2013 IEEE 25th International Conference on Tools with Artificial Intelligence

Explanations and Relaxations for Policy Conflicts in Physical Access Control

Fatih Turkmen, Simon Foley,
Barry O'Sullivan, William Fitzgerald
Cork Constraint Computation Centre
University College Cork
Cork, Ireland

Tarik Hadzic, Stylianos Basagiannis,
Menouer Boubekeur
United Technologies Research Center Ireland
United Technologies Corporation
Cork, Ireland

***Abstract*—Physical access control policies define sets of rules that govern people's access to physical resources such as rooms and buildings. While simple decision-precedence can be used to reconcile different rules that result in conflicting access decisions, the presence of rule conflicts and other rule anomalies can make it difficult for a policy-administrator to comprehend and effectively manage complex policies.**

In this paper we are concerned with discovering conflicts and computing relaxations of access policies in order to eliminate conflicting rule instances. We propose several SAT-based encodings in which these rule conflicts and anomalies are expressed as explanation style problems. Relaxation techniques are in turn used to eliminate these anomalies by recommending what rules have to be revoked or what permissions have to be removed from which rules. Moreover, we discuss a relaxation strategy that preserves most of the access constraints of the original policy. Finally we provide a preliminary performance study of our techniques. Our approach is applicable to access control policies in general.

***Keywords*-Access Control; Policy; Relaxations;**

I. Introduction

Access Control deals with the protection of sensitive resources from unauthorized access. Resources subject to access control could be files and directories in an LDAP directory, protected network ports in a firewall configuration or physical areas protected by physical access control (PAC) systems (involving card readers, electronic locks, alarms etc). In large organizations there could be thousands of files and physical locations to which an access needs to be regulated.

Large organizations such as universities, hospitals or airports can be organized into logical divisions to meet requirements of enterprise policy and the users of the resources are organized into groups for the ease of administration. For instance, a hospital may introduce a division that constitutes operating rooms, organize a set of hospital personnel under the group "surgery" and provide necessary access control in a PAC policy. IT security administration may put files containing sensitive technical information subject to Export Control regulations into a separate folder (or a server) and introduce an "export control required" group, so that all members of the group are not allowed to access folder with export controlled material. In general, we can think of the access policy as a collection of group-based rules which determine whether particular groups of subjects have access to particular groups of resources.

However, the requirements imposed from the enterprise level, the decision choices made to enable easy administration and the current configuration of access control equipment may result with unintended effects, i.e. anomalies, in the access control policy. In particular, rule *conflicts* may occur in situations where for a given subject, one rule permits and another denies access to the same resource. The access control community has developed a number of approaches for conflict resolution, typically in the form of rule *combinators* which determine strategies for aggregation of conflicting decisions, e.g. "deny-overrides", "allow-overrides" or "first applicable" (typically used in network access control). Apart from discovery, finding explanations to an anomaly with reasonable complexity and providing appropriate resolutions to them is an important challenge in access control policy analysis. A policy administrator can be aided with the tools that will help him understand the cause of the anomaly and choose a (optimal) configuration that is anomaly-free. Our observation is that this problem is quite similar to configuration problems existing in constraint satisfiability problem (CSP) research. One could consider a PAC policy as a set of constraint specifications and anomalies as conflicts. With this mapping, existing conflict explanations and constraint relaxation techniques can be applied for preferred policy configurations. Moreover, such an approach leads to several different possibilities on PAC policy encodings such as preserving the operational behaviour or enabling the specification of the rule preferences while eliminating conflicts.

II. Background

A. Group-Based Access Control Model

In our previous work [6], we presented a group-based access control model tailored to physical security and identified several anomalies[1] that may arise in policy specifications. In the model, some subsets of users ($User$) and resources ($Resource$) represent user groups $UGroup = \mathcal{P}(User)$ and

[1]We use the terms anomaly and conflict interchangeably in the paper

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.57

resource groups $RGroup = \mathcal{P}(Resource)$ respectively. The elements of $UGroup$ and $RGroup$ reflect the logical or physical organization of users such as lecturers at a university and resources such as artificial intelligence division in a research lab. The model supports the specification of negative authorizations where a user is not only *allowed* access but also explicitly *denied* access to a resource. Thus possible access control decisions are denoted and given by $Action ::= allow \mid deny$. The benefits of an access control model with negative authorizations have been discussed in the literature widely [1], [11]. Moreover, enabling the specification of negative authorizations allows easy integration with other access control models with more expressive syntax such as Role-based access control (RBAC)[21] and Attribute-based Access Control (ABAC)[12].

