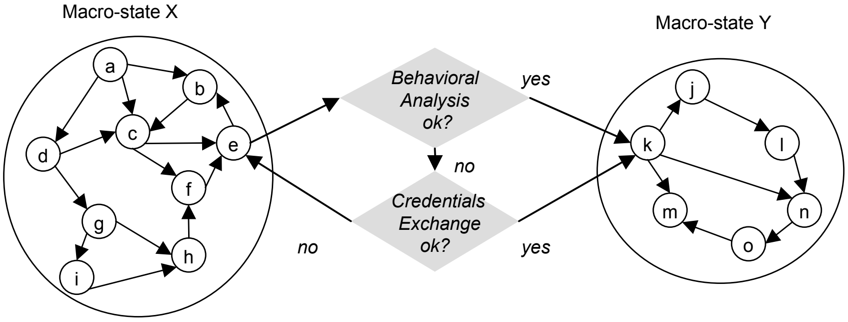

Fig.2: Macro-states x Trust evaluation

Definition 3.2. The specific transitions between the macro states may be captured by the set denominated:

$$MacroTransitions^C = \{t \in Transitions^C \mid \varphi(t) = (a,b) \wedge (a \in m_1,\ b \in m_2 : m_1 \neq m_2)\}$$

Definition 3.3. Let V^C be the subset of *$MacroTransitions^C$* experimented during the user's section in a *C* scenario, let the *BehaviouralTrust* and the *Credential* be function with V^C domain and boolean range, the user's section is set according to the evaluation strategy proposed if the following condition is satisfied :

$$\forall t \in V^C : BehavioralTrust\ (t) \vee Credential(t) = true$$

It is important to stand out that there is a dependence relationship between the result of the evaluation of the behavioral trust and the execution of the evaluation through credentials; the second one goes only in the case of a negative evaluation of the first (figure 2). This kind of approach, practically establishing the credentials switch as a redundancy, has significant impacts in the requirements imposed to the behavioral analysis' mechanisms. The concern with false negatives almost disappears since the non-identification of a user's behavior doesn't cause any degradation in the section in course, it only leads to the solving of the redundant step of credentials' exchange.

On the other hand, a single use of behavioral evaluation would make possible to find out the following anomalous conducts: (i) a system's authenticated user that makes a legitimate use to abuse of the system's resources, (ii) the sporadic use by a colleague at work "that asks to borrow" a workstation, (iii) an automated attack launched by a relatively naive user through a typical sequence of attacks. With the combined use of a credentials' exchange mechanism, the condition (i) naturally looses its' effectiveness, since it is very probable that an authenticated user be also successful in an exchange of credentials.

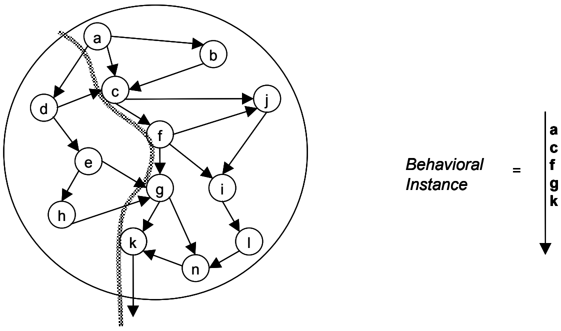

Fig. 3: Example of behavioral instance

4 Behavioral evaluation

This section analyzes the problem of behavioral evaluation through learning, in a way that characterizes and differentiates each individual/system's behavior in terms of a discrete data sequence. The characterization of a user's typical behavior is a great challenge, because, besides the inherent variability, there is a change in the regular used pattern as a natural consequence of the user's absorption of new knowledge. The use of a learning machine allows training a classifier with a user's historical data, so that it is possible to distinguish different behaviors, considering both variability and the evolutive aspect. In this section we examine methods to collect a user's behavioral signature based on learning, and the appropriate definition of similarity according to the required context.

4.1 Behavioral Signature Collecting

Many of the traditional learning approaches is not suitable for behavioral differentiation due to the class of data being processed, i.e. discrete elements with nominal values. Neural networks [13] have proved to be useful to continuous series of numerical values, typically using *Euclidean Distance* for similarity computation, but there is a major limitation for using it in behavioral differentiation: the necessity of retraining for every new user [6]. A very popular and generic class of learning machine techniques is the *Instance-based Learning* (IBL). In this model, a concept is represented implicitly by a set of instances that exemplify it (dictionary). In our situation, it is possible to directly apply a very simple method of the IBL learning model, in which every behavioral instance is directly classified according to the generating user. This way, the behavioral signature is represented by a set of a specific user's behavioral instances (figure 3), generated in every macro-transition.

