Abstract: Disclosed are real-time techniques for determining all access requests to an attribute-based access control policy which evaluate to a given decision, "permit" or "deny". The policy is enforced to control access to one or more resources in a computer network. In one embodiment, a method comprises: (i) receiving a reverse query and a set of admissible access requests, each of which comprises one or more attributes in the policy and values of these; (ii) extracting attributes to which all access requests in the set assign identical values; (iii) reducing the ABAC policy by substituting values for the extracted attributes; (iv) caching the policy as a simplified policy; (v) translating the simplified policy and the given decision into a satisfiable logic proposition; (vi) deriving all solutions satisfying the proposition; and (vi) extracting, based on the solutions, all access requests from the set for which the policy yields the given decision.

[Continued on next page]
Declarations under Rule 4.17:

— of inventorship (Rule 4.17(iv))

Published:

— with international search report (Art. 21(3))
A SYSTEM AND METHOD FOR EVALUATING A REVERSE QUERY

Field of the invention

The invention disclosed herein generally relates to the field of access control (AC), particularly access control to resources in computer systems or computer-aided access control to other types of resources. More precisely, the invention provides improved devices and methods for evaluating a policy for a reverse query, which returns such access requests that evaluate to a given decision.

Background of the invention

An attribute-based AC (ABAC) policy defines access control permissions based on the attributes of the subject, of the resource, and of the action that the subject is to perform on the resource (e.g., read, write). When the policy is enforced in a computer system or computer network, it controls access to entities in the system or network and thereby influences their state of operation. A resource may be, inter alia, a portion of a personal storage quota, a business unit storage quota, an information retrieval system, a (portion of a) database, an online service, a protected webpage or a physical device.

There currently exist general-purpose AC languages that have the richness to express fine-grained conditions and conditions which depend on external data. One particular example of an AC language is the extensible Access Control Markup Language (XACML) which is the subject of standardization work in a Technical Committee within the Organization for the Advancement of Structured Information Standards (see http://www.oasis-open.org). A policy encoded with XACML consists of functional expressions in attribute values, and the return value (decision) of the policy is one of Permit, Deny, Not Applicable, or Indeterminate. An XACML policy can apply to many different situations, that is, different subjects, resources, actions and environments and may give different results for different combinations of these. The XACML specification defines how a policy is evaluated for a request (or access request), particularly what policy attributes are to be evaluated or, at least, which values are required to exist for a successful evaluation to result. Key characteristics of this evaluation process are that the access request (the query against the policy) must describe the attempted access to a protected
resource fully. In practice, it may be that the request is constructed in multiple stages by different components, so that a PEP (Policy Enforcement Point) provides only some initial attribute values and a PDP (Policy Decision Point) or other components can dynamically fetch more attribute values from remote sources as they are needed. Rules in an ABAC policy may be nested in a conditional fashion, so that attribute values - both those provided initially in the access request and those fetched from remote sources - will influence what further rules are to be applied. Based on a policy or policy set (unless otherwise indicated, these terms are used interchangeably herein) that covers a broad range of resources and subjects and a given request, it is often possible to obtain a decision by evaluating only a fraction of all functional expressions in the policy. Conversely, it cannot always be ascertained prima facie whether a request contains enough attribute values to allow a successful policy evaluation.

A reverse query works in the opposite direction. It defines an expected decision and constraints over the set of possible access requests, and is resolved by finding the set of access requests that (a) fulfill all the constraints, and (b) evaluate to the expected decision. Reverse queries have many uses. For instance, they could be used to determine the list of resources that a subject may access, or the list of subjects that may access a resource. Furthermore, many types of policy analyses can be built using reverse queries (e.g., Segregation of Duty validation).

The semantics of an XACML policy $P$ may be given as a function $f_P$ mapping a request to a decision:

$$f_P : \text{Request} \rightarrow \text{Decision}$$

In many situations, however, it is necessary to evaluate the inverse of the policy function,

$$(f_P)^{-1} : \text{Decision} \rightarrow \text{Set(Request)}$$

Given a decision $d$, $(f_P)^{-1}(d)$ is the set of all requests that evaluate to $d$. For example, $(f_P)^{-1}(\text{PERMIT})$ is the set of all requests that are permitted by the policy $P$. Note that $(f_P)^{-1}$ is multi-valued in general, and may be regarded as a mapping from a decision value (Permit, Deny, etc.) to a set of requests.
In many important applications, there is a priori a set \( R \) of interesting requests. For instance, to determine all the users that may "read" a certain file "F", only requests that identify the action as "read" and the resource as "file F" are of interest. In other words, what needs to be computed is actually the intersection of 

\[
(f_p)^{-1}(d) \cap R.
\]

These concepts may be summarized by the following definitions.

**Definition (Reverse Query):** A reverse query is a triple \(< P, d, R >\) where 

- \( P \) is a policy
- \( d \) is a decision
- \( R \) is a set of requests

**Definition (Reverse Query Evaluation):** A reverse query \( < P, d, R > \) is evaluated by computing \((f_p)^{-1}(d) \cap R\), where \( f_p \) is the semantic function associated to policy \( P \).

The evaluation of a reverse query is in general much more demanding, in terms of computing resources, in particular, time, than evaluating a request against a policy. If the set of requests of interest, \( R = \{ r_1, \ldots, r_n \} \), contains a relatively small number of requests then the reverse query \( < P, d, R > \) may be effectively evaluated by computing

\[
f_p(r_1), \ldots, f_p(r_n)
\]

and picking only those requests which evaluate to \( d \). That is, a reverse query can be evaluated by sending each request of interest to the PDP (loaded with the policy \( P \)) and then comparing the returned decision with the expected decision \( d \).

If the set \( R \) is large, however, the method described above becomes impracticable, particularly in situations where the reverse query needs to be evaluated in real-time, e.g., in the context of an interactive system where a user would be waiting in real time for the result of such evaluation.