An access-control rule is a triple : $Rule = UGroup \times RGroup \times Action$ and a policy is a set of rules for a building, $Policy = \mathcal{P}(Rule)$.In some parts of the paper, a rule is represented by using its atomic permissions (authorizations) from the set $A = User \times Resource$. An atomic permission $a(u, r)$ denotes the authorization state of user u and resource r. By expanding each rule r with the Cartesian product of its group and resource elements, we obtain $r.A = \{a_i, \ldots, a_j\}$, the set of atomic permissions associated with r.

Conflicts in PAC Policies: In this paper, our particular focus is the exception anomaly. In the exception anomaly ($\sqcap^{EA}$)), one rule denies (or allows) an authorization that is allowed (resp. denied) by another rule. More formally, given a pair of rules r and s, an exception anomaly (denoted as $r \sqcap^{EA} s$) is defined as follows;

$$r.Action \neq s.Action \wedge r.A \cap s.A \neq \emptyset$$

An exception anomaly is either a sign of simple availability violation, i.e. a user is denied access to a room she is authorized, or simple safety violation, i.e. a user is allowed access to a room she is unauthorized.

Example Policy: Consider that, in the model presented above, we have the following rules:

$$\begin{aligned} r_1 &= \langle \{u1, u2, u3, u4\}, \{df1\}, Allow \rangle \\ r_2 &= \langle \{u1, u3\}, \{df2, df3\}, Deny \rangle \\ r_3 &= \langle \{u2, u3\}, \{df6, df3, df5\}, Allow \rangle \\ r_4 &= \langle \{u2, u4\}, \{df6, df4\}, Allow \rangle \\ r_5 &= \langle \{u3\}, \{df5\}, Deny \rangle \end{aligned}$$

Here df denotes the resources in the physical domain, the "door faces" which uniquely determine the input/output readers in the building. As presented in Section II-A, the users of a rule are part of a group that is created according to their logical or administrative organization. In our example policy, each rule refers to a different group and we have exception anomalies between r_2 and r_3 ($r_2 \sqcap^{EA} r_3$), and r_3 and r_5 ($r_3 \sqcap^{EA} r_5$).

B. Explanations and Relaxations in Constraint Programming

Explanations and relaxations have been mainly considered in the context of interactive product configuration in which a list of options for product selection is given to the customer. Due to customer's budget limitations, an optimum combination of options (according to a cost function) is sought according to user preferences. An over-constrained problem ($\mathcal{P}$) is denoted by the pair $\langle \mathcal{B}, \mathcal{C} \rangle$ where $\mathcal{B}$ is a set of constraints that can not be relaxed, $\mathcal{C}$ is the set of soft constraints. A minimal conflict refers to a subset C of $\mathcal{C}$ in which a removal of a constraint from C makes it consistent. A maximal relaxation R of $\mathcal{P}$ is also a subset of $\mathcal{C}$ in which the addition of a constraint to R makes it inconsistent. Moreover, a conflict C has minimum cardinality if there is no other conflict C^* such that $|C^*| < |C|$ and a relaxation R has maximum cardinality if there is no other relaxation R^* such that $|R^*| > |R|$.

There is a strong duality between the computation of explanations and relaxations. While the literature suggests different explanation algorithms, we discuss our problem by using the QuickXplain algorithm from [13]. The basic explanation algorithm starts with the construction of a relaxation by iteratively removing a constraint α from $\mathcal{C}$ and adding to a set $\mathcal{R}$ after an inconsistency check on the set $B \cup \mathcal{R} \cup \{\alpha\}$ is reached. During the computation of a relaxation when an inconsistency is reached at the constraint α, then it is easy to see that an explanation is a subset of the set $\mathcal{R} \cup \{\alpha\}$.

