Security Analysis of Access Control Policies for Smart Homes

Roberta Cimorelli Belfiore
Anna Lisa Ferrara
r.cimorellibelfio@studenti.unimol.it
annalisa.ferrara@unimol.it
University of Molise
Italy

ABSTRACT

Ensuring security is crucial in smart home settings, where only authorized users should have access to home devices. Over the past decade, researchers have focused on developing access control policies and evaluating their efficacy in preventing unauthorized access. A new variant of Role-Based Access Control (RBAC), called Extended Generalized Role- Based Access Control (EGRBAC), has recently been introduced to capture the intricate user-device-context interactions that are prevalent in smart home environments. In this paper, we demonstrate that the task of analyzing administrative EGRBAC policies for security can be performed by reducing it to the security analysis of administrative RBAC policies. We also conducted a case study on a realistic smart home to prove the viability of our approach with respect to security requirements such as availability and privilege escalation.

CCS CONCEPTS

• **Security and privacy → Access control**.

KEYWORDS

access control, smart homes, automated security analysis

ACM Reference Format:
Roberta Cimorelli Belfiore and Anna Lisa Ferrara. 2023. Security Analysis of Access Control Policies for Smart Homes. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies (SACMAT '23), June 7–9, 2023, Trento, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3589608.3593842

1 INTRODUCTION

With the rise of the Internet of Things (IoT), smart houses are becoming increasingly common. However, the involvement of multiple users, intricate social connections, and interactions with various smart devices has made safeguarding the privacy of residents and the home's resources a crucial concern. To address this issue, sophisticated access control specification and enforcement models are necessary. One such model is the Extended Generalized Role-Based Access Control (EGRBAC) model, specifically designed for access control in the context of smart homes [1].

EGRBAC. EGRBAC builds upon the concepts introduced by the Generalized Role-Based Access Control (GRBAC) model [20] and introduces new concepts, such as *Device Roles*, to provide a more comprehensive approach to access control. In EGRBAC, users are classified as human beings who interact with devices inside the house, while roles represent the relationship between the user and the family, such as parent, child, or guest. EGRBAC utilises the concepts of *Environment Roles* and *Role Pairs* to take into account environmental contexts such as daytime/nighttime and winter/summer. Environment roles are triggered by environmental conditions such as daylight or weather, while role pairs consist of a combination of a role and a subset of environment roles associated with it. For instance, the role *kid* can be associated with the environment role *Entertainment_Time* to form the role pair *(kid, Entertainment_Time)*. Moreover, EGRBAC introduces the concept of *Device Roles*. Device Roles provide a method for classifying permissions of different devices. For instance, one can create a device role called *dangerous devices* to categorize the dangerous permissions of various smart devices, such as turning on the oven, operating the lawn mower, and unlocking/locking the front door. The *Permission to Device-Role Assignment* relation (RPDRA) combines all these components by assigning device roles to role pairs. In this way, it is possible, for instance, to assign kids permission to control entertainment devices during weekend and evenings only.

Administrative EGRBAC. Shakarami and Sandhu proposed an administrative approach that uses role-based modeling to govern EGRBAC [26]. Their methodology includes a framework for managing modifications to the RPDRA assignment relation, which defines a set of administrative actions and prohibited assignments for managing associations between role pairs and device roles. Homeowners often need to establish policies that enable the assignment of role pairs to device roles, based on their association with other device roles. However, to implement this, changes need to be made not only to the RPDRA relation but also to other administrative components, such as the one responsible for assigning permissions to device roles. To simplify the definition of policies for homeowners, who may not necessarily be experts in security policy configuration and management, we propose a slight modification to the way in which the RPDRA relation is modified. Specifically, we include preconditions for assignment actions, following the paradigm used in ARBAC97. This modification allows for the establishment of a hierarchy for device roles and streamlines assignments between a role pair and a device role based on existing device role associations with the role pair. For instance, the policy may require that the maid

SACMAT '23, June 7–9, 2023, Trento, Italy

ACM ISBN 979-8-4007-0173-3/23/06…$15.00
https://doi.org/10.1145/3589608.3593842

during evenings has control over the cleaning devices provided that she has control over the lights.

Security Analysis. Security analysis is essential as it allows to find conditions that can lead to an unwanted authorization state when the access control system evolves via administrative actions. Thus, the purpose of security analysis techniques is to check whether an unwanted authorization state is reachable and whether each reachable state meets certain security or availability requirements. Homeowners design administrative policies to achieve specific security goals; however, mistakes are common and may result in security breaches, such as assigning a role pair to an inappropriate device role. For example, during entertainment time, children may gain access to devices that are restricted to adults. To mitigate these risks, we propose to perform a security analysis of administrative AEGRBAC policies with a specific focus on the *Device Role (DR) reachability problem*. This problem involves determining whether a role pair can be assigned to a given device-role within the administrative model, which can help identify potential security vulnerabilities in the policy design. The Device-Role (DR) reachability problem is informally formulated as follows:

The DR reachability problem. Given an AEGRBAC policy with finite sets of users, role pairs, device roles, administrative actions, an initial configuration of the role pair to device role assignment, and a target device role *goal*, the DR reachability problem asks: *is there a reachable configuration of the access-control system where some role pair is assigned to the given device role goal*?

Contribution. This paper presents a solution to the DR reachability problem through security analysis of Administrative RBAC (ARBAC)[7, 24]. We achieve this by reducing the problem to the role-reachability problem in ARBAC and leveraging existing techniques and tools for ARBAC policy analysis. To demonstrate the feasibility of our approach, we conducted a case study on a smart home based on user requirements. The administrative policies in the model include availability and escalation of privileges constraints on shared resources. We utilize the tool VAC [7][1] to endorse our proof of concept, and our experimental results show that the approach is promising.

Organization: The remainder of this paper is organized as follows. In Section 2 we review the RBAC, the EGRBAC, and the ARBAC models. In Section 3 we present our running example. In Section 4 we describe the administrative EGRBAC model. In Section 5, we address the DR reachability problem by reducing it to the security analysis of ARBAC policies. In Section 6 we show our experimental analysis. We review related works in Section 7.

2 PRELIMINARIES

In this section we recall the RBAC, EGRBAC and the administrative RBAC models[2].

2.1 Administrative Role Based Access Control

An RBAC policy is a tuple $\langle U, R, P, UA, PA\rangle$ where U, R and P are finite sets of *users*, *roles*, and *permissions*, respectively, $UA \subseteq U \times R$ is the *user-role assignment* relation, and $PA \subseteq P \times R$ is the *permission-role assignment* relation. A pair $(u, r) \in UA$ if user u is a member of role r. Similarly, $(p, r) \in PA$ means that members of role r are granted the permission p.

