Anomaly Analysis for Physical Access Control Security Configuration

William M. Fitzgerald, Fatih Turkmen, Simon N. Foley, Barry O'Sullivan
Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork

***Abstract*—Physical Access Controls, such as supervised doors, surveillance cameras and alarms, act as important points of demarcation between physical zones (areas/rooms) of different levels of trust. They do so by controlling personnel flow to and from areas in accordance with the enterprise security policy. A significant challenge in providing physical access control for (restricted) areas is attaining a degree of confidence that a Physical Access Control security configuration adequately addresses the threats. A misconfiguration may result in a threat of unapproved personnel access or the denial of approved personnel access to a restricted zone. In practice, Physical Access Control security configurations typically span multiple zones, involve many users and run to many thousands of access-control rules, and such complexity may increase the likelihood of misconfiguration. In this paper, a formal model for Physical Access Control security configurations is presented. This model, implemented in SAT, captures a number of unique anomalies specific to Physical Access Control domain. A preliminary set of experiments that evaluate our approach is presented.**

I. Introduction

Physical Access Control (*PAC*) systems are specific instances of access control that regulate access to a property, a building or a room. They act as important points of demarcation between physical zones (areas/rooms) of different levels of trust. They do so by controlling *personnel flow to and from* zones in accordance with the enterprise-level security policy. Note, a zone can be considered as a set of doors/panels that collectively form a physical area such as a department. A PAC system is typically composed of a number of interoperating security mechanisms, for example door readers, video surveillance and alarms.

Management of PAC security configurations can be complex, run to many thousands of access-control rules and are typically maintained on an ad-hoc basis [1]. New access-control rules are often added to the PAC security configuration with little regard to how they interoperate with existing access-control rules and likely resulting in an overly-restrictive and/or overly-permissive PAC policy configuration. Similarly, changes to the PAC security configuration of one PAC security mechanism (for example, door reader) may indirectly impact the intent of a configuration of another PAC security mechanism (for example an alarm or reachability to another door reader). The ideal PAC security configuration provides for consistent interoperating PAC security configurations that support valid personnel access, and, preferably, no more and no less.

Consider, as a running example, the following enterprise-level security requirements to "*permit relevant personnel access to the research laboratory and deny all other access*". In practice, deploying access-control rules for access to a zone such as a research laboratory that upholds an enterprise-level security policy, is not simply about making a zone accessible or inaccessible for all personnel. One may wish to deny certain personnel (for example, visitors), only accept access from research personnel, require time-based constraints on authorisation for other personnel (for example contractors). One also has to consider the the various kinds of authentication methods, for example proximity or biometric badges, that various personnel will be required to use for authentication. One may wish to provide entry and exit to the research laboratory using a specified path (a set of doors) and not others. Furthermore, it may also be prudent to provide surveillance to a particular zone or set of zones. In practice, generating a PAC security configuration that is aligned with the enterprise-level security policy is challenging, and is largely dependent on the expert-knowledge of the security administrator.

We argue that the challenges in composing an anomaly-free Physical Access Control security configurations are comparable to those for firewall security configuration. Management of firewall security is complex and error-prone. Typical errors in a firewall security configuration range from incorrect access-control rule ordering (causing conflicts/anomalies) to errors resulting from the poor comprehension of the enterprise-level security policy. For example, two access-control rules are said to *conflict* if they filter the same kinds of packets with contradictory target actions or if one rule is a specialisation of the other where their ordering is incorrect. An incorrect ordering of access-control rules may change the intended semantics of the firewall security configuration, resulting in incorrect enterprise-level security policy enforcement. To avoid misconfiguration, firewall administrators use structural analysis techniques to detect for example *redundancy*, *shadowing*, *correlation* and *spuriousness* anomalies [2]–[4].

In this paper, these well understood firewall structural analysis techniques are taken and applied to the Physical Access Control domain. To the best of our knowledge, this approach provides a basis with which to identify new conflicts not previously considered in other research. For example, the detection of unintended denial of user access to zones due to blocked paths (shadowing anomaly) or unintended user access

to restricted zones (spurious anomaly).

In this paper, we consider the management of PAC security configuration for door access control. Note, future research will consider the inter-operation of other security mechanisms such as alarms and video surveillance.

The contribution of this paper is as followings. A formal model for PAC security configuration with a specific emphasis on door access control is presented. A number of novel configuration anomalies are formally defined for Physical Access Control. Thus, providing a basis for automated detection of PAC security configuration anomalies.

This paper is organised as follows. Section II presents a formal model for PAC security configuration. An example scenario for PAC security configuration is outlined in Section III. Section IV defines the anomalies for PAC security configurations. Section V outlines a preliminary evaluation of our approach where SAT has been used to encode the PAC model. Related research is outlined in Section VI and Section VII concludes the paper.

II. A Security Model for PAC

This section presents a formal model for PAC security configuration. In this paper, we do not capture all of the possible security configuration attributes in PAC systems, rather we focus on a subset and consider door access controls. The aim of this model is to illustrate some of the analysis that can be usefully done over such security configurations. The model is presented in Z notation [5].