Definition 4.1. Let I be a set of indexes; the behavioral instance concept *ic* and the behavioral signature *ac* may be defined as:

$ic = \{e_i \in States^C \mid i \in I\}$

$ac = \{ic_i \mid i \in I\}$

4.2 Behavioral Trust Measure

The degree of similarity S is a function of two behavioral instances which expresses a measure of how alike those behaviors are. We examined several measures for computing the similarity between two discrete-valued temporal instances. Here we describe the measure that we found performs the best on average in empirical evaluations. Basically, this procedure must pinpoints pairs of identical elements and uses a cumulative calculation to give a bigger weight to identical enchained pairs (isentical subsequences). A fundamental requirement for this calculation is no restriction about different sizes between *ic* sequences (i.e.: {a,b,c,d} & {a,g,d}).

Definition 4.2. Let *a* and *b* be two behavioral instances; a preprocessing procedure is captured by the function:

$$\tau(a,b) = (\{z_0, z_1, \ldots z_{m-1}\},\{w_0, w_1, \ldots w_{n-1}\})$$

Where $\{z_0, z_1, \ldots z_{m-1}\}$ is the set of the length of identical subsequences between a and b; $\{w_0, w_1, \ldots w_{n-1}\}$ is the set of the length of different subsequences between a and b; and the behavior of the function τ is expressed by the pseudocode of fig.4.

```
τ(a,b)
Let $p_i$ be any position of an array $i$
Let $w$, $z$ be arrays of variable length
Let $p_a$, $p_b$, $p_w$, $p_z$ equals the first position of
   $a$, $b$, $w$, $z$ respectively
For each $p_a$ in {$p_a$ until last position of $a$}, do
   If $p_a$ equals $p_b$ , then
      If $p_w > 0$, then
         Advances $p_w$ to the next position of $w$
      Add 1 to $p_z$
   Else, then
      If $p_z > 0$, then
         Advances $p_z$ to the next position of $z$
      Let $p_t$ equals the first position of $b$
      Let found equals FALSE
      For each $p_t$ in {$p_t$ until last position of $b$},
      do
         If $p_t$ equals $p_a$, then
            Let $p_b$ equals $p_t$
            If $p_w > 0$, then
               Advances $p_w$ to the next position of $w$
            Let found = TRUE
            Add 1 to $p_z$
            Leave the inner loop
      If found equals TRUE
         Add 1 to $p_w$
   Advances $p_b$ to the next position of array $b$
Return arrays $z$ e $w$
```

Fig. 4: Pseudocode of τ(a,b) function

For example: considering an *ic* pair *{a,g,b,c,d} and {a,b,c,d},* the application of the preprocessing function returns: *(i)* the set {1,3} meaning two identical subsequences: one of 1 element (a) and other of 3 elements (b,c,d); and (ii) the set {1} meaning one different subsequence of 1 element (g).

Definition 4.2. Let $(X,Y) = \tau(a,b)$, the similarity degree *S* between $a = (a_0, a_1, \ldots a_{m-1})$ and $b = (b_0, b_1, \ldots b_{m-1})$ is given by the following trio of functions:

$$\delta(X,Y) = Sum(X) - Sum(Y) \;\mid\; Sum(c_0, c_1, \ldots c_{m-1}) = \sum_{i=0}^{m-1} 2c_i - 1 \qquad (1)$$

$$S'(a,b) = \begin{cases} \dfrac{\delta(\tau(a,b))}{\delta(\tau(a,a))} & if \quad card(a) \geq card(b) \\ \dfrac{\delta(\tau(b,a))}{\delta(\tau(b,b))} & if \quad card(a) < card(b) \end{cases} \tag{2}$$

$$S(a,b) = \frac{S'(a,b)\ +1}{2} \tag{3}$$

The advantage of this type of calculation lies on the possibility to adjust the importance given to sequential states by just exchanging the *Sum* function. Figure 5 shows a comparison between the similarity values obtained from the set of behavioral instances pairs from table 1 for both the linear equation given in (1) and the exponential one defined in (4). It is clear that the linear formula is more sensitive to small differences among sequences, which is the behavior we were looking for.

