Towards Adaptive Access Control

Luciano Argento[1][0000−0001−8869−5035], Andrea Margheri[2][0000−0002−5048−8070], Federica Paci[2][0000−0003−3122−0236], Vladimiro Sassone[2][0000−0002−6432−1482], and Nicola Zannone[3][0000−0002−9081−5996]

[1] University of Calabria
luciano.argento@unical.it
[2] University of Southampton
{a.margheri;f.m.paci;vsassone}@soton.ac.uk
[3] Eindhoven University of Technology
n.zannone@tue.nl

Abstract. Access control systems are nowadays the first line of defence of modern IT systems. However, their effectiveness is often compromised by policy miscofigurations that can be exploited by insider threats. In this paper, we present an approach based on machine learning to refine attribute-based access control policies in order to reduce the risks of users abusing their privileges. Our approach exploits behavioral patterns representing how users typically access resources to narrow the permissions granted to users when anomalous behaviors are detected. The proposed solution has been implemented and its effectiveness has been experimentally evaluated using a synthetic dataset.

Keywords: Access control, machine learning, policy adaptation, insider threat, runtime monitoring

1 Introduction

Data are recognized as the most vital asset of an enterprise and, thus, their protection is of paramount importance. Access control systems are typically employed as the first line of defence for the protection of data as they guarantee that only authorized users can gain access to sensitive resources. Over the past few years, Attribute-Based Access Control (ABAC) [1] has gained in popularity due to its flexibility and expressiveness, allowing the specification of fine-grained and context-aware access control policies.

Despite this flexibility and expressiveness, ABAC (and access control in general) has an intrinsically static nature that makes it difficult to adapt policies in order to timely response to critical events, e.g. a cyber attack. At the same time, policies can become out-of-dated quickly, thus requiring continuous, manual maintenance, which makes policy management and administration a cumbersome and error-prone task. These issues can lead to policy misconfigurations that leave organizations exposed to attacks against data confidentiality and integrity.

A study conduced by SANS in 2017 on threats against sensitive data[4] showed that insider threats are the top concern for organizations, followed by ransomware and denial

[4] `https://www.sans.org/reading-room/whitepapers/analyst/sensitive-data-risk-2017-data-protection-survey-37950`

of service attacks. Insider threats are current or former employees, contractors or business partners who have or had authorized access to the organization's network, system or data, and intentionally exceeds or misuses their privileges in a manner that negatively affected the confidentiality, integrity or availability of the organization's information or information system [2]. Thus, organizations need to reinforce their access control systems with procedures to identify policy misconfigurations that could be exploited by an insider threat and update the policies to prevent such exploitations.

Detecting and preventing insider threats by analyzing access control policies and monitoring user behavior have been an active area of research. Common approaches rest on rule mining techniques to discover harmful exploitable policy faults [3,4] or on monitoring systems based on behavioral models to detect insider threats [5,6,7,8]. Other works [9,5] have also exploited knowledge from access control policies to detect insider threats. However, these approaches only aim at threat detection and do not focus on the adaptation of access control policies. To date, only a few works (e.g., [10,11]) have exploited user behavior to generate and adapt access control policies. However, these approaches either require human intervention for policy update [10] or build models that do not properly discriminate behaviors of different types of users [11].

Contribution. In this paper, we propose an approach based on machine learning to dynamically refine policies to prevent misconfiguration exploitation. The proposed approach allows the refinement of access control rules according to behavioral features monitored at run-time. The designed system, named ML-AC, exploits a white-box decision learning approach whose aim is to learn behavioral profiles of users accessing resources so to accurately refine policies. There might exist different behavioral profiles, here called *classes of interaction*, that can be determined based on the analysis of contextual knowledge that concerns users and resources. Such knowledge has proven to be a valuable source of information for approaches devoted to improve insider threat detection and access control [6,12].

