Methods and Tools for Policy Analysis

AMANI ABU JABAL, MARYAM DAVARI, and ELISA BERTINO, Purdue University, USA
CHRISTIAN MAKAYA, SERAPHIN CALO, and DINESH VERMA, IBM Research, USA
ALESSANDRA RUSSO, Imperial College, UK
CHRISTOPHER WILLIAMS, The Defence Science and Technology Laboratory, UK

Policy-based management of computer systems, computer networks and devices is a critical technology especially for present and future systems characterized by large-scale systems with autonomous devices, such as robots and drones. Maintaining reliable policy systems requires efficient and effective analysis approaches to ensure that the policies verify critical properties, such as correctness and consistency. In this paper, we present an extensive overview of methods for policy analysis. Then, we survey policy analysis systems and frameworks that have been proposed and compare them under various dimensions. We conclude the paper by outlining novel research directions in the area of policy analysis.

CCS Concepts: • Security and privacy → Access control; Authorization; Distributed systems security; Firewalls; Network security;


ACM Reference Format:

1 INTRODUCTION
Advances in robotics and in artificial intelligence, and the explosion of low-cost mobile phones, drones, embedded devices and the Internet of Things (IoT) are leading to the massive deployment of autonomous systems. Such systems will include intelligent autonomous devices able to collaborate, to gather fine-grained information from the operating environments and to physically act in these environments. Many such devices will have very rich cognitive capabilities and be able to control other devices autonomously and will need to interact with other autonomous devices. There needs to be controls in place to manage such interactions such that they are not harmful to any of the individual or collective devices. Devices, especially mobile devices, will move into physically unprotected spaces and will have to interact with “unknown” devices [43]. However in order for autonomous devices to effectively and efficiently carry out their tasks, proper policy-based management is critical. At a higher level, policies can be defined as directives given by a managing
party to one or more managed parties in order to guide their behavior. Policies can be of different types; for example:

- **Constraint policies** govern actions executed by the managed parties, by which different actions are deemed allowed, not allowed, or obligatory. Access control policies [44] represent a well-known example of constraint policies.
- **Goal-based policies** state goals that the managed parties must achieve, e.g., maintain a minimum threshold of utilization or try to finish a task before a specific deadline.
- **Utility-based policies** aim at producing the best outcome according to some value function, such as minimizing the energy consumption.

However, because policies represent the key input for policy-based management, it is critical that they be "correct and fit for their use". Such a requirement has thus motivated research on methods and tools for policy analysis, to determine whether sets of policies verify some given properties (e.g., consistency), and for assessing the impact of policy changes. We believe that in the context of present and future operations for autonomous devices, assuring that policies are correct and adequate to address a large variety of situations is a critical requirement. Policy analysis is also crucial to support policy evolution as the analysis results can provide indications on how to best modify policies to fit continuously changing contexts and situations. In this respect, methods and tools to analyze policies and to support policy evolution and adaptation will be increasingly relevant.

The goal of this survey is thus to provide a comprehensive view of existing methods and tools developed for policy analysis in different domains and for different analysis goals. Moreover, given there are no existing policy analysis tools for autonomous devices in the IoT context and in other contexts, this survey aims at outlining, as conclusions, the key requirements for next-generation policy-based management systems. We emphasize that, as policies will be increasingly crucial for autonomous systems, analysis methods for policies will be critical.

The paper is organized as follows. Section 2 briefly discusses the most relevant policy domains, namely access control, and network management. Section 3 covers the properties that the different analysis methods address, whereas Section 4 presents a comprehensive taxonomy of the analysis methods. Section 5 presents an overview of several well-known approaches and frameworks, while Section 6 compares them under different dimensions, including the analysis methods used. Finally, Section 7 outlines a few novel research directions.

## 2 POLICY DOMAINS

In this section, we briefly discuss policies in the two major areas in which policy-based management has been widely deployed, namely access control and computer network management.

### 2.1 Access Control Policies

Access control policies specify which subject (e.g., user, process, and application) can access which resources (e.g., files) for performing which actions (e.g., read, write). Many access control models have been proposed, including models that take time and location into account [42, 67], and models for privacy-sensitive data [178]. Access control mechanisms are embedded in many different systems, ranging from operating systems to database management systems. In what follows we briefly describe a few relevant models and refer the reader to Bertino et al. [44] for details.

#### 2.1.1 Role-based Access Control (RBAC)

The RBAC model consists of four components [79, 191]: users, roles, permissions, and sessions. A role represents an organizational function within a given domain, such as a coalition. Roles are granted permissions required for the execution of their functions. A permission consists of the specification of a protected object and an action,
The RBAC model definition includes several functions. The user assignment (UA) function specifies which user is assigned to which roles, whereas the permission assignment (PA) function specifies the set of permissions assigned to a role. In a particular system, when a user \( u_i \in U \) becomes active, the user function assigns a session \( s_i \in S \) to \( u_i \), while the roles function maps \( s_i \) to the subset of roles which are associated with \( u_i \). The following definition (adapted from Sandhu et al. [191]) formally defines the RBAC model.

**Definition 1** (RBAC Model [42]). The model consists of the following components.

- \( U, R, P, S \) refer to the set of users, roles, permissions, and sessions, respectively.
- \( PA \) is the permission assignment function that assigns permissions to roles (i.e., \( PA \subseteq R \times P \) and \( PA(r_i) \subseteq P, \forall r_i \in R \)).
- \( UA \) is the user assignment function that assigns users to roles (i.e., \( UA \subseteq U \times R \) and \( UA(u_i) \subseteq R, \forall u_i \in U \)).
- The user function assigns a session to a single user (i.e., \( user : S \rightarrow U | user(s_i) \in U \)).
- The roles function assigns a session to the roles associated with the user activated the corresponding session (i.e., \( roles \subseteq S \times 2^R | roles(s_i) = \{ r | (user(s_i), r) \in UA \} \)).
- \( RH \) is the role hierarchy function (i.e., \( RH \subseteq R \times R \)), which refers to the partially ordered role hierarchy (written \( \geq \)).

The original definition of RBAC does not include the notion of “signed authorization” (i.e., positive or negative authorization). By default, all permissions granted to a role are positive authorization, but negative authorizations are useful when dealing with large sets of protected objects organized according to hierarchies. Signed authorizations have been widely investigated [186], and also introduced in access control systems of commercial products (e.g., the access control model of SQL Server provides the negative authorization by the DENY authorization command SQL [9]). See [41] for a definition of RBAC with signed authorizations.

**Example:** In a healthcare organization, there are different roles including doctor, nurse, manager, and intern. As an example the doctor role is responsible for predefined actions such as performing diagnosis, prescribing medication, ordering laboratory tests, and writing reports. Hence we have

\[
\text{Roles} = \{ \text{doctor, nurse, manager, intern} \}, \quad \text{Action}_{\text{doctor}} = \{ \text{perform, prescribe, order, write} \},
\]

and

\[
\text{Object}_{\text{doctor}} = \{ \text{diagnosis, medication, laboratory test, report} \}.
\]

An example of a signed positive authorization policy is:

\[
R ::= \langle \text{role} : \text{doctor}, \text{action} : \text{prescribe}, \text{object} : \text{medication}, \text{sign} : + > .
\]

RBAC has been extended along different directions. For example, Bertino et al. [42] introduced Temporal-RBAC (TRBAC) to address the challenge of the temporal dependencies among roles. Due to the tremendous increase in location-based services and mobile applications, the GEO-RBAC model was introduced that extends RBAC with geospatial information [67]. In GEO-RBAC, roles are activated based on the geo-location of users. For supporting both spatial and temporal constraints, Kumar and Newman [123] introduced the Spatial-Temporal RBAC model (STRBAC).

2.1.2 Extensible Access Control Markup Language (XACML). XACML [3] is an XML-based language by the OASIS standards organization for the specification of access control policies. In XACML, a policy is organized according to four elements: subject, resource, action, and environment. The subject is the entity requesting access with a particular type of action on a resource (e.g., data, service or system component) within a context (i.e., environment); the policy either decides to allow the request or to deny it. The XACML model can be characterized as an attribute-based access control (ABAC) model since the policy elements are associated with attributes (i.e., properties) which are essential for characterizing subjects and objects and making appropriate decisions based on these attributes. The main concepts of XACML policies are as follows.
• The Policy Set is a set of XACML policies while a Policy comprised Target and a set of Rules.
• The Target Policy specifies the requests which are controlled by a particular policy. The Target specifies the attributes of the subjects, resources, actions, and environments which characterize the access requests.
• Each Rule is mainly composed of two elements; Condition and Effect. The Condition element specifies constraints on the request’s attributes (i.e., subject, resource, action, and environment) while the Effect element specifies whether the request is allowed (Permit) or denied (Deny) based on the condition. A rule contains optionally the Target element which also specifies the requests applicable to the corresponding rule. The result of enforcing a rule on an access request can take one among four possible values: Permit, Deny, Indeterminate (i.e., the decision cannot be made due to an error or some missing value), or Not Applicable (i.e., the request cannot be answered by this service).
• The Rule Combining Algorithm which resolves the case when multiple rules are applicable to a request, but these rules have different effects on the corresponding request. XACML supports several conflict resolution strategies: permit-overrides (i.e., permit the request), deny-overrides (i.e., deny the request), deny-unless-permit (i.e., deny the request unless one of the rules permits it), permit-unless-deny (i.e., permit the request unless one of the rules denies it), first-applicable (i.e., apply the first rule to request in the policy file) and only-one-applicable (i.e., no decision is provided to the request when multiple rules are applicable).
• The Obligation is a function which can be executed before or after the policy is enforced on an access request. An example of such an obligation function is to log information about the corresponding policy and request.