Explanation algorithms have a pivotal function, π, that encapsulates the inconsistency check of a given set of constraints. Specifically, given a set of constraints C;

$$\pi : C \rightarrow \{true, false\}$$

Notice that the function π can perform consistency checks in different forms such as cost function optimization in an optimization problem or conflicting clauses in a propositional satisfiability problem. Hence, its actual functionality is dependent on the type of constraints and it will be discussed in the following sections.

III. Related Work

Anomaly detection and resolution in security policies have been widely discussed in the literature and various techniques have been proposed. Some of the earliest works include [18] and [14] where conflicts are described both syntactically and semantically at a high level and possible resolution strategies are presented. The former proposes to define precedence relations between policies for resolution while the latter uses graph transformations to eliminate conflicts on policies that are specified by using graphs.

Anomaly analysis has been a particular interest in network security research. Various anomalies that can arise in *firewall policies* have been identified and custom algorithms have been proposed for discovery and resolution of the identified

anomalies [2], [8]. Our policies and conflict definitions show similarities to firewall policies but differ in the application domain.

Physical security where only authorized members of an organization are enabled physical access to hardware protected resources is a relatively new research area. Among the few papers available in the literature, [7] presents a formal framework, Access Nets, to model access control in physical spaces. They show how model checking techniques can be applied for verifying physical access control policies by modelling them as a state transition system. The transitions from one state to the other is obtained by several rules including clock ticks. A property such as a violation of access to a room at a given time period can be verified through state exploration.

Within the context of access control, perhaps the closest work to ours is presented in [9] in which the authors discuss anomaly detection and resolution for XACML [19]. The authors employ binary decision diagrams (BDD) for representing the state of a policy or a policy set. Starting from an empty state, they incrementally detect and resolve anomalies at each new rule addition. In our work, we focus on explaining and resolving the anomalies by casting the problem of anomaly detection and resolution as an over-constrained CSP. While we provide minimal explanations for the anomalies and propose possible (maximal) resolutions to them, we leave the final decision to the policy administrator to prevent any lost of intended semantics.

Finally, security analysis of access control systems is an active area of research. Most of the prior work has centered around safety analysis [15] and checking certain security properties of RBAC [16] and eXtensible Access Control Language (XACML) [5], [10]. The main focus in this line of research was to check whether a given model specific access control setting satisfies a given property specification statically and not to explain or refine the access control policies due to some errors.

IV. Explanations and Relaxations for Policy Conflicts

We will refer to the problem of explaining conflicts and relaxing policies for eliminating them as *exception problem* ($ExP(\mathcal{B}, F)$). Notice that each instance of ExP for the same PAC policy is characterized by the constraint definitions. In this section we present two different problem instances for the exception anomaly. The first instance contains only the rules as constraints while the second considers also the authorizations that cause the exception anomaly. We call these two problem instances as *rule-level* and *permission-level* explanations, and provide the constraint encodings in the following sections. Our constraints are specified as part of a SAT formula (i.e. a set of clauses) with hard and soft clauses.

A. Rule-Level Explanations

Rule-level explanations present a coarse-grained view to the conflicts. They contain clauses that encode rule-authorization associations as background constraints ($\mathcal{B}$) and unit clauses that represent rules as soft constraints ($\mathcal{C}$). More formally, for each rule $r \in P$, let x_r and x_a be propositional variables denoting rule r and authorization a respectively, we generate a set of clauses, C_R, as the background constraints.

$$c = \begin{cases} \bigwedge_{a \in r.A} \neg x_r \vee x_a, & \text{if } r.Action = Allow \\ \bigwedge_{a \in r.A} \neg x_r \vee \neg x_a, & \text{if } r.Action = Deny \end{cases}$$

As a result, the exception problem instance for rule-level explanations is given as $ExP(C_R, \{x_{r_1} \dots x_{r_n}\})$. Let $\mathcal{F}$ denote the SAT formula representing an ExP. It is easy to see that a rule-level minimal conflict of the formula $\mathcal{F}$ is a pair of unit clauses representing two rules that conflict with each other. We call the conflicting rules as *rule-level minimal policy conflict* and it is also a cardinality minimal conflict. A maximal relaxation of $\mathcal{F}$ is a subset of unit clauses $\{x_{r_1} \dots x_{r_m}\}$ from $\{x_{r_1} \dots x_{r_n}\}$ such that any addition of a unit clause denoting a rule makes $\mathcal{F}$ inconsistent. From the maximal relaxation of $\mathcal{F}$, we define maximal policy relaxations (MPR).