ML-AC uses access pattern knowledge learned at run-time to introduce controls on behavioral features into access control rules to avoid abuse of granted rights. Behavioral features refine access policies by introducing controls like frequency of access, amount of data, location, etc. By building an access control knowledge model, this work poses the basis towards machine-assisted administration procedures to support timely changing of access rights.

Paper structure. Section 2 provides a motivating example. Section 3 provides an overview of ML-AC and details how machine learning is empowering access control. Section 4 describes the implementation and evaluation of the performance of ML-AC. Section 5 concludes the paper and outlines directions for future work.

2 Motivating Example

To outline our approach, we introduce an ABAC system managing accesses to software projects within an organization where users' permissions depend on their role and on the projects they are assigned to.

Let us assume an access control policy allowing users assigned to role *junior manager* to *read* resources of type $R1$. However, only junior managers working on *ProjectA* and from *Department 1* can actually access those resources. This policy can be represented in the FACPL language [13], an XACML-like ABAC language that we use for its conciseness, as follows:

```
policy policy1 {deny-unless-permit
 rule rule1(permit
  target:
     equal ("read", action/id) && equal ("R1", resource/type)
     && equal ("Junior manager", subject/role)
     && equal ("Department1", subject/department)
     && equal ("ProjectA", subject/project))}
```

Intuitively, our sample policy consists of a `policy` element (*policy1*) comprising a *permit* `rule` (*rule1*) whose `target` defines an access condition built on attributes describing which *action* a certain *subject* can perform on a *resource*.

Let us assume that Bob, a *junior manager* of *Department 1*, attempts to *read* a resource of type $R1$, represented by the following access request (*req1*):

```
(subject/department,"Department 1")
(subject/role, "Junior manager")
(subject/project, "ProjectA")
(action/id="read")
(resource/type,"R1")
```

It is easy to observe that the attributes in the request match the rule target, thus yielding a *permit* decision.[5]

Suppose now that Bob attempts to retrieve a large amount of sensitive project documents without a plausible reason. As the previous request exemplifies, *policy1* would allow him to do so regardless of how many documents he has retrieved. This situation, however, may indicate that the junior manager is abusing his access privileges for personal interests and benefits (e.g., to sell the documents to a competitor).

These insider threats cannot be prevented by existing access control systems. The main problem lies in the fact that access control is static in the sense that the enforced access conditions do not change dynamically according to user behaviour. We argue that *contextual features*, such as the *number of accesses* and *amount of accessed data*, should be taken into account in access decision making. For instance: *Could a user perform multiple read queries in a given time window? Could a user access large amounts of data?* Failing to answer these questions can lead to neglect anomalous behaviour representing the abuse of granted access privileges from insider threats.

To reduce the risks of users abusing their privileges, we need to empower access control with proactive measures that adapt policies according to user behavior. Specifically, our goal is to *dynamically refine access control policies based on user behaviour monitored at run-time by narrowing granted privileges.*

To achieve this goal, we need to equip access control systems with a means to build user profiles representing how users normally access resources and use those profiles to dynamically refine access rules. This requires extracting contextual features that capture

[5] We overlook combining algorithm `deny-unless-permit` as it yields *permit* if the enclosed rule returns *permit*, and yields *deny* otherwise.

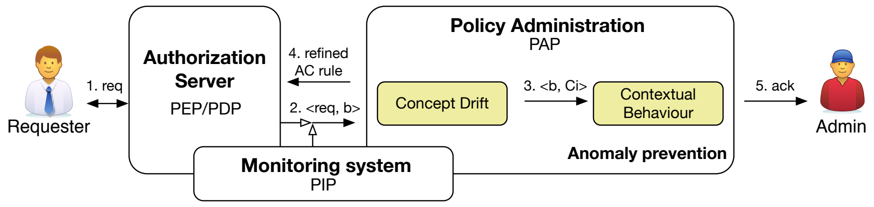


Fig. 1. The ML-AC system

the behavior exhibited by users accessing a specific set of resources by means of a selected set of operations.