Administartive RBAC (ARBAC) policies are usually divided into three languages: one to control the assignment of users to roles (URA), one to control the assignment of permissions to roles (PRA), and the last to control the assignment of roles to roles (RRA). For our purposes, we need to recall the URA module and in the remainder of the paper, with ARBAC we will refer to the ARBAC URA module. URA allows changing the assignment of roles to users UA through assignment/revocation rules executed by administrators, organized into a set of administrative roles AR.

Administrators are allowed to change roles of a user according to a precondition. A *precondition* is a conjunction of literals, where each literal is either in positive form r or in negative form $\neg r$, for some role r in R. A precondition can be partitioned in two sets denoted Pos and Neg, respectively corresponding to the set of roles that appear in positive and negative form in the precondition.

Permission to assign users to roles is specified as:

$$can\_assign \subseteq AR \times 2^R \times 2^R \times R.$$

The meaning of a can-assign tuple $(admin, Pos, Neg, r) \in can\_assign$ is that a member of the administrative role $admin \in AR$ can make a user whose current roles membership satisfies the precondition (Pos, Neg), a member of $r \in R$. In the remainder of the paper we assume that $Pos \cap Neg = \emptyset$.

Permission to revoke users from roles is specified as:

$$can\_revoke \subseteq AR \times 2^R \times 2^R \times R.$$

A tuple $(admin, Pos, Neg, r) \in can\_revoke$ means that a member of the administrative role $admin \in AR$, can revoke the membership of a user u from a role $r \in R$, provided that the role membership of u satisfies the precondition (Pos, Neg).

ARBAC Systems [8]: An ARBAC system is a state transition system that evolves via administrative actions to modify user role assignments. Formally, an ARBAC system is a tuple $S = \langle U, R, AR\ UA, can\_assign, can\_revoke\rangle$ where $\langle U, R, UA\rangle$ represents a RBAC user-to-role assignment policy, AR is the set of administrative roles, and $\langle can\_assign, can\_revoke\rangle$ is an ARBAC model over the set R.

A *configuration* of S is any user-role assignment relation $UR \subseteq U \times R$. A configuration UR is *initial* if $UR = UA$.

Given two S configurations UR and UR', there is a *transition* from UR to UR' with rule $m \in (can\_assign \cup can\_revoke)$, denoted $UR \xrightarrow{m} UR'$, if there is an *administrator ad* and an administrative role *admin* with $(ad, admin) \in UR$ and a user $u \in U$, and one of the following holds:

[**can-assign move**] $m = (admin, P, N, r)$, $P \subseteq \{t \mid (u, t) \in UR\}$, $N \subseteq R \setminus \{t \mid (u, t) \in UR\}$, and $UR' = UR \cup \{(u, r)\}$;

[**can-revoke move**] $m = (admin, r)$, $(u, r) \in UR$, and $UR' = UR \setminus \{(u, r)\}$.

A *run* of S is any finite sequence of S transitions $\pi = c_0 \xrightarrow{m_1} c_1 \xrightarrow{m_2} \ldots c_n \xrightarrow{m_n} c_{n+1}$ for some $n \geq 0$, where c_0 is an *initial* configuration of S. An S configuration c is reachable if c is the last configuration of an S run.

[1] Performance comparisons among existing tools is outside the scope of this paper.
[2] Since the analysis queries do not involve sessions, we do not consider sessions.

U = {alice, james, mary, kate, lucy, john}
R = {kid, parent, babysitter, guest, maid, authority}
D = {DoorLock, TV, DVD, PlayStation, Fridge, WashingMachine, SmartToy, Thermostat, Lights, SurveillanceCameras, SmartRobot, Vacuum Cleaner}
DR = {Entertainment_Devices, Adult_Controlled, Owner_Controlled, Kids_Friendly_Content, Lighting_Devices, Cleaning_Devices, Door_Device}
UA = {(james, kid), (alice, parent), (mary, babysitter), (kate, guest), (lucy, maid), (john, authority)}
OP = {Lock, Unlock, On, Off, Restricted, DisplayFood, PlaySound, Activate, Deactivate, SheduleThermostat, StartRecording, StopRecording, Setting}
ER = {Entertainment_Time, Any_Time, At_Home, Emergency_Time, Wednesday,Friday}

P= { P_1 = {*TV*, *DVD*, *PlayStation*} × {*On*, *Off*, *Restricted*}
P_2 = {*TV*, *DVD*, *PlayStation*} × {*On*, *Off*}
P_3 = {*Fridge*} × {*On*, *Off*, *DisplayFood*}
P_4 = {*DoorLock*} × {*Lock*, *Unlock*}
P_5 = {*WashingMachine*} × {*On*, *Off*}
P_6 = {*Lights*} × {*On*, *Off*}
P_7 = {*SurveillanceCameras*} × {*StartRecording*, *StopRecording*}
P_8 = {*Thermostat*} × {*On*, *Off*, *ScheduleThermostat*}
P_9 = {*Thermostat*} × {*On*, *Off*}
P_{10} = {*SmartToy*} × {*PlaySound*}
P_{11} = {*SmartRobotVacuumCleaner*} × {*On*, *Off*, *Setting*}
P_{12} = {*SmartRobotVacuumCleaner*} × {*On*, *Off*} }

PDRA = $\{P_1 \times Entertainment\_Devices\} \cup \{\{P_2 \cup P_{10}\} \times Kids\_Friendly\_Content\} \cup \{\{P_3 \cup P_4 \cup P_9\} \times Adult\_Controlled\} \cup \{\{P_7 \cup P_8 \cup P_{11}\} \times Owner\_\ Controlled \cup \{P_6 \times Lighting\_Devices\} \cup \{\{P_5 \cup P_{12}\} \times Cleaning\_Devices\} \cup \{P_4 \times Door\_Device\}\}$
RP = { (kid, Entertainment_Time), (parent, Any_Time), (babySitter, Wednesday), (babySitter, Friday), (maid, At_Home), (guest, At_Home), (authority,Emergency_Time) }
RPDRA = {((parent, Any_Time), Adult_Controlled), ((parent, Any_Time), Owner_Controlled), ((babysitter, Friday), Adult_Controlled), ((babysitter, Wednesday), Door_Device), ((kid, Entertainment_Time), Kids_Friendly_Content), ((guest, At_Home), Lighting_Devices), ((maid, At_Home), Cleaning_Devices), ((guest, At_Home), Entertainment_Devices), ((authority, Emergency_Time), Owner_Controlled)}

Figure 2: An EGRBAC policy.

