(54) Title: PERMISSION LEVEL GENERATION BASED ON ADAPTIVE LEARNING

(57) Abstract: The disclosure teaches a system where user access is controlled. The system comprises an access analyzer subsystem and a user control subsystem. The access analyzer subsystem further includes a history machine capable of processing new and historical data related to access attempts, a history database for storing the new and historical data, a hierarchy generator connected to said history machine for generating a user hierarchy related to the historical data, and a permission level generator connected to the history machine and the hierarchy generator. The permission level generator is adapted to generate permission levels for accessing the user subsystem based on the hierarchy and user’s access profile. The user control subsystem further includes an agent adapted to provide an event audit trail to the access analyzer subsystem and receive a permission level from said access analyzer subsystem; and a control unit adapted to provide a security policy to the access analyzer subsystem and receive statistical information and reports from said access analyzer and agent subsystem respectively.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
Permission Level Generation Based on Adaptive Learning

I. Description

I.A. Claim for Priority

This application claims priority from U.S. Provisional Patent Application serial No. 60/226,128 filed on August 18, 2000, incorporated in its entirety by reference herein. This application also claims priority from U.S. Provisional Patent Application serial No. 60/259,575 filed on January 4, 2001, incorporated in its entirety by reference herein.

I.B. Field

This disclosure teaches techniques for automatic generation of permission levels. More specifically, the teachings include, but are not limited to, systems, methods and computer program products for generating permission levels using adaptive learning algorithms.

I.C. Background and Related Art

The following documents provide background for a better understanding of the disclosed teaching, and to that extent, they are incorporated herein by reference.

1. US Patents

4,697,242 Sep 198 Holland et al.
5,115,967 May 1992 Wedekind
5,204,961 Apr 1993 Barlow
5,787,234 Jul 1998 Molloy
5,991,879  Nov 1999  Still
5,996,077  Nov 1999  Williams
6,038,556  Mar 2000  Hutchison
6,082,835  Jul 2000  Brearley
6,098,173  Aug 2000  Elgessy et al
6,134,532  Oct 2000  Lazarus et al.
6,158,010  Dec 2000  Moriconi et al
6,163,383  Dec 2000  Ota et al
6,164,975  Dec 2000  Weingarden et al.
6,167,520  Dec 2000  Touboul
6,186,794  Feb 2001  Brown et al.
6,202,157  Mar 2001  Brownlie et al
6,206,700  Mar 2001  Brown et al.

Adaptive learning is used in several areas and in a wide range of applications. The term adaptive learning could relate to several different domains. It may include areas such as artificial intelligence (AI) on the one hand, and analog integrator circuits on the other. Adaptive learning is widely used in expert systems. Capabilities of adaptive learning could also be used, for example, in vehicle breaking systems, thermostats, teaching systems, and many others.

Adaptive learning generally relies on the capability of accumulating historic information or data over time. This historic information or data is coupled with data relating to the present time to determine a response, or output, that forms the basis for establishing and predicting the current activity of the system.
An adaptive learning system for a thermostat, for example, may "learn" the temperature curve most suitable for the users of a room. This system will then adapt, over time, to respond to various changes that occur during the span of a day, a week, a year and so on. Continued use of such an adaptive learning system can result in significant cost savings as well as overall improvement in system performance.

In certain cases, real-time performance is required from an adaptive learning system. While, in other cases, a delayed response is permissible. For example, a real-time "expert system" is expected to provide an accurate response within a very limited time frame, so that the user does not have to wait a significant amount of time. The same type of constraints would apply for receiving a response in a breaking system for a vehicle. On the other hand, a thermostat could use relatively substantial time to "learn" and respond over time without adversely affecting the performance of the system.

Learning systems and adaptive learning systems are implemented in different ways. Using logic and "IF-THEN" rules is one way of implementing such systems. In some other cases an approach using neural networks is preferred. In yet other cases, a "case by case" method, or "query" based solutions are preferred. Regardless of the method used, the overall function of a learning system and an adaptive learning system, as the titles indicate, is to learn. By such learning, the system becomes capable of generating a next response over time, based on the currently received input and prior history.

The primary purpose of security systems is to prevent unauthorized use of a system's resources. In such security systems, and specifically network security systems, a security policy is usually implemented to prevent unauthorized use of
system resources. Security systems may be implemented by using a "firewall" that prevents unauthorized access by users external to the system. Security systems may also be implemented by a security policy internal to the system organization.

In modern network systems, very often the internal network may be virtual, otherwise known as a virtual private network, or VPN. In such cases the network, though actually physically spread out in various geographic areas, is still considered one network. A security policy to protect such a virtual network may also be necessary. However, conventional systems do not provide the capability of adaptive learning for securing a network system.

It would be advantageous to provide adaptive learning capabilities to such network systems. Specifically, more pronounced advantages can be realized in systems that have a large number of users and/or resources. This is at least because a security policy becomes increasingly difficult to create and maintain accurately and dynamically in such large systems.

II. Summary

To help realize the advantages mentioned above, the disclosed teachings provide a system where user access is controlled. The system comprises an access analyzer subsystem and a user control subsystem. The access analyzer subsystem further includes a history machine capable of processing new and historical data related to access attempts, a history database for storing the new and historical data, a hierarchy generator connected to said history machine for generating a user hierarchy related to the historical data, and a permission level generator connected to the history machine and the hierarchy generator. The
permission level generator is adapted to generate permissions for accessing the user subsystem based on the user hierarchy and the historical access data. The user control subsystem further includes an agent adapted to provide an event audit trail to the access analyzer subsystem and receive a permission level from said access analyzer subsystem; and a control unit adapted to provide a security policy to the access analyzer subsystem and receive a statistical information from said access analyzer subsystem.

Specifically, the agent is a guardian agent further adapted for at least allowing access to a resource in response to an access attempt, the allowing being based on at least said permission level.

Specifically, the agent is a guardian agent further adapted for at least denying access to a resource in response to an access attempt, the denying being based on at least said permission level.

Specifically, the agent is a guardian agent further adapted for at least sending an alarm to said control unit in response to an access attempt, the sending being based on at least said permission level.