Example: A user can create a financial transaction if his credit balance is not lower than the amount of transaction and banking cost. Also, the transaction is valid during the office hours. This policy is modeled in XACML as follows:

```
<Policy PolicyId = "p1" RuleCombinationAlgId = "Deny – Overrides">
  <Rule RuleId = "r1" Effect = "Permit">
    <Target>
      <subject>Bank</subject>
      <resource>Transaction</resource>
      <action>Create</action>
      <condition>transaction_value + banking_cost < credit_balance</condition>
    </Target>
  </Rule>
  <Rule RuleId = "r2" Effect = "Permit">
    <Target>
      <subject>Bank</subject>
      <resource>Transaction</resource>
      <action>Create</action>
      <condition>current_day ∈ [Mon, Fri] ∧ current_time ∈ [08AM, 6PM] →</condition>
    </Target>
  </Rule>
</Policy>
```

We assume that the values of transaction_value, banking_cost, credit_balance, current_day, and current_time are constrained. The type of policy is Deny – Overrides which returns permit if all rule evaluations return permit.
2.2 Network Policies

Network policies are sets of rules composed of conditions, constraints, and settings governing the operations (e.g., access to resources) within a network or across networks. Such rules allow one to specify which subject is authorized to connect to the network and the circumstances under which the subject can or cannot connect. Here, we discuss two types of network policies: firewall and software-defined networking (SDN). We discuss those two types separately for two reasons: a) SDN policies are more general than firewall policies, and b) firewall policies have been the focus of a large body of research. Hence, in order to discuss the firewall policies with adequate depth, we separate its discussion from SDN.

2.2.1 Firewall Policies. Firewall is a network element that filters traffic between network segments [53]. In particular, it filters out packets based on their characteristics and performs actions on the packets that do not match the firewall rules. Subsequently, the firewall contains a list of firewall rules (a.k.a. firewall policies) which specify the matching conditions for the traffic and the actions to be taken against the packets. Firewall rules follow the event-condition-action (ECA) paradigm, where an event (e.g., an incoming packet) triggers the automatic validation of stated conditions and actions. Firewall rules are specified using network fields including the protocol type, the IP address and port number of the source, and the IP address and port number of the destination. Commonly, a firewall rule follows the following format [17]:

\[
\text{Rule} ::= \langle \text{order} \rangle \langle \text{protocol} \rangle \langle \text{srcip} \rangle [\langle \text{srcport} \rangle] \langle \text{dstip} \rangle [\langle \text{dstport} \rangle] \langle \text{action} \rangle .
\]

where \( \text{order} \in \mathbb{Z}^+ \) (i.e., any non-negative integer), \( \text{protocol} \in \{\text{TCP, UDP, } *\} \), \( \text{srcip} \) and \( \text{dstip} \) express respectively source and destination IP address represented (in IPv4 format) as \( \text{address} := x_1.x_2.x_3.x_4 \), \( x_j \in [0, \ldots, 255] \cup \{*\} \), \( \text{action} \in \{\text{allow, deny}\} \), \( \text{port} \in \mathbb{Z}^+ \cup \{*\} \), and \( * \) is used to denote the keyword any or a domain range.

If the traffic matches the specified filtering rule, the action of the rule is executed; otherwise, the next rule, as defined by the order sequence, is executed, and so on.

Example: Consider the following firewall rules: 
\( R_1 := \langle 1 \rangle \langle \text{TCP}, 140.192.37.* \rangle, \text{any}, *.*.*.*.80, \text{deny} > \) and \( R_2 := \langle 2 \rangle : \text{TCP, *.*.*.*.any, 161.120.33.40}, 80, \text{accept} > \).

First, \( R_1 \) (order = 1) denies all traffic from \( \text{srcip} \) (140.192.37.*) and destined to any address on port 80. Then, \( R_2 \) (order = 2) accepts all traffic destined to \( \text{dstip} \) (161.120.33.40) on port 80. These firewall rules with this ordering have the effect of denying all traffic coming from \( \text{srcip} \) (140.192.37.*) and destined to \( \text{dstip} \) (161.120.33.40) on port 80. However, the same traffic will be accepted if the order is reversed.

2.2.2 SDN Policies. Software-defined networking (SDN) is a networking model designed for existing network infrastructures to separate the control logic module from other underlying modules which control the traffic forwarding (i.e., routers and switches) [162, 199]. Subsequently, the policy enforcement and network re-configuration in an SDN architecture are significantly simplified. We refer the reader to Kreutz et al. [122] for a comprehensive survey and details of SDN.

OpenFlow [163] is a well-known technology which standardizes the communication between the switches and the software-based controller in an SDN architecture. OpenFlow specifies the flow table structure that describes how packets are processed in the switches. Specifically, a flow table is composed of a set of flow rules (i.e., policies) and each rule consists of three parts:

- A matching rule which comprises different fields including switch port, source IP address, MAC address and port number; destination IP address, MAC address and port number; and VLAN tag.
- An action which can be either dropping, forwarding, or modifying the matched packets.
- Counters which record statistical data about the matched packets.
The matching process starts with the arrival of a new packet, then flow tables are searched sequentially for a matching rule. OpenFlow enables implementing the SDN policies in an assembly-like machine language. A number of benchmark tools (such as Cbench[2], OFCBenchmark[108], PktBlaster[10]) have been proposed to evaluate the performance of OpenFlow. Also, in order to enhance flexibility, several high-level networking languages have been suggested such as FlowLog [175], Pyretic [169], Frenetic [81], NetCore [168], FML [97], and HFT [78]. There are several open source implementations of OpenFlow Controller as well as commercial versions [5].

To improve OpenFlow, several tools have been developed. FLOWGUARD [99] was designed to detect and resolve security policy violations in OpenFlow-based networks. This tool employs five resolution strategies including flow removing, tendency breaking, update rejecting, flow rejecting, and packet blocking. VANT-GUARD [200] is an extension to OpenFlow; it addresses security challenges in OpenFlow by increasing data plane intelligence. VANT-GUARD protects SDN applications from saturation attacks. It also enhances detection and responsiveness to threats. Other tools, such as VeriCon [29], can verify the correctness of SDN programs on different topologies and for a sequence of network events.

Example: Assume that the information about two subnetworks is as follows: Subnetwork 1 ($S_1$): 100.0.1.0/24 and Subnetwork 3 ($S_3$): 100.0.2.0/24. Mary is a developer (in $S_1$) allowed to access the resources at $S_3$ and the destination TCP port is Telnet. The policy can be written as

$$\text{Rule} := p : \text{path} \mid (S_1,S_3) \land \text{protocol} = TCP \land \text{port} = 24 \land \text{Mary : user } \in \text{Developer}.$$ 

3 ANALYSIS GOALS

Policy analysis is typically carried out for two different purposes: verifying the satisfaction of a set of quality requirements, and designing and organizing a set of policies. In what follows, we describe in more details these policy analysis goals.

3.1 Assessment of Policy Quality

The notion of policy quality was introduced by Bertino et al. [41] as a set of basic policy requirements. Assuring the quality of a set of policies can be stated as the problem of making sure that the set of policies be consistent, minimal, relevant, complete, and correct. They should also minimize the exceptions that may occur at run-time. In what follows, we describe each of these quality requirements.

3.1.1 Consistency. Consistency (CON) refers to making sure that the policy sets do not include policies that contradict each other. For example, in the case of access control, assuring consistency means making sure that a policy set does not include a policy allowing an access request and a policy denying the same request. Inconsistent policies lead to conflicts at policy enforcement. There are different types of conflicts [30, 65, 165] including modality conflicts (an inconsistency in policy specification of two policies which are applied to a request with different signs), conflict of duty (when a system fails to assert the separation of duty principle), conflict of interest, and conflicts of priorities.

3.1.2 Minimality. Minimality (MIN) refers to assuring that the sets of policies do not include redundant policies. For example, consider a file directory and an access control policy stating that a user can read all the files in the directory. Then assigning the same user a read permission on each single file in the directory would result in a policy set with redundant policies. Redundant policies increase the administrative work required to manage policies. For example, if a user is no longer required to access a given object, all the policies controlling such an access must be properly modified, thus increasing the security risk if such changes are not properly managed.
3.1.3 **Relevance.** Relevance (REL) requires that the sets of policies do not contain policies that do not apply to any action executed by the users. In the case of access control, irrelevant policies may undermine security. For example, an attacker may try to compromise a user in order to exploit the permission of this user. Hence, making sure that users do not have permissions for accesses which they are not expected to execute minimizes the risk of such exploitations.

3.1.4 **Completeness.** Completeness (COM) refers to assuring that all actions, executed in the domain controlled by a policy-based management system, are covered by some policies. For example, in the case of access control policies, assuring that a set of policies is complete means assuring that for each access request issued by a subject, there is a policy either allowing the access or denying it. Incomplete policy sets may lead to an unpredictable outcome, and increase the cost of manual security administration.

3.1.5 **Correctness.** Correctness (COR) requires that the policies be free of faults and compliant with their intended goals and system requirements. Ensuring the correctness of policies includes validating their syntax, and verifying their ability to achieve their goals in all possible context and scenarios.

3.2 **Policy Design and Organization**

In many cases, it is critical to maintain an optimal design and organization of policy sets in case of policy evolution or integration of policies from different organizations. To properly re-organize and evolve policy sets, it is often important to assess the similarity of different policies to determine whether one can consolidate similar policies into a single policy. It is also critical to assess the impact of policy modifications.

3.2.1 **Policy Set Structuring.** The policy set structure (PSS) has a primary effect on the cost complexity of policy management and policy enforcement. In particular, organizing the policies and rules in policy sets [151] helps in optimizing policy enforcement. Moreover, assuring optimal structuring for policy components which covers all potential components affects the construction of policy management systems (e.g., role mining in RBAC systems [167]).

3.2.2 **Similarity Analysis.** The policy similarity analysis (SA) is the characterization of the relationships among a set of requests with their corresponding set of enforced policies [126]. The similarity analysis is an important step for policy integration and organization collaboration since the collaboration can be planned based on the similarity analysis results of the collaborators’ policies [160]. Several approaches have been proposed for policy similarity analysis including Mazzoleni et al. [159], Lin et al. [127], Mazzoleni et al. [160], and Lin et al. [126]. In particular, Mazzoleni et al. [160] [126] defined a model for policy similarity based on the reaction of policies to a set of requests while Lin et al. [127] and Lin et al. [126] proposed other models based on the policy structure and components.