Definition 1 (Rule-level MPR): Given a PAC policy P, a rule-level maximal relaxation R of a problem $ExP(C_R, \{x_{r_1} \dots x_{r_n}\})$ is a subset of rules $R \subseteq P$ such that addition of any rule $r \in P \setminus R$ to R introduces a conflict in the policy.

In [20] the worst-case number of maximal relaxations has been observed as $\binom{n}{n/2}$. For our example, the minimal conflicts are unit clauses $\{x_{r_2}, x_{r_3}\}$ and $\{x_{r_3}, x_{r_5}\}$. In what follows, we will sometimes refer to these unit clauses as rules. A rule-level MPR is a possible (among many) policy encoding that eliminates some of the rules that appear in conflicts. For our example policy configuration, example maximal relaxations include $\{x_{r_1}, x_{r_2}, x_{r_4}, x_{r_5}\}$ and $\{x_{r_1}, x_{r_3}, x_{r_4}\}$.

Definition 2 (Rule-level cardinality MPR): Given a PAC policy P, a rule-level cardinality maximal relaxation of a problem $ExP(C_R, \{x_{r_1} \dots x_{r_n}\})$ is a set of rules $R \subseteq P$ such that there is no other maximal relaxation of rules $R^* \subseteq P$ such that $|R^*| > |R|$. A cardinality maximal relaxation can be found by performing a single maximum satisfiability (MaxSAT) solving for π function.

A cardinality MPR is an anomaly-free policy encoding that preserves most of the rules from the original policy. Notice that the maximal relaxation $\{x_{r_1}, x_{r_2}, x_{r_4}, x_{r_5}\}$ is a cardinality maximal relaxation while $\{x_{r_1}, x_{r_3}, x_{r_4}\}$ is not cardinality maximal.

B. Permission-Level Explanations

While rule-level explanations can provide a high-level view of the conflicting rule pairs, an administrator may be interested in a more granular explanation and relaxation of exceptions. In addition to conflicting rule pairs, a

permission-level explanation also contains the authorization that causes the conflict. In order to encode permission-level explanations in SAT, we use the concept of selector variables. A selector variable is a propositional variable that allows enabling and disabling of clauses. For instance, given a clause c in a propositional encoding and y a selector variable, the formula $\neg y \vee c$ enforces c to be true only when y is set to true.

With the selector variable approach, certain clauses can be selected and grouped together to have more meaningful explanations. In our case, we can use a similar encoding to rule-level explanations for rule-authorization associations but introduce a selector variable for each clause. Hence we generate the clauses representing rule-permission associations, denoted as C_A, by using the following encoding:

$$c = \begin{cases} \neg x_r^a \vee \neg x_r \vee x_a & \text{if } r.Action = Allow \\ \neg x_r^a \vee \neg x_r \vee \neg x_a & \text{if } r.Action = Deny \end{cases}$$

Let x_r^a be a selector variable for representing the association between rule r and authorization a and f_r denote the boolean formula $x_r^{a_1} \wedge \ldots \wedge x_r^{a_k}$ for rule $r.A = \{a_1, \ldots, a_k\}$.

Definition 3 (Permission-level Policy Explanations): Given a PAC policy P, permission-level explanations are obtained from the explanation problem instance $ExP(\mathcal{B}, F_A)$ such that the background constraints are $\mathcal{B} = C_A \wedge x_{r_1}, \ldots, \wedge x_{r_n}$ and soft constraints are $F_A = f_{r_1} \wedge, \ldots, \wedge f_{r_n}$.

Notice that the formula representing the set of soft constraints is obtained from the conjunction of selector variables for each rule. Similar to rule-level minimal conflict, a permission-level minimal conflict of $\mathcal{F}$ is a pair of unit clauses (x_r^a, x_s^a) representing the association of a to rules r and s. It is also cardinality minimal conflict. In terms of PAC policies, let RA denote pairs between rules and their permissions such that $RA \subseteq P \times A$. Then a *permission-level minimal policy conflict* is a pair $((r, a), (s, a))$ from RA where $r, s \in P$.