For the sake of exemplification, let us assume that contextual features `feature/NumberOfReadsPerHour` and `feature/BytesReadPerHour` are monitored by the system and can be checked via new attributes in access rules. In particular, it is observed that every hour junior managers typically access at most 14 project documents for a total 345.6 KB. This knowledge can be exploited to refine *policy1* as follows:

```
policy policy2 {deny-unless-permit
 rule rule2 (permit
  target:
    equal ("read", action/id) && equal ("R1", resource/type)
    && equal ("Junior manager", subject/role)
    && equal ("Department1", subject/department)
    && equal ("ProjectA", subject/project))}
    && less-than (feature/BytesReadPerHour, 345.6)
    && less-than (feature/NumberOfReadsPerHour, 14)
```

Intuitively, *policy2* narrows the access conditions of *policy1* through contextual features by imposing additional constraints on how much and how often resources are typically accessed by junior managers. Consider, for instance, the case where Bob attempts to access 50 project documents within 10 minutes. Based on the updated policy, this behavior would be deemed as anomalous and thus denied, preventing Bob to access all documents.

3 Adaptive Access Control

In this section, we introduce ML-AC, a system for adaptive access control that aims to reduce the risks of users abusing their privileges. Figure 1 presents the ML-AC architecture. It comprises the following components:

- *Authorization Server* is a standard ABAC infrastructure à la XACML based on PEP/PDP.
- *Policy Administration Point (PAP)* features the proactive policy refinement functionalities proposed in this work.
- *Monitoring System* supervises the whole system and provides the information needed to build behavioral profiles used by Policy Administration in its refinement process.

Once an access request is received by the Authorization Server (step 1), it is evaluated following typical PEP/PDP evaluation frameworks. If access is granted (i.e., a *permit* decision is returned) by the Authorization Server, the request, together with the corresponding behaviour provided by the Monitoring system, is forwarded to the PAP (step 2) to determine whether it is anomalous. The PAP dynamically refines the access control policies based on the user behavior and enforce them in the Authorisation Server (steps 3-4). Administrators are informed of the changes (step 5).

Our contribution lies in the Policy Administration component. Specifically, we equip PAPs with the *Contextual Behaviour Learning* component, which is responsible to build user profiles according to the monitored behavior and refine access control policies based on those profiles. The *Concept Drift* component detects the evolution of the learned user profiles and inform the Contextual Behavior Learning component when new behaviors are detected (step 3).

In the remaining, we introduce our representation of behavioral models and present the mechanics of the Contextual Behaviour and Concept Drift components.

3.1 Behavioral Model

To build the behavioral models used to identify anomalous accesses, we introduce the notion of *user behavior*. Behaviours, denoted as b, represent how users are *utilizing* resources. They are defined in terms of the attributes forming access requests (i.e., user, resource and action) and of any contextual knowledge features that can be exploited by the access control system for decision making (e.g., working time, working location, types of activities). Formally, a behaviour is defined as:

$$b \triangleq \langle A^u_1, \ldots, A^u_m, A^o_1, \ldots, A^o_h, A^r_1, \ldots, A^r_k, \ldots A^c_1 \ldots A^c_z \rangle$$

where attribute A^u_i describe users, A^o_k operations, A^r_y resources and A^c_z the context. An attribute is an expression of form $A \triangleq \texttt{name}\ \texttt{op}\ value$, where op is a relational operator (e.g., $=, >$) between an attribute name and a value from the attribute's codomain. For instance, the following behaviour (we shorten attribute names for ease)

$$b_1 = \langle \texttt{role} = \textit{jnr-mng}, \texttt{act} = \textit{read}, \texttt{res} = \textit{R1}, \#\texttt{byte} = 250, \#\texttt{read} = 10 \rangle$$

corresponds to the access request reported in Section 2, where $\#\texttt{byte}$ and $\#\texttt{read}$ are contextual features provided by the Monitoring System.