Specifically, the control unit is further adapted for generating reports based on at least said statistical information.

Specifically, the control unit is further capable of generating reports based on at least the alarms.

Another aspect of the disclosed teachings is a system for tracking access attempts. The system comprises a history machine adapted to process new and historical data related to said events; a history database adapted to store the new and historical data; and a hierarchy generator connected to the history machine adapted to generate a hierarchy related to the new and historical data.
Specifically, the further includes a permission level generator connected to
the history machine and said hierarchy generator. The permission level
generator is adapted to generate permissions for accessing system resources
based at least on the hierarchy.

More specifically, the permission level generator comprises a weighted
hierarchy activity calculator connected to a permission level calculator.

More specifically, the weighted hierarchy activity calculator receives at
least a weighted audit trail and a user hierarchy.

More specifically, the permission level calculator generates permission
levels based at least on an input provided from said weighted hierarchy activity
calculator.

Still more specifically, the permission levels are based further on a
security policy.

Still more specifically, said weighted audit trail is determined for a user
accessing a resource in a give access type accounting for all accesses of the
same type in a decreasing value.

Still more specifically, a maximum value for said data is provided.

Still more specifically, said weighted audit trail is determined for a user not
accessing a resource in a give access type by reducing the current said data.

Still more specifically, said reduction is done as a multiplication of said
data by a fractional constant.

Specifically, the history machine generates a weighted audit trail based on
event audit trail and history information provided by the history database.

More specifically, a decay factor is applied to historical data prior to storing
in said history database.
Specifically, the user hierarchy generator further includes an integrator; an aggregator connected to the integrator and further connected to a clustering machine; a clustering machine connected to the aggregator; and a hierarchy accumulator connected to the clustering machine.

More specifically, the integrator receives a weighted audit trail.

More specifically, the integrator calculates a sum total of all accesses of a specific access type made by a user; the access being done to a specific resource.

More specifically, the aggregator calculates a sum total of all accesses of a specific access type made by all users in a cluster, the access being done to a specific resource.

Still more specifically, said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for said user access to said resource and the multiplication of all values relative to optionally normalized individual accesses of said user to said resource multiplied by said maximum value.

Still more specifically, the maximum value is a normalized maximum value

Still more specifically, said value relative to individual accesses is calculated as subtracting the value of said individual access from one. Still more specifically, the aggregator outputs an access information for every access type.

Even more specifically, the access information comprises rows and columns, the rows representing users and said columns representing resources, the access matrix storing data related to the access of a said users to said resources.

More specifically, the clustering machine outputs a preferred clustering assignment. Still more specifically, the clustering machine implements a pairwise
clustering algorithm. Even more specifically, the pairwise clustering algorithm is a greedy pairwise clustering algorithm.

More specifically, the clustering machine includes a similarity matrix generator; and a pairwise clustering generator. Still more specifically, the similarity matrix generator is adapted to determine a commonality between access profiles of a first user and a second user to a resource. The similarity matrix generator is further adapted to determine a first similarity between said first user and said second user, said first similarity being a result of dividing said commonality by the access attempts made by said first user. The similarity matrix generator is further adapted to determine a second similarity between said second user and said first, said second similarity being a result of dividing said commonality by the access attempts made by said second user.

Still more specifically, the similarity matrix generator is adapted to calculate an internal multiplication of user access profiles between a first user and a second user. The similarity matrix generator is further adapted to divide the internal multiplication result by a norm of user access profiles made by the first user multiplied by the norm of access attempts made by the second user.

Yet another aspect of the disclosed teachings is a method for determining a hierarchy of users of a system. The method comprises collecting a weighted audit trail related to access attempts by users to resources belonging to the system. All access attempts of each of a multiple types of access attempts to a given resource by each of the users are integrated. User data is aggregated in accordance with preferred clustering assignments. Clusters are merged into higher level clusters so that each higher level cluster clusters one or more
clusters in an immediate lower level cluster to form a cluster hierarchy. The steps are repeated until all clusters are merged into one cluster.

Specifically, the steps are further repeated periodically to create a temporal hierarchy representation.

Specifically, the aggregation is performed by averaging access data to a resource by all users belonging to a specific cluster.

Specifically, the aggregation is performed by determining the maximum value for access data corresponding to a user belonging to a specific cluster.

More specifically, the aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to optionally normalized individual accesses of said user to said resource multiplied by said maximum value.

More specifically said value relative to individual access is calculated by subtracting the value of said individual access from one.

More specifically, the maximum value is a normalized maximum value.

Specifically, the clustering is performed using a pairwise clustering algorithm. More specifically, the clustering algorithm is a greedy pairwise clustering algorithm.

Specifically, historical data is subjected to a decay.

Specifically, the weighted audit trail is determined for a user accessing a resource in a give access type accounting for all accesses of the same type in a decreasing value.

More specifically, a maximum value for said data is provided.
More specifically, the weighted audit trail is determined for a user not accessing a resource in a given access type by reducing the current said data.

Still more specifically, the reduction is done as a multiplication of said data by a fractional constant.

Still another aspect of the disclosed teachings is a method for determining permission levels for accessing a system. The method comprises collecting data related to access attempts by users to resources belonging to the system. The users are clustered into a user hierarchy. Permission levels are generated based on at least the data and the user hierarchy.

Specifically, the process is repeated periodically. Specifically, the classification of users into a user hierarchy is performed by a subprocess comprising integrating all access attempts of each of a multiple types of access attempts to a given resource by each of said users; aggregating user data in accordance with preferred clustering assignment; merging clusters into higher level clusters, wherein each higher level cluster clusters one or more clusters in an immediate lower level cluster to form a cluster hierarchy; and repeating the clustering until all users are merged into one cluster.

More specifically, the aggregation is performed by averaging access data to a resource by all users belonging to a specific cluster.

More specifically, the aggregation is performed by determining the maximum value for access data corresponding to users belonging to a specific cluster.

More specifically, said value relative to individual accesses is calculated as subtracting the value of said individual access from one.
More specifically, said clustering is performed using a pairwise clustering algorithm.

More specifically, the clustering is performed using a pairwise clustering algorithm. Still more specifically, the pairwise clustering algorithm is a greedy pairwise clustering algorithm.

