# Towards ABAC Policy Mining from Logs with Deep Learning

Decebal Constantin Mocanu
Eindhoven University of Technology
Dep. Of Electrical Engineering
Eindhoven, the Netherlands
d.c.mocanu@tue.nl

Fatih Turkmen
Eindhoven University of Technology
Dep. Of Mathematics and Computer Science
Eindhoven, the Netherlands
f.turkmen@tue.nl

Antonio Liotta
Eindhoven University of Technology
Dep. Of Electrical Engineering
Eindhoven, the Netherlands
a.liotta@tue.nl

## ABSTRACT

Protection of sensitive information in platforms such as the ones offered by smart cities requires careful enforcement of access control rules that denote "who can/cannot access to what under which circumstances". In this paper, we propose our ongoing work on the development of a deep learning technique to infer policies from logs. Our proposal improves the state-of-the-art by supporting negative authorizations (i.e. denied access requests) and different types of noise in logs. A preliminary evaluation of the proposed technique is also presented in the paper.

## Categories and Subject Descriptors

Computing methodologies [**Machine learning approaches**]: [Neural networks]; Security and privacy [**Security services**]: [Access control]

## General Terms

Theory, Algorithms, Security

## Keywords

Deep Learning, Boltzmann Machines, Density Estimation, Attribute-Based Access Control, ABAC Policy Mining

## 1. INTRODUCTION

Smart urban systems such as ACCUS with tightly integrated components will collect and disseminate large amounts of sensitive information from/to citizens, government agencies and commercial/non-commercial organizations. While they will fundamentally improve citizen's life, contribute to preservation of environment and enable sustainability, they will also expose certain security vulnerabilities. Protecting sensitive information from unauthorized access in this context is one of the major obstacles in obtaining full benefit from their deployments. Example consequences of data leakage include financial loss, reputation damage or even social unrest. Authorizations systems ensure that access to sensitive information is strictly regulated through security policies which encode rules on "who can/cannot access to what under which circumstances".

There are different languages to specify security policies and these languages provide different constructs based on underlying access control model when modeling security requirements. In other words, the selection of the policy language and thus the model determine expressiveness in encoding rules and simplicity in administration. Among various models, attribute-based access control (ABAC) [3] has been shown to provide very expressive constructs and various tools have been developed to assist policy specifications with them [14, 13]. In ABAC, access rules are specified by using user and resource attributes. For instance, the "role" of a user in an organization or the "type" of a sensitive resource can be used to identify sets of users/resources in a single rule. The use of attributes not only allows compact encoding of permission assignment to users but also maintains readability.

In order to assist policy administrators when specifying ABAC policies, a particularly useful approach is to infer access control rules from existing logs. Among other information, these logs contain tuples denoting user, sensitive resource, the exercised right and a time-stamp. They may contain access logs that should have not have happened (i.e. under-assignments) in the case of an error in the enforcement or may contain only partial information about the permissions (i.e. over-assignments). To our knowledge, [15] is the first work that discusses policy mining in the context of ABAC. It presents a custom mining algorithm (referred as Xu-Stoller) and its extensive evaluation with both realistic and synthetic policies. However, the approach has two major limitations. On one side it considers only positive authorizations ("who can access to what") whereas many ABAC languages allow also negative authorizations ("who cannot access to what"). The logs may contain denied access requests as well due to auditing purposes which are negative examples in the min-

**Pre-printed version.**
Please cite as: *D.C. Mocanu, F. Turkmen, and A. Liotta: Towards ABAC Policy Mining from Logs with Deep Learning. In proc. of the 18th International Multiconference, IS 2015, Intelligent Systems, Ljubljana, Slovenia.*
http://www.projectaccus.eu/

ing process. On the other side, it provides no support for under-assignment (i.e. noise) case and only partial support for the over-assignment case.