A PAC security configuration assigns badge holders, which identify individual users, to doors in a given zone for which users are permitted access. While a user may hold multiple badges such that access to different zones in a building is possible, we only consider a user that has a single badge. Note, we believe that extending the model for multiple badges is straightforward. Let the basic type $[User]$ represent the set of all possible users. While the majority of the user interfaces available in PAC solutions provide a personalized view to a user, we will be using *groups* that constitute of a set of users. The assumption that the users are grouped also eases the policy administration. The set of all possible groups is defined as

$$Group == \mathbb{P}\, User$$

Let basic types $[Room, Door]$ represent the set of all possible rooms and doors, respectively. While a door $d \in Door$ identifies a unique physical door, it has two faces/sides. Access rules on the door are expressed relative to these faces which are identified by the room that a given side of the door faces. We define

$$DoorFace == (Room \times Door)$$

Thus, the access controls for a door d that connects rooms r and r' are defined in terms of door faces (r,d) and (r',d).

The building topology is defined in terms of a partial function

$$otherside : (Room \times Door) \pfun Room$$

whereby $otherside(r_1, d) = r_2$ means that on the other side of a door d facing room r_1 is the room r_2. We use the constant $\texttt{outside} \in Room$ to denote the outside 'room' of the building. For consistency, $otherside(otherside(r,d),d) = r$ must hold.

For example, the door at the main entrance to the building in Figure 1 can be identified as $(\texttt{outside,D1})$ (the door facing the outside) and we have $otherside(\texttt{outside,D1}) = \texttt{Lobby}$.

In general, there is a path through the building from a starting door d_s in a room r_s through a destination door d_f in room r_f if $existPath((r_s,d_s),(r_f,d_f))$, where

$$existPath : (Room \times Door) \rel (Room \times Door)$$

$$\begin{aligned} &existPath((r_s,d_s),(r_f,d_f)) \Leftrightarrow \\ &\quad \exists\, d_i : Door \bullet \\ &\qquad (r_s,d_s) \in \operatorname{dom} otherside \land \\ &\qquad existPath((otherside(r_s,d_s),d_i),(r_f,d_f)) \end{aligned}$$

Note, that there may exist several paths between the given door faces (denoted by the corresponding room and door). An auxiliary function that checks the existence of a unique path will be introduced further on.

Access controls may be defined for either face of a door and, therefore, an access control is specified in terms of whether a user/group may go through a given door in a given room. A set of these (room,door) faces specifies a Zone:

$$Zone == \mathbb{P}\,(Room \times Door)$$

If no ambiguity can arise then, given $z \in Zone$, $z.Room$ denotes the corresponding room and $z.Door$ denote the corresponding door, respectively. Thus, for example, if $z = (\texttt{Lobby,D1}) \in Zone$ then $z.Door = \texttt{D1}$, and so forth. Note, zones may be classified according to a hierarchical structure.

Conventionally, some PAC systems work on a Close World Assumption (CWA) such that any request for a non-existent permission is denied. However, enabling the specification of *deny* rules (in addition to a default *deny* rule) might be more natural in certain circumstances [6]. For instance, an employee who is on vacation can be denied access to his/her office space for the vacation period, by the addition of a *deny* rule, unless otherwise all rules that allow the user to do so are modified. Moreover, in PAC systems, the logging of events is crucial after a permitted request. Thus, we consider three kind of decisions that define the actions to be taken by the authorizer. Let *Action* denote the different access decisions.

$$Action ::= \texttt{allow} \mid \texttt{deny} \mid \texttt{log}$$

An access-control rule (g,z,a) is a triple specifying that users in group g are granted a access to zone z.

$$Rule == Group \times Zone \times Action$$

As a result, the access control rules in our model can specify whether a group is permitted access (`allow`) to a zone, explicitly denied access (`deny`) to a zone or permitted access to the zone but that the access be logged (`log`).

The rules in a rule pair (r,s) that will be analysed for anomalies/conflicts are categorized as an *upstream* rule and a

downstream rule if their zones are disjoint, $r.Zone \cap s.Zone = \emptyset$. Specifically, r is an upstream rule if its zone contains a door face or set of door faces that control inward bound personnel flow in the direction of the door face or set of door faces controlled by (downstream) rule s. For instance, a rule r that contains `D1` would be the upstream rule when analysed with a rule s that has `D6` in the zone definition for a personnel flow path from `Outside` to `OSLab`.

A PAC security configuration/policy is the set of access-control rules in place for a building.

$$Config : \mathbb{P}\, Rule$$

For the sake of clarity, we use the name of a defining type as an accessor function for access-control rule attributes. In particular, given $r \in Rule$ then $r = (r.Group, r.Zone, r.Action)$.

A configuration query $mayPass(u, r, d)$ returns the access-decision as to whether a user u in room r is permitted to pass through door d. In evaluating a query there may be a number of matching access-control rules in which the user is a member of the group and the room-door pair is in a zone. In the event that the matching rules specify conflicting access decisions a precedence ordering is assumed over *Actions*. This (total) precedence ordering provides an upper bound operator max.

We define

$$mayPass : User \times Room \times Door \rightarrow Action$$

$$\begin{aligned} mayPass(u, r, d) = \max(\{g : Group;\ z : Zone;\ a : Action \\ \mid (g, z, a) \in Config \wedge \\ u \in g \wedge (r, d) \in z \\ \bullet\ a\} \cup \{\texttt{deny}\}) \end{aligned}$$

where the order between decisions is `deny` $\geq$ `log` $\geq$ `allow` and in the absence of matching rules, the default decision is `deny`.