3.2.3 **Change Impact Analysis.** Change impact analysis (CIA) refers to assessing and evaluating the extent of the change on the specifications of a set of policies by identifying the potential consequences and estimating the risks associated with the change. Thus, it provides an accurate understanding of the implications of a proposed change. For this sake, CIA evaluates the changes among two versions of a policy by providing a set of counterexamples that show semantic differences between the two policies.
4 TAXONOMY OF ANALYSIS METHODS

Various methods have been proposed for policy analysis which we review in what follows. Fig. 1 shows a taxonomy of these methods.

![Fig. 1. Taxonomy of the Methods Used in Policy Analysis](image)

4.1 Formal Methods (FM)

A large body of research exists in the area of formal methods for policy analysis based on a variety of formalisms, including event calculus and argumentation-based reasoning. In what follows, we briefly summarize the formal methods most commonly used for policy analysis.

4.1.1 Matrix Based Methods. A matrix is a collection of elements organized in rows and columns. A matrix is denoted by its dimensions (i.e., a matrix $A$ which composed of $m$ rows and $n$ columns denoted as $m \times n$ matrix), and an element in $A$ is denoted as $a_{i,j}$.

In the context of policy analysis, matrices are used as a representation mechanism for the policy components. In particular, Boolean matrices (a.k.a. binary matrix or logical) are mainly used to relate two finite sets according to 0 or 1 values. The existence of a relationship between two corresponding elements in the set is denoted by 1 and 0 otherwise. For example, in RBAC [212], the user-role assignment is modeled by matrix $A$ whose dimensions $m \times k$ and the role-permission association is modeled by matrix $B$ whose dimensions $k \times n$. The Boolean multiplication of $A$ and $B$ (see Definition 2) returns the full set of user permissions.

**Definition 2** (Boolean Matrix Multiplication [212]). A Boolean matrix multiplication between Boolean matrices $A \in \{0, 1\}^{m \times k}$ and $B \in \{0, 1\}^{k \times n}$ is $A \otimes B = C$, where $C$ is in space $\{0, 1\}^{m \times n}$ and

$$c_{ij} = \bigvee_{l=1}^{k} (a_{il} \wedge b_{lj}).$$

Moreover, the complete set of role hierarchies in RBAC can be represented using the transitive closure of roles [100]. For example, let $R$ denote a set of roles and $H$ denote the role hierarchy defined over $R$. Thus, the pair $(r_i, r_j) \in H$ implies that the role $r_j$ is a child of the role $r_i$.

The transitive closure of $H$ on $R$ (see Definition 3) is the relation $H^+$ such that $(r_i, r_j) \in H^+$ means there are one or more hierarchy levels between the roles $r_i$ and $r_j$. 

ACM Computing Surveys, Vol. 1, No. 1, Article . Publication date: July 2018.
Abductive reasoning is a form of logical inference which starts with an observation of the system and the premises given, whereas the truth of the deductive reasoning’s conclusion is definite. Induction can be defined inductively by $H^i$ is the $i^{th}$ power of $H$, defined inductively by $H^1 = H$, and for $i > 0$, $H^{i+1} = H \otimes H^i$.

Several algorithms for policy analysis have been proposed that use different forms of matrices (e.g., conflict matrix [218], and access control matrix [95]) while other analysis frameworks use matrices for policy representation in conjunction with other analysis techniques [212].

### 4.1.2 Reasoning Methods

There are various forms of logical reasoning including deductive, inductive, and abductive reasoning. In the context of policy analysis, deduction reasoning is the process of using one or more premises about the system behavior together with the history of events occurring in the system to reach to a certain logical conclusion about the state of the system properties. Induction can be defined as the derivation of a general conclusion about the system behavior based on premises on the observed history of events and the state of system properties. Abductive reasoning is a form of logical inference which starts with an observation of the system behavior to provide the best explanation using the sequence of events to reach a state of system properties. The conclusion of both inductive and abductive reasoning might be uncertain based on the premises given, whereas the truth of the deductive reasoning’s conclusion is definite.

In the context of policy analysis, some approaches [210] use reasoning associated with SMT solver (see Section 4.2.1). While other approaches [31, 34, 54] used reasoning utilizing the policy representation by Event Calculus (see Section 4.1.3). Another reasoning-based approach to analyze policies was proposed by Halpern and Weissman [91] based on a first-order logic approach which helps in reasoning about policies and preserving traceability.

### 4.1.3 Event Calculus

The event calculus (EC) is a logic-based formal model for expressing and reasoning about the occurrence of events. Being able to express the dynamic aspects of events, EC has been used for representing the specification of policies and analyzing them (e.g., [31, 34, 54]).

EC has many variants, but the one widely used for policy analysis is the variant proposed by Russo et al. [189]. The specifications of this variant comprise a set of time-varying properties referred to as fluents, a set of positive integers referred to as timepoints, and a set of actions referred to as events. This variant also includes predicates describing the order of action, such as Happens, Initiates, Terminates and HoldsAt, and some auxiliary predicates such as Clipped and Declipped (see Table 1).

### 4.1.4 Argumentation Methods

Argumentation is a method for reaching conclusions through logical reasoning on conflicting information [214]. In the artificial intelligence area, Dung "Methods and Tools for Policy Analysis: 9"
proposed an abstraction argumentation framework, which was extended later to logic-based [45] and value-based [38] frameworks. In Dung [74]’s framework, argumentation is formally defined as a pair <T, A>, where T is a set of abstract arguments that represent data or a proposition and A is a binary attaching relation on the subsets of T where the relation shows the conflicts between the arguments. In particular, for any two arguments t1, t2 ∈ T, we say that t1 attacks t2 when the relation (t1, t2) ∈ A. Argumentation reasoning is given through the notion of an admissible argument as defined in Definition 4. In the framework <T, A>, T is represented in a logic language L.

Definition 4 (Admissibility of Arguments [33]). Given an argumentation framework <T,A>, an argument t1 ∈ T is admissible ⇔ (t1,t1) ∉ A (i.e., t1 does not attack itself) ∧ (∀tj | tj ∈ T ∧ (tj,t1) ∈ A ∧ i ≠ j → (tj, t1) ∈ A) (i.e., for any other argument which attacks t1, t1 counter-attacks it).

In the context of policy analysis, argumentation policies can be modeled as arguments. The attack relation defined over the set of arguments serves as a criterion to deal with possible conflicts among policies and to select a set of compatible policies that achieve a set of requirements. Argumentation has been used for analyzing firewall policies [25, 32, 33] and access control policies [46, 47, 180].

4.2 Model Checking (MC) Methods

The model checking technique has been introduced to exhaustively and automatically verify the correctness of properties and specifications of concurrent finite-state systems. To achieve this goal, a mathematical representation is used to characterize the system’s model and specifications. A typical model checking system can be described by two components: a) the preprocessor which extracts a set of states S from a program and models them using a state transition graph M, and b) the engine which takes the state transition graph M and a temporal formula f and determines whether the formula is true or false (i.e., counterexample).

Model checking has some advantages compared to other verification techniques, such as automated theorem proving and proof checking [60]. However, the size of a state graph can be exponential which is the main drawback of model checking. Therefore several techniques, such as symbolic model checking, bounded model checking (e.g., SAT), compositional reasoning, abstraction, and partial order reduction, have been proposed to address such an issue [77]. In those methods, the model checking problem reduces to a graph search problem and binary decision diagrams (BDD) or other temporal logic methods can be used for the traversal of the state space. Model checking has been widely used for policy analysis to detect for example conflicts between access control policies or security errors in access control policies.

In what follows, we describe model checking methods that have been applied for policy analysis.

4.2.1 SAT and SMT Solvers. The Boolean satisfiability (SAT) problem is the problem of evaluating the satisfiability of a given Boolean formula. The SAT problem is formulated in Definition 5.

Definition 5 (The SAT problem [63]). Given a Boolean formula φ composed of

- n Boolean variables: x1, x2, ..., xn;
- m Boolean connectives: any Boolean function such as ∧ (AND), ∨ (OR), ¬ (NOT), → (implication), ⇔ (if and only if); and
- Parentheses: One pair of parentheses per a Boolean connective.

The Boolean satisfiability problem posed on φ aims at finding some values that can be assigned to the variables composing φ to make its value true. If such values exist, then φ is satisfiable. Otherwise, φ is unsatisfiable.

The SAT problem is known to be NP-hard [62]. However, there has been tremendous progress (see [225]) in developing efficient mechanisms for the SAT problem. These algorithms are known
as SAT solvers. Some SAT solvers are complete (e.g., Davis-Putnam-Logemann-Loveland (DPLL) algorithm [69, 70]), while others are stochastic (e.g., random walk based algorithms such as WALKSAT [195]). For any given instance of a SAT problem, a complete solver is able to find the variable values that solve the problem in case it is satisfiable or proves that it is unsatisfiable. Meanwhile, a stochastic SAT solver is not able to prove the unsatisfiability for some instances.

Advances in SAT have made SAT solvers an attractive underlying reasoner for policy analysis w.r.t. propositional logic formulas used to model various access control policies. However, native SAT solvers do not support efficient reasoning over non-Boolean variables, such as temporal constraints, which play a significant role in analyzing the correctness of policies. These non-Boolean variables are often left uninterpreted, hence restricting the analysis capabilities [210]. An extension of SAT is Satisfiability Modulo Theories (SMT) [35] which provides the Boolean SAT representation with additional operators such as linear arithmetic and equality. Consequently, various SMT solvers were proposed (e.g., Z3 [72] and openSMT [7]) that support a more efficient fine-grained analysis than native SAT-based policy analysis tools.

Since policies are essentially a set of constraints over a set of resources, the policy analysis can be intuitively mapped into logic formulas. As a consequence, satisfiability checking can be employed to validate requirements, such as security requirements, over a given set of policies. Several algorithms and frameworks for policy analysis have been developed which use SAT [101, 111, 126] or SMT [21, 22, 109, 210] solvers.