Definition 2.1 (Role-reachability Problem [8]). *For any role $r \in R$, r is* reachable *in an ARBAC system S if there is an S reachable configuration UR such that $(u, r) \in UR$, for some $u \in U$. Given an ARBAC system S over the set of roles R and a target role* `goal` $\in R$, *the* role-reachability problem *asks whether* `goal` *is reachable in S.*

2.2 Extended Generalized RBAC

Ameer et al. proposed the Extended Generalized RBAC Model (EGRBAC), a fine-grained access control paradigm specifically designed for smart home environments in which the scope of control is at the device-operation level [1].

An EGRBAC policy is a tuple $\langle U, R, UA, D, OP, P, DR, PDRA, EC, ER, EA, RP, RPDRA\rangle$ where U and R are finite sets of *users* and *roles*, respectively. A user is a person who interacts in permitted activities with smart home devices. A role represents the role of a user within the family (e.g., parents, children, neighbors, etc.) and the set $UA \subseteq U \times R$ is the *user-role assignment* relation.

D, OP, P and DR are sets of devices, operations, permissions and device roles, respectively. Smart home appliances such as a smart TV belong to the set of devices D. The set of operations OP are actions that might be performed on devices according to manufacturer specifications. Permissions $P \subseteq D \times OP$ are pairs (device, operation). The relation $PDRA \subseteq P \times DR$ is the many to many *permissions-device to roles assignment*. EC are used to record environmental context, including time and place, and then activates or deactivates Environment Roles (ER) in response. $EA \subseteq 2^{EC} \times ER$ is a many to many subsets of environment conditions to environment roles assignment. $RP \subseteq R \times 2^{ER}$ is a set of role pairs specifying all permissible combinations of a user role and subsets of environment roles. $RPDRA \subseteq RP \times DR$ is the many to many *role pairs-device roles* assignment.

3 RUNNING EXAMPLE

In the paper by Shakarami and Sandhu [26], an EGRBAC policy was presented, which we expanded upon by incorporating insights from several case studies in the literature [1, 3, 4, 11, 12, 16, 21]. In this section, we describe a small excerpt of it, depicted in Figure 2, that we will use throughout the paper as a running example. The policy has been crafted to meet certain key needs, including: granting children limited access to entertainment devices, enabling parents to have full access to all devices in the home at all times, providing the babysitter with control over adult-controlled devices such as adjusting the thermostat or unlocking the front door, giving the housekeeper access to cleaning devices, allowing all residents to utilize lighting devices, and permitting police, fire department, or medical personnel to use the devices in emergency situations. The full policy is available in the supplemental material.

User-Role Assignment UA. In our example, users *alice, james, mary, kate, lucy,* and *john* are assigned to certain roles based on their status within the family/household. For example, the pair *(alice, parent)* belongs to the UA user-role assignment relationship, which means that *alice* is assigned the role *parent*. In addition to the roles *parent, child, babysitter, guest* and *maid*, the policy also includes the role *authority* which is a law enforcement role needed to allow police, firefighters or medics to use certain devices during an emergency.

Devices, Permissions, and Operations. Our running example considers 11 devices, including *TV*, *Fridge* and *Lights*, each associated with their respective manufacturer-defined operations. For example, most devices involve the operations *on* and *off*. Taking into account the operations that the different devices can perform,

the policy distinguishes 12 sets of permissions. For example, both *Lights* and *Fridge* devices can perform *on* and *off* operations, while *Fridge* also has *DisplayFood* functionality.

The set of Device Roles DR. The running examples has 7 device roles: *Entertainment Devices, Adult Controlled, Owner Controlled, Kids Friendly Content, Lighting Devices, Cleaning Devices, Door Device.* The *Entertainment_Devices* role contains permissions to interact with devices such as TV, DVD, PlayStation. *Owner_Controlled* includes permission to control homeowner devices, such as the safe, security cameras, and burglar alarm. Only the homeowner can have control of these devices. *Lighting_Devices* contains permissions to interact with lights in various rooms. As analyzed in [12], access control policies related to lights are the most permissive. In particular, it was pointed out that many users allow access to lighting devices to people physically present inside the house. The *Adult_Controlled* device role encapsulates permission to control devices such as the oven, thermostat, and smoke alarm. This device role includes permissions that can be granted to anyone as long as they are an adult; they do not have to be the homeowner. We also defined a separate device role, *(Door_Device)*, to unlock the front door so as to ensure the least privilege principle. *Cleaning_Devices* contains permission to control devices such as washing machine or the SmartRobot Vacuum Cleaner.

The Permission Device-Role Assignment Relation. PDRA relates permissions to device roles, e.g. P_1 will be associated to *Entertainment_Devices*, while P_2 will be linked to *Kids_Friendly _Content* in such a way that a kid can be assigned only to content that is appropriate for them. The two different permissions for the robot, P_{11} and P_{12}, can be assigned to different device roles, e.g., assign P_{12} to *Cleaning_Devices* device role. Thus, it is possible to turn the robot on/off, but extra access to robot setting would not be provided.

The set of Environment Roles ER. The policy has 6 environment roles: *Entertainment_Time, Any_Time, At_Home, Emergency _Time, Wednesday,* and *Friday. Entertainment_Time* is active when both the environment conditions *weekends* and *evenings* are active; *Any_Time* should always be active; *At_Home* will be activated when the condition *at_home* is triggered (e.g. by detecting WiFi connection); *Emergency_Time* will be triggered by the environmental condition *emergency* (e.g. when alarm is activated);
As an example, two days of the week active with the condition *certain_day*, *Wednesday* and *Friday*, are given.

The set of Role-Pairs RP. The running example has 7 role pairs. The role kid has been associated with the environment role *Entertainment_Time* to form the role pair *(kid, Entertainment_Time)*. In this way we can assign kids permission to control entertainment devices during weekend, evenings only. Parents, must have access to all IoT devices regardless of location and time. The role is associated with the environment role *Any_Time* that is always active, forming the role pair *(parent, Any_Time)*. External users such as a guest or a maid, can only interact with the devices if at home. The *At_Home* environment role is activated if wi-fi connection is detected; we form the following role pairs: *(maid, At_Home)* and *(guest, At_Home)*. To manage emergency situations we introduced the *Emergency_Time* environment role that can be associated with the authority to allow police, firefighters, or medics to use the devices during an emergency, *(authority, Emergency_Time)*. We have also defined an environment role to differentiate permissions based on the current day. For example, we have the environmental roles *Wednesday* and *Friday* that can be coupled with the role babysitter resulting in *(babySitter, Wednesday)* and *(babySitter, Friday)*. The same role can be associated with different environment roles according to the desired day.