Behaviors are grouped into so-called *class of interaction*s. Each class represents a group of homogeneous behaviours (i.e., whose representation involves the same set of attribute name) that are considered normal. We assume that the set of initial classes are defined according to the access policy rules, hence on the basis of the controls on user, action and resource attributes. Changes at runtime to the classes are managed by the Concept Drift component (Section 3.3).

3.2 Contextual Behavior

Our approach builds behavioral profiles of normal accesses in the form of class of interactions. To determine to which class of interaction a given behavior belong, we relies on Random Forest (RF) [14], where each RF is used to characterize a class of interaction. Based on this matching, the knowledge on the corresponding contextual features is used to refine access control policies.

Learning Practicalities. Given a class of interactions C_i, our goal is to recognize whether a user behavior b_i is similar to those represented by C_i or not, i.e. whether b_i is anomalous. Practically, being b_i a potentially anomalous behavior, we cannot assume it will always match (the attributes of) C_i. It follows that this problem cannot be addressed as a multi-class classification (labels would be represented by the classes of interactions and our goal would be to determine if the label of a test sample b_i is C_i or another C_j, with $i \neq j$), but as a *One Class Classification* (OCC) problem [15] for each of class of interaction C_i individually.

Solutions for the OCC problem are numerous. However, being our goal to obtain learning outputs that can be used to refine policies, we opted for the white-box approach of RF. Specifically, each RF is an ensemble of *Decision Tree* (DT) each of which models the conditions on the attributes identifying the normal behaviors of a class. Each DT produces an output of the form of $antecedent \Rightarrow consequent$ rules [16]. The antecedent consists of logical conjunctions stating under which conditions a behavior can be classified as *normal* or *anomalous*, as indicated by the consequent.

It is worth noting that each DT produces its own output, leading to potential inconsistencies. For instance, let us suppose a RF with three DTs. Given a behavior b, each DT produces its own decision, e.g. $DT_1(b) = 1$, $DT_2(b) = 0$ and $DT_3(b) = 1$, where 1 and 0 denote normal and anomalous behavior, respectively. To solve this problem RF relies on a majority vote algorithm among DTs; hence b is classified as normal. Notably, a final decision can always be guaranteed by using a known RF voting solution [17].

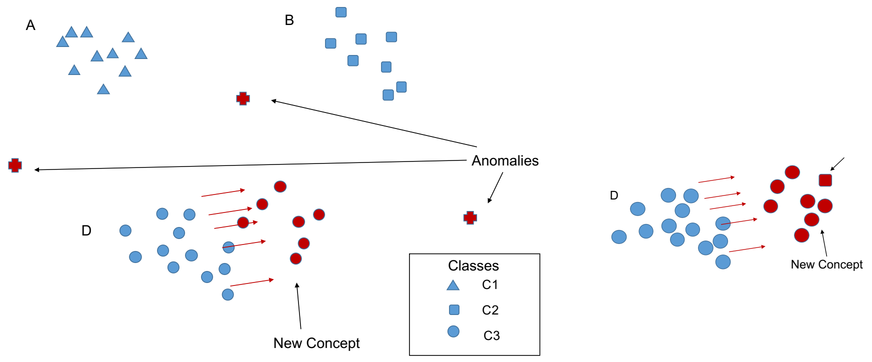
Access Control Refinement. The DT outputs, hereafter called *ML-rules*, is used to bridge from the machine learning world to the actual refinement of access control policies. Practically, they encode the obtained knowledge in terms of additional conditions to add to the access control policy. The syntax of ML-rules is defined by the following grammar:

$$\begin{array}{rlrl} \textit{ML-rule} & ::= \textit{Antec} \Rightarrow \textit{Consec} & & \\ \textit{Antec} & ::= \textit{Cnd}\ \ \{\&\&\ \textit{Cnd}\} & \textit{Cnd} & ::= \texttt{name}\ \textit{op}\ \texttt{value} \\ \textit{Op} & ::= = \mid > \mid < \mid \leq \mid \geq \mid\ \texttt{in} & \textit{Consec} & ::= \texttt{true}\ \mid\ \texttt{false} \end{array}$$