Definition 4 (Permission-level MPR): Given a PAC policy P, a permission-level MPR, R, of a problem $ExP(\mathcal{B}, F_A)$ is a subset R of RA such that any permission $a \in r.A$ in the rule r where $(r, a) \in RA \setminus R$ introduces a conflict in the policy.

Definition 5 (Permission-level cardinality MPR): Given a PAC policy P, a permission-level cardinality MPR, R, of a problem $ExP(C_R, \{x_{r_1} \ldots x_{r_n}\})$ is a set of rule-permission associations $R \subseteq RA$ such that there is no other maximal relaxation of rules $R^* \subseteq RA$ such that $|R^*| > |R|$. The cardinality maximal relaxation can be found by performing a single maximum satisfiability (MaxSAT) solving for π function.

Compared to rule-level cardinality MPR, permission-level cardinality MPR presents a more granular resolution to conflicts in the PAC policy. It introduces some minimum number of changes to rules that are involved in conflicts.

V. Automated Optimal Resolutions with Relaxations

Relaxations refer to alternative policy encodings that are anomaly-free. In majority of the cases, the intended semantics of a policy specification is not provided formally and is unavailable to the resolution process a priori. Different strategies can be followed when proposing alternative policy encodings. There has been different proposals to anomaly resolution in policy analysis. For instance, [9] requires an access control decision (i.e. allow or deny) from the user for each conflicting authorization segment. An alternative technique is available in [4] where the correct authorization setting is learned from past actions (i.e. logs of access).

In this section we present a different strategy that preserves operational behavior of a PAC policy while resolving anomalies with relaxations. In what follows, we extend the respective Exp problem instances defined in Section IV for this purpose and present an algorithm for post processing the relaxation.

A. Preserving Operational Behaviour

In this strategy, the policy resulting from anomaly resolution must maintain the operational behaviour of the original PAC system. More specifically, a negative authorization (deny) must remain negative and a positive authorization (allow) must remain positive after relaxation. Moreover, there can be authorization requests that are covered by the policy but responded by the default behaviour of the PAC system. Even though it is possible to allow access by default in most of the commercially available PAC systems, we assume that the default behaviour of a door is deny. This is a reasonable assumption and it simplifies our problem as we can now consider that an authorization is either positive or negative operationally. Note that, it is easy to enumerate all positive and negative authorizations by simply querying the system that operates based on the precedence $deny \succ allow$.

Definition 6: (Operational PAC Policy and Equivalence). Given a PAC policy P, the respective operational policy, denoted as $P^{\triangleright}$, is a pair $\langle A^+, A^- \rangle$ that divides the total permission space A into disjoint authorization sets that contain positive, and negative authorizations. We say that two policies P and Q are *operationally equivalent*, denoted as $P \equiv Q$ if the following conditions hold:

$$P^{\triangleright}.A^+ = Q^{\triangleright}.A^+ \quad \wedge \quad P^{\triangleright}.A^- = Q^{\triangleright}.A^-$$

In an ideal policy, the intersection of authorization spaces should be empty, $A^+ \cap A^- = \emptyset$ so that the given policy specification is equivalent to its operational counterpart. When relaxing a policy with operational behaviour preservation we are interested in maintaining the authorization set A^+ and A^- intact. We can achieve this by adding a unit clause to background constraints of permission-level explanation problem for each authorization. Let x_a denote an authorization $a \in A$. The background constraints $\mathcal{B}'$

for the operational behaviour preserving permission-level relaxations are obtained from background constraints $\mathcal{B}$ of permission-level explanations as follows:

$$\mathcal{B}' = \mathcal{B} \wedge \bigwedge_{a \in A^+} x_a \wedge \bigwedge_{a \in A^-} \neg x_a$$

Obviously, any solution to the relaxation problem RexP($\mathcal{B}'$, $\mathcal{C}$) is a possible policy encoding that does not consider the logical organization of rules. The solution suggest modifications to some rules (i.e. authorization removal) which in return presents a consistent policy that preserves most of the policy intact. However, one could be interested in maximizing the preservation of logical structures of rules, i.e. minimizing the number of new rules of groups, by further processing a maximal solution to RexP. For this purpose we define an operation ($\ominus$) that removes a permission from a rule in an optimized way.