**Summary of the invention**

It is in view of the above mentioned problems that the present invention has been made.
A method for real-time evaluation of a reverse query to an attribute-based access control policy, ABAC policy, (P) which is enforced to control access to one or more resources in a computer network, executes on a system comprising a processing means, preferably a computer connected to a data network. In accordance with a first aspect of the invention, the method comprises the steps of:

i) receiving a reverse query indicating a given decision (d), which is one of permit access and deny access, and a set (R) of admissible access requests, each of which comprises one or more attributes appearing in the ABAC policy and explicit values assigned to these;

ii) extracting attributes to which all access requests in the set (R) assign identical values;

iii) reducing the ABAC policy at least by substituting values for the extracted attributes;

iv) caching the policy after said reducing as a simplified policy (P');

v) translating the cached simplified policy (P') and the given decision (d) into a satisfiable logic proposition in Boolean variables \( \{v_i \mid i = 1, 2, \ldots\} \);

vi) deriving all variable assignments \( \{c_j = \{v_i = x_j1, v_i = x_j2, \ldots, j \leq i \leq 1, 2, \ldots\}\} \) satisfying the logic proposition; and

vii) extracting, based on the variable assignments thus derived, all access requests from the set (R) for which the ABAC policy (P) yields the given decision (d) (referred to below as "valid requests").

The method alleviates the problems associated with the prior art, since inter alia the reduction of the ABAC policy considerably reduces the computational load on the processing means.

In a second aspect, the invention provides a system operable to evaluate a reverse query, defining an expected decision, and a subset of a set of admissible requests, over a policy in real time. The system comprises a first storing means operable to store policies. The system also comprises a partial request generation means operable to construct a partial request from the subset of the set of possible requests (cf. step ii). Furthermore, the system also comprises a policy decision means connected to the partial request generation means, and to the first storing means and operable to partially evaluate the policy over the partial request, resulting in a simplified policy (cf. steps iii, iv). The system also comprises a translation means connected to the policy decision means and operable to trans-
late the simplified policy, the subset of the set of possible requests, and the expected decision into a propositional logic formula (cf. step v). Furthermore, the system also comprises an analyzing means connected to the translation means and operable to analyze the propositional logic formula in order to determine a sequence of one or more conditions over requests (cf. step vi). The conditions in the sequence are sufficient in the sense that each corresponds to a variable assignment that satisfies the logic proposition. The order of the conditions in the sequence is not important. The system also comprises a conversion means connected to the analyzing means, and operable to convert the sequence of conditions to a set of valid requests contained in the subset, and evaluate the expected decision (cf. step vii).

The main advantage with this system is that it can evaluate a reverse query in real time. A further advantage with this system is that it can make the evaluation of a reverse query in real time even if the subset of requests contains a large number of requests.

Step ii in the method may be performed by (and corresponding modules in the system may be adapted for) studying each of the attributes appearing in the requests in the set (R) to discover whether equal or different values are assigned to them. Attributes with equal values in all requests are extracted. An attribute to which some requests assign values and some do not is preferably not extracted. Step ii may be refined further by forming subsets within the set (R), wherein a first group of attributes has identical values in a first subset, a second group of attributes - possibly overlapping with the first group - has identical values in a second subset, and so forth. The reduction will then lead to a corresponding number of simplified policies, so that the method bifurcates into several branches, the results of which are gathered to form the final result.

In step iv, the reduced policy is cached and forms a simplified policy. Since the simplified policy can be represented as a smaller set of processing instructions (code), it is in general more economical to evaluate. As far as the XACML context is concerned, it is noted that the simplified policy may be represented in the same form as the original policy. However, the simplified policy may in some cases require a richer representation than standard XACML, possibly including a number of new quasi-error states stemming from the fact that some attributes have not yet been evaluated (substituted). For instance, a situation may
arise in which a rule cannot be evaluated for lack of target or lack of values as-
sumed by the attributes appearing in a condition in the rule. The simplified policy
may then contain an indication that the rule is indeterminate, that is, temporarily
overriding the standard evaluation rules, which may specify for this situation that
an evaluation error is to be signaled. This is useful since it may turn out, when
combining algorithms in the policy are applied, that the sub-tree in which this rule
is located is inconsequential to the policy evaluation, so that this sub-tree may be
eliminated from the simplified policy. It is noted that if the simplified policy is not
represented in standard XACML, evaluation engines adapted for standard XACML
may need to be modified to evaluate a simplified policy.

(The concept of a simplified policy and the technique of partial evaluation
have been described in more detail in the applicant's earlier applications. Using
the terminology of these earlier applications, the initial request may be regarded as
a partial request, and the step of reducing may be regarded as a partial evaluation
of the policy.)

Step iv may be further refined by evaluating attributes, for which the val-
ues to be substituted are found using implicit references defined by values given in
a request together with the structure of the policy. For instance, the nationality of a
subject may be retrievable from a database using the name of the subject as a
key.

Step vi may be performed by means of a SAT solver, i.e., a functional enti-
ty implemented in hardware and/or software and configured to input a satisfiable
logic expression and to output, in response thereto, a solution (or a variable as-
signment, or a set of values of variables in the expression) for which the logic ex-
pression evaluates to true. Many SAT solvers are known in the art and available
from commercial or non-commercial providers. The form in which the logical ex-
pression is to be input may vary between different SAT solvers, and this is prefer-
ably taken into account when step v is implemented, so that compatibility can be
ensured. In general, an ABAC policy encoded in one of the customary or standard-
ized languages is not compatible. Some SAT solvers are configured to return one
solution even though the logic expression may evaluate true for several variable
assignments. It is desirable in step vi to exhaust the set of solutions; this may be
achieved invoking the SAT solver repeated times, wherein a condition excluding
the previous solution is added to the logical expression before a new repetition is
initiated. As SAT solvers are typically highly specialized for the task of finding a variable assignment satisfying the expression, step vi can be completed very efficiently.