4.2.2 Alloy. Alloy is a declarative logical language for expressing structural constraints and behavior of a system [1, 106]. The language is simple and based on the standard first-order logic. Its syntax uses the following notations:

- **Signatures**: A signature is a data type in Alloy. It can be considered equivalent to a class in object-oriented languages since a signature can be instantiated.
- **Relations**: A relation is a tuple which maps instances of signatures to each other.
- **Functions**: A function maps one instance of a signature to an instance of another signature.

When modeling the system constraints using the Alloy language, the Alloy Analyzer [107] is executed to find structures that satisfy the constraints. Thus, the Alloy Analyzer can be used both to explore the model by generating sample structures, and to check the properties of the model by generating counterexamples. The core of the analyzer engine is reduced to a Boolean SAT solver.

Since policy constraints can be modeled using the Alloy language, several frameworks for policy analysis use Alloy as underlying building block [94, 96, 115, 149, 182, 193].

4.2.3 Binary Decision Diagrams. A binary decision diagram (BDD) is a data structure for representing a Boolean function that takes Boolean arguments as inputs and produces a boolean output [16]. A BDD is represented as a rooted, directed, acyclic graph \( G(N, E) \) composed of nodes \( N \) and edges \( E \). The set of nodes \( N \) are categorized into two types: decision nodes corresponding to the Boolean arguments, and terminal nodes corresponding to the Boolean decision (0-terminal and 1-terminal). The edges represent the Boolean values branching from nodes. Consider the Boolean function \( f \) defined in Eq. (1) [16]. Fig. 2 shows two examples of BDD representing \( f \). Given \( n \) arguments in a Boolean function, there are \( 2^n \) different possible inputs; thus an instance of BDD contains at most \( 2^n \) possible paths in correspondence to the truth table for \( f \) (e.g., Fig. 2a). For efficiency, the BDD is designed with the minimum number of paths by omitting the redundant test of boolean arguments (e.g., Fig. 2b).

\[
f = A \lor \neg B \land C
\] (1)
The BDD has been extended to the multi-terminal BDD (MTBDD) [82] where a terminal node is allowed to have a finite set of values instead of binary values (i.e., 0 or 1). The MTBDD structure is also known as algebraic decision diagram (ADD) [28]. The implementation of both BDD and MTBDD is provided with the CUDD package [203]. BDD and MTBDD are used as the underlying representation of policies which facilitates the analysis process (e.g., [51, 80, 126, 181, 223]).

4.3 Data Mining (DM) Methods

Data mining is a technique for exploring massive datasets and finding interesting trends or patterns to guide decision for further analysis. Data mining algorithms can be classified as supervised and unsupervised. Supervised algorithms require a learning stage on a historical dataset for building a model which summarizes the identified patterns. Unsupervised algorithms detect patterns in datasets without the need of the learning phase. Examples of supervised algorithms include classification while unsupervised algorithms include association rule mining and clustering. Some of the data mining methods are described below.

4.3.1 Association Rule Mining (ARM). ARM [13] is a technique for detecting statistical relations (referred to as association rules defined in Definition 6 (adapted from Tan et al. [208])) between items in a database. The mechanism aims at discovering all rules which satisfy a certain threshold on two metrics: support and confidence. The support metric indicates how often a rule applies to a given dataset and hence it is used to eliminate uninteresting rules and capture the frequency of the rule. The confidence metric represents the quality of the rule. Hence, it measures both the reliability of the inference made by the rule and the strength of the relation between the items sets. The inference indicated by the association rule suggests a strong co-occurrence relationship between items in the antecedent and consequent of the rule. There are various algorithms designed for ARM including the Apriori Algorithm [14], FP-growth [92], and Eclat [224].

Definition 6 (Association Rule [208]). Given a set of items $I = \{i_1, i_2, ..., i_n\}$ and a database $D$ composed of a set of transactions $T$ where each transaction $t \in T$ is composed of a set of items $t \subseteq I$, an association rule is an implication expression of the form $X \rightarrow Y$ where $X$ and $Y$ are disjoint subsets of $I$ (i.e., $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$). The association rule $X \rightarrow Y$ has two metrics:

- The support value $s$ which denotes the fraction of transactions containing both $X$ and $Y$ in the whole database $D$ (i.e., $s(X \rightarrow Y) = p(X \cup Y)$); and
- The confidence value $c$ which denotes the fraction of transactions that contain both $X$ and $Y$ among the transactions that contain $X$ (i.e., $c(X \rightarrow Y) = p(X \cup Y) / p(X)$).

Then, in an association rule ($X \rightarrow Y$), (where $X$ and $Y$ are attribute values), a confidence of the rule is conditional probability of $X$ given $Y$, $Pr(X|Y)$ and support is the prior probability of $X$ and $Y$.
The ARM-based policy analysis techniques explore a large dataset to identify interesting rules for further analysis. For example, Ma et al. \cite{148} used ARM in the context of policy analysis to classify constraints in RBAC. Cardinality constraints on the number of users, roles, and permissions are represented in the terms of association rules. Bauer et al. \cite{37} employed ARM to anticipate misconfiguration in access control policies. By applying ARM to the history of accesses, association rules are identified (the rules are mined by the Apriori algorithm \cite{14}). Based on the rules, the data is analyzed and potential mis-configurations of access control policies are detected. Golnabi et al. \cite{84} proposed an ARM-based technique to detect firewall anomalies defined in terms of the superset, subset, and correlation relationship among criteria. Firewall log files are mined to extract attributes (such as protocol, direction, source IP, destination IP, source port, and destination port).

### 4.3.2 Clustering

Clustering is a mechanism for distributing a set of objects into groups (referred to as clusters) where the intra-similarity among the objects of a group is higher than the inter-similarity with the other groups. The clustering technique defines the groups of objects based on some measure of inherent similarity or distance (e.g., Euclidean and Manhattan distance). Each cluster has a centroid object which identifies the corresponding cluster. After constructing the clusters, a new object is assigned to the cluster whose centroid is the closest one.

Well-known clustering algorithms include k-Means Lloyd \cite{144}, Mean-shift \cite{61}, and spectral clustering \cite{176}. In addition to the basic clustering techniques, hierarchical clustering builds a hierarchy of clusters. Hierarchical clustering follows two strategies: bottom-up approach (referred as agglomerative \cite{202}), and top-down approach (divisive \cite{116}). Several open-source libraries (e.g., scikit-learn \cite{179}, Weka \cite{90}) provide implementations in different languages of these algorithms.

Clustering-based approaches partition access control policies into clusters. For example, for XACML policies in distributed applications, after extracting the rules in each policy, similar rules (identified based on some similarity scores) are assigned to a cluster \cite{15, 40}. Each cluster is separately analyzed for quality (such as redundancy, and inconsistency). A k-means clustering method \cite{217} has been proposed by Marouf et al. \cite{150, 151} to optimize the evaluation of XACML policies in the case of large numbers of requests. This approach clusters similar subjects (i.e., subjects in one cluster share a large number of policies relevant to all of them) to find relevant policies to incoming requests with the best rule orders. The best ordering is dynamically identified based on incoming requests, history of requests, and executions. Moreover, Benkaouz et al. \cite{40} proposed a K-Nearest Neighbors-based technique for RBAC policies to enhance flexibility and reduce policy dimensionality in large-scale applications.

### 4.3.3 Data Classification

Classification is a learning technique for assigning an item to a set of predefined classes or groups (a.k.a labels). Classification algorithms include two stages: training and prediction. In the training stage, the algorithm discovers the patterns of the attributes characterizing the items for each predefined class and summarizes these patterns in a training model. In the prediction stage, the algorithm utilizes the training model to predict the class to a new item. Data classification techniques can handle complicated Boolean expressions, missing and continuous attributes. Unlike the formal logic approaches for inconsistency detection, classification techniques do not suffer from exponential growth and computational complexity \cite{198}. In the data classification techniques, each rule \((R_i)\) is represented by a set of attributes \((A_1, A_2, \ldots, A_n)\) based on which decision are made (e.g., role, subject) and a class of the rule (\(C\)) (e.g., allowed, denied), \((R_i : A_1 \land A_2 \land \ldots \land A_n \rightarrow C)\). Data classification techniques, such as C4.5 \cite{185}, ID3 \cite{184}, Limited Search Induction Algorithm (LSIA) \cite{50}, and ASSISTANT\'86 \cite{52} (with some modifications), are
mainly used for detecting incompletenesses [196, 197] and inconsistencies [27, 196, 198], which are formally defined in Definition 7 and 8.

**Definition 7 (Incompleteness[197])**. Given a role \( R_i \), a set of attributes \( A_k \in A \), all values assigned to \( A_k (Y(A_k)) \), and the \( A_k \) value \( \nu(R_i,A_k) \), \( \mathcal{R} \) is incomplete with respect to \( A_k \) if:

\[
\bigcup_{i=1,\ldots,m} \nu(R_i,A_k) \subset Y(A_k).
\]

**Definition 8 (Mutually Inconsistent[196])**. Given roles \( R_i, R_j \in \mathcal{R} \), and the \( A_k \) value for the roles \( R_i (\nu(R_i,A_k)) \) and \( R_j (\nu(R_j,A_k)) \) are mutually inconsistent if and only if:

1. \( \forall A_k \in A \), \( \nu(R_i,A_k) \cap \nu(R_j,A_k) \neq 0 \), and
2. \( \nu(R_i,\text{Permission}) \neq \nu(R_j,\text{Permission}) \).

### 4.3.4 Role Mining

Role mining (RM) is an application of data mining specifically designed for transforming a non-RBAC system into an RBAC system by finding user roles in the observed policies [212]. The RM problem is formally defined in Definitions 10 and 9. RM is a tool for examining whether roles are appropriately assigned to existing functions and business processes.

**Definition 9 (Role Mining Problem [212])**. Given a set of users \( U \), a set of permissions \( P \), and a user-permission assignment \( UPA \), find the minimal set of roles \( R \) where user-to-role assignment \( UA \), a role-to-permission assignment \( PA \), and user-to-permission assignment \( UPA \) are \( \delta \)-consistent (i.e., \( \delta = 0 \)).