$$Sum(c_0,\ c_1,\ \dots\ c_{m-1}) = \sum_{i=0}^{m-1} 3^{c_i - 1} \tag{4}$$

Table 1. Similarity for linear and exponential function Sum

a	*a g b c d*	*a g b c d*	*a g a b d d*	*a b c d e f g h*	*a a b a b c d*
b	*a b c d*	*g a b c d*	*a b c d*	*d e f g h*	*a b c d*
Linear	*0.7775*	*0.8885*	*0.7272*	*0.6330*	*0.9230*
Exponential	*0.5555*	*0.5679*	*0.5102*	*0.5164*	*0.5212*

Using *S* (similarity) is possible to calculate three independent factors that need to be considered in trust : *Comparative similarity*, *Intra-similarity* and *Inter-similarity*.

Comparative Similarity (*Scomp*) – represents the similitude between the current collected instance and the set of instances that form the signature. Essentially, its value mirrors how close this behavior is from the previously captured ones. To perform this calculation, a *similarity* function is applied between the current behavioral instance and every instance that form the signature, retaining the maximum value obtained:

$$Scomp^M = \max\{S(ic^M_{cur}, ic^M_i), \forall ic^M_i \in ac^M\} \tag{5}$$

where ic_{cur} denotes the current computational instance, and ac^M denotes the behavioral signature of the macro-transition M.

Intra-Similarity (Sintra) – is related to the quality of the user's signature, being completely independent of the current behavioral instance sample. This represents if a user has a well-formed behavior (when signature instances are repeated, or slightly different), or the opposite (when all signature instances are very different amongst themselves). Naturally, a bad-formed signature makes the user's behavioral validation process difficult. To calculate *Sintra*, we calculate the mean among every resulting values of the similarity function between all 2 to 2 arrangements of the signature instances:

$$Sintra^M = \frac{\displaystyle\sum_{\forall ic^M_n \in ac^M} \sum_{\forall ic^M_m \in ac^M} S(ic^M_m, ic^M_n)}{A_{card(ac^M),2}} \tag{6}$$

Inter-Similarity (Sinter) – represents the quality of a user signature in function of the complete set of signatures (from different users) associated to the same macro-transition. A given behavioral instance can be extremely similar to a well-formed signature, and even so, not be trusted due to a possible similarity with other existent signatures (from several users from the same scenery). Signature similarity makes the user differentiation process difficult. *Sinter* reflects the similarity between a given signature and the signature most “alike” from the full set of signatures, and is expressed by the following pair of functions:

$$\Phi(ac^M, ac_i^M) = \frac{\sum_{\forall ic_n^M \in ac^M} \max\{S(ic_n^M, ic_m^M), \forall ic_m^M \in ac_i^M\}}{card(ac^M)} \qquad (7)$$

$$Sinter(ac^M) = 1 - \max\{\Phi(ac^M, ac_i^M), \forall ac_i^M \in U^M - \{ac^M\}\} \qquad (8)$$

where U^M denotes the full set of signatures associated to M from a scenario C.

Finally, trust calculation (*Trust*) is expressed by the product of these three factors:

$$Trust^M = Scomp^M * Sintra^M * Sinter^M \qquad (9)$$

Given that *trust* can be quantified, all there is left is to establish a minimal acceptance level *TrustRef* to evaluate the function *BehavioralTrust* (definition 3.3):

$$BehavioralTrust(M) = \begin{cases} True & if \quad Trust^M \geq TrustRef \\ False & if \quad Trust^M < TrustRef \end{cases} \qquad (10)$$

5 Experimental evaluation

The performance of a mechanism such as this is strongly influenced by the kind of application and the variety of intrinsic behaviors. Empirical analysis, performed on concrete usage examples as test environments, have been largely used for testing purposes in similar proposals, as [9, 13]. The major problem of this kind of approach is the danger of selecting an extremely inappropriate environment, generating a possible false negative evaluation of the mechanism; or, on the other hand, an extremely appropriate one, which would also lead to a false conclusion of a questionable effectiveness, certainly unadvisable to be generalized. We chose not to develop a perfectly suitable website model for the simulation, but rather to perceive how the mechanism would behave in different environments that were not initially devised to support macrostates or trust evaluation systems.