Specifically, the generation of permission levels is performed using a subprocess comprising giving each of said users a weighted access profile to a resource based on said data and said user hierarchy; determining a range of said weighted access profiles to said resource and dividing it into a predetermined number of levels; and assigning for each of said users, a permission level based on which of said predetermined levels said each of said users corresponds to.

More specifically, the range of said weighted access attempts is scaled to a logarithmic scale.

Specifically, historical data is subjected to a decay factor.

More specifically, said weighted audit trail is determined for a user accessing a resource in a give access type accounting for all accesses of the same type in a decreasing value.

Still more specifically, a maximum value for said data is provided.

Still more specifically, the weighted audit trail is determined for a user not accessing a resource in a give access type by reducing the current said data.

Still more specifically, the reduction is done as a multiplication of said data by a fractional constant.

Still another aspect of the disclosed teachings include computer program products that include computer readable media with instructions, the instructions being capable of enabling a computer to implement the above methods.
The above summaries are merely meant to provide a guidance for a better understanding of the disclosed teachings and are not intended to be limiting the scope of the claims in any manner.

III. Brief Description of the Drawings

The disclosed teachings and techniques are described in more details using embodiments thereof with reference to the attached drawings in which:

FIG. 1 – is a block diagram illustrating an architecture used in conjunction with the present invention.

FIG. 2 – is an exemplary block diagram of an access analyzer.
FIG. 3 – is an exemplary block diagram of a hierarchy generator.
FIG. 4 – is an exemplary access matrix indicating users’ accesses to a resource.
FIG. 5 – is an exemplary hierarchical clustering example.
FIG. 6 – is an exemplary clustering machine using a pairwise clustering unit.
FIG. 7 – is an exemplary similarity matrix.
FIG. 8 – is an exemplary block diagram of a permission levels generator.

IV. Detailed Description of the Embodiments

FIG. 1 is a schematic illustration of a security system 10 in accordance with one embodiment that uses the disclosed teachings. This architecture is fully described in the co-pending US patent application filed on the same date herewith, entitled ”An Adaptive System and Architecture for Access Control”, and
which is assigned to common assignee as the present application, and is hereby incorporated by reference in its entirety, for all it discloses. The architecture illustrated in FIG. 1 includes at least three elements: an agent 100, a control unit 110 and an access analyzer 120. The relationship between the three elements are now described.

Agent 100 monitors access to the resources. Such access to the resource is also called access attempt 108. Agent 100 provides a periodic historical information of access attempts to the access analyzer 120. Such periodic historical information is also known as event audit trail (EAT) 102. Further, agent 100 also provides alarms (104) to the control unit 120. Still further, if agent 100 functions as a guardian agent, it enforces access rights to system resources by permission or denial 106 of access to a system resource or resources in response to access attempt 108.

Access analyzer 120 analyzes EAT 102, possibly using a first security policy 122. Access analyzer 120 responds, periodically, with a list of permission levels 134, to agent 100. Access analyzer 120 also provides statistical information 132 to control unit 110. Based on the permission levels received from access analyzer 120, and possibly using a second security policy 122’ received from control unit 110, a guardian agent 100 is capable of enforcing the security control of the resources. The control is implemented by permitting, alerting, denying, or otherwise controlling access to at least one resource within the overall system.

Although only one guardian agent, access analyzer and control unit are shown in the embodiment described in FIG.1, the disclosed teaching also contemplates using any combination of one or more agents, one or more access
analyzers and one or more control units.

The first security policy 122 and the second security policy 122’ may be identical, and may resides in the control unit 110. The second security policy 122’ may be provided to agent 100 through access analyzer 120.

The disclosed teachings include using techniques in unsupervised learning for learning an access control policy; from the access history of users, for implementing access analyzer 120. The information is gathered by agent 100 as a sequence of access attempts 108. An access attempt may include, but not limited to, a user, a location, a resource, an access type and time. The user may be an identification of the user that initiated the access event. The location may be an Internet protocol (IP) address, console name, terminal identifier, and so on, from where an access attempt 108 is being made. The resource may be an identification of the resource that was accessed, or attempted to be accessed, by the user. The access type may be of the type of access attempt 108 that was made by the user. Examples of access types include read, write, execute, or others, and any combination thereof. The time may be the exact time when the access event took place and may include among others the time of day, day of the week, etc.

The information gathered by the agent 100 is provided periodically by it to access analyzer 120 as event audit trail 102. It should be clear to a skilled artisan that the information gathering could be easily extended to accommodate additional information to be used by the learning system.

FIG. 2 is a schematic block diagram of an embodiment of access analyzer 120 that uses the disclosed teachings. Access analyzer 120 includes a history machine (HM) 210, a history database (HDB) 220, a user hierarchy generator
(UHG) 240, and a permission level generator (PLG) 230. HM 210 processes newly accepted event audit trail 102 and combines it with the history of event audit trail collected in the past that are stored in HDB 220. HM 210 outputs a weighted audit trail 250 to both UHG 240 and PLG 230. HM 210 further stores the newly accepted event audit trail 102 in HDB 220 for future use. The history is represented and collected as data \( d_{ijk} \). It relates to an activity of a user \( u_i \) on a resource \( r_j \) performing activity \( a_k \).

In one embodiment using the disclosed teachings, the history collection is continued and each \( d_{ijk} \) regardless of the time at which it is collected has the same weight.

However, in an alternate embodiment a decay factor \( \alpha \) is introduced. This decay factor ensures that the weight of an older \( d_{ijk} \) is reduced over time. Hence, HM 210 stores in HDB 220 the updated value of \( d_{ijk} \) that is calculated according to the following equation:

\[
d_{ijk} = \alpha * d_{ijk}^0 + (1-\alpha) * d_{ijk}^1
\]

Where the superscript "0" denotes data for a current event audit trail and the superscript "1" denotes data for a historical event audit trail. The constant \( \alpha \) is a value between 0 and 1 that denotes how fast the decay occurs. The higher the value of \( \alpha \), the higher the importance give to current values of \( d_{ijk} \) compared to the old value.