Alternatively, step vi is carried out using techniques based on the theory of binary decision diagrams (BDDs) and Reduced-Order BDDs (ROBDDs), as outlined in B. Akers, "Binary Decision Diagrams", *IEEE Trans. Comp.*, vol. C-27, no. 6 (1978), p. 509 and R. E. Bryant, "Graph-based algorithms for Boolean function manipulation", *IEEE Trans. Comp.*, vol. C-35, no. 8 (1986), p. 677. A plurality of references in this area describe algorithms for translating a logic expression into a BDD or ROBDD, and there exist further algorithms for deriving exhaustive sets of solutions. These solutions may then be converted back into a form that will allow those access requests in the set (R) for which the policy yields the given decision to be extracted. Implementations of such algorithms as libraries of computer-executable code can be retrieved from commercial or non-commercial sources. For example, the library JavaBDD can be retrieved from [http://sourceforge.net](http://sourceforge.net), [http://sourceforge.net/projects/javabdd](http://sourceforge.net/projects/javabdd), [http://javabdd.sourceforge.net/](http://javabdd.sourceforge.net/) or archived versions of these pages. A further BDD package is BuDDy, downloadable from [http://buddy.sourceforge.net](http://buddy.sourceforge.net). Compared with the SAT-solver approach to carrying out step vi, the translation (step v) may require a slightly greater effort if BDDs are used; step vi may on the other hand execute more efficiently.

As the person skilled in computer science will realize when studying this disclosure, there are further options for implementing step vi. Since generally a faster evaluation happens at the price of a more involved translation process, and conversely, the selection of a particular implementation may depend on facts related to the intended application or use, including the complexity of the concerned ABAC policy and its degree of nesting.

A further advantage in this context is achieved if the system also comprises a policy information means which is connected to the policy decision means, and the conversion means and is operable to handle a set of attributes.

Furthermore, it is an advantage in this context if the partial request generation means also is operable to examine the subset of possible requests in order to determine (I) the set (D) of attributes that are associated with exactly the same set of values in all requests of the subset (R); (II) the set (A) of attributes that are
absent in all requests of the subset (R); and (III) the set (U) of all other attributes
not included in any of the sets (D or A) of attributes.

A further advantage in this context is achieved if the partial request generation
means also is operable to, by using the sets (D, A and U) of attributes, define
the partial request which (IV) associates to each attribute in the set (D) of attributes,
the exact set of values associated to it by any request in the subset (R); (V) marks all attributes in the set (A) as not present; and (VI) leaves all attributes in
the set (U) undefined.

Furthermore, with particular reference to step v, it is an advantage in this
context if the translation means also is operable to represent the simplified policy
with a tree structure, and to, whenever a sub-tree represents a Boolean expres-
sion that compares an attribute with a fixed value, replace the whole sub-tree by a
variable.

A further advantage in the context of step v is achieved if the translation
means also is operable to, from each condition node and downwards, whenever a
sub-tree represents a Boolean expression, but at least one of its children evalu-
ates to a non-Boolean value, replace the whole sub-tree by a variable.

Furthermore, it is an advantage in the context of step v if the system also
comprises a second storing means connected to the translation means, and to the
conversion means, and operable to store the correlation between each variable
and the sub-tree it has replaced.

A further advantage in the context of step v is achieved if the variables can
hold a value from the set, representing the values true, false and indeterminate.
This may be achieved by substituting two associated Boolean variables for each
such variable capable of assuming three values. As such, (1,1) may signify a true
state, (0,0) may signify a false state, and (0,1) may signify that the variable is in-
determinate. Similarly, for a tree (representing a policy or policy set) or sub-tree (rep-
resenting a subordinate policy or policy set), one may substitute two associated
Boolean values representing (1,0) - permit, (0,0) - deny, (1,1) - indeterminate,
(0,1) - not applicable.

Furthermore, it is an advantage in this context if the conversion means a l-
so is operable to, given a request r, determine if it fulfills any condition, and if it
does, then r is added to the set R of valid requests.
The above mentioned problems are also solved with a method for evaluating a reverse query, defining an expected decision, and a subset (R) of a set of possible requests, over a policy (P) in real-time. The method is performed with the aid of a system. The method comprises the steps:

- with the aid of a first storing means, comprised in the system, to store policies;
- with the aid of a partial request generation means, comprised in the system, to construct a partial request from the subset (R) of the set of possible requests;
- with the aid of a policy decision means connected to the partial request generation means, and to the first storing means, to partially evaluate the policy over the partial request resulting in a simplified policy;
- with the aid of a translation means, connected to the policy decision means, to translate the simplified policy, the subset of the set of possible requests, and the expected decision into a propositional logic formula;
- with the aid of an analyzing means connected to the translation means, to analyze the propositional logic formula in order to determine a sequence of conditions over requests; and
- with the aid of a conversion means, connected to the analyzing means, to convert the sequence of conditions to a set of valid requests contained in the subset (R), and evaluate to the expected decision.

The main advantage with this method is that it can evaluate a reverse query in real-time. A further advantage with this method is that it can make the evaluation of a reverse query in real-time even if the subset of requests contains a large number of requests.

A further advantage in this context is achieved if the method also comprises the step:

- with the aid of a policy information means, comprised in the system, and connected to the policy decision means, and the conversion means, to handle a set of attributes.

Furthermore, it is an advantage in this context if the method also comprises the steps:

- with the aid of the partial request generation means, to examine the subset (R) of possible request in order to determine
- (I) the set \( D \) of attributes that are associated with exactly the same set of values in all requests of the subset \( R \);
- (II) the set \( A \) of attributes that are absent in all requests of the subset \( R \); and
- (III) the set \( U \) of all other attributes not included in any of the sets \( D \) or \( A \) of attributes.

A further advantage in this context is achieved if the method also comprises the steps:
- with the aid of the partial request generation means, by using the sets \( D, A \) and \( U \) of attributes, to define the partial request, which
- (IV) associates to each attribute in the set \( D \) of attributes, the exact set of values associated to it by any request in the subset \( R \);
- (V) marks all attributes in the set \( A \) as not present; and
- (VI) leaves all attributes in the set \( U \) undefined.

Furthermore, it is an advantage in this context if the method also comprises the steps:
- with the aid of the translation means, to represent the simplified policy with a tree structure;
- whenever a sub-tree represents a Boolean expression that compares an attribute with a fixed value; and
- to replace the whole sub-tree by a variable.

A further advantage in this context is achieved if the method also comprises the step:
- with the aid of the translation means, from each condition node and downwards, whenever a sub-tree represents a Boolean expression, but at least one of its children evaluates to a non-Boolean value, to replace the whole sub-tree by a variable.

Furthermore, it is an advantage in this context if the method also comprises the step:
- with the aid of a second storing means, comprised in the system, and connected to the translation means, to store a correlation between each variable and the sub-tree it has replaced.

