7 EVALUATION RESULTS

This section presents the results of experiments comparing our algorithms with Bui et al.'s FS-SEA* algorithm [7] and Iyer et al.'s algorithm [21]. DTRM and DTRM$^-$ are implemented in Python, except that phase 2 step 2 (merge and simplify rules) uses the Java code from Bui et al.'s implementation of FS-SEA*, available at [30]. Experiments were run on Windows 10 on an Intel i7-6770HQ CPU. In summary, we find that: (1) compared with FS-SEA*, our algorithms are comparably effective at discovering the desired ReBAC rules, and are significantly faster, with the speedup exceeding 10× for several datasets and generally increasing with policy size, hence expected be even larger for the large datasets arising in practice; and (2) compared with Iyer et al.'s algorithm, our algorithms are several times faster, and produce policies that are the same size or smaller (fewer rules) and more similar to the original policies.

7.1 Comparison with FS-SEA*

We compared DTRM and DTRM$^-$ with FS-SEA* using the datasets described in Section 6.1. We use the same path length limits (*cf.* Section 5.1) as in [7, 10]. For the case studies, we generated policies with varying size (of the object model): $N = 10, 15, 20, 25, 30, 35$ for eWorkforce and $N = 75, 100, 125, 150, 175$ for e-document. For each size, we generated 5 pseudo-random object models. For synthetic policies, we generated two families of policies. Synthetic policies are designated by syn_N_M, where N is the object model size parameter, and M is the number of rules. The first family consists of 5 sets of $M = 20$ synthetic rules, and object models with sizes $N = 20, 25, 30$ (one of each size); we chose $M = 20$ because it is the average number of rules in the sample policies and case studies. The second family consists of sets of $M = 10, 30, 40$ synthetic rules (one of each size), and 5 object models with size $N = 20$. We ran DTRM, DTRM$^-$, and FS-SEA* on all of them, and average the results for the five policies with the same N and M. The standard deviations are reasonable, indicating that averaging over 5 object models for each data point is sufficient to obtain meaningful results.

7.1.1 Policy Similarity and WSC. All three algorithms always mine policies that grant exactly the same authorizations as the input ACL policies and thus achieve perfect *semantic similarity* for all datasets.

All algorithms achieve similar *syntactic similarity* when comparing mined rules with simplified original rules, as explained in Section 6. The minimum, median, and maximum (over all datasets) syntactic similarity achieved by each algorithm are: 0.91, 0.98, 1.0 for FS-SEA*; 0.90, 0.98, 1.0 for DRTM; and 0.90, 0.97, 1.0 for DTRM$^-$. The syntactic similarity achieved by DTRM and DTRM- are usually the same or better than that achieved by FS-SEA*, and in the worst cases in Table 1, are at most 2% and 4% lower, respectively. DTRM$^-$ achieved slightly lower syntactic similarity since the input policies do not use any negative atomic condition/constraint.

We report results for WSC in terms of the ratio of the WSC of the policy mined by DTRM or DTRM$^-$ to the WSC of the policy mined by FS-SEA*; thus a ratio below 1 means that DTRM or DTRM$^-$ produce a more concise policy than FS-SEA*. The minimum, median, and maximum (over all datasets) of this ratio are: 0.79, 1.0, 1.21 for DRTM, and 0.86, 1.01, 1.32 for DTRM$^-$. WSC of policies mined by DTRM$^-$ is not smaller than WSC of policies mined by DTRM, even though theoretically negation could allow more concise policies. This indicates that DTRM$^-$ sometimes produces policies that use negation even when it is not beneficial. This is not surprising, because when constructing the decision tree, the algorithm does not have a preference for using or avoiding negation.

Detailed results for policy similarity appear in Table 1. Detailed results for WSC appear in [6]. We conclude that all three algorithms produce policies with similar quality according to all three metrics.