**Definition 10 (\( \delta \)-Consistency [212])**. A given user-to-role assignment \( UA \), role-to-permission assignment \( PA \) and user-to-permission assignment \( UPA \) are \( \delta \)-consistent if and only if:

\[
\| M(UA) \otimes M(PA) - M(UPA) \|_1 \leq \delta
\]

where \( M(UA), M(PA), \) and \( M(UPA) \) denote the matrix representation of \( UA, PA \) and \( UPA \), respectively. \( \delta \)-consistency bounds the degree of difference between the \( UA, PA \) and \( UP \).

RM is an NP-hard problem [212] which has been addressed by various algorithms. In particular, Schlegelmilch and Steffens [194] designed the ORCA role mining tool which is based on a hierarchical clustering on permissions. Another algorithm is RoleMiner (proposed by Vaidya et al. [213]) which is based on subset enumeration. Moreover, Molloy et al. [167] designed a role mining algorithm which considers the semantic meanings of the roles. In addition to being a method for policy analytics, RM can be used to migrate from or to RBAC model [207, 222].

### 4.4 Graph-Based Modeling (GM) Methods

In the context of policy analysis, a set of policies can be represented as a graph (i.e., nodes connected with edges) or a tree (a special form of a graph) which helps to efficiently query, analyze, and verify the corresponding policies by utilizing the graph operations. In particular, graph traversal, graph union, graph intersection, and graph difference are used to analyze properties of policies. The preparatory step of the analysis process is to transform a given policy set into a single graph (referred as Policy Graph) or tree (referred as Policy Tree). Transforming the set of policies into a graph includes modeling the policy components (e.g., roles in RBAC policies) into nodes and constructing a path (i.e., a sequence of edges connecting certain nodes) for every policy. The transformation includes multiple graph operations such as traversing and updating the graph. The graph-based policy model has been used for ensuring that the specifications of the policies comply with their goals and the system requirements [17–20, 23, 24, 41, 71, 205, 221].
4.5 Mutation Testing (MT) Methods

Mutation testing [73] is a technique for software testing which involves generating several versions of the software by making one small change at a time. Mutation testing uses mutation operators which differ based on the corresponding programming language. Each generated version is called a mutant, and the goal is to create test cases that are able to detect all faults in the mutants. A major limitation of mutation testing is its computational cost. Hence, several algorithms have been proposed for reducing the number of mutants without affecting the effectiveness of tests. This problem is referred to as “Mutant Reduction Problem” [113] as described in Definition 11. Several techniques were introduced to address this problem including mutant sampling [11, 49], mutant clustering [102], selective mutation [158], and higher order mutation [112].

Definition 11 (The Mutant Reduction Problem [113]). For a given set of mutants $M$, and a set of tests $T$, $MS_T(M)$ denotes the mutation score of the test set $T$ applied to mutants $M$. Find a subset of mutants $M'$ from $M$, where $MS_T(M) \approx MS_T(M')$.

Mutation testing has been adopted for policy testing and verification [154–157, 172, 173, 183] in three consequent steps: 1) inject faults into the original policies to generate a set of mutant policies; 2) create various scenarios to enforce both original and mutant policies; and 3) measure the percentage of mutant policies whose enforcement output is different from their original policies. The percentage calculated in the last step is referred to as detection percentage of mutant policies which indicates the correctness level of the policy set. Consequently, mutation testing helps in discovering certain scenarios where the corresponding policies are specified incorrectly. To generate mutant policies for mutation testing, it is crucial to design an efficient mutation mechanism (known as mutation operators). Several researchers [76, 125, 170, 171] defined generic operators which are independent of the policy model.

5 SYSTEMS AND FRAMEWORKS FOR POLICY ANALYSIS

In this section, we survey several well-known systems and frameworks which address the challenge of policy analysis in various domains.

5.1 Environment for XACML-based Policy Analysis and Management (EXAM)

EXAM is a comprehensive framework for policy analysis and it is composed of five components: policy annotator, policy filter, policy similarity analyzer, and policy integration framework. The policy annotator pre-processes policies by adding annotations to represent the semantics of the policies [126]. The policy filter evaluates the similarity of each pair of policies using a policy similarity measure [127]. The similarity measure for a pair of policies produces a similarity score that is used for reducing the number of analyzed policies by eliminating the low-score policy pairs. The policy similarity analyzer supports three types of analysis queries: a) policy metadata queries which are related to the metadata associated with the policies; b) policy content queries which are related to the content of policies; and c) policy effect queries which are related to the interactions among the policies and their outcomes. The analyzer component is based on two methods: MTBDD and SAT solver. The policy integration represents policies in the MTBDD model to support algebraic operations defined by the fine-grained integration algebra (FIA) [187].

5.2 Margrave

Margrave is a software suite for analyzing XACML policies [80]. It uses the MTBDD structure as the underlying representation of policies. Margrave has two components: verification and change impact analysis. The verification component is an engine for querying policies and evaluating a particular property. The change impact analysis component uses a decision diagram, referred to as
a *change-analysis decision diagram*, to analyze the changes among a pair of policies and outlining the semantic differences between them. The analysis results are represented by MTBDD which enables exploring them by different verifications queries.

### 5.3 MlsconfiguRAtion ManaGeR (MIRAGE)

MIRAGE is a management framework for the analysis of network policies deployed in network devices (e.g., firewalls, intrusion detection systems, and VPN routers) [83]. The framework detects anomalies and verifies consistency, relevancy, and correctness of network security policies by using bottom-up and top-down approaches. The bottom-up approach analyzes the configuration errors in deployed components and the top-down approach refines global security policies into configuration of security components. These approaches are mainly based on representing policies using a formal model and utilizing boolean functions and reasoning methods. MIRAGE is composed of four services: intra-component analysis, inter-component analysis, aggregation mechanism, and refinement mechanism. The intra-component analysis detects the inconsistencies between the policies of a single security appliance, while the inter-component analysis detects the inconsistencies between the policies of different security appliances. The aggregation mechanism merges the policies of all security appliances into one global minimal consistent set of policies. The refinement mechanism deploys the aggregated global set of policies to newly added security appliances.

### 5.4 Policy Conflict Analysis for Quality of Service Management

Charalambides et al. [54] proposed a policy management technique to handle conflicts in the network management domain with a focus on Quality of Service (QoS) management. It classifies conflicts into application-specific (e.g., routing conflict) and domain-independent (e.g., redundancy and mutual exclusion) conflicts. To detect conflicts, it uses abduction and explanation of conflicts occurrence. The detection approach concentrates on policies for network dimensioning (ND) (i.e., which is a component of the TEQUILA framework). It defines conflict rules for the Ponder [68] language; it then searches the policy repository to check if there is a policy violating conflict rules.

### 5.5 A Toolkit for Firewall Modeling and Analysis (FIREMAN)

FIREMAN [223] is a framework for firewall modeling and analysis. FIREMAN provides static analysis methods to detect mis-configurations, inconsistencies, redundancies, and irrelevancies of firewall rules by modeling them in binary decision diagrams (BDDs). FIREMAN can be used for both individual and distributed firewall systems. Thus, it supports analysis services at various levels: intra-firewall, inter-firewall, and cross-path. The analysis framework adopted by FIREMAN is a bottom-up strategy by employing a local analysis for each individual firewall and then performing a global analysis considering the interactions between the distributed set of firewalls.

### 5.6 FLCheck

FLCheck is a framework to specify, enforce, and verify access control policies [51]. FLCheck utilizes BDD for performing the analysis of access control policies. In particular, it analyzes the RBAC properties (e.g., consistency, safety, and domain dependent conditions) and indicates how domain-dependent assumptions can be merged into the check without a need to modify policies.

### 5.7 Event Calculus Framework for Policy Specification and Analysis

Bandara et al. [34] proposed an approach to formalizing policy specifications and system behavior rules using event calculus. Through the proposed formal representation and using abductive reasoning techniques, it performs policy analysis to generate a refined set of policies while ensuring correctness, consistency, minimality, and completeness of the generated policy set. The approach
aims to achieve consistency by detecting different actual and potential types of conflicts: modality conflict, conflict of duty, conflict of interest, and conflicts of priorities.

5.8 Analysis tools for SELinux security policy

Different policy analysis tools, such as SLAT [89] and PAL [192], have been developed to analyze the security policies of SELinux [145]. Security-Enhanced Linux Analysis Tools (SLAT) [89] utilizes formal methods and model checking techniques to verify that SELinux policies achieve their intended security goals. It translates access control policies written in SELinux’s policy language into logic programs then uses query evaluations for analysis. Policy Analysis using Logic-Programming (PAL) [192] uses an information flow model of SLAT for policy analysis. Functionalities of SLAT and PAL are similar while they are different with respect to the query languages. To write queries SLAT uses a special-purpose language determining information flow paths between security contexts. PAL, unlike SLAT, is implemented in XSB (i.e., a logic-programming and deductive database system based on tabled resolutions) [8]. To improve the understandability of SELinux policy analysis results Xu et al. [220] developed a visualization tool based on both adjacency metrics and semantic substrates (i.e., a visualization technique that uses the user-defined semantic substrate to generate graph layouts) [201]. The framework enables administrators to detect possible policy violations by running visualization-based queries on the policies.

5.9 Graph-based Policy Analysis (GPA)

GPA [221] is a framework for analyzing information flow of network, representing policy queries, and identifying integrity violation of SELinux policies. GPA displays policy layouts by using two visual techniques: semantic substrates and adjacency matrices. It allows system administrators to apply a specific query on policies. Then, a policy violation graph (generated by the framework) displays violations recognized by an integrity model. The integrity model is developed based on the Trusted Computing Base (TCB), Transaction Procedure in the Clark-Wilson security model, and Biba concepts. GPA introduces filtering and ignoring techniques to remove violations from policy graphs.

5.10 Tierless Programming and Reasoning for Software-Defined Networks

Nelson et al. [175] proposed an SDN-based declarative programming language, called Flowlog. Flowlog enables the writing of SDN-policies as well as their analysis. In particular, it allows converting Flowlog ruleset to Alloy (see Section 4.2.2) specifications. Upon compiling Flowlog rules into Alloy, their correctness is verified using the Alloy analyzer.