Definition 7: (Safe Permission Removal). An authorization a can be safely removed from a rule $r \in P$, denoted as $r.A \ominus a$, if the removal of r from P and addition of rule/s obtained from $(r.A \ominus a)$ to P keep the operational policy of P intact, i.e. $P \setminus \{r\} \cup \{(r.A \ominus a)\} \equiv P$.

By using Definition 7, multiple objectives can be easily integrated when proposing modifications to rules that appear in conflicts. We will present one possible algorithm for the implementation of $\ominus$ where we minimize the number of *rules*, *groups* and *resource groups* respectively. The algorithm considers authorizations ($User \times Resource$) as a table where rows are users and columns are door faces. Moreover, the cardinality MPRs (abbreviated as *sol*) of an explanation problem instance is represented as a mapping function, $sol : Rule \rightarrow \mathcal{P}A$.

Step 1 Identify the rows (i.e. users) without any authorization to be removed and create a rule (succeeding rule r') from the respective users and door faces.

Step 2 Among the remaining rows, group rows that have the same columns to be removed and order them according to number of common columns (i.e. door faces). For each group of rows, create a new rule with the rest of the columns.

Step 3 Create a new rule for each remaining row from Step 2.

Figure 1 illustrates the execution of the algorithm on a rule r given on the left for possible authorization removals. Both examples result with two rules due to our optimization strategy on the number of rules.

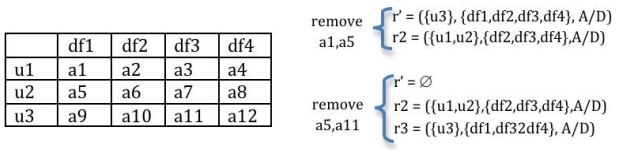

Figure 1. Resolution Example

VI. Evaluation

We have performed some preliminary experiments to obtain indications on computational costs of finding anomalies (i.e. conflicts) and their resolutions with relaxations. Our experimental test-beds include both randomly generated synthetic policies that vary in size and real-world policies from our university (UCC). Synthetic policies contain disjoint user/resource pairs so that they do not contain anomalies initially. In the experiments with UCC policies, we determined three different PAC instances according to logical and physical organization of the university: UCC-PL, UCC-NU and UCC-CS. The details of the policies are summarized in the Table I. As shown in the table UCC policies are rather flat in organizing resources. There is only one resource (door face) in each resource group. Note that our UCC policies contain only allow rules initially.

Table I
Policy sets for the experiments

	#Rule	#User	#Resource	#UGroup	#RGroup
Synthetic-1	100	500	20	20	15
Synthetic-2	500	2500	60	100	45
Synthetic-3	1000	5000	100	200	75
UCC-PL	42	1425	6	17	6
UCC-NU	208	1907	4	52	4
UCC-CS	236	1342	35	74	35

According to the binary decision diagram (BDD) based technique proposed in [9], an important factor of the performance of anomaly detection and resolution is the number of disjoint authorization spaces. Disjoint authorization spaces are obtained from rules or their fractions that are applicable to distinct requests. The list of disjoint authorization spaces can be considered as a policy encoding (i.e. rules) in which the rules specify disjoint permission sets. Thus we list the number of disjoint authorization spaces associated with each policy setting in our results. Other important factors include the number of rules and permissions (users and door face pairs) as they determine the size of SAT formula.

While the computation of explanations and relaxations in the context of SAT is an active area of research and there are multiple computations of interest, we consider two of them more relevant in our context: enumeration of conflicts and computation of cardinality MPRs. Enumeration of conflicts, known as minimally unsatisfiable subset (MUS) in SAT is inherently complex when completeness is sought. Thus different algorithms have been proposed that enable a trade-off between completeness and quicker MUS enumeration. We employed one such tool, MARCO [17], that enumerates most of the MUSes quickly at first and then computes maximally satisfiable subsets (MSS) to check if there remains any MUS not found. MARCO enables the specification of a limit on the number of obtained subsets, either MUS or MSS, and when that limit is reached the solver returns all found subsets and terminates. In our

experiments we exploit this feature and set the subset limit to 40 for all experiments in order to keep a balance between computational complexity and completeness. For computing cardinality MPRs (MaxSAT) we used a solver, IncMaxSatz, from Sixth MaxSAT competition [3].