An alternate technique for implementing a weighted audit trail would be to introduce "charging" and "discharging" effects as follows:

If a current audit trail contains access of type \( a_k \) to resource \( r_j \) by a user \( u_i \),
then:

\[ d_{yk} = d_{yk}^1 + \beta_k (\theta_k - d_{yk}^1) \]

Otherwise:

\[ d_{yk} = \gamma_k d_{yk}^0 \]

where,

\( \beta_k, \gamma_k \) represent charging and discharging coefficients respectively for access type \( a_k \) and

\[ 0 \leq \beta_k \leq 1 \text{ and } 0 \leq \gamma_k \leq 1 \], in a typical example \( \beta_k = 0.5 \) and \( \gamma_k = 0.9 \)

\( \theta_k \) represents the maximum value for \( d_{yk} \).

While in this embodiment a couple of specific techniques for handling the impact of historical are implemented, a skilled artisan will know that other approaches for implementing the decay of historical data may be used without deviating from the spirit of the disclosed teachings. All such decay techniques are believed to be within the scope of the disclosed teachings.

The weighted audit trail 250, which is basically a reflection of each user’s access patterns, is used by UHG 240 to generate the user hierarchy based on the user pattern of accesses to system resources. An exemplary implementation of UHG 240 using the disclosed teachings is described herein with reference to FIG. 3. The basic elements that are included in UHG 240 are Integrator 310, aggregator 320, clustering machine (CM) 330 and hierarchy accumulator 340. Integrator 310 operates on data points \( d_{yk} \) and integrates all the data points that correspond to a certain access type. While one example is provided in this disclosure, other structures of hierarchy, such as a hierarchy tree is also possible. A tree hierarchy differs from a clustering hierarchy, in the sense that in
the tree hierarchy each node represents a split of clusters and further that the leaves are at different levels.

An access type may be, for example, a read access, a write access, and others as separate or combinations thereof. It should be noted that for the purpose of hierarchy identification the access types of the users can be integrated by performing, for example the following calculation:

\[ d_y = \sum_{k=1}^{K} \mu_k d_{ik} \]

where \( \mu_k \) is the importance of a particular access type. For example, write may be more significant than read.

In the case where the weighted event audit trail is updated using a “charging” and “discharging” function, the integration is performed using the following equation:

\[ d_{cj} = 1 - \prod_{n \in C} (1 - d_y) \]

Where the \( d_{cj} \) corresponds to the value of \( d \) for a cluster \( C \) and resource \( j \), and the \( d \) values are normalized to the range between 0 and 1.

It should be noted that instead of \( d_{ijk} \) another factor \( d_{ijld} \) could be used where \( l \) represents a specific time slot. In such a case \( d_{ijld} \) would represent an activity of a user \( u_i \) on a resource \( r_j \) performing activity \( a_k \) during a time slot. The equations for handling the impact of historical data and integration can be appropriately modified without undue experimentation by a skilled artisan.

Aggregator 320 accumulates the data corresponding to a user relating to the resources the user is attempting to access. It further handles the feedback provided by CM 330. Such feedback necessitates the recalculation of the data as
cluster assignments take place and lower level clusters are clustered into higher level clusters.

An example implementation for the operation of aggregator 320 is to average the data $d_{ij}$ for all data belonging to all users of a specific cluster particular value of "$i". Concept of a cluster is described subsequently in detail with reference to CM 330 and FIG.4.

In another implementation, a maximum value of all $d_{ij}$ corresponding to users of a specific cluster is chosen by the aggregator as an aggregate value of the data. A skilled artisan could without undue experimentation implement further examples of such aggregation functions without deviating from the spirit of the disclosed teachings.

The user access profiles are an output of aggregator 320 and may be organized as an access matrix correlating clusters of users to resources as in the exemplary matrix of Fig. 4. In this access matrix, the rows indicate the users and the columns indicate resources. The values in each $ij$ combination is related to the access attempts by a user "$i" to a resource "$j". This data can be used by CM 330 to define the way in which users $u_i$ should be clustered. As can be clearly seen by the feedback mechanism, the process is iterative. A feedback is provided to aggregator 320 allowing for a further higher level cluster assignment that factors in the feedback relative to the current clustering status. However, output of aggregator 320 may be organized in other structures, more specifically as a tree of resources, such as would be the case when handling files within a sequence of file folders.

CM 330 may cluster users $u_1$ through $u_5$, in the example of Fig. 4, as described in FIG. 5. In a hypothetical scenario, CM could use a first lower level
cluster assignment that clusters users $u_1$ with $u_2$ 520, users $u_4$ with $u_6$ 540, and leaves user $u_3$ 530 on its own. As the clustering process of CM 330 contains a feedback loop, the current lower level clustering assignment is fed back to aggregator 320 that updates the input matrix to CM 330. As a result of the new input, CM 330, may have a new upper level cluster assignment where lower level cluster 520 and lower level cluster 530 are further assigned to one upper level cluster 510, and cluster 540 is left on its own. In the next step round of clustering, upper level cluster 510 and cluster 540 are assigned one super level cluster 500. The super level 500 represents the case of all the users belonging to the system 10.

One exemplary implementation of CM 330 using the disclosed teachings is described in Fig. 6 where a similarity matrix generator (SMG) 610 is connected to a pairwise clustering generator (PCG) 620. In one embodiment of the CM 330 using the disclosed teachings a greedy pairwise clustering generator (GPCG) is used. A GPCG is the subject matter of a co-pending US patent application filed on the same date herewith, entitled “A System and Method for a Greedy Pairwise Clustering”, and which is assigned to common assignee as the present invention, and is hereby incorporated by reference in its entirety, for all it discloses.

However, the PCG requires a similarity matrix as an input. Therefore, the output of aggregator 320 must be preprocessed in order to generate the similarity matrix. The generation of the similarity matrix is performed by SMG 610 receiving the access profiles as its input. The similarity between two users $u_i$ and $u_m$ can be determined in several ways. One possible way is to calculate the number of shared resources between $u_i$ and $u_m$ divided by the total number of occurrences for $u_i$. This can be summarized in the following formula:
\[ \text{Sim}(u_1, u_m) = \frac{\text{com}(u_1, u_m)}{|u_1|} \]

The matrix of FIG. 4 can be used in conjunction with the above-mentioned formula to generate the similarity matrix of Fig. 7. The results of the self-similarity, i.e., the similarity of \( u_i \) to itself, is clearly "1".