5.11 Access Control Policy Evaluator and Generator (AcPeg)

AcPeg [227] is a Java-based framework which enables analyzing policies using a model-checking algorithm [226]. The algorithm uses the symbolic model-checking approach [164] to support various goals, including checking whether the policies provide enough permissions to the users and revealing weaknesses of the policies. For analyzing access control policies with this framework, they have to be specified in a formal language, called RW (Read-Write) [87]. However, the framework also supports the translation of the description of a policy from the RW language into the XACML format.

5.12 MOHAWK

Jayaraman et al. [110] introduced a framework, called MOHAWK, which enables error detection in ARBAC [190] access control policies. The policy analysis algorithm supported by MOHAWK combines both an abstraction-refinement and a bounded model checking techniques. The algorithm
first applies abstraction on a set of input policies (with successive refinements, if necessary) to
efficiently evaluate an abstract form of a policy. The level of abstraction can be configured based on
the scale of the access control policies (e.g., for a small number of policies the user can configure the
tool to abstract less). After the abstraction step, the algorithm applies a model checking for verifying
the abstract version of policy against a set of security properties. For this purpose, MOHAWK
translates the abstract form of the policy to the specifications of NuSMV [6]. NuSMV is a model
checking library that utilizes both types of model checking techniques; BDD and SAT.

5.13 Firewall Policy Advisor (FPA)

FPA [17–20] is a Java-based tool for analyzing firewall policies using a tree-based method. FPA
supports two approaches for analyzing policies: intra-firewall and inter-firewall. The intra-firewall
analysis approach evaluates policies within a single firewall while the inter-firewall analysis
assesses the policies between inter-connected firewalls. The analysis approaches aims at detecting
inconsistencies, redundancies, and irrelevancies among the policy set. In addition, FPA includes an
editor which supports managing firewall policy rules. It also displays the appropriate order of the
added or modified rules and the effect of the removed rules on the policy set.

5.14 Structured Firewall Design

A firewall is often designed as a sequence of rules leading to three main problems, namely con-
sistency, completeness, and compactness [85]. To address these problems, Gouda and Liu [85]
proposed a method called structured firewall design based on a firewall design diagram (FDD)
instead of a sequence of rules which often conflict with each other. A firewall design diagram
(FDD) is an acyclic and directed graph. Two algorithms, FDD reduction and FDD marking, are used
to combine rules together. A firewall compaction algorithm to remove the redundant rules has
also been proposed [129, 141]. The FDD is used to represent the semantics of a firewall. The FDD
reduction algorithm is used to reduce the number of decision paths (i.e., generated rules) in an
FDD. In the proposed method, the user only deals with the firewall decision diagram, which is a
formal specification of the firewall. This requires the user to know the formal specification of the
firewall, making the method less generic. Moreover, the FDD reduction is based on graph traversal
method that may be time consuming if the FDD specified by the user is very large.

Due to a large number of firewall rules in a firewall, analyzing and understanding the firewall
rules are difficult tasks. An effective way to analyze and understand deployed firewall rules is by
issuing queries. An SQL-like query language called Structured Firewall Query Language (SFQL) has
thus been proposed [138, 140]. Furthermore, an algorithm for efficiently processing firewall queries
has also been proposed by Liu and Gouda [138]. The FDD is used as the core data structure by the
query processing algorithm and engine.

The correctness of firewall rules and configuration are crucial for an effective security perimeter
deployment. However, ensuring the correctness of the firewall policies is a challenging task, since
a firewall may have a large number (e.g., hundreds to few thousands) of rules and firewall rules are
often added at different times by different firewall administrators for various reasons. To address
this challenge, Hwang et al. [104] proposed a systematic structural testing approach. The proposed
method is based on the firewall policy coverage concept, used to test firewall policy’s structural
entities (i.e., rules, predicates, and clauses). In other words, the method allows one to check if each
entity is specified correctly. Three structural coverage measurements, which monitor whether rules,
predicates, or clauses are covered when evaluating packets against the policy under test, have been
declared [104]: (1) rule coverage measurement is the percentage of the number of covered rules (i.e.,
predicates being evaluated to true) in a policy; (2) predicate coverage measurement is the percentage
of the number of covered predicates (i.e., predicates being evaluated to true or false); (3) clause
coverage measurement is the percentage of the number of covered true or false values of clauses. Instead of exhaustively testing all possibilities, these measurements enable the testing to cover only specific entities.

A configuration of firewall very often leads to errors (i.e., misconfiguration). Such misconfiguration, considered as a policy fault, either creates security holes that allow malicious traffic into private networks or blocks legitimate traffic and disrupts normal services. In other words, a faulty firewall policy mis-classifies some packets thus leading to unexpected decisions. In [58, 59], a fault model is proposed for firewall policies including five types of faults: wrong order, missing rules, wrong decisions, wrong predicates, and wrong extra rules. For each type of fault, an automatic correction technique is proposed. An approach that employs these techniques is proposed to automatically correct all or part of the misclassified packets of a faulty firewall policy.

5.15 Gorgias-B

Gorgias-B [4, 204] is a tool that combines argumentation reasoning with preference-based rules and abductive logic programming for making informed decisions. It can help the users not only to define their decision policy but also execute scenarios to test it. Gorgias-B has been used for conflicts resolution in various applications such as data access control [4], autonomous systems (e.g., Drones), and firewall configuration management [33].

6 COMPARISON

In this section, we compare 132 research publications describing either a system, framework, or a research approach for policy analysis. Our comparison considers three dimensions (see Table 3): policy domain, analysis goals, and analysis methods. The abbreviations for both policy analysis goals and methods used in the comparison table (i.e., Table 3) are listed in Table 2.

![Fig. 3. Statistics about the 132 Research Publications Describing the Systems and Frameworks Compared in Table 3](image)

1If the field related to the analysis goals (i.e., policy quality or policy design) is marked as "N/A", this means that the policy analysis approach proposed by the corresponding research work does not address that goal.
Fig. 3 shows some statistics of the surveyed papers. Among the surveyed papers, access control is the policy domain which is the focus of the majority of the papers. In particular, RBAC and XACML are the models which are widely considered. Papers related to the network policy domain consider mainly the firewall rules model. Thus, research is needed for methods and tools to analyze other network policy models, such as the rule models of SDN. With respect to the analysis goals, most systems and frameworks focus on consistency and correctness. Therefore, research is needed on methods for analyzing completeness and relevancy which are critical in dynamic contexts. Also, analysis techniques addressing policy design and organization (e.g., similarity analysis and change impact analysis) need to be further explored. With respect to the analysis methods, the most used techniques are model checking and formal methods since these methods can be used for almost all the analysis goals. Data mining is the least used technique because is computing intensive, time-consuming, and error-prone.

Fig. 4. Association among the Analysis Goals Addressed by the Policy Analysis Systems and Frameworks Described in the Surveyed Papers

Fig. 5. The Distribution of the Analysis Methods Used for Each Analysis Goal Addressed by the Policy Analysis Systems and Frameworks Described in the Surveyed Papers

None of the policy analysis systems and frameworks is able to support all the analysis goals. In particular, none of the systems is able to support the analysis of policies with respect to all the policy quality requirements. In the access control policy domain, Bertino et al. [41] consider all the
quality requirements except correctness, while in the network policy domain, the Mirage [83] and FPA [17–20] frameworks consider all the quality requirements except completeness. Fig. 4 shows the association ratios of the analysis goals in the surveyed publications. All analysis goals are tightly associated with consistency, especially it is the focus of the majority of the papers, except for the goal of change-impact analysis that is tightly associated with correctness. This is due to the similarity between the analysis goals of both correctness and change-impact analysis in terms of purposes and underlying algorithms. In addition, all the goals are associated loosely with the similarity analysis and policy set structuring goals. In particular, the policy set analysis goal is associated with only three other goals: consistency, completeness, and minimality.

Fig. 5 shows the distribution of the methods used for verifying each goal for the policy analysis systems described in the surveyed papers. In general, every goal is assessed by various methods. Consistency, correctness, minimality, and completeness are mainly evaluated by model checking and formal methods because these methods are comprehensive and easy. Hence, policy analysis approaches using model checking and formal methods can be a baseline or benchmark when developing new systems. Other analysis goals (i.e., change-impact analysis, relevancy, and policy set structuring) are analyzed using other techniques which suitably fit the nature of these goals. For example, the change-impact analysis is evaluated using mutation testing since it is the most relevant method in order to apply various changes on the available set of policies to determine the impact on the quality of the analyzed policies.

![Fig. 6. The Distribution of Techniques per Analysis Methods Used by the Policy Analysis Systems and Frameworks Described in the Surveyed Papers](image-url)
diagrams are widely used because of their ability of representing all possible states of policies, hence reducing the ambiguities which arise in policy enforcement systems and resulting in accurate and comprehensive analysis results.