The Role-Pair Device-Role Assignment Relation. The role pair *(parent, Any_Time)* can use *Owner_Controlled* and *Adult_Controlled* device roles without environmental restrictions. Guests or neighbors should not have any form of authorization or access to IoT devices located in the home, except for what relates to entertainment or lighting. So we can have *((guest, At_Home), Lighting_Devices)*, *((guest, At_Home), Entertainment_Devices)* in the set RPDRA. The babysitter must be able to use the functions of adult-controlled devices, such as unlocking doors, turning the oven on or off, and controlling the thermostat. However, you do not want to grant the babysitter unnecessary access, e.g., changing the thermostat program. The maid should have access to cleaning devices, such as the washing machine, vacuum cleaner, dishwasher, etc., but she does not need to use the oven or thermostat, we then add *((maid, At_Home), Cleaning_Devices)* in the RPDRA relation.

4 AN ADMINISTRATIVE MODEL FOR EGRBAC

Shakarami and Sandhu [26] proposed a role-based administrative model for the EGRBAC operational model. It consists of several administrative components. Our attention is focused on the one responsible for assigning role pairs to device roles (i.e., for changes to the RPDRA assignment relation). In order to allow this component to implement policies that enable the assignment of role pairs to device roles based on their associations with other device roles, we propose a slightly modified approach for updating the RPDRA relation. Our modification involves including preconditions for assignment actions, which follows the paradigm employed in ARBAC97.

For ease of presentation, below we only consider the elements of the model which are needed for our analysis. Let AEGRBAC = $\langle EGRBAC, AUser, AR, AUA, assignRPDR, revokeRPDR\rangle$ where:

- $EGRBAC = \langle U, R, UA, D, OP, P, DR, PDRA, EC, ER, EA, RP, RPDRA\rangle$ is an EGRBAC policy described in section 2.2.
- AR is a set of administrative roles.
- $AUser \subset U$ is a set of administrative users.
- $AUA \subset AUser \times AR$ is a many-to-many assignment relationship between administrative users and administrative roles.
- The authorization functions:
  - $assignRPDR(auser, ar, rp, Pos, Neg, dr)$, means that an administrative user $auser \in AUser$ with the administrative role $ar \in AR$ can assign the role pair $rp \in RP$ to the device role $dr \in DR$ if the precondition $(Pos, Neg) \subseteq 2^{DR} \times 2^{DR}$ is met. The precondition is a conjunction of literals where each literal is in a positive form dr (when $dr \in Pos$) or a negative form $\neg dr$ (where $dr \in Neg$) for some dr in DR.

– *revokeRPDR*(*auser*, *ar*, *rp*, *dr*), means that an administrative user *auser* ∈ *AUser* with the administrative role *ar* ∈ *AR* can remove the pair (*rp*, *dr*) from the set RPDRA.

Since our analysis focuses on variation in the RPDRA relationship, we consider the tuple ⟨*U*, *R*, *UA*, *D*, *OP*, *P*, *PDRA*, *EC*, *ER*, *EA*⟩ as fixed. For ease of presentation in the remainder of the paper we refer to an EGRBAC policy as the tuple ⟨*DR*, *RP*, *RPDRA*⟩ and assume that the other components are fixed and implicit. Unlike ARBAC models, the EGRBAC-based administrative model in the assignRPDR function explicitly specifies which administrative user is assigning the permission. For our DR reachability security analysis we do not take into account the set of users and we assume that in each administrative role AR there is always an assigned user, (*admin* ∈ AUser) such that we have one administrator (*Admin* ∈ AR) who can perform any action. From now on we refer to this model as a tuple *AEGRBAC* = ⟨*admin*, *Admin*, *RP*, *DR*, *RPDRA* *assignRPDR*, *revokeRPDR*⟩.

4.1 Administrative Use Case

In [26] Shakarami and Sandhu presented a case study by defining both an operational and an administrative policy for smart home. Building upon such a policy we have developed an AEGRBAC policy for smart home. In this section we describe a small portion of this policy to be used as a running example (see Figure 3).

Administrative User AUser, Administrative Roles AR and AUA assignment relation. *admin* ∈ AUser is a user with administrative authorizations. *Admin* ∈ AR is an administrative role that can perform any action within a smart home. The *admin* user is assigned to the administrative role *Admin*.

The set of Role-Pairs RP and Device-Roles DR. The sets of role pairs and device roles are the same as those already described in the running example in section 3.

The Authorization Function AssignRPDR. As an example we list just a few of the assign functions defined in our policy: ⟨*Admin, (babySitter,{Friday}), ¬Adult_Controlled, Door_Device*⟩ means that *Admin* assigns the babysitter who comes on Friday permission to control the door if she is not in possession of the adults' devices; We may want to assign different permissions based on the day; we want the babysitter on Wednesday to be able to use the children's devices, this results in the function ⟨*Admin, (babySitter,{Wednesday}), Lighting_Devices, Kids_Friendly_Content*⟩.
⟨*Admin, (parent,{Any_Time}), -, Adult_Controlled*⟩ allows a parent to control the adult devices without restrictions; ⟨*Admin, (guest,{At_Home}), Door_Device, Lighting_Devices*⟩ the *Admin* assigns a guest permission to control the lights if they have access to the door.
⟨*Admin, (kid,{Entertainment_Time}), ¬ Entertainment_Devices, Kids_Friendly_Content*⟩, the *Admin* allows the child to control the devices appropriate for him and finally ⟨*Admin, (maid,{At_Home}), Door_Device & Lighting_Devices, Cleaning_Devices*⟩ leads the maid to control the cleaning devices only if they have control over the door and the lights.
The Authorization Function RevokeRPDR. The running example has 7 *RevokeRPDR* functions one for each *AssignRPDR* function.

4.2 Device Role Reachability

AEGRBAC Systems: An AEGRBAC system can be considered as a state transition system that evolves via administrative actions. Formally it can be defined as a tuple: $S = \langle admin, Admin, RP, DR, RPDRA, assignRPDR, revokeRPDR\rangle$.

A *configuration* of S is any $RPDR \subseteq RP \times DR$. A configuration is *initial* if $c_0 = (RP_0, DR_0) \in RPDR_0$ where in c_0 we have the initial assignments between role pairs and device roles.