To overcome the aforementioned limitations, this paper is an early report on how deep learning performs for ABAC policy mining in the case of only positive authorizations. The approach has two phases. The first phase consists in generalizing the knowledge from the logs yielding a good set of candidates rules in a binary vector format. The idea is to obtain insight about certain parameters that are significant in obtaining good rules. In the second phase the target will be to transform the set of candidate rules from the binary vector format to the format acceptable by Xu-Stoller and compare them. In this paper, we focus on the first phase, and we make use of the excellent capabilities of Restricted Boltzmann Machines (RBMs) as density estimators to propose a technique that is capable to produce a set of suitable candidate rules based on the knowledge extracted by the RBMs from the processed logs. More exactly, in our previous work we showed that the reconstruction error (i.e. the difference between a data point and its *reconstruction* made by the RBM model) of various type of RBMs may be used as an excellent similarity measure to find the closest clusters of Markov Decision Processes (MDPs) and we demonstrate the advantage of using it in transfer learning [1], or to assess in an objective manner the quality of impaired images [7, 9] (i.e. to estimate automatically how humans would perceive the degradation level of impaired images in comparison with their unimpaired version). Herein, we show that by using the generative power of RBMs and the previous mentioned reconstruction error, we may generalize and create a set of good candidates rules from a small amount of ABAC logs.

The remaining of this paper is organized as follows. Section 2 provides the problem definition and insights on ABAC policy mining for the benefits of the non-specialist reader, while Section 3 details our proposed method. Section 4 shows a preliminary evaluation of our approach, while Section 5 concludes the paper and presents further research directions.

## 2. PROBLEM DEFINITION

Given a set of users ($U$), resources ($R$) and operations ($O$) that users can perform on resources, an attribute-based access control system contains two types of attributes $A_u$ and $A_r$ for users and resources respectively. An attribute $a$ of a user or resource $x$ may have an empty value (denoted $\perp$) or a set of values from its domain $D_a$ and this value is denoted by the attribute assignment relation $d(x, a)$.

An ABAC rule $\langle e, o\rangle$ specifies an expression that determines its applicability and an operation. For instance, a rule

$$\langle role \in \{nurse, doctor\} \wedge resource \in \{PatRec\}, read\rangle$$

imposes that only nurses or doctors can read a patient record. For simplicity of presentation, an expression is considered to be a total mapping, $e : A \mapsto \{2^{D_A} \setminus \{\perp\}, \top\}$, that maps each attribute $a$ to either a subset of values from its domain ($D_a$) or to a special value $\top$. The value $\top$ means that the expression does not specify any constraint on the value of the attribute. If the triple $\langle u, r, o\rangle$ denotes a request by user $u$ to perform operation $o$ on resource $r$ and $e$ is an expression then for each attribute $a$ in $A_u$ or $A_r$ either $e(a) = \top$ or $(d(u, a) \subseteq e(a) \wedge d(r, a) \subseteq e(a))$ holds in order for $e$ to be satisfied.

We only consider positive authorizations in this paper and the order of rules in the policy does not matter.

An ABAC policy is a sequence of rules $\{r_1, \ldots, r_n\}$ that encodes authorizations of users over resources. The access control system receives a request, evaluates it against the policy and logs the permitted requests with a time stamp. Thus the problem we consider in this paper is formulated as follows:

Definition 2.1 (ABAC Policy Mining Problem). *Given a log $L$ where each entry is of the form $\langle u, r, o, t\rangle$ denoting the fact that user $u$ performed $o$ on resource $r$ at time $t$, users $U$, resources $R$, operations $O$, attributes $A$ and attribute assignment relation $d$, an ABAC policy mining problem amount to finding a policy that maximizes a policy quality metric.*

There are various policy quality metrics that help to compare policies. An example metric, which is also considered in our work, is weighted structured complexity (WSC) [10] where the expression and the operation in a rule are given weights. More specifically, the complexity of a rule $\langle e, o\rangle$ is calculated as $WSC(e) + WSC(o)$ where the WSC of an expression or an operation is given as the number of atomic elements contained in them. The overall WSC of a policy is the sum of WSCs of all its elements.