7.1.2 Running Time. We report results for running time as the speedup relative to FS-SEA*, i.e., the ratio of the running time of each algorithm to the running time of FS-SEA*. Detailed results appear in Table 1. The results are summarized in the stacked bar chart in Figure 3. Each bar has three segments, representing three overlaid bars, each corresponding to an algorithm. The total height (as measured on the y-axis) of the top of each segment is the speedup of that algorithm. The first (black) segment is for FS-SEA*, so it always has height 1. The second (white) segment is for DTRM. The third (shaded) segment is for DTRM$^-$. For example, if DTRM achieved speedup 2.2 and DTRM$^-$ achieved speedup 4.4 for some policy, then the top of the black segment would be at height 1, the top of the white segment at height 2.2 (hence the white segment would be 1.2 units long), and the top of the shaded segment at height 4.4. This stacked bar chart format is suitable for reporting the speedups because, in all of our experiments, DTRM$^-$ is faster than DTRM, and DTRM is faster than FS-SEA*. The bars within each cluster other than the sample policy cluster are ordered left-to-right by increasing policy size, specifically by object model size for the e-doc., eWorkforce, and syn clusters, and by number of rules for the syn_20 cluster. Observe that speedup generally increases from left to right within those clusters, i.e., generally increases with policy size. A main reason that speedups for e-document and synthetic policies are larger than for eWorkforce and most sample policies is that the former policies have a larger number of rules per $\langle C_s, C_r, a\rangle$ tuple (explained in Section 5.1). DTRM (and DTRM$^-$) achieve larger speedups for such policies, because FS-SEA* repeats its expensive processing (feature selection and evolutionary search) for each generated rule, while DTRM performs its expensive processing (tree construction) once per $\langle C_s, C_r, a\rangle$ tuple and can quickly extract multiple rules from a tree.

Experiments with Sample Policies. DTRM and DTRM$^-$ spend most of the time in phase 1 to learn decision trees. The averaged running times spent on phase 2 are less than 1 second for EMR_15 and projectmangagment_5, and are less than 3 seconds for healthcare_5 and university_5. DTRM and DTRM$^-$ are faster than FS-SEA* on all of these policies. The average speedup is 2.17 for DTRM and 2.38 for DTRM$^-$. DTRM has similar running time as DTRM$^-$ on the sample policies, since only a few negative features are generated when learning decision trees for these policies, and they are not useful and hence are removed in the "*merge and simplify rules*" phase, so the negative feature elimination step in DTRM has no work to do.

Experiments with Case Study Policies. For eWorkforce, the average speedup is 1.70 for DTRM and 1.96 for DTRM$^-$. The negative feature elimination step in DTRM has little effect on the speedup, since the decision trees generated from the first phase do not contain many negative features. For e-document, the average speedup is 2.92 for DTRM and is 8.13 for DTRM$^-$, and the speedup for DTRM$^-$ increases with policy size. The difference in speedup is larger for