A further advantage in this context is achieved if the variables can hold a value from the set, representing the values true, false and indeterminate.
Furthermore, it is an advantage in this context if the method also comprises the steps:
- with the aid of the conversion means, given a request \( r \in R \), to determine if it fulfills any condition; and
- if it does, to add \( r \) to the set of valid requests.

The above mentioned problems are also solved with at least one computer program product. The at least one computer program product is/are directly loadable into the internal memory of at least one digital computer, and comprises software code portions for performing the steps of the method according to the present invention when the at least one product is/are run on the at least one computer.

The main advantage with this computer program product is that it can evaluate a reverse query in real-time. A further advantage with this product is that it can make the evaluation of a reverse query in real-time even if the subset of requests contains a large number of requests.

It will be noted that the term "comprises/comprising" as used in this description is intended to denote the presence of a given characteristic, step or component, without excluding the presence of one or more other characteristics, features, integers, steps, components or groups thereof.

**Brief description of the drawings**

Embodiments of the invention will now be described with a reference to the accompanying drawings, on which:

- Fig. 1 is a block diagram of a system operable to evaluate a reverse query, defining an expected decision, and a subset of a set of possible requests, over a policy in real-time according to the present invention;
- Fig. 2 is a flow chart of a method for evaluating a reverse query, defining an expected decision, and a subset of a set of possible requests, over a policy in real-time according to the present invention;
- Fig. 3 schematically shows a number of computer program products according to the present invention;
- Fig. 4 illustrates an exemplifying AC policy architecture; and
- Fig. 5 is a tree representation of an ABAC policy set.
Detailed description of the preferred embodiments

In fig. 1 there is disclosed a block diagram of a system 10 operable to evaluate a reverse query, defining an expected decision (d), and a subset (R) of a set of possible requests, over a policy (P) in real-time according to the present invention. The system 10 comprises a first storing means 12 operable to store policies. Furthermore, the system 10 also comprises a partial request generation means 14 operable to construct a partial request \( r_{par} \) from the subset (R) of the set of possible requests. As is apparent in fig. 1, the system 10 also comprises a policy decision means 16 connected to the partial request generation means 14, and to the first storing means 12, and operable to partially evaluate the policy (P) over the partial request \( r_{par} \) resulting in a simplified policy (P'). Furthermore, the system 10 also comprises a translation means 20 connected to the policy decision means 16, and operable to translate the simplified policy (P'), the subset (R) of the set of possible requests, and the expected decision (d) into a propositional logic formula (F). The system 10 also comprises an analyzing means 18 connected to the translation means 20, and operable to analyze the propositional logic formula (F) in order to determine a sequence \( [c_i, ..., c_k] \) of conditions over requests, preferably sufficient conditions. As also is apparent in fig. 1, the system 10 also comprises a conversion means 22 connected to the analyzing means 18, and operable to convert the sequence \( [c_i, ..., c_k] \) of conditions to a set of valid requests contained in the subset (R), and evaluate to the expected decision (d).

According to one alternative, the system 10 can also comprise a policy information means 30 operable to handle a set of attributes 32. The policy information means 30 is connected to the policy decision means 16, and to the conversion means 22. These connections are disclosed in fig. 1 with broken lines, because these elements are not mandatory in the system 10.

According to another alternative, the partial request generation means 14 is also operable to examine the subset (R) of the set of possible requests in order to determine (I) the set (D) of attributes that are associated with exactly the same set of values in all requests of the subset (R); (II) the set (A) of attributes that are absent in all requests of the subset (R); and (III) the set (U) of all other attributes not included in any of the sets (D or A) of attributes.
According to a further alternative, the partial request generation means 14 is also operable to, by using the sets (D, A and U) of attributes, define the partial request \( r_{\text{partial}} \), which (IV) associates to each attribute in the set (D) of attributes, the exact set of values associated to it by any request in the subset (R); (V) marks all attributes in the set (A) as not present; and (VI) leaves all attributes in the set (U) undefined.

According to yet another alternative, the translation means 20 is also operable to represent the simplified policy \( P' \) with a tree structure, and to, whenever a sub-tree represents a Boolean expression that compares an attribute with a fixed value, replace the whole sub-tree by a variable \( v_i \).

According to another alternative, the translation means 20 is also operable to, from each condition node and downwards, whenever a sub-tree represents a Boolean expression, but at least one of its children evaluates to a non-Boolean value, replace the whole sub-tree by a variable \( v_i \).

As also is apparent in fig. 1, the system 10 can also comprise a second storing means 24 operable to store the correlation between each variable \( v_i \) and the sub-tree it has replaced. The second storing means 24 is connected to the translation means 20, and to the conversion means 22. These connections are disclosed in fig. 1 with a broken line, because these elements are not mandatory in the system 10.

Furthermore, according to another alternative, the variables \( v_i \) can hold a value from the set \( \{ T, F, \perp \} \), representing the values true, false and indeterminate.

According to yet another alternative, the conversion means 22 is also operable to, given a request \( r \in R \), determine if it fulfills any condition \( c_i \), and if it does, then \( r \) is added to the set of valid requests.

In fig. 2 there is disclosed a flow chart of a method for evaluating a reverse query, defining an expected decision \( c_i \), and a subset \( R \) of a set of possible reverse requests, over a policy \( P \) in real-time according to the present invention. The method is performed with the aid of a system 10 (see fig. 1). The method begins at block 50. The method continues, at block 52, with the step: with the aid of a first storing means 12, comprised in the system 10, to store policies. Thereafter, the method continues, at block 54, with the step: with the aid of a partial request generation means 14, comprised in the system 10, to construct a partial request \( r_{\text{partial}} \).
tiai) from the subset (R) of the set of possible requests. The method continues, at
block 56, with the step: with the aid of a policy decision means 16 connected to the
partial request generation means 14, to partially evaluate the policy (P) over the
partial request \( (r_{\text{par}, \text{tiai}}) \) resulting in a simplified policy \( (P') \). Thereafter, the method
continues, at block 58, with the step: with the aid of a translation means 20, con-
connected to the policy decision means 16, to translate the simplified policy \( (P') \), the
subset (R) of the set of possible requests, and the expected decision (d) into a
propositional logic formula \( (F) \). The method continues, at block 60, with the step:
with the aid of an analyzing means 18 connected to the translation means 20, to
analyze the propositional logic formula \( (F) \) in order to determine a sequence \([c_i, \ldots, c_j]\) of conditions over requests. Thereafter, the method continues, at block 62,
with the step: with the aid of a conversion means 22, connected to analyzing
means 18, to convert the sequence \([c_i, \ldots, c_j]\) of conditions to a set of valid re-
quests contained in the subset (R), and evaluate to the expected decision (d). The
method is completed at block 64.