*Noise and Incompleteness in the Log.* There are certain types of problems that a policy mining approach must handle. A log $L$ may contain permissions that are not supposed to be granted (under-assignment). Another problem is the completeness of the log. A log may lack certain permissions that have not been exercised (over-assignments).

## 3. APPROACH

In this section, firstly, we present the mathematical details of RBMs, and secondly, we describe in details the proposed approach.

### 3.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines(RBMs) [11] are energy-based models capable to perform unsupervised learning. These models are stochastic with stochastic nodes and layers, making them less vulnerable to local minima [12]. Furthermore, due to their architecture and neural configurations, RBMs and their variants posses excellent generalization capabilities [2, 8].

Formally, an RBM has a visible binary layer $\mathbf{v} = [v_1, .., v_{n_v}]$, where $\forall i, v_i \in \{0, 1\}$ and $v_{n_v}$ represents the total number of visible neurons and a hidden binary layer $\mathbf{h} = [h_1, .., h_{n_h}]$, where $\forall j, h_j \in \{0, 1\}$ and $v_{n_h}$ is the total number of hidden neurons, as shown in Fig. 1. The visible layer encodes the data directly, while the hidden one increases the learning capacity by enlarging the class of distributions that can

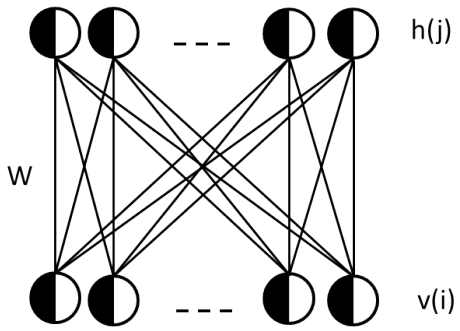



Figure 1: RBM general architecture, where $\mathbf{v}(i)$, $\mathbf{h}(j)$, and $\mathbf{W}$ represent the visible neurons, hidden neurons and the undirected connections between the neurons from different layers, respectively.

be represented to an arbitrary complexity. The total energy (i.e. a cost function over all neurons and connections between them) of an RBM is given by equation 1.

$$E(v,h) = -\sum_{i,j} v_i h_j W_{ij} - \sum_i v_i a_i - \sum_j h_j b_j \qquad (1)$$

where $W_{ij}$ denotes the connection between visible neuron $i$ and hidden neuron $j$, $a_i$ is the bias for visible neuron $i$ and $b_j$ is the bias for hidden neuron $j$. The term $\sum_{i,j} v_i h_j W_{ij}$ represents the total energy between neurons from different layers, $\sum_i v_i a_i$ represents the energy of the visible layer and $\sum_j h_j b_j$ the energy of the hidden layer. The inference for the hidden neuron $j$ is done by sampling from a sigmoid function, $p(h_j = 1|\mathbf{v}, \mathbf{W}, \mathbf{b}) = sigm(b_j + \sum_i v_i W_{ij})$. The inference for the visible unit $i$ is done by sampling also from a sigmoid function, $p(v_i = 1|\mathbf{h}, \mathbf{W}, \mathbf{a}) = sigm(a_i + \sum_j h_j W_{ij})$.

In order to maximize the likelihood of the model, the gradients of the energy function with respect to the parameters (i.e. weights, biases) have to be calculated. Unfortunately, in all types of RBMs the maximum likelihood can not be straightforwardly applied due to intractability problems. As a solution to these problems, Contrastive Divergence (CD) algorithm to train RBMs, was introduced by Geoffrey Hinton in [6].