Note, that the current model assumes that access rules are expressed in no particular order and that a *mayPass* query is made across the entire set of access rules. Similar interpretations are commonly used in conventional access control policies, for example, [7]–[9]. An alternative interpretation would be to treat a PAC security configuration as a *sequence* of access rules whereby for a given (user, room, door) triple, the query is tested against each access-control rule, starting from the first, in sequence, and the first rule that matches gives the result that is returned. Similar interpretations are used in firewall policy rules [2], [10]–[12].

III. PAC Security Configuration

Consider as a running example, the building topology for our computer laboratory used to construct a PAC security configuration (Figure 1). The main entrance (`D1`) leads to a lobby where there are two doors to enter the computer science facility. Computer science facility has a meeting room that is protected with an anti-passback mechanism. The lobby leads to an open space which is also accessible from the meeting room. Individual research group locations for Software Engineering (`SELab`), Security (`SecLab`) and Operating Systems (`OSLab`) are also protected with access control. Note, the $*$ in Figure 1 for `D2` and `D3` represents doors with two readers (one on each side).

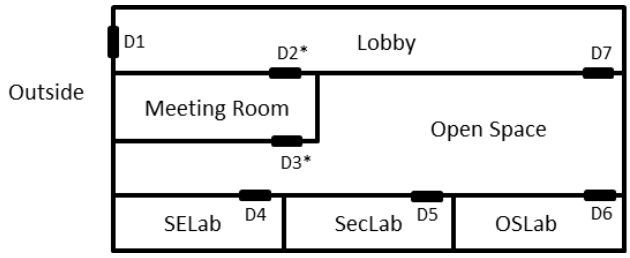

Fig. 1. Example Building Topology

As part of the SAT implementation outlined in Section V, a graph is constructed for all possible paths between a starting position (*Out most*) for example `Outside` and an end position (*In most*) for example `SecLab` within a given building topology (Figure 2). Nodes represent the door faces with readers and edges represent the room that a given door face leads to. Note, nodes *Out most* and *In most* do not represent physical nodes (doors), rather they are markers to state a starting and an end position.

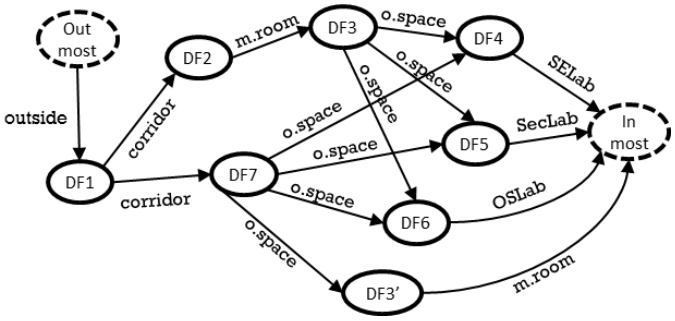

Fig. 2. Example Graph Of Possible Paths Between Two Nodes

The following is an example PAC security configuration. All employees are permitted access to the `Lobby` and `OpenSpace` areas and we have the rule:

$$(\{\texttt{alice}, \texttt{bob}, \texttt{bill}\}, \{(\texttt{Outside}, \texttt{D1}), (\texttt{Lobby}, \texttt{D7})\}, \texttt{allow}) \in Config$$

Eve and Clare are contractors and are permitted access to the `Lobby` area

$$(\{\texttt{eve}, \texttt{clare}\}, \{(\texttt{Outside}, \texttt{D1})\}, \texttt{allow}) \in Config$$

while Clare and Bob both work on a security project which gives them access to the `SecLab`

$$(\{\texttt{bob}, \texttt{clare}\}, \{(\texttt{Lobby}, \texttt{D7}), (\texttt{OpenSpace}, \texttt{D5})\}, \texttt{allow}) \in Config$$

Lastly, Bob and Bill have management roles and have access to the `MeetingRoom`

$$(\{\texttt{bob}, \texttt{bill}\},\ \{(\texttt{Lobby}, \texttt{D2}), (\texttt{OpenSpace}, \texttt{D3})\}, \texttt{allow}) \in \mathit{Config}$$

Note, that the PAC security configuration defined by this set of access-control rules is incomplete and contains a number of inconsistencies. For example, while employees may enter the `Lobby` area by `D1`, no access-control rule specifies that they are permitted to exit the `Lobby` area. The next section considers a range of such anomalies/inconsistencies that may exist in PAC security configurations.

IV. Anomalies in PAC Security Configuration

A. Building Topology Anomalies

One might like to check whether there is a physical path from the outside to every room in the building.

$$\forall r : \mathit{Room} \bullet \exists d, d' : \mathit{Door} \bullet \mathit{existPath}((\texttt{outside}, d), (r, d'))$$

While every room may be physically accessible in principle, it is possible that there is a room that may not be accessed by any user in practice as a result of the PAC security configuration. Let $\mathit{mayPass}^*(u, (r_s, d_s), (r_f, d_f))$ indicate whether a user starting in Room r_s can gain access, from door d_s, to the door d_f in room r_f.