According to one alternative, the method also comprises the step: with the
aid of a policy information means 30, comprised in the system 10, and connected
to the policy decision means 16, and to the conversion means 22, to handle a set
of attributes 32.

According to another alternative, the method also comprises the steps:
with the aid of the partial request generation means 14, to examine the subset (R)
of possible requests in order to determine

- (I) the set \( (D) \) of attributes that are associated with exactly the same set of
  values in all requests of the subset (R);

- (II) the set \( (A) \) of attributes that are absent in all requests of the subset (R);

  and

- (III) the set \( (U) \) of all other attributes not included in any of the sets \( (D \text{ or } A) \)
of attributes.

According to yet another alternative, the method also comprises the steps:
with the aid of the partial request generation means 14, by using the sets \( (D, A \)
and \( U) \) of attributes, to define the partial request \( (r_{\text{par}, \text{tiai}}) \), which

- (IV) associates to each attribute in the set \( (D) \) of attributes, the exact set of
  values associated to it by any request in the subset (R);

- (V) marks all attributes in the set \( (A) \) as not present; and
- (VI) leaves all attributes in the set (U) undefined.

Furthermore, according to another alternative, the method also comprises the steps:
- with the aid of the translation means 20, to represent the simplified policy
  \((\mathcal{P'})\) with a tree structure;
- whenever a sub-tree represents a Boolean expression that compares an attribute with a fixed value; and
- to replace the whole sub-tree by a variable \((v_i)\).

According to a further alternative, the method also comprises the step:
- with the aid of the translation means 20, from each condition node and downwards, whenever a sub-tree represents a Boolean expression, but at least one of its children evaluates to a non-Boolean value, to replace the whole sub-tree by a variable \((v_i)\).

According to another alternative, the method also comprises the step:
- with the aid of a second storing means 24, comprised in the system 10, and connected to the translation means 20, and to the conversion means 22 to store a correlation between each variable \((v_i)\) and the sub-tree it has replaced.

According to yet another alternative, the variables \((v_i)\) can hold a value from the set \(\{T, F, -1\}\), representing the values true, false and indeterminate.

Furthermore, according to another alternative, the method also comprises the steps:
- with the aid of the conversion means 22, given a request \(r \in \mathbb{R}\), to determine if it fulfills any condition \(c_i\); and
- if it does, to add \(r\) to the set of valid requests.

The systems and methods disclosed hereinabove may be implemented as software, firmware, hardware or a combination thereof. In a hardware implementation, the division of tasks between functional units referred to in the above description does not necessarily correspond to the division into physical units; to the contrary, one physical component may have multiple functionalities, and one task may be carried out by several physical components in cooperation. Certain components or all components may be implemented as software executed by a digital signal processor or microprocessor, or be implemented as hardware or as an application-specific integrated circuit. Such software may be distributed on computer
readable media, which may comprise computer storage media (or non-transitory media) and communication media (or transitory media). As is well known to a person skilled in the art, the term computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by a computer. Further, it is well known to the skilled person that communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media.

In fig. 3, some computer program products 302₁, ..., 302ₙ according to the present invention are schematically shown. In fig. 3, n different digital computers 300₁, ..., 300ₙ are shown, where n is an integer. In fig. 3, n different computer program products 302₁, ..., 302ₙ are shown, here shown in the form of floppy/compact discs. The different computer program products 302₁, ..., 302ₙ are directly loadable into the internal memory of the n different computers 300i₁, ..., 300ₙ. Each computer program product 302ᵢ₁; ..., 302ₙ comprises software code portions for performing all the steps according to fig. 2, when the product/products 302ᵢ₁, ..., 302ₙ is/are run on the computers 300i₁, ..., 300ₙ. The computer program products 302₁, ..., 302ₙ may, for instance, be in the form of diskettes, RAM discs, magnetic tapes, magneto-optical discs or some other suitable products.

Figure 4 is a generalized block diagram of the XACML architecture 100, although simplified, according to the prior art. As stated before, XACML is an access control policy language. An attempt to access a resource 102 is represented as a "Request", which lists attributes of the subject 104, the resource 102, the action and the environment 106. An attribute is an identifier, a data type and a value. It can also be described as a variable with a name (the identifier), a data type and a value. Most facts relating to the subject 104, the resource 102, the action and the environment 106 can be described in terms of attributes.
The request is constructed by a PEP 108. The purpose of a PEP 108 is to
guard access to a resource 102 and only let authorized users through. The PEP
108 itself does not know who is authorized, but it submits the request to a PDP
110, which contain policies governing what requests are to be permitted or denied,
respectively. The PDP 110 evaluates the policies and returns a permit/deny re-
sponse to the PEP 108. The PEP 108 then either lets the access proceed or stops
it. As already noted, the PEP (Policy Enforcement Point) may provide only some
initial attribute values and the PDP (Policy Decision Point) or other components
may dynamically fetch more values from remote sources as they are needed. If all
necessary values cannot be retrieved, the policy evaluation may return an output
to the effect that the policy is indeterminate or not applicable in the circumstances,
or an error message.

A purpose of this architecture is to establish separation of concerns, that
is, to differentiate between policy decision making and policy enforcement. En-
forcement is by its nature specific to a particular resource 102, while a decision
engine can be made general-purpose and reusable.

In general, policies can be nested to form a policy set, as may be visual-
ized in a tree form of the type shown in figure 5. The combination of different poli-
cies is governed by combining algorithms, which define what policy takes prece-
dence over another policy. The node "PolicySet 1" in figure 5 is of the "deny-
overides" type, so that a "Deny" decision from one of the three sub-trees will take
precedence; hence, the "deny-overides" operator acts as logical AND. In contrast,
decisions produced by sub-trees which connect at a "permit-overrides" node are
subject to logical OR.