Table 2. Symbols for both Policy Analysis Goals and Methods

<table>
<thead>
<tr>
<th>Policy Analysis Goals</th>
<th>Policy Analysis Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>CON</td>
<td>Consistency</td>
</tr>
<tr>
<td>COM</td>
<td>Completeness</td>
</tr>
<tr>
<td>MIN</td>
<td>Minimality</td>
</tr>
<tr>
<td>REL</td>
<td>Relevance</td>
</tr>
<tr>
<td>COR</td>
<td>Correctness</td>
</tr>
<tr>
<td>PSS</td>
<td>Policy Set Structuring</td>
</tr>
<tr>
<td>SA</td>
<td>Similarity Analysis</td>
</tr>
<tr>
<td>CIA</td>
<td>Change Impact Analysis</td>
</tr>
</tbody>
</table>

Table 3. Comparison of Systems and Frameworks for Policy Analysis

<table>
<thead>
<tr>
<th>Publications</th>
<th>Policy Domain</th>
<th>Policy Quality</th>
<th>Policy Design</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIRAGE [83]</td>
<td>Network - Firewall</td>
<td>CON, MIN, REL, COR</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>TEQUILA [54, 55]</td>
<td>Network</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>FM - Event Calculus, FM - Reasoning</td>
</tr>
<tr>
<td>Wu et al. [218]</td>
<td>Access Control</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>FM - Matrix based</td>
</tr>
<tr>
<td>Vaidya et al. [212, 213]</td>
<td>RBAC</td>
<td>N/A</td>
<td>PSS</td>
<td>FM - Matrix based, DM - Role Mining</td>
</tr>
<tr>
<td>Bandara et al. [34]</td>
<td>Access Control, Management</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Event Calculus, FM - Reasoning</td>
</tr>
<tr>
<td>Bandara et al. [31]</td>
<td>Network</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Event Calculus, FM - Reasoning</td>
</tr>
<tr>
<td>Bandara et al. [32, 33]</td>
<td>Network - Firewall</td>
<td>CON, MIN, COR</td>
<td>N/A</td>
<td>FM - Argumentation</td>
</tr>
<tr>
<td>Bandara et al. [30]</td>
<td>Access Control</td>
<td>CON, COR</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Turkmen et al. [210]</td>
<td>XACML</td>
<td>CON, COR</td>
<td>CIA</td>
<td>FM - Reasoning, MC - SMT Solver</td>
</tr>
<tr>
<td>Applebaum et al. [25]</td>
<td>Network - Firewall</td>
<td>MIN</td>
<td>N/A</td>
<td>FM - Argumentation</td>
</tr>
<tr>
<td>Boella et al. [46, 47]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Argumentation</td>
</tr>
</tbody>
</table>

Continued on the next page
Table 3 – continued from the previous page

<table>
<thead>
<tr>
<th>Publications</th>
<th>Policy Domain</th>
<th>Policy Quality</th>
<th>Policy Design</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moffett and Sloman [165]</td>
<td>Management</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Kolovski et al. [121]</td>
<td>XACML</td>
<td>CON, MIN, COR</td>
<td>SA</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Lin et al. [127]</td>
<td>XACML</td>
<td>N/A</td>
<td>SA</td>
<td>GM - Tree, DM - Clustering</td>
</tr>
<tr>
<td>Lupu and Sloman [146]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>McDaniel and Prakash [161]</td>
<td>Access Control</td>
<td>CON, COR</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Craven et al. [64, 65]</td>
<td>XACML</td>
<td>CON, COM, MIN</td>
<td>SA</td>
<td>FM - Event Calculus, FM - Reasoning</td>
</tr>
<tr>
<td>Halpern and Weissman [91]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Benferhat et al. [39]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Cuppens et al. [66]</td>
<td>Rule-BAC, Or-BAC</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Adi et al. [12]</td>
<td>CA-BAC</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Wang et al. [216]</td>
<td>Access Control</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Sun et al. [206]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Liu [130]</td>
<td>Network - Firewall</td>
<td>MIN, COR</td>
<td>CIA</td>
<td>MC - Decision Diagrams, FM - Reasoning</td>
</tr>
<tr>
<td>Gupta et al. [88]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>EXAM [126]</td>
<td>XACML</td>
<td>N/A</td>
<td>SA</td>
<td>MC - SAT Solver, MTBDD</td>
</tr>
<tr>
<td>Margrave [80]</td>
<td>XACML</td>
<td>CON</td>
<td>CIA</td>
<td>MC - MTBDD</td>
</tr>
<tr>
<td>FIREMAN [223]</td>
<td>Network</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>MC - BDD</td>
</tr>
<tr>
<td>FLCheck [51]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>MC - BDD</td>
</tr>
<tr>
<td>Pina Ros et al. [181]</td>
<td>XACML</td>
<td>MIN</td>
<td>N/A</td>
<td>MC - BDD</td>
</tr>
<tr>
<td>Hughes and Bul-tan [101]</td>
<td>XACML</td>
<td>COR</td>
<td>N/A</td>
<td>MC - Alloy, MC - SAT Solver</td>
</tr>
<tr>
<td>Jeffrey and Samak [111]</td>
<td>Network</td>
<td>COR</td>
<td>N/A</td>
<td>MC - SAT Solver</td>
</tr>
<tr>
<td>Alberti et al. [21, 22]</td>
<td>RBAC</td>
<td>MIN, COR</td>
<td>N/A</td>
<td>FM - Reasoning, MC - SMT Solver</td>
</tr>
<tr>
<td>MOHAWK [110]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>MC - BDD, MC - SAT Solver</td>
</tr>
</tbody>
</table>

Continued on the next page
Table 3 – continued from the previous page

<table>
<thead>
<tr>
<th>Publications</th>
<th>Policy Domain</th>
<th>Policy Quality</th>
<th>Policy Design</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jayaraman et al. [109]</td>
<td>Network</td>
<td>CON, COR</td>
<td>CIA</td>
<td>MC - SMT Solver</td>
</tr>
<tr>
<td>Mankai and Logrippo [149]</td>
<td>XACML</td>
<td>CON</td>
<td>N/A</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Hassan et al. [96]</td>
<td>XACML</td>
<td>CON</td>
<td>N/A</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Power et al. [182]</td>
<td>RBAC</td>
<td>COR</td>
<td>CIA</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Karimi et al. [114], Karimi and Cowan [115]</td>
<td>RBAC, XACML</td>
<td>CON, COR</td>
<td>N/A</td>
<td>MC - Alloy, MC - SAT Solver, MC - BDD</td>
</tr>
<tr>
<td>Schaad and Moffett [193]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Haraty and Naous [94]</td>
<td>Network</td>
<td>CON</td>
<td>N/A</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Guttman et al. [89]</td>
<td>Access Control</td>
<td>COR</td>
<td>N/A</td>
<td>MC - BDD</td>
</tr>
<tr>
<td>Tanvir and Tripathi [209]</td>
<td>RBAC</td>
<td>CON, COM, COR</td>
<td>N/A</td>
<td>MC - SAT Solver</td>
</tr>
<tr>
<td>AcPeg [226, 227]</td>
<td>XACML</td>
<td>COM, COR</td>
<td>N/A</td>
<td>MC - BDD</td>
</tr>
<tr>
<td>Ma et al. [147]</td>
<td>Access Control</td>
<td>CON, COM, COR</td>
<td>N/A</td>
<td>MC - SAT Solver</td>
</tr>
<tr>
<td>Hu and Ahn [98]</td>
<td>RBAC</td>
<td>CON, COR</td>
<td>N/A</td>
<td>MC - Alloy, MC - SAT Solver, FM - Reasoning</td>
</tr>
<tr>
<td>Hwang et al. [105]</td>
<td>XACML</td>
<td>COR</td>
<td>N/A</td>
<td>MC - SAT Solver, MC - BDD</td>
</tr>
<tr>
<td>PoliVer Koleini and Ryan [120]</td>
<td>Access Control</td>
<td>COR</td>
<td>N/A</td>
<td>FM - Reasoning, MC - BDD</td>
</tr>
<tr>
<td>Ngo et al. [177]</td>
<td>XACML</td>
<td>MIN, COR</td>
<td>N/A</td>
<td>MC - MIDD</td>
</tr>
<tr>
<td>Nelson et al. [175]</td>
<td>Network - SDN</td>
<td>COR</td>
<td>N/A</td>
<td>MC - Alloy</td>
</tr>
<tr>
<td>Mazzoleni et al. [159, 160]</td>
<td>XACML</td>
<td>N/A</td>
<td>SA</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Golnabi et al. [84]</td>
<td>Network - Firewall</td>
<td>MIN</td>
<td>N/A</td>
<td>DM - Association Rule Mining</td>
</tr>
<tr>
<td>Bauer et al. [37]</td>
<td>RBAC</td>
<td>CON, COR</td>
<td>N/A</td>
<td>DM - Association Rule Mining</td>
</tr>
<tr>
<td>Ma et al. [148]</td>
<td>RBAC</td>
<td>COM, COR</td>
<td>N/A</td>
<td>DM - Association Rule Mining</td>
</tr>
<tr>
<td>Marouf et al. [150, 151]</td>
<td>XACML</td>
<td>N/A</td>
<td>PSS</td>
<td>DM - Clustering</td>
</tr>
<tr>
<td>Ait El Hadj et al. [15]</td>
<td>XACML</td>
<td>MIN</td>
<td>N/A</td>
<td>DM - Clustering</td>
</tr>
<tr>
<td>Shaikh et al. [196, 197, 198]</td>
<td>Access Control</td>
<td>CON, COM, MIN</td>
<td>N/A</td>
<td>DM - Classification</td>
</tr>
<tr>
<td>Aqib and Shaikh [27]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>DM - Classification</td>
</tr>
</tbody>
</table>