$$\begin{array}{|l}
\mathit{mayPass}^* : \mathit{User} \leftrightarrow (\mathit{Room} \times \mathit{Door}) \leftrightarrow (\mathit{Room} \times \mathit{Door}) \\
\hline
\mathit{mayPass}^*(u, (r_s, d_s), (r_f, d_f)) \Leftrightarrow \\
\quad \exists d_i : \mathit{Door} \bullet \\
\qquad (r_s, d_s) \in \operatorname{dom} \mathit{otherside} \land \\
\qquad \mathit{mayPass}(u, r_s, d_s) = \texttt{allow} \\
\qquad \mathit{mayPass}^*(u, (\mathit{otherside}(r_s, d_s), d_i), (r_f, d_f))
\end{array}$$

We then can then specify that access to every room is permitted by the policy:

$$\forall r : \mathit{Room} \bullet \exists u : \mathit{User};\ d, d' : \mathit{Door} \bullet \mathit{mayPass}^*(u, (\texttt{outside}, d), (r, d'))$$

A similar check can be specified that determines whether a user (permitted) in a particular room can exit the building.

$$\forall r : \mathit{Room};\ u : \mathit{User};\ d, d' : \mathit{Door} \bullet \mathit{mayPass}^*(u, (\texttt{outside}, d), (r, d')) \Rightarrow \exists d'' : \mathit{Door} \bullet \mathit{mayPass}^*(u, (r, d'), (\texttt{outside}, d''))$$

Note that the definition of $\mathit{mayPass}^*$ permits the exit path (per the policy) to be different to the entry path.

B. Redundant and Conflicting Rules

Simple Anomaly analysis examines the relationship that access-control rules have with one another without considering topology information. While the access-control rules on an individual basis may be compliant with the enterprise-level security policy requirements, the relationships between the access-control rules themselves may introduce a scenario such that the overall PAC security configuration is inconsistent with the enterprise-level security policy. For example, the following PAC security configuration:

- Rule A: Main entrance door remain locked except for scheduled times.
- Rule B: Main entrance door remain locked except for scheduled times where scheduled times are activated by first-unlock.

is ambiguous whereby Rule A is inconsistent with Rule B and vice versa. The following is an formal model of access-control rule anomaly analysis definitions with respect to door access control.

a) ***Redundancy Anomaly****:* Given access-control rules $r, s : \mathit{Rule}$ then access-control rule r is redundant to access-control rule s, denoted as $r \sqsubseteq^{ra} s$, if the *Group* and *Zone* attribute fields of r are subsumed by those of s and both r and s have the same actions.

$$\begin{array}{|l}
_ \sqsubseteq^{ra} _ : \mathit{Rule} \leftrightarrow \mathit{Rule} \\
\hline
\forall g_1, g_2 : \mathit{Group};\ z_1, z_2 : \mathit{Zone};\ a_1, a_2 : \mathit{Action} \bullet \\
\quad (g_1, z_1, a_1) \sqsubseteq^{ra} (g_2, z_2, a_2) \\
\quad \Leftrightarrow g_1 \subseteq g_2 \land z_1 \subseteq z_2 \land a_1 = a_2
\end{array}$$

For example, consider the following access-control rules $r, s : \mathit{Rule}$:

$$r = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$
$$s = (\{\texttt{alice}, \texttt{bob}, \texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$

then access-control rule r is considered *redundant* to s where $r.\mathit{Group} = \{\texttt{alice}, \texttt{bob}\}$ is subsumed by $s.\mathit{Group} = \{\texttt{alice}, \texttt{bob}, \texttt{eve}\}$, $r.\mathit{Zone} = \{(\texttt{Lobby}, \texttt{D7})\}$ equals $s.\mathit{Zone} = \{(\texttt{Lobby}, \texttt{D7})\}$ and $r.\mathit{Action} = \texttt{allow} = s.\mathit{Action} = \texttt{allow}$.

While redundancy does not influence the semantics of a physical access control policy, having to reason over additional and unnecessary rules may result in a performance cost. Resolution of redundancy conflicts corresponds to removing the redundant access-control rule.

b) ***Exception Anomaly****:* Given access-control rules $r, s : \mathit{Rule}$ then access-control rule r is an exception to access-control rule s, if the *Group* and *Zone* attribute fields of r are subsumed by those of s and both r and s have different actions.

$$\begin{array}{|l}
_ \sqsubseteq^{ea} _ : \mathit{Rule} \leftrightarrow \mathit{Rule} \\
\hline
\forall g_1, g_2 : \mathit{Group};\ z_1, z_2 : \mathit{Zone};\ a_1, a_2 : \mathit{Action} \bullet \\
\quad (g_1, z_1, a_1) \sqsubseteq^{ea} (g_2, z_2, a_2) \\
\quad \Leftrightarrow g_1 \subseteq g_2 \land z_1 \subseteq z_2 \land a_1 \neq a_2
\end{array}$$

For example, consider the following access-control rules $r, s : Rule$:

$$r = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$
$$s = (\{\texttt{alice}, \texttt{bob}, \texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$

then access-control rule r is considered an exception to access-control rule s where $r.Group = \{\texttt{alice}, \texttt{bob}\}$ is subsumed by $s.Group = \{\texttt{alice}, \texttt{bob}, \texttt{eve}\}$, $r.Zone = \{(\texttt{Lobby}, \texttt{D7})\}$ equals $s.Zone = \{(\texttt{Lobby}, \texttt{D7})\}$ and $r.Action = \texttt{allow}$ is not equal to $s.Action = \texttt{deny}$.