The first step was to find ways to adjust the requirements of the trust evaluation model in a regular website log. The following items had to be assessed consistently: (i) States and Macrostates definition and (ii) Users and user's behavioral instances

Each possible state *e* of the website was defined as a HTML or PHP web page, logged via a HTTP GET requests. Other requests, mostly for images and stylesheets, were discarded. As no user authentication was provided, an user *u* was defined as any known static IP address. Also, since no macrostate boundaries were available, every state *e* was defined as part of the same macrostate *M* and the following conventions were adopted: (i) behavioral instances with less than 5 states would not be considered; and (ii) a behavioral instance is said to have ended when more than 30 minutes have passed from the last state included in the instance until the next state entered by the same user. Those rules suffice the need to distinguish between behavioral instances of the same user and prevent the model from analyzing instances too small to be meaningful.

We have experimented with our approach on logs collected at the Computing Center Department, Federal University of Rio de Janeiro. We examined the Web-server logs of different applications through the trust model perspective to see what kind and quality of usage patterns were available.

The first website collected (UFRJ virtual library - www.bibvirtuais.ufrj.br) was not appropriate for not showing the actual IP addressess of the machines that made the HTTP requests. The second one (UFRJ Architecture and Urbanism school - www.fau.ufrj.br), although well structured (for macrostate adaptation), has a poor navigation diversity to generate different signatures (very low *Sint*er value). The last website (UFRJ Libraries and Information System - www.sibi.ufrj.br) showed none of the above problems and therefore was chosen for the model evaluation. We collected access logs from march 31st to september 6th 2006 (160 days), and then devised a parser to (i) anonymize all entries, replacing IP addresses and webpages by index numbers; (ii) remove all but those IP addresses known to be unique to a single user's computer; (iii) remove any sequential state repetitions (page reloads), as the model does not predict state transitions to itself; (iv) ignore requests to non-existent webpages, considering only the ones with a server return status of 200 (OK) or 304 (Not Modified); (v) ignore instances with less than 5 states; and (vi) ignore signatures with less than 5 instances, the empirical minimum value established for the evaluation. The result was 42 valid behavioral signatures files ready to be tested by the model prototype. The *SIntra* values (figure 5) give a clear understanding of whether the signatures are good or not, for example: user 34 has little difference between its internal behavioral instances, while user 38 had most of its instances remarkably differentiated among them.

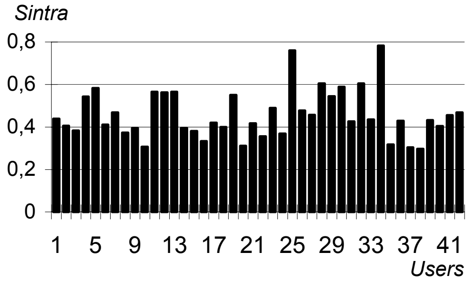

Fig.5: *Sintra* measures

Trust Ref	***Users Accepted***	***False Pos***	***False Neg***
0.070	42 (100%)	39 (93%)	0 (0%)
0.120	29 (69%)	3 (7%)	13 (31%)
0.150	16 (38%)	1 (2%)	26 (62%)

Table 2. *TrustRef* Evaluation

To simulate the user's input, we retrieved a random behavioral instance from inside each user's own signature database, leaving it with *n – 1* instances, with *n* being the original number of instances inside the signature. The simulation evaluated every user's input against each signature, and calculated the Trust value for each case. The purpose was to see whether the highest Trust value was indeed from the same user as the signature in question.

Figure 6 shows a comparison between the mean values of the actual *Trust* evaluation obtained for each signature against the best false positive result for that signature (i.e. input from a different user that achieved the best Trust result against that signature). It is relevant to notice that, of all simulations performed with those 42 signatures, absolutely none held a user whose Trust evaluation was higher than the actual owner of the signature being tested. Another pertinent issue lies on the significantly low Trust values obtained, mostly because the experiment had rather low Sinter values, with a mean value of 0.365, indicating little difference between signatures. Even so, it was still possible to differentiate behaviors from the Trust results and establish possible positions for a TrustRef mark to be set, considering the desired amount of false positives and false negatives, as shown in Table 2.