In Contrastive Divergence, learning follows the gradient of:

$$CD_n \propto D_{KL}(p_0(\mathbf{x})||p_\infty(\mathbf{x})) - D_{KL}(p_n(\mathbf{x})||p_\infty(\mathbf{x})) \qquad (2)$$

where, $p_n(.)$ is the resulting distribution of a Markov chain running for $n$ steps, and $D_{KL}(.||.)$ represents the Kullback-Leibler divergence. To find the update rules for the parameters of RBMs we have to calculate the derivatives of the energy function from equation 1 with respect to those parameters (i.e. $\mathbf{W}$, $\mathbf{a}$ and $\mathbf{b}$). Since the visible units are conditionally independent given the hidden units and vice versa, learning can be performed using one step Gibbs sampling, which practically has two half-steps: (1) update all the hidden units, and (2) update all the visible units. Thus, in $CD_n$ the weight updates are done as follows: $W_{ij}^{\tau+1} = W_{ij}^{\tau} + \alpha\left(\left\langle\langle h_j v_i\rangle_{p(\mathbf{h}|\mathbf{v};\mathbf{W})}\right\rangle_0 - \langle h_j v_i\rangle_n\right)$ where $\tau$ is the training epoch, $\alpha$ is the learning rate, and the subscript $_{(n)}$ indicates that the states are obtained after $n$ iterations of Gibbs sampling from the Markov chain starting at $p_0(\cdot)$. For a more comprehensive discussion about RBM and CD, the interested reader is referred to [2].

## 3.2 Policy mining procedure

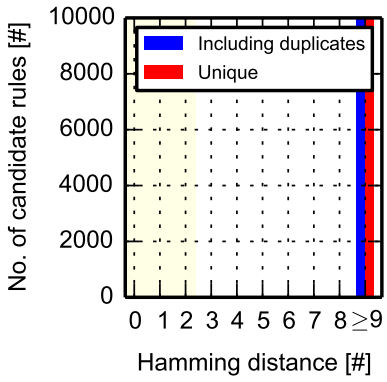
```
1  %% initialization;
2  Transform logs L in binary logs L^b;
3  Initialize RBM.W, RBM.a, RBM.b ← N(0, 0.01), learning rate;
4  Train RBM on L^b;
5  %% get the set of hidden configurations H^{L^b} from the binary logs;
6  %% find the maximum reconstruction error mHD on the binary logs;
7  Initialize the set H^{L^b} empty;
8  mHD=0;
9  for each l ∈ L^b do
10     RBM.v=l;
11     RBM.h=RBM.inferHiddenLayer(RBM.v, RBM.W, RBM.a, RBM.b);
12     Add RBM.h to H^{L^b};
13     RBM.v=RBM.inferVisibleLayer(RBM.h, RBM.W, RBM.a, RBM.b);
14     mHD=max(mHD,computeHammingDist(l,RBM.v));
15 end
16 %% generate the set of good candidate rules R^b with the RBM;
17 Initialize the set of candidate rules R^b empty;
18 for all trials do
19     select h randomly from H^{L^b};
20     RBM.h=h+N(0, σ); % σ = 0.7 seems to be a good choice (Section 4);
21     RBM.v=RBM.inferVisibleLayer(RBM.h, RBM.W, RBM.a, RBM.b);
22     possibleRule=RBM.v;
23     RBM.h=RBM.inferHiddenLayer(RBM.v, RBM.W, RBM.a, RBM.b);
24     RBM.v=RBM.inferVisibleLayer(RBM.h, RBM.W, RBM.a, RBM.b);
25     if (possibleRule ∉ R^b) then
26         if (computeHammingDist(possibleRule,RBM.v)≤mHD) then
27             Add possibleRule to R^b
28         end
29     end
30 end
```

**Algorithm 1:** High level pseudo-code to generate the set of good candidate rules. The rules are generated from the true probabilities of the hidden neurons (inferred from the binary logs $L^b$) altered by a Gaussian noise $\mathcal{N}(0, \sigma)$.