The above *exception* anomaly definition may be further constrained such that one can determine whether the access-control rule causing the exception anomaly is of the kind that permits the action that the generalized access-control rule is denying or vice versa. Given rules r and s then a *permit-exception* conflict occurs iff $(r \sqsubseteq^{ea} s \land s.Action \neq \texttt{deny})$, and a *deny-exception* conflict occurs iff $(r \sqsubseteq^{ea} s \land s.Action \neq \texttt{allow})$.

Resolution of such an anomaly may be to remove the exception access-control rule or leave the access-control rule as it was intended.

c) ***Correlation Anomaly****:* Two access-control rules $r, s : Rule$ are said to be correlated (symmetric relation), denoted $r \sqsubseteq^{ca} s$, where

$$\_ \sqsubseteq^{ca} \_ : Rule \leftrightarrow Rule$$
$$\forall g_1, g_2 : Group;\ z_1, z_2 : Zone;\ a_1, a_2 : Action \bullet$$
$$(g_1, z_1, a_1) \sqsubseteq^{ca} (g_2, z_2, a_2)$$
$$\Leftrightarrow ((g_1 \supset g_2 \land z_1 \subset z_2) \lor (g_1 \subset g_2 \land z_1 \supset z_2)$$
$$\land\ a_1 \neq a_2$$

Consider the following access-control rules $r, s : Rule$:

$$r = (\{\texttt{alice}, \texttt{bob}\},$$
$$\{(\texttt{Outside}, \texttt{D1}), (\texttt{Lobby}, \texttt{D7}), (\texttt{SecLab}, \texttt{D4})\}, \texttt{deny})$$
$$s = (\{\texttt{alice}, \texttt{bob}, \texttt{eve}\}, \{(\texttt{Outside}, \texttt{D1})\}, \texttt{allow})$$

where $r.Group = \{\texttt{alice}, \texttt{bob}\}$ is subsumed by $s.Group = \{\texttt{alice}, \texttt{bob}, \texttt{eve}\}$ and $r.Zone = \{(\texttt{Outside}, \texttt{D1}), (\texttt{Lobby}, \texttt{D7}), (\texttt{SecLab}, \texttt{D4})\}$ subsumes the attribute field $s.Zone = \{(\texttt{Outside}, \texttt{D1})\}$ and both r and s have opposing actions.

The conjunction of access-control rules r and s may be interpreted in one of two ways. In the case where access-control rule r is given precedence over s (where a `deny` action is default security policy) implies an implicit access-control rule x where $x.Group = \{\texttt{alice}, \texttt{bob}\}$, $x.Zone = \{(\texttt{Outside}, \texttt{D1})\}$ and x.Action=`deny`. However, if access-control rule s is given precedence over r (where a `allow` action is default security policy) implies an implicit access-control rule x where $x.Group = \{\texttt{alice}, \texttt{bob}\}$, $x.Zone = \{(\texttt{Outside}, \texttt{D1})\}$ and x.Action=`allow`. The result is that users `alice` and `bob` are either denied access to `Lobby` instead of being allowed or allowed access to `Lobby` instead of being denied.

Such an anomaly is a paradox and resolution (if any) through removal of an access-control needs careful consideration. Another resolution without the removal of a single access-control is to replace both access-control rules with two new access-control rules re-written without the correlation anomaly.

C. Topology Anomalies

Access-control rules that define user authorisation for one zone versus another zone may appear disjoint and not related (for example access-control rules r and s illustrated below). By considering additional external information such as the enterprise-level security requirements and/or the building's room topology, an additional set of access-control rule relationships may be defined. In this paper, the building's topology where a path between a given set of rooms is considered in conjunction to the access-control rules themselves.

As a running example, consider the following scenario. A path to `SecLab` is to first enter `Lobby` from `Outside` through `D1`, then `OpenSpace` through `D7` and finally `SecLab` through `D5`. Therefore, generating a suitable room PAC security configuration requires permitting a set of users access to all rooms in a given path of rooms that lead to the room where access is being sought. The following is an example PAC security configuration, *Config*, that permits user `alice` access to `SecLab`.

$$Config = \{r, s, w\}$$

where

$$r = (\{\texttt{alice}\}, \{(\texttt{Outside}, \texttt{D1})\}, \texttt{allow})$$
$$s = (\{\texttt{alice}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$
$$w = (\{\texttt{alice}\}, \{(\texttt{OpenSpace}, \texttt{D5})\}, \texttt{allow})$$

d) ***Shadowing Anomaly****:* An upstream access-control rule is denoted by s_{up} and a downstream access-control rule is denoted by r_{down}. Given access-control rules $r_{down}, s_{up} : Rule$, access-control rule r_{down} is shadowed (overly restricted) by access-control rule s_{up}, if the attribute field of $r_{down}.Group$ is a subset of or a superset of the corresponding attribute field of access-control rule s_{up}, the attribute field of $r_{down}.Zone$ is disjoint from the corresponding attribute field of access-control rule s_{up} where s_{up} is denying what access-control rule r_{down} is intending to `allow` and a path exists between rooms encoded within rules r_{down} and s_{up}.