The experiments were conducted on a Linux computer with Intel-Core 3.40GHz processor and 8GBs of memory.

Conflicts: To consider the effect of anomaly existence in a given policy, we introduced a fixed number of new deny rules in the policies. When adding new deny rules, first a new user group is created, if none exists already, that contains one randomly selected user from the existing allow rules. Then a random resource group is selected from the existing resource groups and a deny rule is constructed by using the selected user/resource group pairs. For the experiments with synthetic policies, these two steps allowed us to keep the number of conflicts (MUSes) in the respective SAT formula within a range. More specifically, given n deny rules introduced we expect n conflicts for rule-level conflicts, and a minimum of n and a maximum of $2n$ conflicts in synthetic policies.

For UCC policies, keeping the number of conflicts under control is relatively difficult since there are overlapping user groups. Because resource groups contain only a single resource in UCC policies, the addition of a deny rule corresponds to the introduction of a single negative authorization that may conflict with several other rules. As a result, the number of conflicts is unknown due to random construction of a deny rule and the overlapping user groups in policies. To cope with this issue, we considered 40 as a reasonable limit on the number of subsets (MUS or MSS) for enumerating conflicts. By setting a limit on the number of subsets we speeded up the enumeration while loosening the completeness.

A. Results

The experimental results are given in Table II. Because there is no consensus on the possible number of conflicts (the column #Deny Rule) in access control policies, we considered a ratio of the existing rule number. More specifically, 5% for Synthetic-1 policy and 2% for Synthetic-2 and Synthetic-3 policies. A similar pattern has also been followed for UCC policies. Notice that we generated different policy sets for rule-level and permission-level explanations. As expected, due to disjoint rules generated randomly, the number of disjoint authorization spaces for synthetic policies are close to actual number of rules. This does not hold for UCC policies due to overlapping user groups.

Generally, the time necessary for enumerating conflicts is almost proportional to the size of the SAT formula encoding synthetic policies. It takes around 3 seconds to enumerate less than 40 conflicts for policies with 100 rules and containing 5 conflicting rule pairs. The time rises to almost 20 seconds for 1000 rules and 20 conflicting rules. Notice that the inherent complexity of enumeration of all conflicts is apparent in the results. In fact if completeness is sought, then a direct approach that iterates through rules (or permissions) to find the conflicting pairs may perform better. However this approach may hinder the possible benefits (e.g. preferences) of well-established explanation frameworks.

The results of UCC policies show a similar pattern to synthetic policies. Moreover, the performance of our approach does not appear to be affected from complex user group organizations. Specifically, the conflict enumeration of UCC-CS policies which has more elements in the product $|User| \times |Resource|$, i.e. a more scattered policy, is as efficient as the enumeration on a relatively less scattered policy UCC-NU.

Computation of cardinality MPRs is generally quite efficient. In both types of cardinality MPRs, it is at the levels of milliseconds and requires less than 1 second for 1000 rules. However, a further processing of the obtained cardinality MPR may be needed most of the time since it may exclude rules that are important (rule-level) or modify rules in such a way that requires logical changes to policy (permission-level).

VII. Conclusions And Future Work

Conflict detection and resolution in access control policies enable the maintenance of correctness in rule specifications. In this paper, we proposed a CSP based technique that exploits explanations and relaxations of over-constrained problems for the discovery and resolution of conflicts in physical access control policies. We considered coarse-grained explanations that work at the level of rules and fine-grained explanations that work at the level of permissions. The former explanations lack conflicting permission details but are expected to be less in the number and more understandable. The latter explanations may be numerous but provide more granular information about conflicts. In addition to basic conflict discovery and resolution, we demonstrated how optimal conflict resolution can be built on top of our technique. Optimal conflict resolution opens a door to wider application scenarios where different objectives such as preserving the logical policy structures may be sought.

We experimentally evaluated the computational cost of enumerating all conflicts and resolving them while maintaining most of the policy intact. We believe that there is room for optimizations and observe that the enumeration of conflicts in SAT is an active area of research. As future work, we plan to investigate methods of improving performance and develop different optimal conflict resolution strategies with different objectives. Example objectives include the preservation of logical organization of rules or satisfaction of user preferences on rules or permissions. Finally, we would like to apply the concepts discussed in this paper to various access control languages such as XACML.