where `true` (resp., `false`) identifies a normal (resp., anomalous) behavior. The antecedent consists of a conjunctive sequence of conditions, whereas the consequent is a Boolean value stating whether a behavior satisfying the antecedent is normal or anomalous. For instance, the following ML-rule represents the refinement leading from *policy1* to *policy2* of the example

$$\begin{array}{l} \texttt{role} = \textit{jnr-mng}\ \&\&\ \texttt{act} = \textit{read}\ \&\&\ \texttt{res} = \textit{R1} \\ \&\&\ \#\texttt{read} < 14\ \&\&\ \#\texttt{byte} < 345.6 \end{array} \quad \Rightarrow \quad \texttt{true}$$

Therefore, ML-rules are used to transfer the contextual knowledge learned by the RFs into the access control policies. Policy refinement occurs on the basis of the conditions on the contextual features present in an ML-rule (e.g., the #`read` and #`byte` above). Practically, the access control rules to refine correspond to those matching the conditions on user, action and resource attributes present in the ML-rule. The refined rules contain the additional controls on the contextual features as per the example of *policy2*.



(a) Detection of a new concept along with anomalies (b) Suspicious instances

Fig. 2. Detection of concept drift

3.3 Concept Drift

After the behavioral models have been build, the system starts monitoring the evolution of user behaviors to detect concept drifts in order to maintain the RF models accurate over time. It follows from the RF design that concept drifts can only be detected on the contextual features.

To this aim, we rely on *Olindda* [18], a clustering-based approach similar to the one followed by BBNAC [11]. Olindda uses the well-known *k-means* algorithm (or one of its extensions, e.g. *k-modes* for categorical features) to cluster behaviors of the classes of interactions and detect the emergence of new classes (aka new clusters) based on the distance among clusters.

Figure 2(a) depicts a scenario where concept drift can be observed. For instance, given three classes of interactions C_1, C_2 and C_3, at the beginning just three groups of user behaviors, respectively the clusters *A*, *B* and *D* of blue-filled shapes, are identified. After a certain amount of time, the behaviors in the cluster D change to the point that concept drift is detected: the red-filled shapes. Therefore, these behaviors are used by the RF modelling the class C_3 to update its knowledge accordingly. Additionally, when behavior like those represented by red cross shapes are observed, i.e. not forming any cluster, Olindda deems them as anomalies. This is the key assurance to avoid RF to refine access control policies with conditions allowing anomalous behaviors.

Notably, emerging concepts may overlap among classes leading to difficulties in their classification. For instance, Figure 2(b) depicts new concepts emerging from the cluster D that present similarities to both cluster D (circle shapes) and cluster B (square shape). All emerging concepts like the latter are treated as anomalies.

Divergence in clusters. A key aspect of concept drift is the emergence of sub-clusters within an existing cluster. Given a class C_i, it may happen that a subset of its users start behaving significantly different from the remaining users, over time. This would lead to the discovery of new classes of interactions detailing different behavioral profiles.

Our approach aims to build classes of interactions that are characterized by behaviors that are very similar to each other. Therefore, in the light of the new discovered

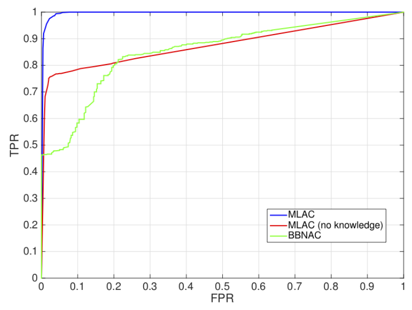


Fig. 3. Comparison among ML-AC, *BBNAC* and ML-AC$_{nok}$

sub-clusters, we derive two new classes of interaction from C_i, namely C_i^n and C_i^o, representing behavior related to the new and old concepts, respectively. This has the consequence of introducing a new RF modelling the new class. In order to keep updating the policy refinements generated by the initial C_i, it is required a layered RF modelling able to discriminate between the two new classes. Further details are left to future work.