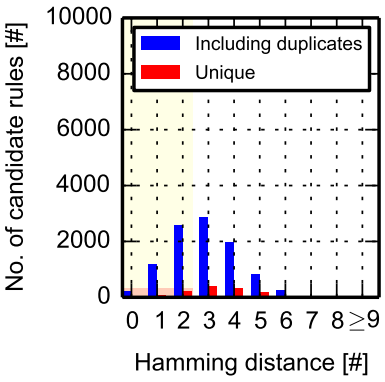
To obtain the generalized set of the ABAC candidate rules, the first step is to represent each entry of the logs $L$ in a binary vector $\mathbf{l}$ to be easy understandable by RBMs. Let $L^b$ denote this transformed set of binary logs. The elements of each binary vector obtained $\mathbf{l} \in L^b$ will have the following unveiled meaning $[l^1_{A^1_u}, .., l^{|D^1_u|}_{A^1_u}, l^1_{A^2_u}, .., l^{|D^2_u|}_{A^2_u}, .., l^1_{A^{|A_u|}_u}, .., l^{|D^{|A_u|}_u|}_{A^{|A_u|}_u}, l^1_{A^1_r}, .., l^{|D^1_r|}_{A^1_r}, l^1_{A^2_r}, .., l^{|D^2_r|}_{A^2_r}, .., l^1_{A^{|A_r|}_r}, .., l^{|D^{|A_r|}_r|}_{A^{|A_r|}_r}, l^1_O, .., l^{|O|}_O]$, where $l^1_{A^i_u}, .., l^{|D^i_u|}_{A^i_u}$, $\forall i$, represent each discrete value from the domain ($D^i_u$) of user attribute $A^i_u$, and $|A_u|$ is the total number of user attributes, $l^1_{A^i_r}, .., l^{|D^i_r|}_{A^i_r}, \forall i$, represent each discrete value from the domain ($D^i_r$) the resource attribute $A^i_r$, and $|A_r|$ is the total number of resource attributes, while $l^i_O, \forall i$, represent all the possible discrete values of the operations set and $|O|$ is the total amount of operations. Furthermore, for each binary vector $\mathbf{l} \in L^b$ each element $l_i \in \mathbf{l}, \forall i$ is set to 1 if the original entry in the logs $L$ contains the corresponding attribute (or operation) value and it is set to 0 otherwise.

The second step consists in generating good candidate rules using RBMs capabilities, as briefly explained in Alg. 1 and detailed below. Firstly, the set of binary logs is used to train an RBM (Alg. 1, lines 3-4). As a consequence, the trained RBM will learn to incorporate well the hidden distribution of the logs. The straight forward solution to obtain the set of good candidates rules from the trained RBM would be: (1) compute the maximum reconstruction error obtained by any of the binary logs $L^b$ passed through the

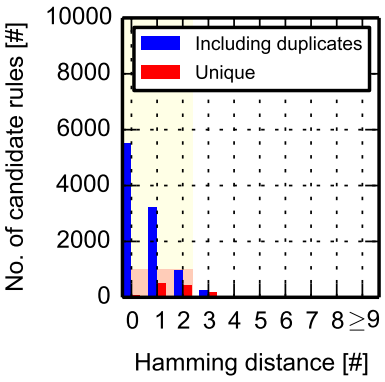
For any data point the reconstruction error is represented by the difference between the data point under scrutiny and



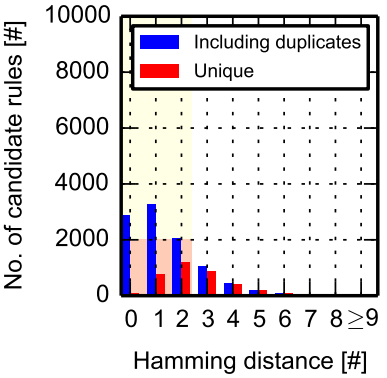

(a) Randomly generated candidate rules.



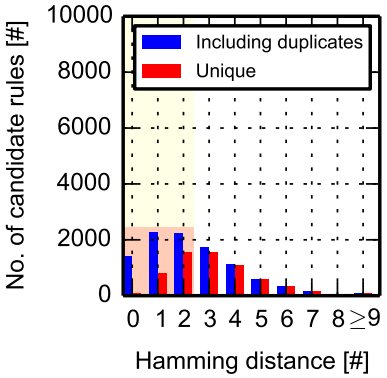

(b) Candidate rules generated directly from random generated probabilities of the hidden neurons.