At the lowest level of the tree shown in figure 5, there are rules including
effects (e.g., "Deny") and underlying conditions formulated in terms of attributes,
for instance, "subject-nationality != 'US'", where "subject-nationality" is a subject
attribute and "US" is a constant. At several nodes above the lowest level, there are
conditions labelled "Target: ", which indicate the requests for which the sub-tree
having its root at that node is applicable. For example, if the condition "document-
stage = 'draft'" evaluates to "false", then Policy 3, including Rule 6, is not appli-
cable and can be excluded from evaluation.

Further embodiments
1. A system (10) operable to evaluate a reverse query, defining an expected decision (d), and a subset (R) of a set of possible requests, over a policy (P) in real-time, said system (10) comprising a first storing means (12) operable to store policies, wherein said system (10) also comprises a partial request generation means (14) operable to construct a partial request (r_{par \text{\text{iai}}}) from said subset (R) of said set of possible requests, a policy decision means (16) connected to said partial request generation means (14), and to the first storing means (12), and operable to partially evaluate said policy (P) over said partial request (r_{par \text{\text{iai}}}) resulting in a simplified policy (P'), a translation means (20) connected to said policy decision means (16), and operable to translate said simplified policy (P'), said subset (R) of said set of possible requests, and said expected decision (d) into a propositional logic formula (F), an analyzing means (18) connected to said translation means (20), and operable to analyze said propositional logic formula (F) in order to determine a sequence [c_i, ..., c_k] of conditions over requests, and a conversion means (22) connected to said analyzing means (18), and operable to convert said sequence [c_i, ..., c_k] of conditions to a set of valid requests contained in said subset (R), and evaluate to said expected decision (d).

2. A system (10) operable to evaluate a reverse query over a policy (P) in real-time according to embodiment 1, wherein said system (10) also comprises a policy information means (30) connected to said policy decision means (16), and said conversion means (22), and operable to handle a set of attributes (32).

3. A system (10) operable to evaluate a reverse query over a policy (P) in real-time according to embodiment 1 or 2, wherein said partial request generation means (14) also is operable to examine said subset (R) of possible requests in order to determine (I) the set (D) of attributes that are associated with exactly the same set of values in all requests of said subset (R); (II) the set (A) of attributes that are absent in all requests of said subset (R); and (III) the set (U) of all other attributes not included in any of the sets (D or A) of attributes.

4. A system (10) operable to evaluate a reverse query over a policy (P) in real-time according to embodiment 3, wherein said partial request generation means (14) also is operable to, by using said sets (D, A and U) of attributes, define said
partial request \( (r_{\text{partial}}) \), which (IV) associates to each attribute in said set \( (D) \) of attributes, the exact set of values associated to it by any request in said subset \( (R) \); (V) marks all attributes in said set \( (A) \) as not present; and (VI) leaves all attributes in said set \( (U) \) undefined.

5. A system \( (10) \) operable to evaluate a reverse query over a policy \( (P) \) in real-time according to any one of embodiments 1-4, wherein said translation means (20) also is operable to represent said simplified policy \( (P') \) with a tree structure, and to, whenever a sub-tree represents a Boolean expression that compares an attribute with a fixed value, replace said whole sub-tree by a variable \( (v_i) \).

6. A system \( (10) \) operable to evaluate a reverse query over a policy \( (P) \) in real-time according to embodiment 5, wherein said translation means (20) also is operable to, from each condition node and downwards, whenever a sub-tree represents a Boolean expression, but at least one of its children evaluates to a non-Boolean value, replace said whole sub-tree by a variable \( (v_i) \).

7. A system \( (10) \) operable to evaluate a reverse query over a policy \( (P) \) in real-time according to embodiment 5, or 6, wherein said system \( (10) \) also comprises a second storing means (24) connected to said translation means (20), and to said conversion means (22), and operable to store the correlation between each variable \( (v_i) \) and said sub-tree it has replaced.

8. A system \( (10) \) operable to evaluate a reverse query over a policy \( (P) \) in real-time according to embodiment 7, wherein said variables \( (v_i) \) can hold a value from the set \( \{T, F, \perp\} \), representing the values true, false and indeterminate.

9. A system \( (10) \) operable to evaluate a reverse query over a policy \( (P) \) in real-time according to any one of embodiments 5-8, when dependent on embodiment 5, wherein said conversion means (22) also is operable to, given a request \( r \in R \), determine if it fulfills any condition \( c_i \), and if it does, then \( r \) is added to said set of valid requests.
10. A method for evaluating, with the aid of a system (10), a reverse query, defining an expected decision (d), and a subset (R) of a set of possible requests, over a policy (P) in real-time, said method comprises the steps:
   - with the aid of a first storing means (12), comprised in said system (10), to store policies;
   - with the aid of a partial request generation means (14), comprised in said system (10), to construct a partial request (r_{pa,tia}) from said subset (R) of said set of possible requests;
   - with the aid of a policy decision means (16) connected to said partial request generation means (14), and to said first storing means (12), to partially evaluate said policy (P) over said partial request (r_{pa,tia}) resulting in a simplified policy (P');
   - with the aid of a translation means (20), connected to said policy decision means (16), to translate said simplified policy (P'), said subset (R) of said set of possible requests, and said expected decision (d) into a propositional logic formula (F);
   - with the aid of an analyzing means (18) connected to said translation means (20), to analyze said propositional logic formula (F) in order to determine a sequence [c_1, ..., c_k] of conditions over requests; and
   - with the aid of a conversion means (22), connected to said analyzing means (18), to convert said sequence [c_1, ..., c_k] of conditions to a set of valid requests contained in said subset (R), and evaluate to said expected decision (d).

11. A method for evaluating a reverse query over a policy (P) in real-time according to embodiment 10, wherein said method also comprises the step:
   - with the aid of a policy information means (30), comprised in said system (10), and connected to said policy decision means (16), and said conversion means (22), to handle a set of attributes (32).