Given two S configurations $c = (RP, DR) \in RPDR$ and $c' = (RP', DR') \in RPDR'$, there is a *transition* from c to c' with rule $m \in (assignRPDR \cup revokeRPDR)$, denoted $c \xrightarrow{m} c'$, if one of the following holds:

[**assignRPDR move**] $m = (admin, Admin, rp, Pos, Neg, dr)$, the role pairs-device roles assignment relation where $Pos \subseteq \{dr \mid (rp, dr) \in RPDRA\}$, $Neg \subseteq \{DR \setminus Pos\}$ and $RPDRA' = RPDRA \cup \{(rp, dr)\}$;

[**revokeRPDR move**] $m = (admin, Admin, rp, dr)$, the role pair-device role revoke relation and $RPDRA' = RPDRA \setminus \{(rp, dr)\}$;

A *run* of S is any finite sequence of S transitions $\pi = c_0 \xrightarrow{m_1} c_1 \xrightarrow{m_2} \ldots c_n \xrightarrow{m_n} c_{n+1}$ for some $n \geq 0$, where c_0 is an *initial* configuration of S. An S configuration c is reachable if c is the last configuration of an S run.

Definition 4.1. *(*DR reachability Problem*)*
Let $S = \langle admin, Admin, RP, DR, RPDRA, assignRPDR, revokeRPDR\rangle$ *be an AEGRBAC system. For any device role* $dr \in DR$, dr *is* reachable *in* S *if there is an* S *reachable configuration* $c \subseteq RP \times DR$ *such that* $(rp, dr) \in c$. *The* DR reachability problem *asks whether a device-role* `dr_goal` $\in DR$ *is reachable in* S.

5 THE ANALYSIS

In this section, we reduce the DR reachability problem to the role-reachabilty problem in administrative RBAC [8, 15], thus, enabling the use of techniques and tools, designed to address the role-reachabilty problem in administrative RBAC, to solve the DR reachability problem.

We now show our reduction from the DR reachability problem in *AEGRBAC* to the role-reachability problem in *ARBAC*.

5.1 From AEGRBAC to ARBAC

The idea behind the transformation is as follows: in ARBAC we track users and roles while in AEGRBAC the goal is to track the association between role pairs and device roles so it becomes quite obvious that in the transformation each role pair in AEGRBAC must be translated to an ARBAC user and each device role must correspond to a role. Although the assign function serves a similar purpose in both AEGRBAC and ARBAC, there is a fundamental difference in its semantics. In AEGRBAC, the role pair to which the device role should be assigned is explicitly specified, while in ARBAC, any authorized administrator can assign any user to the target role. To address this issue in our transformation, we adopt an approach where each role pair rp is linked to a user u_{rp} as well as a dummy role r_{rp}, which is exclusively assigned to u_{rp}. By associating the particular user with the given role pair, we ensure

AUser admin;
AR Admin;
AUA (admin, Admin);
RP (parent,Any_Time), (maid,At_Home), (guest,At_Home), (babySitter,Friday), (babySitter,Wednesday), (kid,Entertainment_Time);
DR Owner_Controlled, Adult_Controlled, Kids_Friendly_Content, Entertainment_Devices, Lighting_Devices, Cleaning_Devices, Door_Device;
RPDRA ⟨(parent,Any_Time), Owner_Controlled⟩
RevokeRPDR
⟨admin, Admin, (babySitter,Friday), Door_Device⟩
⟨admin, Admin, (parent,Any_Time), Owner_Controlled⟩
⟨admin, Admin, (babySitter,Wednesday), Kids_Friendly_Content⟩
⟨admin, Admin, (guest,At_Home), Lighting_Devices⟩
⟨admin, Admin, (kid,Entertainment_Time), Kids_Friendly_Content⟩
⟨admin, Admin, (maid,At_Home), Cleaning_Devices⟩
AssignRPDR
⟨admin, Admin, (babySitter,Friday), ¬Adult_Controlled, Door_Device⟩
⟨admin, Admin, (parent,Any_Time), -, Adult_Controlled⟩
⟨admin, Admin, (guest,At_Home), Door_Device, Lighting_Devices⟩
⟨admin, Admin, (kid,Entertainment_Time), ¬Entertainment_Devices, Kids_Friendly_Content⟩
⟨admin, Admin, (babySitter,Wednesday), Lighting_Devices, Kids_Friendly_Content⟩
⟨admin, Admin, (maid,At_Home), Door_Device & Lighting_Devices, Cleaning_Devices⟩

Figure 3: AEGRBAC use case

the precise identification of the role pair to which a device role should be assigned.

Definition 5.1. *(Transformation)*
Let $S = \langle AUser, AR, AUA, RP, DR, RPDRA\ assignRPDR, revokeRPDR\rangle$ *be an AEGRBAC system where* $AR = \{Admin\}$, $AUser = \{admin\}$, $AUA = \{(admin, Admin)\}$.
We construct a corresponding ARBAC system $S' = \langle U, R, AR, UA, can\_assign, can\_revoke\rangle$ *as follows:*

1. *add user admin to* U;
2. *add the tuple* $(admin, Admin)$ *to* UA;
3. *for each role pair* $rp \in RP$:
   - *add the user* u_rp *to* U;
   - *add the dummy role* r_rp *to* R;
4. *for each device role* $dr \in DR$
   - *add the role* r_dr *to* R;
5. *for each* $(rp, dr) \in RPDRA$
   - *add the tuple* $(u\_rp,\ r\_dr)$ *in* UA;
6. *for each* $rp \in RP$
   - *add the tuple* $(u\_rp,\ r\_rp)$ *in* UA;
7. *for each* $(admin, Admin, rp, Pos\_dr, Neg\_dr, dr) \in assignRPDR$
   - *add* $(Admin, r\_rp \cup Pos, Neg, r\_dr)$ *into can_assign where* $Pos = \{r\_dr\ |dr \in Pos\_dr\}$ *and* $Neg = \{r\_dr\ |dr \in Neg\_dr\}$;
8. *for each* $(Admin, rp, dr) \in revokeRPDR$
   - *add* $(admin, Admin, r\_dr)$ *into can_revoke where* $dr = r\_dr$.

Before proceeding with the proof we give the following definition.

Definition 5.2 (**Matching Configurations**). *Let* $S = \langle admin, Admin, RP, DR, RPDRA\ assignRPDR, revokeRPDR\rangle$ *be an AEGRBAC system and let* $S' = \langle U, R, AR, UA, can\_assign, can\_revoke\rangle$ *be the corresponding ARBAC system obtained from the transformation in Definition 5.1. We say that two configurations* c *in* S *and* c' *in* S' *are matching if for each* $rp \in RP$ *and each* $dr \in DR$ *it holds that*

$$(rp,\ dr) \in c \iff (u\_rp,\ r\_dr) \in c'.$$

We first need the following lemma.