However, the similarity between two users shown in FIG.7, \( u_1 \) to \( u_2 \), is calculated as follows. In this example, Resources \( R_3 \) through \( R_7 \), a total of five, are accessed by both \( u_1 \) and \( u_2 \). User \( u_1 \) alone, however, accesses a total of seven of the resources, namely resources \( R_3 \) through \( R_7 \). The ratio between the commonly used resources five, and that individually used by \( u_1 \), seven, determines the similarity between \( u_1 \) and \( u_2 \), which is 5/7 (0.714). Similarly, the similarity of \( u_2 \) to \( u_1 \) is calculated. Resources \( R_3 \) through \( R_7 \), a total of five, are accessed by both \( u_2 \) and \( u_1 \). User \( u_2 \), however, accesses a total of six of the resources, namely resources \( R_3 \) through \( R_7 \) as well as \( R_9 \). The ratio between the commonly used resources, five, and the total accesses by \( u_2 \), six, determines the similarity between \( u_2 \) and \( u_1 \), which is 5/6 (0.833).

In an alternate embodiment using the disclosed teaching, similarity may be the result of the internal multiplication relative to \( u_i \) and \( u_m \) divided by the norms for \( u_i \) and \( u_m \), as in the formula below:

\[ \text{Sim}(u_1, u_m) = \frac{(u_1 \cdot u_m)}{|u_1| \cdot |u_m|} \]

A skilled artisan can without undue experimentation use other ways of calculating similarity calculations for creating a similarity matrix without deviating from the spirit of the disclosed teachings. It should also be clear that instead of a similarity matrix, a dissimilarity matrix, representing the dissimilarity between
two users, may be used. If a dissimilarity matrix is to be used, a minimum rather then maximum scores will be used by PCG 620 to determine the clustering assignment.

Permission levels 134 are generated by PLG 230 based upon at least weighted audit trail 250 and user hierarchy 260. In an alternate embodiment using the disclosed teachings security policy 122 is also an input to PLG 230. PLG 230, as shown in FIG.8, includes a weighted hierarchical activity calculator (WHAC) 810 and a permission level calculator (PLC) 820. The number of permission levels may be determined by a user or administrator of system 10, depending on the level of granularity and control necessary to be achieved, and is designated as $N_{PL}$.

WHAC calculates the activity type $a_k$ on a resource $r_j$ for all the users of the system. Naturally, the activity will range from a very low activity $a_k$ of some users on a resource $r_j$, to a very high activity $a_k$ on a resource $r_j$, of other users. It is important that users sharing a cluster, whether within the same cluster, or through the hierarchy of clusters, may also get a permission to use certain resources. To ensure this, a weighted value resulting from the access of a user or a cluster of users to a resource, is also provided to other users sharing the cluster. In other words, an access by a user to a resource will result in a certain weight given to such access attempt to all other users that have a relationship with that user. Such a relationship could be by being in the same cluster or through the hierarchy of clusters. The weighted contribution of the relationship reduces relative to the distance of the relationship.

The concept of generating a weighted user profile is described herein. Recall that $d_{ij}$ is related to the activity of user $u_i$ with resource $r_j$. A weighted
number $d'_{ij}$ is calculated in which all clusters that contain the user (at all different levels of the hierarchy) will have an influence.

To decide what is contributed from a cluster $C$ that contains $u_i$, all the activities related to the members of the cluster need to be aggregated. Depending on the way the raw data is collected, different mechanisms may be chosen. A mechanism similar to the one used in aggregator (before finding the next level of the hierarchy) could be used. Also, when weighting the influence of a cluster, one may take into account only the new members of the cluster. It should be noted that the newer members of the cluster correspond to those that are not members of the clusters that contain $u_i$ at lower levels of the hierarchy. This will ensure that every user will exercise influence only once.

Finally, the influence of a cluster $C$ on $d'_{ij}$ is weighted according to the level of $C$ in the hierarchy, where the cluster that immediately contains $u_i$ is at level 1, the cluster that immediately contains this cluster is at level 2, and so forth. Since a logarithmic scale is used to analyze the weighted profiles, a reasonable choice for the weight of the cluster $c$ at level $i$ should exponentially decrease with $i$, e.g. $\frac{1}{2^i}$.

As described in the application, to generate the permission levels, the weighted profiles $d'_{ij}$ are used rather than the profiles $d_{ij}$ themselves.

A logarithmic scale is proposed to measure activity $a_k$ on resource $r_j$. Such a scale is suggested due to the significant range of activity levels between users. Hence, a user may access his own files very frequently, while other users may not access it at all. The logarithmic range is then divided by $N_{PL}$, creating $N_{PL}$.
levels. A user $u_i$ having an activity level $a_{kl}$ on resource $r_j$, will fall within one of the levels, designated as $l$. The adaptive permission level $L_{ijk}$, which is the permission level for a user $u_i$ to access a resource $r_j$ for the purpose of an access $a_{kl}$ is calculated as follows:

$$L_{ijk} = \frac{N_{pl} - l}{N_{pl}}$$

In the alternate embodiment where security policy 122 is also used as an input, an access attempt 108 may be defined as: a) having an access permission regardless of the results of the access analyzer 120; or b) having no access permission regardless of the results of the access analyzer 120; or c) being an adaptive rule, the permission level is dependent on the results of access analyzer 120. Therefore the values for $L_{ijk}$ may be "1", "0", or the results of the formula above, respectively.

It should be noted that a user may receive a permission level other than zero to access a resource, even if the resource was never used by this user. Such a situation arises if another user in his cluster, or a joined cluster of a higher level, has accessed that resource. A way of implementing this feature by WHAC 810 is by assigning a decreasing resource access weight along the hierarchy to clusters of a user who used a certain resource.

Assume that the contribution of a cluster to a user is derived from the maximal access value of users within the cluster. Referring to Fig. 5, assume further that a user $u_i$ accessed $r_j$, which no other user has accessed.