Continued on the next page
### Table 3 – continued from the previous page

<table>
<thead>
<tr>
<th>Publications</th>
<th>Policy Domain</th>
<th>Policy Quality</th>
<th>Policy Design</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schlegelmilch and Steffens [194]</td>
<td>RBAC</td>
<td>N/A</td>
<td>PSS</td>
<td>DM - Role Mining, DM - Clustering</td>
</tr>
<tr>
<td>Molloy et al. [167]</td>
<td>RBAC</td>
<td>N/A</td>
<td>PSS</td>
<td>DM - Role Mining</td>
</tr>
<tr>
<td>Dunlop et al. [75]</td>
<td>Management</td>
<td>CON</td>
<td>N/A</td>
<td>DM - Classification</td>
</tr>
<tr>
<td>Mukkamala et al. [174]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>DM - Role Mining</td>
</tr>
<tr>
<td>Basile et al. [36]</td>
<td>Network - Firewall</td>
<td>CON, MIN</td>
<td>N/A</td>
<td>DM - Classification</td>
</tr>
<tr>
<td>Xu et al. [221]</td>
<td>Access Control</td>
<td>COR</td>
<td>CIA</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Staniford-Chen et al. [205]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Alves and Fernández [23, 24]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Bertino et al. [41]</td>
<td>RBAC</td>
<td>CON, COM, MIN, REL</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Davy et al. [71]</td>
<td>Network</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Al-Shaer et al. [17–20]</td>
<td>Network - Firewall</td>
<td>CON, MIN, REL, COR</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Sarna-Starosta and Stoller [192]</td>
<td>Access Control</td>
<td>CON, COM, COR</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Koch et al. [118]</td>
<td>LBAC</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Russello et al. [188]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Mohan et al. [166]</td>
<td>XACML</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Huang et al. [100]</td>
<td>RBAC</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Bravo et al. [48]</td>
<td>Access Control</td>
<td>CON</td>
<td>N/A</td>
<td>GM - Graph</td>
</tr>
<tr>
<td>Aqib and Shaikh [26]</td>
<td>Access Control</td>
<td>CON, COR</td>
<td>N/A</td>
<td>GM - Tree</td>
</tr>
<tr>
<td>Martin et al. [152–156]</td>
<td>XACML</td>
<td>COR</td>
<td>CIA</td>
<td>MT</td>
</tr>
<tr>
<td>Pretschner et al. [183]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>MT</td>
</tr>
<tr>
<td>Mouelhi et al. [170, 171, 172, 173]</td>
<td>RBAC, XACML, OrBAC</td>
<td>CON, COR</td>
<td>N/A</td>
<td>MT</td>
</tr>
<tr>
<td>Masood et al. [157]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>MT</td>
</tr>
<tr>
<td>Elrakaiby et al. [76]</td>
<td>RBAC</td>
<td>COR</td>
<td>CIA</td>
<td>MT</td>
</tr>
<tr>
<td>Le Traon et al. [125]</td>
<td>RBAC</td>
<td>COR</td>
<td>N/A</td>
<td>MT</td>
</tr>
</tbody>
</table>

Continued on the next page
### Table 3 – continued from the previous page

<table>
<thead>
<tr>
<th>Publications</th>
<th>Policy Domain</th>
<th>Policy Quality</th>
<th>Policy Design</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xu et al. [219]</td>
<td>XACML</td>
<td>COR</td>
<td>N/A</td>
<td>MT</td>
</tr>
<tr>
<td>Gorgias-B Spanoudakis et al. [204]</td>
<td>Access Control, Network - Firewall</td>
<td>CON</td>
<td>N/A</td>
<td>FM - Reasoning</td>
</tr>
<tr>
<td>Gouda and Liu [85, 86]</td>
<td>Network - Firewall</td>
<td>CON, COM, MIN</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu and Gouda [135, 137]</td>
<td>Network - Firewall</td>
<td>MIN</td>
<td>CIA</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu and Gouda [138]</td>
<td>Network - Firewall</td>
<td>CON</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu [128]</td>
<td>Network - Firewall</td>
<td>COR</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu et al. [136, 139, 141]</td>
<td>Network - Firewall</td>
<td>MIN</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu and Chen [131, 132]</td>
<td>Network - Firewall</td>
<td>CON, COM</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu et al. [133, 134]</td>
<td>XACML</td>
<td>CON, COM</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Liu et al. [142, 143]</td>
<td>Network - Firewall</td>
<td>CON, COM, MIN</td>
<td>N/A</td>
<td>MC - Decision Diagram</td>
</tr>
<tr>
<td>Hwang et al. [103, 104]</td>
<td>Network - Firewall</td>
<td>COR</td>
<td>CIA</td>
<td>MT</td>
</tr>
<tr>
<td>Chen et al. [58, 59]</td>
<td>Network - Firewall</td>
<td>CON, COM, COR</td>
<td>CIA</td>
<td>MC - Decision Diagram, MT</td>
</tr>
<tr>
<td>Khakpour and Liu [117], Liu and Khakpour [140]</td>
<td>Network - Firewall</td>
<td>CON, COM</td>
<td>N/A</td>
<td>MC - Decision Diagram, FM - Matrix based</td>
</tr>
<tr>
<td>Chen et al. [56, 57]</td>
<td>Network - Firewall</td>
<td>CON, COM, MIN</td>
<td>PSS</td>
<td>MC - Decision Diagram</td>
</tr>
</tbody>
</table>

### 7 RESEARCH DIRECTIONS

The area of methods, systems, and frameworks for policy analysis has been widely investigated, especially in the context of access control systems and network management systems. However, as our previous section has shown, there are no comprehensive systems able to support a variety of analysis with different goals. Most systems focus on some specific type of analysis with specific goals, for example consistency analysis. It is thus clear that more general policy analysis systems are needed. In what follows we discuss several relevant directions for such analysis systems and methods.

#### 7.1 New Goals for Policy Analysis

Policy analysis is an area in which there are still important open research directions specifically motivated by the deployment of policy-based management in the context of distributed systems consisting of autonomous intelligent devices. A first relevant research direction is related to the
identification of analysis goals for such contexts. Two important goals are the assessments of enforceability and risk. Enforceability refers to an assessment of the feasibility and the cost of enforcing a set of policies. For example, a policy whose enforcement requires access to top secret information may be very difficult to enforce in an insecure enforcement environment as the transfer of the top secret information into the enforcement system may not be possible. This means that this policy may be difficult to enforce and may require a specialized partitioned architecture which may thus be expensive. Assessing the risks arising from the use of specific policies is critical and non-trivial as risks are usually application and context dependent. Further, the risk appetite of system owners could change. Therefore risk assessment methods are required able to assess policies with respect to specific applications and also able to continuously assess risks to deal with changing contexts and situations. Thus, policy changes may be required in response to such a changing risk and a risk appetite.

7.2 New Methods for Policy Analysis

New methods for policy analysis ought to take into account policy goals of enforceability and risk, and evaluate them in the presence of uncertainty or unpredictability of the context in which systems operate. They cannot, therefore, be restricted to only traditional analysis tasks, nor they can assume “perfect” inference about properties of the environment and of the system state. The new methods of analysis methods will need to be more stochastic in nature in both the representation of the new policy goals and the analysis process. For example, analysis methods based on probabilistic model checking techniques (e.g., PRISM [124]) could be used to support automated verification of quantitative analysis of enforceability and risk. Probabilistic model checking can support the analysis of several types of probabilistic models (e.g., discrete-time Markov chains, probabilistic automata, and probabilistic timed automata), with respect to properties expressed in probabilistic temporal logic (PCTL). Risk scenarios could be modeled using Markov chain (MC) models of stochastic (discrete) risk-related policy-driven scenarios, and probabilistic model checking could be used to evaluate expected risk implied by these models, by expressing these goals using probability formalisms (e.g., PCTL [93]), and evaluating the models against these properties. A similar approach can be used for the analysis of the enforceability of policies. Markov chain models are normally extended with notions of costs and rewards used to estimate resources and their usage. The challenge is how to express policy specifications into these models, for the purpose of analysis, and what types of properties to express in order to capture notions of risk and enforceability. A recently proposed technique that combines probabilistic and abductive inference for reasoning about most likely explanations of violations of given goals [211] could also provide an alternative underpinning framework for new methods of policy analysis. Such methods would constitute a natural extension of established state-of-the-art policy analysis techniques based on abductive inference to the context of uncertainty and risk. Both proposals would cover the case of static analysis of policies with respect to new classes of goals. Analysis methods based on run-time executions of autonomous intelligent systems could also be envisaged. These would rely upon analysis of outcomes of enforced policies. Advanced machine learning algorithms could be used to recognize patterns of risk and/or (lack of) enforceability of policies through past execution traces. Policies could then be analyzed with respect to these learned patterns and outcomes of the analysis could be used to inform ways to revise and adapt the policies. The trained machine learning architectures could also be applied to new (unseen) situations to predict future goal violations.

7.3 New Analysis Frameworks for Distributed Systems

Another interesting research direction is about policy analysis in large-scale distributed systems and refers to whether policy analysis (and possibly consequent policy evolution) must be carried
centrally or locally at different subsystems. Ensuring that a set of policies is of high quality at a
global level may be difficult if at all possible, as different portions of the system may be characterized
by different access patterns and different contexts. On the other hand, carrying out policy analysis
according to a distributed strategy so that policies are analyzed at different subsystems may result
in policy analysis and evolution that are optimal with respect to local contexts but not optimal with
respect to a more global level. A possible approach is to have a flexible analysis infrastructure able
to support and possibly combine both approaches depending on the specific requirements of the
system of interest. This is particularly relevant in dynamic network environments (e.g., coalitions
and federated networks) that may involve network entities from different parties.

7.4 Next Generation Policy-based Management for Autonomous Devices

Finally, a newly emerging approach in the area of policy-based management is represented by the
notion of generative policies [215] that has been proposed as the next generation policy-based
management approach for cognitive autonomous devices and IoT devices. In a generative policy
framework, devices are given policy templates and they can autonomously instantiate and refine
such templates according to their own missions and contexts. The generative policy framework
will require novel analysis techniques to analyze policy templates and how these templates are
instantiated by different devices in different contexts and for different missions (e.g., cognitive
and distributed collaborative applications) and compare different instantiations to determine for
example the optimal ones for specific contexts. Since the generative policy approach brings the
ability of generating policies based on the contexts, an analysis of the change impact might be
needed to assess the potential consequences and the risks associated with the change. An efficient
mechanism is required for conducting such a dynamic policy analysis for cognitive autonomous
devices with limited storage and computation capabilities. Also for IoT devices that have limited
computing capabilities, some of the functions for policy generation may have to be delegated to
more powerful devices and systems, such as edge servers. For those devices, an important issue is
to analyze policies in order to determine the costs of policy enforcement with respect for example
to energy and storage.

ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of
Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in
this document are those of the authors and should not be interpreted as representing the official
policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government,
the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized
to reproduce and distribute reprints for Government purposes notwithstanding any copyright
notation hereon.

REFERENCES


[92] J. Han, J. Pei, and Y. Yin. 2000. Mining Frequent Patterns without Candidate Generation. In SIGMOD. Vol. 29. ACM, 1–12.


ACM Computing Surveys, Vol. 1, No. 1, Article . Publication date: July 2018.


Methods and Tools for Policy Analysis