Lemma 5.1. *Let* $S = \langle admin, Admin, RP, DR, RPDRA\ assignRPDR, revokeRPDR\rangle$ *be an AEGRBAC system and let* $S' = \langle U, R, AR, UA, can\_assign, can\_revoke\rangle$ *be the corresponding ARBAC system obtained from the transformation of Definition 5.1. A configuration* c *is reachable in* S *iff there exists a reachable configuration* c' *in* S' *such that* c *and* c' *are matching.*

Proof. $\Longrightarrow$ We first prove that if c is a reachable configuration in S then there exists a reachable configuration c' in S' such that c and c' are matching. Let π be an S run having c as last configuration. The proof follows by induction on the length n of π. When $n = 0$, it is easy to see that the thesis follows by construction (see Definition 5.1). Assume by inductive hypothesis that the thesis holds for some $i = n > 0$. Let now consider a run of length $i+1$, $\pi = c_0 \xrightarrow{m_1} c_1 \xrightarrow{m_2} \ldots c_i \xrightarrow{m_{i+1}} c_{i+1}$. From the inductive hypothesis, there exists a run $\pi' = c'_0 \xrightarrow{m'_1} c'_1 \xrightarrow{m'_2} \ldots \xrightarrow{m'_t} c'_t$ in S' for some $t \geq 0$ such that a pair (ρ, δ) belongs to c_i iff the pair (u_ρ, r_δ) belongs to c'_t.
Since the move m_{i+1} can be either an *assignRPDR* or a *revokeRPDR*, we distinguish the following two cases:

AssignRPDR move. Let $m_{i+1} = (admin,\ Admin,\ rp,\ Pos\_dr,\ Neg\_dr,\ dr) \in assignRPDR$. Thus, $c'_{i+1} = c_i \cup \{(rp,\ dr)\}$. From step 7 of Definition 5.1 there exists a rule $m'_{t+1} = (Admin,\ r\_rp \cup Pos,\ Neg,\ r\_dr) \in can\_assign$ where $Pos = \{r\_\delta\ |\delta \in Pos\_dr\}$ and $Neg = \{r\_\delta\ |\delta \in Neg\_dr\}$. Notice that if m_{i+1} can be fired then for each device-role $\delta \in Pos\_dr$ the role-pair $(rp, \delta) \in c_i$ while for each device-role $\delta \in Neg\_dr$ the role-pair $(rp, \delta) \notin c_i$. From inductive hypothesis if $\delta \in Pos\_dr$ then $r\_\delta \in Pos$ while if $\delta \in Neg\_dr$ then $r\_\delta \in Neg$, thus if m_{i+1} can be fired by adding the pair (rp, dr) to c_{i+1} then m'_{t+1} can be fired to add user u_rp to role r_dr. Indeed, user u_rp belongs to role r_rp by construction (see step 3 of Definition 5.1) and no rule exists to revoke this membership. Moreover, by construction no other rule exists to add any other user to role

rp. Therefore, move m'_{t+1} adds the pair (rp, dr) to c'_{t+1}. Hence, c_{i+1} and c'_{t+1} are matching configurations and the thesis holds.

RevokeRPDR move. Let $m_{i+1} = (admin, Admin, rp, dr) \in revokeRPDR$. Thus, $c'_{i+1} = c_i \setminus \{(rp,\ dr)\}$. From step 8 of Definition 5.1 there exists a rule $m'_{t+1} = (Admin, r\_dr) \in can\_revoke$. From inductive hypothesis if $(rp,\ dr) \in c_i$ then $(u\_rp,\ r\_dr) \in c_t$. Therefore, m'_{t+1} can be applied to revoke user u_rp from role r_dr. Hence, c_{i+1} and c'_{t+1} are matching configurations and the thesis holds.

$\Longleftarrow$ We now prove that if we have a configuration c' in S' then we have a configuration c in S such that S and S' are equivalent.

The proof proceeds by induction on the length n of the run. When $n = 0$, the thesis follows by construction of Definition 5.1. By inductive hypothesis assume the thesis for some $n = i > 0$. Let now consider a run of length $i + 1$, $\pi' = c'_1 \xrightarrow{m'_1} c'_2 \xrightarrow{m'_2} \ldots \xrightarrow{m'_{t-1}} c'_{t+1}$ in S'. From the inductive hypothesis, there exists a run $\pi = c_1 \xrightarrow{m_1} c_2 \xrightarrow{m_2} \ldots c_i \xrightarrow{m_i} c_i$ in S for some $i >= 0$ such that (u_rp, r_dr) belongs to c'_t iff (rp, dr) belongs to c_i, for some $rp \in RP$.

We distinguish two cases, one where $m' \in can\_assign$ and one where $m' \in can\_revoke$:

[**can_assign move**] Let $m'_t = (admin,\ Admin,\ r\_rp \cup Pos,\ Neg,\ r\_dr) \in can\_assign$ such that $UA' = (UA \cup \{(u\_rp,\ r\_dr)\})$. We construct $\pi = c_1 \xrightarrow{m_1} c_2 \xrightarrow{m_2} \ldots c_i \xrightarrow{m_i} c_{i+1}$ where $m_i = (admin, Admin,\ rp, Pos\_dr,\ Neg\_dr,\ dr) \in assignRPDR$ such that $RPDRA' = RPDRA \cup \{(rp,\ dr)\}$. Notice that m_i belongs to $assignRPDR$ (see step 6 of Definition 5.1) and can_assign function keeps track of the role pair rp thanks to the introduction of the dummy role r_rp (see step 2 of Definition 5.1); thus, S and S' are still matching.

[**can_revoke move**] Let $m'_t = (admin,\ Admin,\ r\_dr) \in can\_revoke$ such that $UA' = UA \setminus \{(u\_rp,\ r\_dr)\}$. We construct $\pi = c_1 \xrightarrow{m_1} c_2 \xrightarrow{m_2} \ldots c_i \xrightarrow{m_i} c_{i+1}$ where $m_i = (admin,\ Admin,\ rp,\ dr) \in revokeRPDR$ such that $RPDRA' = RPDRA \setminus \{(rp,\ dr)\}$. Notice that m_i belongs to $revokeRPDR$ (see step 7 of Definition 5.1); thus, S and S' are still matching. □

We are now ready to prove our main theoretical result. In particular, we show that the DR reachability problem of Definition 4.1 can be reduced to the role-reachability problem of Definition 2.1.