$$\_ \sqsubseteq^{sa} \_ : Rule \leftrightarrow Rule$$
$$\forall g_1, g_2 : Group;\ z_1, z_2 : Zone;\ a_1, a_2 : Action \bullet$$
$$(g_1, z_1, a_1) \sqsubseteq^{sa} (g_2, z_2, a_2)$$
$$\Leftrightarrow ((g_1 \subseteq g_2 \lor g_1 \supseteq g_2) \land (z_1 \nsubseteq z_2 \land z_1 \not\supseteq z_2)) \land$$
$$a_1 \neq a_2 \land a_2 = \texttt{deny} \land$$
$$\exists d_1, d_2 : Door;\ r_1, r_2 : Room \bullet$$
$$(r_1, d_1) \in z_1 \land (r_2, d_2) \in z_2 \land$$
$$existPath((r_1, d_1), (r_2, d_2))$$

Consider the following access-control rules $r_{down}, s_{up} : Rule$ and given there exists a path between `OpenSpace` and `SecLab`, then r_{down} is shadowed by s_{up}. That is, user `alice` is denied access (along a path) to `SecLab`.

$$s_{up} = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$r_{down} = (\{\texttt{alice}\}, \{(\texttt{SecLab}, \texttt{D5})\}, \texttt{allow})$$

Note, in the current security PAC model, shadowing is only considered between access-control rules within a single path and not across multiple paths. Another path may exist from an outmost room to an inmost room. For example, an alternative path from `Outside` to `OSLab` using doors `D1`, `D2`, `D3` and `D6` with the correct access controls may exist.

The following access-control rules, demonstrates a *partial* shadowing conflict as a consequence of $r_{down}.Group \supseteq s_{up}.Group$ where only user `alice` (not `bob`) has inadvertently been denied access to `SecLab` along the path defined as $((\texttt{Lobby}, \texttt{D7}), \texttt{OpenSpace})$ and $((\texttt{OpenSpace}, \texttt{D5}), \texttt{SecLab})$

$$s_{up} = (\{\texttt{alice}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$r_{down} = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{SecLab}, \texttt{D5})\}, \texttt{allow})$$

If one was to adopt the view that all enterprise-level downstream access-control security requirements are to have precedence over all enterprise-level upstream access-control security requirements then the upstream low-level access-control rule causing the shadowing should be modified as to remove the shadowing anomaly. Note, this view assumes that the low-level downstream access-control rule has been written correctly.

e) ***Spurious Anomaly****:* Given access-control rules $r_{down}, s_{up} : Rule$, access-control rule s_{up} is spurious (overly permissive) to access-control rule r_{down} , if the attribute field of $s_{up}.Group$ is a subset of or a superset of the corresponding attribute field of access-control rule r_{down}, the attribute field of s_{up} is disjoint from the corresponding attribute field of access-control rule $r_{down}.Zone$ where s_{up} is allowing what access-control rule r_{down} is intending to `deny` and a path exists between rooms encoded within rules s_{up} and r_{down}.

$$\_ \sqsubseteq^{spa} \_ : Rule \leftrightarrow Rule$$

$$\begin{aligned}
&\forall\, g_1, g_2 : Group;\ z_1, z_2 : Zone;\ a_1, a_2 : Action \bullet \\
&\quad (g_1, z_1, a_1) \sqsubseteq^{spa} (g_2, z_2, a_2) \\
&\quad \Leftrightarrow ((g_1 \subseteq g_2 \lor g_1 \supseteq g_2) \land (z_1 \nsubseteq z_2 \land z_1 \nsupseteq z_2)) \land \\
&\qquad a_1 \neq a_2 \land a_1 = \texttt{deny} \land \\
&\qquad \exists\, d_1, d_2 : Door;\ r_1, r_2 : Room \bullet \\
&\qquad (r_1, d_1) \in z_1 \land (r_2, d_2) \in z_2 \land \\
&\qquad existPath((r_1, d_1), (r_2, d_2))
\end{aligned}$$

Consider the following access-control rules $r_{down}, s_{up} : Rule$ and given there exists a path between `OpenSpace` and `SecLab` $((\texttt{OpenSpace}, \texttt{D5}), \texttt{SecLab})$, then s_{up} is spurious to r_{down} (in terms of unintended door reachability). That is, user `bob` is permitted access to `Openspace` (s_{up}) where `bob` has the potential to exploit a single door (`D5`) to gain access to `SecLab`.

$$s_{up} = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$
$$r_{down} = (\{\texttt{bob}\}, \{(\texttt{OpenSpace}, \texttt{D5})\}, \texttt{deny})$$

In practice, one may like to implement a security in depth approach, where for example `SecLab` may be categorized as a critical room such that it requires access to two or more doors in a given path. Note, access-control rule s_{up} is spurious only in the context of access-control rule r_{down}. However, access-control rule s_{up} when considered in isolation, maybe correctly upholding an enterprise-level security requirement defining who may access `OpenSpace`. If as with the shadowing anomaly, one adopts the view that the downstream access control rule should have precedence over an upstream access-control rule, then access-control rule s_{up} will require modification as to exclude user `bob` from `OpenSpace` thereby providing defense in depth to `SecLab`.