4 Evaluation

To evaluate the effectiveness of our approach, we have implemented it[6] and performed experiments using a synthetic dataset. The goal of the experiments is to demonstrate the benefits of combining domain context knowledge (inferred using machine learning) with knowledge based on access control rules (used to create classes of interaction) for the detection of anomalies. In particular, we assess the detection accuracy of ML-AC by analyzing the number of true and false positive. This will allows us to demonstrate how the refinement based on this detection approach will prevent anomalies to happen a priori. Note that we do not evaluate the use of machine learning to classify user behavior, being this extensively reported in the literature (e.g., [19]).

In the experiments, we compared our approach for anomaly prevention with two other approaches.We took BBNAC [11], the closest approach from the literature. To avoid bias in the experimentation, we re-implemented BBNAC in Matlab.[7] Additionally, we performed a comparison with ML-AC$_{nok}$, a variant of ML-AC where a priori knowledge on the policy is not used. This allows us to better assess the role of contextual knowledge.

Dataset. We generated a synthetic dataset whose data instances represent user behaviors, under the assumption that there are two classes of interactions. The behaviors are represented as three-dimensional points and based on numerical features. We generated over 3000 behaviors, with almost an equal number of normal and anomalous instances. Intuitively, about 2000 behaviors were used for training, while the rest for testing.

Results. To evaluate the accuracy of the approaches, we computed the ROC curve for the three evaluated approaches (Figure 3). The ROC curve is a graphical plot we use to evaluate how well the classifier we defined distinguishes between normal and anomalous

[6] The tool is freely available at `https://github.com/cybersoton/ml-ac/`

behaviors, based on a varying discrimination threshold (i.e., the value used to deem a behavior anomalous). Specifically, it shows the true positive rate or TPR (y-axis) against the false positive rate or FPR (x-axis) with respect to different threshold settings. These performances can also be reduced to a single scalar value, called *area under the ROC curve* (AUC), which represents the probability that a classifier will rank a randomly chosen anomalous instance higher than a randomly chosen normal instance. This gives better insights on the number of false positive detected.

As can be seen in Figure 3, ML-AC achieves the best performance. ML-AC significantly surpasses the others due to the use of a priori knowledge on the policy. This allows ML-AC to create more accurate profiles and hence to discern more precisely among groups of behaviors. Specifically, we have that the AUC drops from the 99% of ML-AC to the 87% and 85% of ML-AC$_{nok}$ and BBNAC, respectively. The main cause is that both BBNAC and ML-AC$_{nok}$ classify most of the anomalous behaviors of each class as a normal behavior of the other class.

5 Conclusion

In this paper we proposed ML-AC, an approach to refine and update access policies in order to eliminate policy misconfigurations that can be exploited by insider threats. ML-AC builds behavioral models representing the normal usage of resources and exploits these models at run-time to prevent anomalous accesses. Our approach has been implemented and validated using a synthetic dataset.

Future work. There are a number of interesting aspects to address as future work. Firstly, we plan to conduct more extensive experiments based on real-life datasets and measure the number of refinement applied in practice to the access control policies.

Moreover, we would like to study further the presence of an adversarial attacker [20], who may try to deceive the machine learning algorithm in order to bypass security controls. ML-AC employs different machine learning algorithms to achieve its goals, therefore it may incur in the risk of being subject to adversarial attacks. The definition of class of interaction may constitute a starting point for devising strategies to hinder typical attacks such as causative and exploratory. The fact that user behaviors are split based on the classes, may make the task of deceiving ML-AC quite difficult. Many anti-adversarial algorithms have been proposed in literature, like those presented in [11] that effectively mitigate threats related to concept drift.