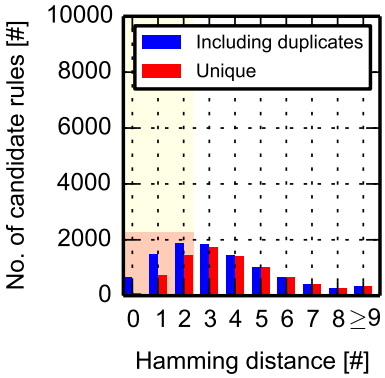

(c) Candidate rules generated from the hidden neurons set to the true probabilities of the hidden layer inferred from randomly chosen binary logs $L^b$ plus noise $\mathcal{N}(0, 0.3)$.




(d) Candidate rules generated from the hidden neurons set to the true probabilities of the hidden layer inferred from randomly chosen binary logs $L^b$ plus noise $\mathcal{N}(0, 0.5)$.




(e) Candidate rules generated from the hidden neurons set to the true probabilities of the hidden layer inferred from randomly chosen binary logs $L^b$ plus noise $\mathcal{N}(0, 0.7)$.




(f) Candidate rules generated from the hidden neurons set to the true probabilities of the hidden layer inferred from randomly chosen binary logs $L^b$ plus noise $\mathcal{N}(0, 0.9)$.

Figure 2: Histograms of rules generation successful rate. For each plot we made 10000 trials to generate rules. The light yellow area represents the target zone, while the light red area represents the number of unique good candidate rules for the method under scrutiny. The blue bar represents the total number of generated rules for which the reconstruction error is expressed by the x-axis, while the red bar shows the total number of unique generated rules for each possible value of the reconstruction error (i.e. x-axis). The highest number of good unique candidate rules is generated by the method from subplot (e).

trained RBM, and set it as a threshold (Alg. 1, lines 8-15); (2) pass all the possible combination of rules through the RBM and consider as good candidates rules those ones for which the reconstruction error is smaller or equal with the previous computed threshold. For the sake of brevity, for a better understanding on how the reconstruction error may reflect if some data points belong to the hidden distribution incorporated by an RBM type model, the interested reader is referred to [1, 7]. However, due to the fact that the total number of possible combination of rules is $2^{n_v}$ (i.e. exponential with the number of visible neurons), enumeration of all candidate rules does not represent a feasible solution. As an alternative, we propose to make use of the generative properties of RBMs and to sample rules from the trained RBM in a controlled manner (Alg. 1, lines 19-22). This yields to a reduced amount of rules drawn from a distribution close enough to the one incorporated in the trained RBM, and which can be assessed further-on using the reconstruction error procedure (Alg. 1, lines 23-29). To sample in a controlled manner from a trained RBM, one can start from a specific configuration of the hidden neurons to infer the values of the visible ones. Depending on how the configuration of the hidden neurons is chosen the amount of generate rules may differ, as we will show in Section 4.

the expected values of the visible neurons, inferred from the expected values of the hidden ones, which initially are inferred from the visible neurons set to the data point itself.

## 4. PRELIMINARY EVALUATION

In this section, we evaluate the effectiveness of the proposed approach to generate a set of good candidates rules. For this, we make use of the *healthcare* dataset from [15]. On this specific dataset, after we transformed the original logs in binary logs we obtained a vector of 46 binary values to cover all user attributes, resource attributes, and operations. Thus, in the RBM we set the number of visible neurons to 46, and the number of hidden neurons to 40 to ensure enough representational power. The learning rate was set to 0.001 and we trained the RBM model until it converges (i.e. 200 training epochs). After training, the RBM obtained was used to generate new candidates rules. In all experiments, we used the Hamming distance [5] to measure the reconstruction error.

The biggest reconstruction error obtained by passing any of the binary logs $\mathbf{l} \in L^b$ through the trained RBM was 2.