Theorem 1. *Let $S = \langle admin,\ Admin,\ RP,\ DR,\ RPDRA, assignRPDR, revokeRPDR\rangle$ be an AEGRBAC system and let $S' = \langle U,\ R,\ AR, UA, can\_assign, can\_revoke\rangle$ be the corresponding ARBAC system of Definition 5.1. The target device-role* `dr_goal` *is reachable in S iff the target role* `goal` *is reachable in S'.*

$\Longrightarrow$ We first show that if $(rp,$ `dr_goal`$) \in RPDRA$ is reachable in S then the role goal = `r_dr_goal` is reachable in S'. Let $\pi = c_1 \xrightarrow{m_1} c_2 \xrightarrow{m_2} \ldots c_{n-1} \xrightarrow{m_{n-1}} c_n$ be a run of S such that $(rp,$ `dr_goal`$)$ belongs to c_n. The thesis follows from Lemma 5.1 which states that there exists a run $\pi' = c'_1 \xrightarrow{m'_1} c'_2 \xrightarrow{m'_2} \ldots c'_{t-1} \xrightarrow{m'_{t-1}} c'_t$ in S' such that S and S' are matching, thus if $(rp,$ `dr_goal`$) \in RPDRA$ then $(u\_rp,$ `r_dr_goal`$) \in UA$.

$\Longleftarrow$ We now show that if the role goal = `r_dr_goal` is reachable in S' such that $(u\_rp, r\_$`dr_goal`$) \in UA$ then $(rp,$ `dr_goal`$) \in RPDRA$ is reachable in S. Let $\pi' = c'_1 \xrightarrow{m'_1} c'_2 \xrightarrow{m'_2} \ldots c'_{t-1} \xrightarrow{m'_{t-1}} c'_t$ be a run of S' such that $(u\_rp,$ `r_dr_goal`$)$ belongs to c_t. The thesis follows from Lemma 5.1 which states that there exists a run $\pi = c_1 \xrightarrow{m_1} c_2 \xrightarrow{m_2} \ldots c_{n-1} \xrightarrow{m_{n-1}} c_n$ in S such that S and S' are matching, thus if $(u\_rp,$ `r_dr_goal`$) \in UA$ then $(rp,$ `dr_goal`$) \in RPDRA$.

6 EXPERIMENTAL RESULTS

For the experimental analysis we considered the full policy case study we constructed. We converted the AEGRBAC policy to the corresponding ARBAC policy following the transformation in Section 5.1 and continued the analysis by using the tool VAC [7], an automated tool for the security analysis of role-reachability properties in administrative role-based access control. The resulting ARBAC policy has a total of 162 authorization functions (118 can_assign and 44 can_revoke), 44 roles and 16 users.

The primary objective of the Smart Home policy's security requirements is to ensure the safety of household members. This involves verifying that a specific category of users (role pairs) cannot access devices outside their jurisdiction, to prevent potential privilege escalation. Availability properties are also checked to confirm that a role pair has access to all necessary devices before granting access to a new device. Table 1 provides a summary of our experimental findings, including the time taken by the VAC tool to conduct the analysis and whether a given device role was deemed reachable or unreachable.

Our experiments were carried out on a MacBook Pro with an Apple M1 Chip, running a Kali Virtual Machine with 8Gb RAM.

In the first experiment, we examined whether a child could access devices controlled by an adult, and found that the Adult Controlled role was unreachable. The second experiment assessed whether a guest could control the owner's devices, and the analysis determined that the target role was unreachable. The third experiment focused on whether a maid could use cleaning devices without access to the room being cleaned, and the policy stipulated that the maid must have permission to control both the door and lighting devices before being granted access to the cleaning devices. The fourth and fifth experiments checked whether a babysitter or a guest could access a child's smart toy when the policy required that only family members and babysitters could control the device. As indicated in Table 1, the analysis terminated in a few seconds, regardless of whether the target pair was reachable or not. For benchmarks with a reachable target, the analysis took less than 2 seconds, including the generation of a counterexample. We tested the policy against 21 queries, the results of the full set of experiments can be found in the extended abstract.

Experiment	Time	Result
1. Kid to AdultControlled	28.85s	U
2. Guest to OwnerControlled	1.91s	U
3. Maid to CleaningDevices	1.11s	R
4. BabySitter to KidsFriendlyContent	1.17s	R
5. Guest to KidsFriendlyContent	1.69s	U

Table 1: Experimental results

7 RELATED WORK

Smart homes and IoT. The Internet of Things (IoT) in smart homes has been extensively researched by security experts, with a focus on identifying security and privacy vulnerabilities [13, 28]. Many researchers have also analyzed IoT frameworks to assess security challenges and design issues [5, 6, 19, 22, 28]. Access control is considered one of the key security services in IoT, and has been the subject of significant research. Ouaddah et al. [23] have conducted extensive research on access control in IoT environments.

Extended Generalized Role- Based Access Control, has been introduced to capture the intricate user-device-context interactions that are prevalent in smart home environments [1]. A recent study by Sandhu et al. [2] proposes an attribute-based access control model for smart home IoT (HABACα).

Security analysis. The first access control model that takes evolving policies into consideration is the HRU model, as noted in [18]. This model uses a standard access control matrix to represent the authorization state, allowing for the addition of subjects and objects. However, the reachability problem in HRU is undecidable. Li et al. [17] investigated role-based policies that allow principals to add or remove role-membership rules, and studied the complexity of specific analysis queries in such a system. Sasturkar et al. [25] proved that reachability in ARBAC policies is Pspace-complete, while Stoller et al. [27] showed that mixed roles are the main source of intractability in computational complexity. Jayaraman et al. proposed Mohawk, a tool for finding errors and proving correctness in complex ARBAC policies [14], while Ferrara et al. presented VAC, an automatic and scalable tool for the reachability problem of ARBAC policies with an unbounded number of users [7–10]. Ranise et al. proposed the ASASPXL tool, which can analyze large ARBAC policies [24].

Acknowledgments: This work was partially supported by project *VITALITY Ecosystem, Spoke 1 MEGHALITIC* under the NRRP MUR program funded by the EU - NGEU and by INDAM-GNCS 2023.