V. Implementation and Evaluation

The security model presented in Section II was implemented and the anomalies were encoded in SAT. We developed an anomaly discovery algorithm similar to the ones in firewall systems. Our algorithm, summarized in Algorithm 1, uses SAT solving (denoted as *solve*()) as a sub-procedure to reason about two given rules for an anomaly. Note that, SAT has been successfully used to perform analysis over large firewall rule-sets [13], [14]. We have not provided the details of our SAT encoding due to space limitations but necessary explanations are provided when necessary. In simple terms, we have followed a monolithic approach in the encoding such that a large part of the clauses corresponds to our policy model and basic predicates in anomaly definitions, and is shared among all anomaly encodings. In this way, the actual query, checking whether there is an anomaly between a rule pair, is created by forcing the satisfiable assignment of the necessary predicates for the anomaly under consideration.

In order to reduce the overhead caused by translation of a rule pair, topology information and the anomaly in SAT, we introduced two types of clauses: once-clauses and pair-clauses. The former contains all the static information that is available in a PAC configuration and represents the basic relations such as subsumption between elements (e.g. usersets) of a PAC setting or inequality of decisions in rules. The latter contains the actual rule pair under consideration, and starting and ending nodes (e.g. Outmost and Inmost) of the directed graph that represents building topology.

Our algorithm also uses helper methods such as *store*() and *extractPath*() for reporting purposes. In general, the time complexity associated with the discovery algorithm is characterized with the following formula:

$$\frac{t \times (n \times n - 1)}{2}$$

where t is the average time for analyzing all anomalies combined and n is the number of rules. As can be noted from the formula, the driving factor in the overall complexity is the number of rules.

A. Evaluation

We carried out a set of preliminary experiments to evaluate the effectiveness of our approach. The experimental setup was constructed as follows. Experiments were performed on a computer with an Intel i7 3.40GHz processor and 8GB of RAM. The building topology consisted of 50 rooms and 60 doors where the maximum number of paths between the nodes

Algorithm 1: Anomaly Discovery

```
   Input  : PAC Configuration (Topology, List of Readers,
            Policy), List of Anomalies
   Output: Rule Pair, Anomaly
1  Generate a directed graph from topology and readers
2  Generate "once-clauses"
3  for i = 1 to | R | −1 do
4      for j = i + 1 to | R | do
5          foreach anomaly ∈ Anomalies do
6              Generate "pair-clauses" for R[i] and R[j]
7              cls[][] ← once-clauses ∪ pair-clauses ∪
               "anomaly"-unit-clause
8              solution[] ← solve (cls)
9              if solution ≠ null then break
10             switch anomaly do
11                 case redundancy and exception anomalies
12                     store ("Anomaly:"+anomaly+R[i] + R[j])
13                     Generate "pair-clauses" for R[j] and R[i]
14                     cls[][] ← once-clauses ∪ pair-clauses ∪
                       "anomaly"-unit-clause
15                     sol[] ← solve (cls)
16                     if solution ≠ null then
17                         store ("Anomaly:"+ anomaly+R[j] +
                           R[i])
18                 case correlation
19                     store ("Anomaly:"+ anomaly+R[i] + R[j]
20                 case topology anomalies
21                     store ("Anomaly:"+ anomaly+R[i] + R[j]
                       + extractPath (solution))
22 if ¬ anomalyFound then return "No Anomaly"
```

Outmost and Inmost (see Figure 2) was 20. Note, the number of door faces was selected randomly from 120, that is, each face of a door. The number of users was set to 100.

For each experiment, involving an access-control rule-set of different size, five different PAC configurations were randomly generated and their average has been used as the final result. Specifically, for the experiment that involved 10 rules we obtained five random PAC configurations with different building topologies and policy configurations. Note that, we measured the times at each multitude of 10. That is, the first experiment was for a PAC security configuration that consisted of 10 access-control rules, the second for 20 access-control rules and so forth.

While the performance of our approach is acceptable for modest number of rules, we believe certain optimizations can be applied to scale for very large number of rules. For instance an alternative SAT encoding that addresses anomalies individually rather than a monolithic approach as ours may prove to be more efficient. Our approach on the other hand enables other types of queries that an individual anomaly encoding may not offer. We are currently investigating optimization possibilities for our current SAT encoding of the model and alternative encodings that would allow more efficient analysis.

VI. Related Work

While firewall security configuration analysis has been extensively researched [2], [10]–[12], research on Physical Access Control has been limited. In [15], the authors use *association rule mining* to detect misconfigurations in a PAC security configurations. Their approach identifies potential misconfigurations/inconsistencies by comparing and analysing the history of previous PAC security configurations with the current PAC security configuration. The anomalies presented in our paper build upon those defined within the firewall domain, for example [2], [11], [16] and are specialized for the Physical Access Control domain.

Discovering anomalies in access control policies has also attracted some attention [17]. The authors tackle a similar problem to that presented in this paper but for eXtensible Access Control Markup Language (XACML). They present a binary decision diagram (BDD) based technique to check whether an anomaly, such as redundancy, exists. Property analysis on access control policies has been mainly centered around XACML and RBAC. A SAT encoding for analysing properties of XACML policies is presented in [18].

VII. Conclusion

This paper considered the management of anomaly-free PAC security configurations. A formal model for PAC security configuration that governs user access through doors was presented. Drawing upon analysis techniques for firewall security configuration (for example [2], [10]–[12]) comparisons where made with PAC security configurations. This facilitated the key contribution of this paper where a number of configuration anomalies have been identified, for example a *shadowing* anomaly. The formal model and anomalies have been encoded in SAT. This provided a basis for an automated testing of PAC security misconfiguration.