To assess which is the best controlled manner to generate rules with a high chance of belonging to the same distribution incorporated in the trained RBM, we have considered 6 slightly different scenarios, as depicted in Fig. 2. In one scenario we have generated rules completed randomly, while in the other 5 the rules were generated from the true probabilities of the hidden neurons (inferred from the binary logs $L^b$) altered by a Gaussian noise $\mathcal{N}(0, \sigma)$ with a scenario specific standard deviation, while keeping the constraint $0 \leq h_j \leq 1, \forall j$. For each scenario, we have made 10000 trials to generate rules. Because different trials may yield duplicate rules, the goal was to find the method which is capable to generate the highest number of unique rules which passed through the trained RBM have a reconstruction error (i.e. Hamming distance) smaller or equal with 2. As it is reflected in Fig. 2e, by setting the values of the hidden neurons in the generative phase to the true probabilities of the hidden layer, inferred with the trained RBM from randomly chosen binary logs, and adding them a random Gaussian noise with a standard deviation of 0.7, yields the highest amount of unique good rules (i.e. 2423). Remarkable, in the extreme case of generating rules completely randomly (i.e. Fig. 2a), none of them was successful.

# 5. CONCLUSIONS AND FUTURE WORK

In this paper, we reported our initial findings about the use of RBM for ABAC policy mining when there are only positive examples. Our overall idea is to exploit certain features (e.g. inductive bias [4]) offered by neural networks to overcome limitations of existing approaches to policy mining problem in particular when the logs contain negative authorizations or large amount of noise with varying levels of incompleteness. As further work, we intend to implement the second phase of our approach where we compare the prediction accuracy of our technique to Xu-Stoller according to different policy quality metrics. This will involve evaluations with different real-world policies and supporting more complex policies that contain expressions of different types.

# 6. ACKNOWLEDGMENTS

This work has been supported by ARTEMIS project ACCUS (Adaptive Cooperative Control in Urban sub-Systems), Grant agreement n. 333020 (http:/projectaccus.eu/).

# 7. REFERENCES

[1] H. B. Ammar, E. Eaton, M. E. Taylor, D. C. Mocanu, K. Driessens, G. Weiss, and K. Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[2] Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, Jan. 2009.

[3] O. X. T. Committee, 2013. eXtensible Access Control Markup Language (XACML).

[4] M. Craven and J. W. Shavlik. Using neural networks for data mining. *Future Generation Comp. Syst.*, 13(2-3):211–229, 1997.

[5] R. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[6] G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, Aug. 2002.

[7] D. Mocanu, G. Exarchakos, H. Ammar, and A. Liotta. Reduced reference image quality assessment via boltzmann machines. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 1278–1281, May 2015.

[8] D. C. Mocanu, H. Bou-Ammar, D. Lowet, K. Driessens, A. Liotta, G. Weiss, and K. Tuyls. Factored four way conditional restricted boltzmann machines for activity recognition. *Pattern Recognition Letters*, 2015.

[9] D. C. Mocanu, G. Exarchakos, and A. Liotta. Deep learning for objective quality assessment of 3d images. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 758–762, Oct 2014.

[10] I. Molloy, Y. Park, and S. Chari. Generative models for access control policies: applications to role mining over logs with attribution. In *17th Symposium on Access Control Models and Technologies (SACMAT)*, pages 45–56, 2012.

[11] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, 1987.

[12] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research*, 12:1025–1068, 2011.

[13] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone. Analysis of XACML policies with SMT. In *4th International Conference on Principles of Security and Trust - (POST) Proceedings*, pages 115–134, 2015.

[14] F. Turkmen, S. N. Foley, B. O'Sullivan, W. M. Fitzgerald, T. Hadzic, S. Basagiannis, and M. Boubekeur. Explanations and relaxations for policy conflicts in physical access control. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013*, pages 330–336, 2013.

[15] Z. Xu and S. D. Stoller. Mining attribute-based access control policies from logs. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 276–291, 2014.