12. A method for evaluating a reverse query over a policy (P) in real-time according to embodiment 10 or 11, wherein said method also comprises the steps:
   - with the aid of said partial request generation means (14), to examine said subset (R) of possible requests in order to determine
     - (I) the set (D) of attributes that are associated with exactly the same set of values in all requests of said subset (R);
     - (II) the set (A) of attributes that are absent in all requests of said subset (R); and
- (III) the set \( (U) \) of all other attributes not included in any of the sets \( (D \) or \( A ) \) of attributes.

13. A method for evaluating a reverse query over a policy \( (P) \) in real-time according to embodiment 12, wherein said method also comprises the steps:

- with the aid of said partial request generation means \( (14) \), by using said sets \( (D, A \) and \( U ) \) of attributes, to define said partial request \( (r_{partial}) \), which
- (IV) associates to each attribute in said set \( (D) \) of attributes, the exact set of values associated to it by any request in said subset \( (R); \)
- (V) marks all attributes in said set \( (A) \) as not present; and
- (VI) leaves all attributes in said set \( (U) \) undefined.

14. A method for evaluating a reverse query over a policy \( (P) \) in real-time according to any one of embodiments 10-13, wherein said method also comprises the steps:

- with the aid of said translation means \( (20) \), to represent said simplified policy \( (P') \) with a tree structure;
- whenever a sub-tree represents a Boolean expression that compares an attribute with a fixed value; and
- to replace said whole sub-tree by a variable \( (v_i) \).

15. A method for evaluating a reverse query over a policy \( (P) \) in real-time according to embodiment 14, wherein said method also comprises the step:

- with the aid of said translation means \( (20) \), from each condition node and downwards, whenever a sub-tree represents a Boolean expression, but at least one of its children evaluates to a non-Boolean value, to replace said whole sub-tree by a variable \( (v_i) \).

16. A method for evaluating a reverse query over a policy \( (P) \) in real-time according to embodiment 14, or 15, wherein said method also comprises the step:

- with the aid of a second storing means \( (24) \), comprised in said system \( (10) \), and connected to said translation means \( (20) \), and to said conversion means \( (22) \), to store a correlation between each variable \( (v_i) \) and said sub-tree it has replaced.
17. A method for evaluating a reverse query over a policy (P) in real-time according to embodiment 16, wherein said variables (v) can hold a value from the set \{T, F, L\}, representing the values true, false and indeterminate.

18. A method for evaluating a reverse query over a policy (P) in real-time according to any one of embodiments 14-17, when dependent on embodiment 14, wherein said method also comprises the steps:
- with the aid of said conversion means (22), given a request \( r \in R \), to determine if it fulfills any condition \( c \); and
- if it does, to add \( r \) to said set of valid requests.

19. At least one computer program product \( (102_1, \ldots, 102_n) \) directly loadable into the internal memory of at least one digital computer \( (100i, \ldots, 100_n) \), comprising software code portions for performing the steps of embodiment 10 when said at least one product \( (102_1, \ldots, 102_n) \) is/are run on said at least one computer \( (100_1, \ldots, 100_n) \).

Further embodiments of the present invention will become apparent to a person skilled in the art after studying the description above. Even though the present description and drawings disclose embodiments and examples, the invention is not restricted to these specific examples. For instance, the invention can be applied to control access to resources outside the context of computing; as an example, access to the premises in a building can be controlled if suitable identification means (e.g., card readers, biometric sensors, which identify a person as a subject in a guarding system) and actuators (e.g., electrically controllable door locks) are provided and are communicatively connected to a computer system for enforcing the AC policy. Numerous modifications and variations can be made without departing from the scope of the present invention, which is defined by the accompanying claims. Any reference signs appearing in the claims are not to be understood as limiting their scope.
1. A computer-implemented method for real-time evaluation of a reverse query to an attribute-based access control policy, ABAC policy, (P) which is enforced to control access to one or more resources in a computer network, said method comprising the steps of:
   i) receiving a reverse query indicating a given decision (d), which is one of permit access and deny access, and a set (R) of admissible access requests, each of which comprises one or more attributes appearing in the ABAC policy and explicit values assigned to these;
   ii) extracting attributes to which all access requests in the set (R) assign identical values;
   iii) reducing the ABAC policy at least by substituting values for the extracted attributes;
   iv) caching the policy after said reducing as a simplified policy (P');
   v) translating the cached simplified policy (P') and the given decision (d) into a satisfiable logic proposition in Boolean variables (ν, i = 1, 2, ...);
   vi) deriving all variable assignments (Cj = [ν, = xj1, ν2 = xj2, ...J, j = 1, 2, ...) satisfying the logic proposition; and
   vii) extracting, based on the variable assignments thus derived, all access requests from the set (R) for which the ABAC policy (P) yields the given decision (d).

2. The method of claim 1, wherein step ii includes examining the set (R) of access requests in order to determine:
   (I) a first set (D) of attributes that are associated with exactly the same set of values in all requests of said subset (R);
   (II) a second set (A) of attributes that are absent in all requests of said subset (R); and
   (III) a third set (U) of all other attributes not included in any of the sets (D or A) of attributes.

3. The method of claim 2, wherein step iii includes using said sets (D, A, U) of attributes to generate a partial access request, which:
(IV) assigns, to each attribute in said first set (D) of attributes, the exact set of values associated to it by any request in said set (R); and

(VI) leaves all attributes in said set (U) undefined.

4. The method of claim 3, wherein the partial access request further (V) marks all attributes in said set (A) as not present.

5. The method of any of the preceding claims, wherein step iv includes caching the simplified policy (P') represented with a tree structure,

   wherein step v includes, whenever a sub-tree in the tree structure represents a Boolean expression that compares an attribute with a fixed value, replacing said whole sub-tree by a variable (v,).

6. The method of any of the preceding claims, wherein step iv includes caching the simplified policy (P') represented with a tree structure,

   wherein step v includes, whenever a sub-tree in the tree structure represents a Boolean expression but at least one of its children evaluates to a non-Boolean value, replacing said whole sub-tree by a variable (v,).

7. The method claim 5 or 6, wherein, in step v, the variable replacing the sub-tree comprises two associated Boolean variables for representing more than two possible values of the sub-tree.