REFERENCES

[1] Safwa Ameer, James Benson, and Ravi Sandhu. 2020. The EGRBAC Model for Smart Home IoT. In *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. 457–462. https://doi.org/10.1109/IRI49571.2020.00076

[2] Safwa Ameer, James Benson, and Ravi Sandhu. 2022. An Attribute-Based Approach toward a Secured Smart-Home IoT Access Control and a Comparison with a Role-Based Approach. *Information* 13, 2 (2022), 60.

[3] Bejarano Andrés, Fernández Alejandra, Jimeno Miguel, Salazar Augusto, and Wightman Pedro. 2016. Towards the Evolution of Smart Home Environments: A Survey. *International Journal of Automation and Smart Technology* 6, 3 (2016). https://www.ausmt.org/index.php/AUSMT/article/view/1039

[4] Diane Cook, G. Youngblood, E.O. III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. 2003. MavHome: An agent-based smart home. *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications, PerCom 2003*, 521 – 524. https://doi.org/10.1109/PERCOM.2003.1192783

[5] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security Analysis of Emerging Smart Home Applications. 636–654. https://doi.org/10.1109/SP.2016.44

[6] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 531–548.

[7] Anna Lisa Ferrara, P. Madhusudan, Truc Lam Nguyen, and Gennaro Parlato. 2014. VAC - Verifier of Administrative Role-based Access Control Policies. In *CAV*.

[8] Anna Lisa Ferrara, P. Madhusudan, and Gennaro Parlato. 2012. Security Analysis of Role-Based Access Control through Program Verification. In *IEEE Computer Security Foundation*, Stephen Chong (Ed.). IEEE, 113–125.

[9] Anna Lisa Ferrara, P. Madhusudan, and Gennaro Parlato. 2013. Policy Analysis for Self-administrated Role-Based Access Control. In *TACAS*. 432–447.

[10] Anna Lisa Ferrara, Anna Cinzia Squicciarini, Cong Liao, and Truc L. Nguyen. 2017. Toward Group-Based User-Attribute Policies in Azure-Like Access Control Systems. In *31st Annual IFIP WG 11.3 Conference, DBSec 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10359)*. Springer, 349–361.

[11] Victoria Haines, Val Mitchell, Catherine Cooper, and Martin Maguire. 2007. Probing user values in the home environment within a technology driven Smart Home project. *Personal and Ubiquitous Computing* 11 (06 2007). https://doi.org/10.1007/s00779-006-0075-6

[12] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking Access Control and Authentication for the Home Internet of Things ({{{{{IoT}}}}}). In *27th USENIX Security Symposium (USENIX Security 18)*. 255–272.

[13] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (Xi'an, China) *(ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 461–472. https://doi.org/10.1145/2897845.2897886

[14] Karthick Jayaraman, Mahesh V. Tripunitara, Vijay Ganesh, Martin C. Rinard, and Steve J. Chapin. 2013. Mohawk: Abstraction-Refinement and Bound-Estimation for Verifying Access Control Policies. *ACM Trans. Inf. Syst. Secur.* 15, 4 (2013), 18. https://doi.org/10.1145/2445566.2445570

[15] Somesh Jha, Ninghui Li, Mahesh V. Tripunitara, Qihua Wang, and William H. Winsborough. 2008. Towards Formal Verification of Role-Based Access Control Policies. *IEEE Trans. Dependable Sec. Comput.* 5, 4 (2008), 242–255. https://doi.org/10.1109/TDSC.2007.70225

[16] David Leake, Ana Maguitman, and Thomas Reichherzer. 2006. Cases, Context, and Comfort: Opportunities for Case-Based Reasoning in Smart Homes. *Lecture Notes in Computer Science* 4008, 109–131. https://doi.org/10.1007/11788485_7

[17] Ninghui Li and Mahesh V. Tripunitara. 2004. Security analysis in role-based access control. In *9th ACM SACMAT*. ACM, 126–135. https://doi.org/10.1145/990036.990058

[18] W.L. Ruzzo M.H. Harrison and J.D. Ullman. 1999. Protection in Operating Systems. *Communications of the ACM Trans. Inf. Syst. Secur.* 2, 1 (1999), 105–135. https://doi.org/10.1145/300830.300839

[19] Philipp Morgner, Stephan Mattejat, and Zinaida Benenson. 2016. All your bulbs are belong to us: Investigating the current state of security in connected lighting systems. *arXiv preprint arXiv:1608.03732* (2016).

[20] M.J. Moyer and M. Abamad. 2001. Generalized role-based access control. In *Proceedings 21st International Conference on Distributed Computing Systems*. 391–398. https://doi.org/10.1109/ICDSC.2001.918969

[21] Chris Nugent, Dewar Finlay, Richard Davies, Haiying Wang, Huiru Zheng, Josef Hallberg, Kåre Synnes, and Maurice Mulvenna. 2007. homeML – An Open Standard for the Exchange of Data Within Smart Environments, Vol. 4541. 121–129. https://doi.org/10.1007/978-3-540-73035-4_13

[22] Temitope Oluwafemi, Tadayoshi Kohno, Sidhant Gupta, and Shwetak Patel. 2013. Experimental Security Analyses of {Non-Networked} Compact Fluorescent Lamps: A Case Study of Home Automation Security. In *LASER 2013 (LASER 2013)*. 13–24.

[23] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. 2017. Access control in the Internet of Things: Big challenges and new opportunities. *Computer Networks* 112 (2017), 237–262.

[24] Silvio Ranise, Anh Tuan Truong, and Alessandro Armando. 2012. Boosting Model Checking to Analyse Large ARBAC Policies. In *Security and Trust Management - 8th International Workshop, STM*. 273–288. https://doi.org/10.1007/978-3-642-38004-4_18

[25] Amit Sasturkar, Ping Yang, Scott D. Stoller, and C. R. Ramakrishnan. 2006. Policy Analysis for Administrative Role Based Access Control. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19) 2006*. 124–138. https://doi.org/10.1109/CSFW.2006.22

[26] Mehrnoosh Shakarami and Ravi Sandhu. 2021. Role-Based Administration of Role-Based Smart Home IoT. In *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems* (Virtual Event, USA) *(SAT-CPS '21)*. Association for Computing Machinery, New York, NY, USA, 49–58. https://doi.org/10.1145/3445969.3450426

[27] Scott D. Stoller, Ping Yang, C. R. Ramakrishnan, and Mikhail I. Gofman. 2007. Efficient policy analysis for administrative role based access control. In *Proc. of the 2007 ACM Conference on Computer and Comm. Security, CCS*. ACM, 445–455. https://doi.org/10.1145/1315245.1315300

[28] Blase Ur, Jaeyeon Jung, and Stuart Schechter. 2013. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*, Vol. 29. 209–218.