Future research will extend this model to consider other attributes of users and security mechanisms and their constraints. In this paper, anomaly detection was considered. Future research shall investigate (semi-) automated PAC security misconfiguration resolution.

One may avoid much of the anomalies/conflicts, if each access-control rule is defined as a permission for a single user to access to a single zone. However, such singleton-based access-control rules are not reflective of modern enterprise hierarchical structures. For example enterprises are composed of departments and have different kinds of personnel (employees, contractors, visitors). In practice, access-control rules are defined to with respect to groups of users that may have permission to access a room or set of rooms (zones).

Given that conflict-free singleton-based access-control rules are impractical, future research will consider physical access control policy relaxation techniques [19]. Relaxation of access-control rules will generate PAC security configurations that are anomaly-free. However, these PAC security configurations while conflict-free may not be optimal.

For example, given the following conflicting access-control rules:

$$r = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$s = (\{\texttt{alice}, \texttt{bob}, \texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$

a relaxation of those access-control rules may produce the following anomaly-free PAC security configurations.

PAC security configuration a : *Config* such that $a = \{r\}$:

$$r = (\{\texttt{alice}, \texttt{bob}, \texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$

where security configuration a adopts a deny by default policy. Thereby unintentionally denying user `eve`.

PAC security configuration b : *Config* such that $b = \{r, s\}$:

$$r = (\{\texttt{alice}, \texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$s = (\{\texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$

PAC security configuration c : *Config* such that $c = \{q, r, s\}$:

$$q = (\{\texttt{alice},\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$r = (\{\texttt{bob}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{deny})$$
$$s = (\{\texttt{eve}\}, \{(\texttt{Lobby}, \texttt{D7})\}, \texttt{allow})$$

In this example, PAC security configuration b may be the optimal solution depending on the security policy requirements. As a consequence, optimisation of access-control rule relaxations will also be considered in future research.

REFERENCES

[1] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea, "Real life challenges in access-control management," *27th international conference on Human factors in computing systems (CHI), ACM, Boston, MA, USA*, April 2009.

[2] W. M. Fitzgerald and S. N. Foley, "Analysis of Firewall Configuration using an Ontology Engineering Approach," *Semantic Web – Interoperability, Usability, Applicability, IOS Press Journal, Under Review*.

[3] W. M. Fitzgerald and S. N. Foley, "Reasoning about the security configuration of san switch fabrics," *4th IEEE Symposium on Configuration Analytics and Automation (SafeConfig), Arlington, VA, USA*, October 2011.

[4] W. M. Fitzgerald and S. N. Foley, "Aligning Semantic Web Applications with Network Access Controls," *International Journal on the Development and Application of Standards for Computers, Software Quality, Data Communications, Interfaces and Measurement,Computer Standards & Interfaces, Volume 33, Issue 1, Elsevier*, January 2011.

[5] J. M. Spivey, *Z Notation - a reference manual (2. ed.)*, ser. Prentice Hall International Series in Computer Science. Prentice Hall, 1992.

[6] *Cisco Physical Access Manager User Guide*, Release 1.3.0 ed., Cisco.

[7] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38 –47, feb 1996.

[8] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, "Flexible support for multiple access control policies," *ACM Trans. Database Syst.*, vol. 26, no. 2, pp. 214–260, 2001.

[9] S. F. Sabrina De Capitani di Vimercati and P. Samarati, "Access control policies and languages," *Int. J. Computational Science and Engineering*, vol. 3, no. 2, 2007.

[10] E. S. Al-Shaer, H. H. Hamed, R. Boutaba, and M. 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.

[11] F. Cuppens, N. Cuppens-Boulahia, and J. García-Alfaro, "Detection and Removal of Firewall Misconfiguration," *IASTED International Conference on Communication, Network and Information Security (CNIS), Phoenix, AZ, USA*, November 2005.

[12] M. Abedin, S. Nessa, L. Khan, and B. M. Thuraisingham, "Detection and Resolution of Anomalies in Firewall Policy Rules," *Data and Applications Security XX, 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Sophia Antipolis, France*, July 2006.

[13] A. Jeffrey and T. Samak, "Model checking firewall policy configurations," *Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY) Washington, DC, USA*, 2009.

[14] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, "The Margrave Tool for Firewall Analysis," *24th USENIX Large Installation System Administration Conference (LISA), San Jose, CA, USA*, November 2010.

[15] L. Bauer, S. Garriss, and M. K. Reiter, "Detecting and resolving policy misconfigurations in access-control systems," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, 2011.

[16] E. S. Al-Shaer and H. H. Hamed, "Discovery of Policy Anomalies in Distributed Firewalls," *23rd Conference on Computer Communications INFOCOM, Hong Kong*, March 2004.

[17] H. Hu, G.-J. Ahn, and K. Kulkarni, "Anomaly discovery and resolution in web access control policies," in *SACMAT*, 2011, pp. 165–174.

[18] G. Hughes and T. Bultan, "Automated verification of access control policies using a sat solver," *STTT*, vol. 10, no. 6, pp. 503–520, 2008.

[19] D. Lesaint, D. Mehta, B. O'Sullivan, L. Quesada, and N. Wilson, "Context-sensitive call control using constraints and rules," *16th international conference on Principles and practice of constraint programming, Scotland*, 2010.