Notwithstanding, each user will still receive a contribution of that access based on its cluster proximity. Hence, cluster 520 will have a weight of, for example,
0.5, cluster 510 a weight of 0.25, and cluster 500 a weight of 0.125. Therefore, user \( u_1 \) will have an overall access weight of \( 1+0.5+0.25+0.125 = 1.875 \), while user \( u_2 \) will have an access weight of \( 0.5+0.25+0.125 = 0.875 \). User \( u_3 \) will have an access weight of \( 0.25+0.125 = 0.375 \), while users \( u_4 \) and \( u_5 \) will have an access weight of 0.125 only.

If a four level permission system - with permissions of “1”, “0.75”, “0.5”, and “0.25” is used - then \( u_1 \) may have a permission level of “1”, \( u_2 \) a permission level of “0.75”, \( u_3 \) a permission level of “0.5”, while \( u_4 \) and \( u_5 \) will have a permission level of “0.25” each. In conjunction with the threshold values defined in security policy 122, an agent 100 operating as a guardian agent may permit or deny certain access attempts. For example, if the threshold for an access to resource \( r_1 \) is defined at a level of “0.6” only users \( u_1 \) and \( u_2 \) will be allowed to access this resource.

The purpose of access analyzer 120 is to deduce from event audit trail 102 a number in the range from 0 to 1, to be assigned to each potential access attempt 108. The number assigned to a potential access event \( (u, r, a, t) \), where “\( u \)” denotes a user(s), “\( r \)” denotes a resource(s), “\( a \)” denotes an access attempt(s), and “\( t \)” denotes a time(s), can be interpreted as the probability that a user \( u \) should be permitted to make an access of type “\( a \)” to a resource “\( r \)” at a time “\( t \)”. A person skilled in the art would be able to add parameters to the definition of a potential event. In one embodiment of this invention, the parameter “\( l \)”, denoting a location(s) may be used. The higher this probability, the more confident system 10 is that access attempt 108 should be permitted. By allowing system 10 to autonomously learn the characteristics of access attempts 108, the system administrators are freed from the need to manually
maintain security policy 122 of a complex enterprise wide system, as would otherwise be required in conventional systems.

Further, as users are grouped into clusters with similar access patterns, access analyzer 120 provides a generalization mechanism from access attempts made by individual users to access attempts made by clusters of users. In other words, a user may get permissions to resources he has never accessed in the past provided that there is strong evidence in the hierarchy data that his cluster should get permission to access that resource. This generally happens when other members of the cluster have accessed that resource in the past. Using this mechanism, the accesses made by a user may influence the permissions assigned not only to each member of his basic cluster, but also influence the permissions assigned to members of clusters at higher levels.

Furthermore, using the method described hereinabove, a hierarchy of clusters was produced, in other words several levels of clusters. The most general cluster, the cluster at the top of the hierarchy includes all the users, that may be used to allow the most liberal access to resources of system 10. The bottommost level has every user in a different cluster. In between there are several levels of partitions of users into clusters, wherein each cluster assignment is a refinement of the previous cluster assignment.

A person skilled in the art could easily adapt this invention for use in other fields other the access control. One specific use is to learn the informal hierarchy of an organization and its temporal dynamics.

It should be clear to a skilled artisan that the disclosed teaching can be implemented using hardware, software or a combination thereof. It can also be implemented using any type of computer including PCs, workstations and
mainframes. The disclosed teaching also includes computer program products that includes a computer readable medium with instructions. These instructions can include, but not limited to, source code, object code and executable code and can be in any higher level language, assembly language and machine language. The computer readable medium is not restricted to any particular type of medium but can include, but not limited to, RAMS, ROMs, hard disks, floppies, tapes and internet downloads.

Other modifications and variations to the disclosed teachings will be apparent to those skilled in the art from the foregoing disclosure and teachings. Thus, while only certain embodiments using the disclosed teachings have been specifically described herein, it will be apparent that numerous modifications may be made thereto without departing from the spirit and scope of the disclosed teachings.
What is claimed is:

1. A system with controlled user access, the system comprising:
   an access analyzer subsystem; and
   a user control subsystem;
   the access analyzer subsystem further including:
   a history machine capable of processing new and historical data related to access attempts;
   a history database for storing said new and historical data;
   a hierarchy generator connected to said history machine for generating a hierarchy related to said historical data;
   a permission level generator connected to said history machine and said hierarchy generator adapted to generate permissions for accessing said user subsystem based on said hierarchy, and
   the user control subsystem further including:
   an agent adapted to provide an event audit trail to said access analyzer subsystem and receive a permission level from said access analyzer subsystem; and
   a control unit adapted to provide a security policy to said access analyzer subsystem and receive a statistical information from said access analyzer subsystem.

2. The system of claim 1, wherein said agent is a guardian agent further adapted for at least allowing access to a resource in response to an access attempt, the allowing being based on at least said permission
level.

3. The system of claim 1, wherein said agent is a guardian agent further adapted for at least denying access to a resource in response to an access attempt, the denying being based on at least said permission level.

4. The system of claim 1, wherein said agent is a guardian agent further adapted for at least sending an alarm to said control unit in response to an access attempt, the sending being based on at least said permission level.

5. The system of claim 1, wherein said control unit is further adapted for generating reports based on at least said statistical information.

6. The system of claim 1, wherein said control unit is further capable of generating reports based on at least said alarms.

7. A system for tracking access attempts, the system comprising:
   a history machine adapted to process new and historical data related to said events;
   a history database adapted to store said new and historical data;
   and
   a hierarchy generator connected to said history machine adapted to generate a hierarchy related to said new and historical data.

8. The system of claim 7, further including:
a permission level generator connected to said history machine and said hierarchy generator, said permission level generator being adapted to generate permissions for accessing system resources based at least on said hierarchy.

9. The system of claim 8, wherein said permission level generator comprises a weighted hierarchy activity calculator connected to a permission level calculator.

10. The system of claim 9, wherein said weighted hierarchy activity calculator receives at least a weighted audit trail and a user hierarchy.