8. The method of claim 7, said possible values being "true", "false" and "indeterminate".

9. The method of any of the preceding claims, wherein step v includes storing a correlation between each variable (v,) and the sub-tree which it replaces.

10. The method of any of the preceding claims, wherein step vii includes assessing, for each request (r) in the set (R) of requests, whether it corresponds to
any of the variable assignments satisfying the logic expression; and, if it does, extracting this access request.

11. The method of any of the preceding claims, wherein step vi is performed by means of a SAT solver.

12. The method of any of claims 1 to 10, wherein step v includes representing at least part of the simplified policy (P') as a binary decision diagram; and wherein step vi includes deriving all paths in the binary decision diagram evaluating to either true or false.

13. The method of any of the preceding claims, wherein step iii includes:

determining an implicit reference defined by a value of one of said extracted attributes together with the ABAC policy; and

fetching, in accordance with this implicit reference, at least one attribute value from a remote source.

14. A computer program product comprising a data carrier storing computer-executable instructions for performing the method of any of the preceding claims.

15. A computer system (10) configured for real-time evaluation of a reverse query to an attribute-based access control policy, ABAC policy, which is enforced to control access to one or more resources in a computer network, wherein the reverse query indicates a given decision (d), which is one of permit access and deny access, and a set (R) of admissible access requests, each of which comprises one or more attributes appearing in the ABAC policy and explicit values assigned to these, said computer system comprising:

a data memory (12) operable to store one or more ABAC policies;

a partial request generation means (14) operable to construct, based on said set (R) of admissible access requests, a partial request \( r_{\text{partial}} \) assigning values only to attributes associated with identical values throughout the set (R) of admissible requests;
a policy decision means (16) connected to said partial request generation means (14) and to the electronic storing means (12), and operable to evaluate said policy (P) for said partial request (r_{par, tial}), thereby yielding a simplified policy (P');

a translation means (20), connected to said policy decision means (16) and operable to translate said simplified policy (P'), said subset (R) of said set of possible requests, and said given decision (d) into a satisfiable logic proposition (F) in Boolean variables (v_i, i = 1, 2, ...);

an analyzing means (18), connected to said translation means (20) and operable to analyze said propositional logic formula (F) in order to determine a sequence ([c_1, ..., c_k]) of conditions over requests, each condition defining a variable assignment (c_i = [v_1 = x_1, v_2 = x_2, ...], j = 1, 2, ...); and

a conversion means (22) connected to said analyzing means (18), and operable to extract, based on said sequence [c_1, ..., c_k] of conditions, valid requests contained in said subset (R) which evaluate to said given decision (d).
Fig. 4
Fig. 5
### INTERNATIONAL SEARCH REPORT

**PCT/SE201 1/050955**

#### A. CLASSIFICATION OF SUBJECT MATTER

**IPC:** see extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

#### B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

**IPC:** G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE, DK, FI, NO classes as above

Electronic database consulted during the international search (name of database and, where practicable, search terms used)

EPO-Internal, PAJ, WPI data, COMPENDEX, INSPEC

#### C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>US 20070078840 A1 (STERN AARON A ET AL), 5 April 2007 (2007-04-05); abstract; paragraphs [0016]-[0019]; claim 1</td>
<td>1-15</td>
</tr>
<tr>
<td>X</td>
<td>WO 20061 18659 A2 (MICROSOFT CORP), 9 November 2006 (2006-11-09); abstract; paragraphs [0005]-[0008], [0033]-[0037]; claims 1-9</td>
<td>1-15</td>
</tr>
</tbody>
</table>

**X** Further documents are listed in the continuation of Box C.  

**X** See patent family annex.

---


d “T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

e “E” earlier application or patent but published on or after the international filing date

c “L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

---

Date of the actual completion of the international search: 16-04-2012

Date of mailing of the international search report: 17-04-2012

Name and mailing address of the ISA/SE

Patent och registreringsverket

Box 5055

S-1 02 42 STOCKHOLM

Facsimile No. +46 8 666 02 66

Authorized officer

Hanna Jansdotter

Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 2009)
### DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 201 001 53695 A1 (BUSSARD LAURENT ET AL), 17 June 2010 (2010-06-17); abstract</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 20090265780 A1 (KORKUS OHAD ET AL), 22 October 2009 (2009-10-22); abstract</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>WO 201 015 1493 A2 (MICROSOFT CORP), 29 December 2010 (2010-12-29); abstract</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Continuation of: second sheet
International Patent Classification (IPC)

G06F 17/30 (2006.01)
G06F 21/00 (2006.01)

Download your patent documents at www.prv.se
The cited patent documents can be downloaded:
  • From "Cited documents" found under our online services at www.prv.se (English version)
  • From "Anforda dokument" found under "e-tjanster" at www.prv.se (Swedish version)

Use the application number as username. The password is VXQQLPMWNG.

Paper copies can be ordered at a cost of 50 SEK per copy from PRV InterPat (telephone number 08-782 28 85).

Cited literature, if any, will be enclosed in paper form.
<table>
<thead>
<tr>
<th>Country</th>
<th>Application Number</th>
<th>Filing Date</th>
<th>Classification</th>
<th>Filing Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>20070078840</td>
<td>05/04/2007</td>
<td>A1</td>
<td>NONE</td>
</tr>
<tr>
<td>US</td>
<td>20070078874</td>
<td>05/04/2007</td>
<td>US</td>
<td>74721 30</td>
</tr>
<tr>
<td>WO</td>
<td>20061 18659</td>
<td>09/1 1/2006</td>
<td>US</td>
<td>7533088</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>US</td>
<td>2006025341</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>US</td>
<td>2008001 6033</td>
</tr>
<tr>
<td>US</td>
<td>201 001 53695</td>
<td>17/06/2010</td>
<td>WO</td>
<td>201 0074855</td>
</tr>
<tr>
<td>US</td>
<td>20090265780</td>
<td>22/1 0/2009</td>
<td>NONE</td>
<td>07701 9</td>
</tr>
<tr>
<td>WO</td>
<td>201 015 1493</td>
<td>29/1 2/2010</td>
<td>AR</td>
<td>201 0263086</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AU</td>
<td>2761 970</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TW</td>
<td>201 10 1769</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>US</td>
<td>201 00332601</td>
</tr>
</tbody>
</table>