Table II
EXPERIMENT RESULTS

	#Deny Rules	#Disjoint Authorization Spaces		Time (s) - (<40) Conflicts		Time (s) - Cardinality MPR	
		Rule-level	Permission-level	Rule-Level	Permission-level	Rule-level	Permission-level
Synthetic-1	5	100	102	2.96	2.99	0.048	0.04
Synthetic-2	10	500	500	12.55	13.29	0.308	0.240
Synthetic-3	20	1003	1002	19.98	19.22	0.864	0.832
UCC-PL	5	72	74	1.06	2	0.016	0.02
UCC-NU	10	267	266	5.24	4.82	0.1	0.304
UCC-CS	10	351	352	6.7	5.16	0.96	0.248

A. Acknowledgements

We thank to Mark Liffiton for providing a version of MARCO that supports partial MaxSAT input and the SAT group in University College Dublin (UCD) for useful discussions.

REFERENCES

[1] Mohammad A. Al-Kahtani and Ravi S. Sandhu. Rule-based rbac with negative authorization. In *ACSAC*, pages 405–415, 2004.

[2] Ehab S. Al-Shaer, Hazem H. Hamed, Raouf Boutaba, and Masum Hasan. Conflict Classification and Analysis of Distributed Firewall Policies. *IEEE Journal on Selected Areas in Communications, Issue: 10, Volume: 23, Pages: 2069 - 2084*, October 2005.

[3] Josep Argelich, Chu Min Li, Felip Many, and Jordi Planes. Sixth MaxSAT competition, February 2011.

[4] Lujo Bauer, Scott Garriss, and Michael K. Reiter. Detecting and resolving policy misconfigurations in access-control systems. *ACM Trans. Inf. Syst. Secur.*, 14(1), 2011.

[5] Kathi Fisler, Shriram Krishnamurthi, Leo A. Meyerovich, and Michael Carl Tschantz. Verification and change-impact analysis of access-control policies. In *ICSE*, pages 196–205, 2005.

[6] William M. Fitzgerald, Fatih Turkmen, Simon N. Foley, and Barry O'Sullivan. Anomaly analysis for physical access control security configuration. In *7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–8, 2012.

[7] Robert Frohardt, Bor-Yuh Evan Chang, and Sriram Sankaranarayanan. Access nets: Modeling access to physical spaces. In *VMCAI*, pages 184–198, 2011.

[8] Joaquín García-Alfaro, Nora Boulahia-Cuppens, and Frédéric Cuppens. Complete analysis of configuration rules to guarantee reliable network security policies. *Int. J. Inf. Sec.*, 7(2):103–122, 2008.

[9] Hongxin Hu, Gail-Joon Ahn, and Ketan Kulkarni. Anomaly discovery and resolution in web access control policies. In *SACMAT*, pages 165–174, 2011.

[10] Graham Hughes and Tevfik Bultan. Automated verification of access control policies using a sat solver. *STTT*, 10(6):503–520, 2008.

[11] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[12] Xin Jin, Ram Krishnan, and Ravi S. Sandhu. A unified attribute-based access control model covering dac, mac and rbac. In *DBSec*, pages 41–55, 2012.

[13] Ulrich Junker. Quickxplain: Preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004.

[14] Manuel Koch, Luigi V. Mancini, and Francesco Parisi-Presicce. Conflict detection and resolution in access control policy specifications. In *FoSSaCS*, pages 223–237, 2002.

[15] Manuel Koch, Luigi V. Mancini, and Francesco Parisi-Presicce. A graph-based formalism for rbac. *ACM Trans. Inf. Syst. Secur.*, 5(3):332–365, 2002.

[16] Ninghui Li and Mahesh V. Tripunitara. Security analysis in role-based access control. *ACM Trans. Inf. Syst. Secur.*, 9(4):391–420, 2006.

[17] Mark H. Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple muses quickly. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pages 160–175, 2013.

[18] Emil Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Trans. Software Eng.*, 25(6):852–869, 1999.

[19] Organization for the Advancement of Structured Information Standards (OASIS). eXtensible Access Control Markup Language (XACML) Version 3.0, January 2013.

[20] Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu. Representative explanations for over-constrained problems. In *AAAI*, pages 323–328, 2007.

[21] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, feb 1996.