11. The system of claim 9, wherein said permission level calculator generates permission levels based on at least an input provided from said weighted hierarchy activity calculator.

12. The system of claim 11, wherein said permission levels are based further on a security policy.

13. The system of claim 7, wherein said history machine generates a weighted audit trail based on event audit trail and history information provided by said history database.

14. The system of claim 13, wherein a decay factor is applied to historical data prior to storing in said history database.
15. The system of claim 13, wherein said weighted audit trail is determined for a user accessing a resource in a given access type accounting for all accesses of the same type in a decreasing value.

16. The system of claim 15, wherein a maximum value for said data is provided.

17. The system of claim 13, wherein said weighted audit trail is determined for a user not accessing a resource in a given access type by reducing the current said data.

18. The system of claim 17, wherein said reduction is done as a multiplication of said data by a fractional constant.

19. The system of claim 7, wherein said user hierarchy generator further includes:
   an integrator;
   an aggregator connected to said integrator and further connected to a clustering machine; and
   a clustering machine connected to said aggregator;
   a hierarchy accumulator connected to said clustering machine.

20. The system of claim 19, wherein said integrator receives a weighted audit trail.
21. The system of claim 20, wherein said integrator calculates a sum total of all accesses of a specific access type made by a user, said access being done to a specific resource.

22. The system of claim 19, wherein said aggregator calculates a sum total of all accesses of a specific access type made by all users in a cluster, said access being done to a specific resource.

23. The system of claim 19, wherein said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to normalized individual accesses of said user to said resource multiplied by said maximum value.

24. The system of claim 23, wherein said value relative to individual accesses is calculated as subtracting the value of said individual access from one.

25. The system of claim 19, wherein said aggregator outputs access information for every access type.

26. The system of claim 25, wherein said access information comprises a matrix of rows and columns, said rows representing users and said columns representing resources, said matrix storing data related to the
access of said users to said resources.

27. The system of claim 19, wherein said clustering machine outputs a preferred clustering assignment.

28. The system of claim 27, wherein said clustering machine implements a pairwise clustering algorithm.

29. The system of claim 28, wherein said pairwise clustering algorithm is a greedy pairwise clustering algorithm.

30. The system of claim 19, wherein said clustering machine comprises:
   
a similarity matrix generator; and

   a pairwise clustering generator.

31. The system of claim 30, wherein said similarity matrix generator is adapted to determine a commonality between access profiles of a first user and a second user to a resource, said similarity matrix generator is further adapted to determine a first similarity between said first user and said second user, said first similarity being a result of dividing said commonality by the access attempts made by said first user and said similarity matrix generator being further adapted to determine a second similarity between said second user and said first, said second similarity being a result of dividing said commonality by the access attempts made by said second user.
32. The system of claim 30, wherein said similarity matrix generator is adapted to calculate an internal multiplication of user access profiles between a first user and a second user, said similarity matrix generator being further adapted to divide the internal multiplication result by a norm of user access profiles made by said first user multiplied by the norm of access attempts made by said second user.

33. A method for determining a hierarchy of users of a system, said method comprising:
   (a) collecting a weighted audit trail related to access attempts by users to resources belonging to the system;
   (b) integrating all access attempts of each of a multiple types of access attempts to a given resource by each of said users;
   (c) aggregating user data in accordance with preferred clustering assignments;
      (d) merging clusters into higher levels clusters, wherein each higher level cluster clusters one or more clusters in an immediate lower level cluster to form a cluster hierarchy; and
   (e) repeating step (d) until all clusters are merged into one cluster.

34. The method of claim 33, wherein the steps are repeated periodically to create a temporal hierarchy representation.

35. The method of claim 33, wherein said aggregation is performed by averaging access data to a resource by all users belonging to a specific
36. The method of claim 33, wherein said aggregation is performed by determining the maximum value for access data corresponding to one of all users belonging to a specific cluster.

37. The method of claim 33, wherein said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to individual accesses of said user to said resource multiplied by said maximum value.

38. The method of claim 33, wherein said clustering is performed using a pairwise clustering algorithm.

39. The method of claim 38, wherein said pairwise clustering algorithm is a greedy pairwise clustering algorithm.

40. The method of claim 33, wherein historical data is subjected to a decay.

41. The method of claim 33, wherein said weighted audit trail is determined for a user accessing a resource in a give access type accounting for all accesses of the same type in a decreasing value.
42. The method of claim 41, wherein a maximum value for said data is provided.

43. The method of claim 33, wherein said weighted audit trail is determined for a user not accessing a resource in a given access type by reducing the current said data.

44. The method of claim 43, wherein said reduction is done as a multiplication of said data by a fractional constant.

45. A method for determining permission levels for accessing a system, said method comprising:
(a) collecting data related to access attempts by users to resources belonging to the system;
(b) clustering users into a user hierarchy;
(c) generating permission levels based on at least said data and said user hierarchy.

46. The method of claim 45, wherein the process is repeated periodically.

47. The method of claim 45, wherein said classifying users into a user hierarchy is performed by a subprocess comprising:
(b)(1) integrating all access attempts of each of a multiple types of access attempts to a given resource by each of said users;
(b)(2) aggregating user data in accordance with preferred clustering assignment;
(b)(3) merging clusters into higher levels clusters, wherein each higher level cluster clusters one or more clusters in an immediate lower level cluster to form a cluster hierarchy; and
(b)(4) repeating step (b)(3) until all users are merged into one cluster.

48. The method of claim 47, wherein said aggregation is performed by averaging access data to a resource by all users belonging to a specific cluster.

49. The method of claim 47, wherein said aggregation is performed by determining the maximum value for access data corresponding to one of all users belonging to a specific cluster.

50. The method of claim 47, wherein said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to normalized individual accesses of said user to said resource multiplied by said maximum value.

51. The method of claim 50, wherein said value relative to individual accesses is calculated as subtracting the value of said individual access from one.
52. The method of claim 47, wherein said clustering is performed using a pairwise clustering algorithm.