Another aspect concerns classes of interactions. It might not always be possible to clearly define those classes based only on the knowledge derived from the context and access control policies. There may be some classes that should be merged or split, hence we plan to design a preprocessing step to support the classes definition process.

References

1. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Attribute-based access control. IEEE Computer **48**(2) (2015) 85–88
2. Silowash, G., Cappelli, D., Moore, A., Trzeciak, R., Shimeall, T., Flynn, L.: Common sense guide to mitigating insider threats. Technical report (2012)

3. Hwang, J., Xie, T., Hu, V., Altunay, M.: Mining likely properties of access control policies via association rule mining. In: Data and Applications Security and Privacy, Springer (2010) 193–208
4. Bauer, L., Garriss, S., Reiter, M.K.: Detecting and resolving policy misconfigurations in access-control systems. ACM Trans. Inf. Syst. Secur. **14**(1) (2011) 2:1–2:28
5. Park, J.S., Giordano, J.: Role-based profile analysis for scalable and accurate insider-anomaly detection. In: Proceedings of International Conference on Performance, Computing, and Communications, IEEE (2006) 7–pp
6. Maloof, M., Stephens, G.: Elicit: A system for detecting insiders who violate need-to-know. In: Recent Advances in Intrusion Detection, Springer (2007) 146–166
7. Legg, P.A., Buckley, O., Goldsmith, M., Creese, S.: Caught in the act of an insider attack: Detection and assessment of insider threat. In: Proceedings of International Symposium on Technologies for Homeland Security, IEEE (2015) 1–6
8. Alizadeh, M., Peters, S., Etalle, S., Zannone, N.: Behavior Analysis in the Medical Sector: Theory and Practice. In: Proceedings of Symposium on Applied Computing, ACM (2018)
9. Hu, N., Bradford, P.G., Liu, J.: Applying role based access control and genetic algorithms to insider threat detection. In: Proceedings of the Annual Southeast Regional Conference, ACM (2006) 790–791
10. Costante, E., Fauri, D., Etalle, S., den Hartog, J., Zannone, N.: A hybrid framework for data loss prevention and detection. In: Proceedings of IEEE Security and Privacy Workshops, IEEE (2016) 324–333
11. Frias-Martinez, V., Sherrick, J., Stolfo, S.J., Keromytis, A.D.: A network access control mechanism based on behavior profiles. In: Proceedings of Annual Computer Security Applications Conference, IEEE (2009) 3–12
12. Hummer, M., Kunz, M., Netter, M., Fuchs, L., Pernul, G.: Adaptive identity and access management contextual data based policies. EURASIP Journal on Information Security **2016**(1) (2016) 19
13. Margheri, A., Masi, M., Pugliese, R., Tiezzi, F.: A Rigorous Framework for Specification, Analysis and Enforcement of Access Control Policies. IEEE Transactions on Software Engineering (2017) Preprint at `doi:10.1109/TSE.2017.2765640`.
14. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
15. Tax, D.M.J.: One-class classification: Concept-learning in the absence of counter-examples. PhD thesis, University of Delft (2001)
16. Quinlan, J.R.: Generating production rules from decision trees. In: Proceedings of International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc. (1987) 304–307
17. Andrzejak, A., Langner, F., Zabala, S.: Interpretable models from distributed data via merging of decision trees. In: Proceedings of Symposium on Computational Intelligence and Data Mining, IEEE (2013) 1–9
18. Spinosa, E.J., de Leon, F., Ponce, A., Gama, J.: Novelty detection with application to data streams. Intelligent Data Analysis **13**(3) (2009) 405–422
19. Nellikar, S., Nicol, D.M., Choi, J.J.: Role-based differentiation for insider detection algorithms. In: Proceedings of Workshop on Insider Threats, ACM (2010) 55–62
20. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.: Adversarial machine learning. In: Proceedings of Workshop on Security and Artificial Intelligence, ACM (2011) 43–58