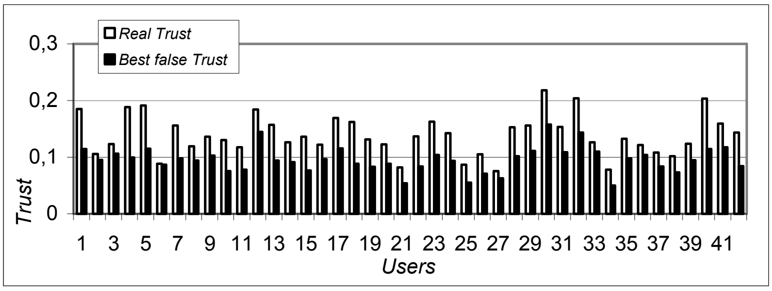

Fig. 6: Real Trust x Best false Trust

5 Conclusions and Future Work

This paper described a proposal for an integrated use of the concept of trust and access control management in secure Web applications. The originality of the approach lies on the employ of a user’s behavioral evaluation mechanism (via Web navigation track) through a learning machine. The result of this analysis is used in the trust evolution process to replace, or complement, the classic use of mechanisms for credentials exchange. An important contribution of this work is the similarity measure between two representative samples of a user's behavior that, unlike the usual, compares behavioral sequences of different lengths.

It is also noticeable the extent of the proposed heuristics in the calculation of the trust level of a behavioral instance, which takes into account three different factors: (i) comparative similarity – relationship between the current behavioral instance and the signature (behavior resemblance), (ii) intra-similarity – relationship between behavioral instances that form the signature (quality of the signature) and (ii) inter-similarity – relationship between the different existing signatures (signature differentiability).

The performance of the proposed mechanism is well characterized by experimental evaluation that, besides attesting the viability of its utilization in the behavioral differentiation context, give some important subsides for the establishment of a minimal trust level *TrustRef.*

An open question in the proposed approach, and subject of ongoing works, concerns the necessity of the use of data reduction techniques, once the signatures store all of a user's past behavioral instances. For certain applications, that allow a great variability of behaviors, the signature size may grow considerably. It is under study the viability of the replacement of a set of similar behavioral instances for generic models that capture a certain degree of variability. Another work in progress tries to characterize a timetable of the behavior of a certain user, allowing the removal of old behaviors from his signature that should not reoccur.

6 References

1. J. Lopez, R. Oppliger and G. Pernul, Authentication and authorization infrastructures (AAIs): a comparative survey, *Computers & Security*, 23 - 2004, Elsevier, pp. 578-590.
2. H. Skogsrud, B. Benatallah and F. Casati, Model-Driven Trust Negotiation for Web Services, *IEEE Internet Computing*, 1089-7801/03, Nov/Dec 2003, pp. 45-52.
3. C. Platzer, Trust-based Security in Web Services, Master‘s Thesis, Information Systems Institute, Technical University of Vienna, Austria, 2004.
4. J. Bacon, K. Moody and W. Yao, Access Control and Trust in The Use of Widely Distributed Services, *Software-Practice Experience*, 33, 2003, pp. 375–394.
5. R. Tatyana, L. Zhou, C. Neuman, T. Leithead and K.E. Seamons, Adaptive trust negotiation and access control, In tenth ACM symposium on Access control models and technologies, ACM Press, Stockholm, Sweden, 2005.
6. F. Monrose and A. Rubin, Authentication via Keystroke Dynamics, In Fourth ACM Conference on Computer and Communication Security - CCS 97, Zurich, Switzerland, 1997, pp. 48-56,
7. A. Guven, and I. Sogukpinar, Understanding Users' Keystroke Patterns for Computer Access Security, *Computers & Security*, Elsevier, Vol. 22-8, 2003, pp. 695-706.
8. A. Peacock, X. Ke and M. Wilkerson, Typing Patterns: A Key to User Identification, *IEEE Security & Privacy*, September/October, 2004, pp. 40-47.
9. M. Pusara and C.E. Brodley, (2004). "User Re-Authentication via Mouse Movements, In CCS Workshop on Visualization and Data Mining for Computer Security -VizSEC/DMSEC'04*,* ACM press, Washington, DC, USA, October, 2004.
10. T. Lane, and C. Brodley, Temporal Sequence Learning and Data Reduction for Anomaly Detection, *ACM Transactions on Information and System Security*, Vol. 2, No. 3, August, 1999, pp. 295–331.
11. D.W. Aha, D. Kibler and M.K Albert, Instance-based learning algorithms", *Machine Learning*, Kluwer Academic Publishers, Vol. 6, No 1, January, 1991, pp. 37–66.
12. M. El-Ramly and S. Stroulia, Analysis of Web-usage behavior for focused Web sites: a case study", *Journal of Software Maintenance and Evolution: Research and Practice*, No. 16, 2004, pp. 129–150.
13. T. Lane, "Machine learning techniques for the computer security". Ph.D. thesis, Purdue University, 2000.