53. The method of claim 52, wherein said pairwise clustering algorithm is a greedy pairwise clustering algorithm.

54. The method of claim 45, wherein said generation of permission levels is performed using a subprocess comprising:

   (c)(1) giving each of said users a weighted access profile to a resource based on said data and said user hierarchy;

   (c)(2) determining a range of said weighted access profile to said resource and dividing it into a predetermined number of levels; and

   (c)(3) assigning for each of said users, a permission level based on which of said predetermined levels said each of said users corresponds to.

55. The method of claim 54, wherein said range of said weighted access attempts is scaled to a logarithmic scale.

56. The method of claim 45, wherein historical data is subjected to a decay factor.

57. The method of claim 45, wherein said weighted audit trail is determined for a user accessing a resource in a give access type accounting for all accesses of the same type in a decreasing value.
58. The method of claim 57, wherein a maximum value for said data is provided.

59. The system of claim 45, wherein said weighted audit trail is determined for a user not accessing a resource in a give access type by reducing the current said data.

60. The method of claim 59, wherein said reduction is done as a multiplication of said data by a fractional constant.

61. A computer program product determining a hierarchy of users of a system, said program product comprising a computer-readable medium, said medium comprising instructions that enable a computer to implement a process, said process comprising:
(a) collecting data related to access attempts by users to resources belonging to the system;
(b) integrating all access attempts of each of a multiple types of access attempts to a given resource by each of said users;
(c) aggregating user data in accordance with preferred clustering assignments;
(d) merging clusters into higher levels clusters, wherein each higher level cluster clusters one or more clusters in an immediate lower level cluster to form a cluster hierarchy; and
(e) repeating step (d) until all clusters are merged into one cluster.
62. The computer program product of claim 61, wherein the process is repeated periodically for the purpose of determining a temporal hierarchy.

63. The computer program product of claim 61, wherein the output of step (d) represents an output related to a specific hierarchy level.

64. The computer program product of claim 61, wherein said aggregation is performed by averaging access data to a resource by all users belonging to a specific cluster.

65. The computer program product of claim 61, wherein said aggregation is performed by determining the maximum value for access data corresponding to one of all users belonging to a specific cluster.

66. The computer program product of claim 61, wherein said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to individual accesses of said user to said resource multiplied by said maximum value.

67. The computer program product of claim 66, wherein said value relative to individual accesses is calculated as subtracting the value of said individual access from one.
68. The computer program product of claim 61, wherein said clustering is performed using a pairwise clustering algorithm.

69. The computer program product of claim 68, wherein said pairwise clustering algorithm is a greedy pairwise clustering algorithm.

70. The computer program product of claim 61, wherein historical data is subjected to a decay.

71. The computer program product of claim 61, wherein said weighted audit trail is determined for a user accessing a resource in a given access type accounting for all accesses of the same type in a decreasing value.

72. The computer program product of claim 71, wherein a maximum value for said data is provided.

73. The computer program product of claim 61, wherein said weighted audit trail is determined for a user not accessing a resource in a given access type by reducing the current said data.

74. The computer program product of claim 73, wherein said reduction is done as a multiplication of said data by a fractional constant.

75. A computer program product for determining permission levels for accessing a system, said program product comprising a computer-
readable medium, said medium comprising instructions that enable a computer to implement a process, said process comprising:

(a) collecting data related to access attempts by users to resources belonging to the system;
(b) clustering users into a user hierarchy;
(c) generating permission levels based on at least said data and said user hierarchy.

76. The computer program product of claim 75, wherein the process is repeated periodically for the purpose of updating said permission levels.

77. The computer program product of claim 75, wherein said classifying users into a user hierarchy is performed by a subprocess comprising:

(b)(1) integrating all access attempts of each of a multiple types of access attempts to a given resource by each of said users;
(b)(2) aggregating user data in accordance with preferred clustering assignment;
(b)(3) merging clusters into higher levels cluster, wherein each higher level cluster clusters one or more clusters in an immediate lower level cluster to form a cluster hierarchy; and
(b)(4) repeating step (b)(3) until all users are merged into one cluster.

78. The computer program product of claim 77, wherein said aggregation is performed by averaging access data to a resource by all users belonging to a specific cluster.
79. The computer program product of claim 77, wherein said aggregation is performed by determining the maximum value for access data corresponding to one of all users belonging to a specific cluster.

80. The computer program product of claim 77, wherein said aggregator calculates the contribution of all users within a cluster to access to a resource as the difference between the maximum value for such user access to said resource and the multiplication of all values relative to individual accesses of said user to said resource multiplied by said maximum value.

81. The computer program product of claim 80, wherein said value relative to individual accesses is calculated as subtracting the value of said individual access from one.

82. The computer program product of claim 77, wherein said clustering is performed using a pairwise clustering algorithm.

83. The computer program product of claim 82, wherein said pairwise clustering algorithm is a greedy pairwise clustering algorithm.

84. The computer program product of claim 75, wherein said generation of permission levels is performed using a subprocess comprising:
   (c)(1) giving each of said users a weighted access profile to a resource based on said data and said user hierarchy;
   (c)(2) determining a range of said weighted access profile to said
resource and dividing it into a predetermined number of levels; and
(c)(3) assigning for each of said users, a permission level based on
which of said predetermined levels said each of said users
corresponds to.

85. The computer program product of claim 84, wherein said range of
said weighted access attempts is scaled to a logarithmic scale.

86. The computer program product of claim 75, wherein historical data
is subjected to a decay factor.

87. The computer program product of claim 75, wherein said weighted
audit trail is determined for a user accessing a resource in a give access
type accounting for all accesses of the same type in a decreasing value.

88. The computer program product of claim 87, wherein a maximum
value for said data is provided.

89. The computer program product of claim 75, wherein said weighted
audit trail is determined for a user not accessing a resource in a give
access type by reducing the current said data.

90. The method of claim 89, wherein said reducing is performed as a
multiplication of said data by a fractional constant.
91. The method of claim 37, wherein said value relative to individual access is calculated by subtracting the value of said individual access from one.

92. The system of claim 23 wherein the maximum value is a normalized maximum value.

93. The method of claim 50 wherein the maximum value is a normalized maximum value.
FIG. 7

Weighted Audit Trail 250

Weighted Hierarchical Activity Calculator

Permission Level Calculator

Permission Levels 134

FIG. 8