Policy	Syntactic Similarity						Running Time (sec)							
	FS-SEA*		DTRM		DTRM$^-$		FS-SEA*		DTRM		SpdUp	DTRM$^-$		SpdUp
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ		μ	σ	
EMR_15	0.99	0.01	0.99	0.01	0.99	0.01	96	7.37	56	0.30	1.70	53	5.55	1.82
healthcare_5	1.00	0.00	1.00	0.00	1.00	0.00	111	14.54	80	41.07	1.39	70	29.68	1.57
project-mgmt._5	1.00	0.00	1.00	0.00	1.00	0.00	6	0.45	2	1.62	3.88	2	0.39	4.07
university_5	1.00	0.00	1.00	0.00	1.00	0.00	271	21.98	159	64.02	1.70	131	44.32	2.07
e-doc._75	0.93	0.02	0.90	0.01	0.90	0.01	696	133.88	296	42.57	2.35	121	14.90	5.75
e-doc._100	0.94	0.01	0.91	0.03	0.90	0.02	1734	542.88	650	64.94	2.67	250	19.14	6.94
e-doc._125	0.93	0.01	0.92	0.01	0.93	0.02	3516	1415.93	1200	276.13	2.93	481	24.52	7.31
e-doc._150	0.91	0.01	0.94	0.01	0.94	0.00	6068	1202.25	2292	323.12	2.65	735	58.44	8.25
e-doc._175	0.92	0.01	0.93	0.01	0.93	0.01	15218	4535.45	3823	460.36	3.98	1227	68.07	12.41
eWorkforce_10	0.97	0.01	0.98	0.01	0.97	0.01	70	9.32	50	5.80	1.41	48	0.69	1.47
eWorkforce_15	0.95	0.02	0.98	0.02	0.97	0.02	287	37.68	182	13.62	1.58	176	2.06	1.63
eWorkforce_20	0.92	0.03	0.98	0.02	0.96	0.02	669	89.99	426	94.17	1.57	319	5.16	2.10
eWorkforce_25	0.95	0.04	0.97	0.02	0.95	0.02	1750	294.25	946	280.87	1.85	745	10.67	2.35
eWorkforce_30	0.97	0.02	0.97	0.02	0.95	0.02	3113	725.28	1492	312.45	2.09	1378	17.42	2.26
syn_20_10	0.99	0.00	0.99	0.01	0.99	0.01	938	348.24	166	52.16	5.66	142	12.42	6.61
syn_20_20	0.98	0.01	0.98	0.02	0.97	0.04	3129	887.18	309	88.93	10.11	256	24.97	12.22
syn_20_30	0.99	0.00	0.99	0.01	0.98	0.03	6303	1258.03	379	88.73	16.65	317	27.25	19.86
syn_20_40	0.99	0.01	0.98	0.02	0.97	0.03	11169	2812.94	435	69.75	25.67	370	14.52	30.21
syn_25_20	1.00	0.00	0.98	0.02	0.97	0.04	6494	2283.31	571	142.31	11.38	485	42.43	13.40
syn_30_20	1.00	0.00	0.99	0.01	0.99	0.01	11161	3396.60	898	82.72	12.43	861	75.63	12.96
syn_35_20	0.99	0.01	0.99	0.01	0.99	0.01	21758	7355.68	1419	138.17	15.33	1416	114.47	15.36

Table 1: Comparison of DTRM, DTRM$^-$, and FS-SEA*. μ and σ are the mean and standard deviation, respectively. SpdUp is the speed up, computed as the ratio of the running time of each of our algorithms to the running time of FS-SEA*.

e-document, because more negative features are generated in phase 1, so the negative feature elimination step in DTRM takes longer.

DTRM and DTRM$^-$ have lower average speedups on sample policies and eWorkforce, compared with the other policies (discussed next), because these policies are simpler, allowing FS-SEA* to have relatively good running time on them. In particular, FS-SEA* needs only one or a few iterations of feature selection and evolution to learn the rules for a given combination of subject type, resource type, and action, whereas for the more complicated policies, FS-SEA* typically needs more such iterations.

Experiments with Synthetic Policies. In experiments with the first family of synthetic policies, with $M = 20$ rules and varying object model size, the average speedup is 12.31 for DTRM and 13.49 for DTRM$^-$. For both DTRM and DTRM$^-$, the speedup generally increases with object model size; the 3% dip from syn_25_20 to syn_30_20 is not statistically significant (it's less than the σ).

In experiments with the second family of synthetic policies, with object model size $N = 20$ and varying number of rules, the average speedup is 17.48 for DTRM and 20.76 for DTRM$^-$. The speedups of both DTRM and DTRM$^-$ significantly increase with the number of rules: for DTRM, speedup increases from 5.66 with 10 rules to 25.67 with 40 rules; for DTRM$^-$, speedup increases from 6.61 with 10 rules to 30.21 with 40 rules.

7.2 Comparison with Iyer et al.'s Algorithm

We compare DTRM and DTRM$^-$ with Iyer et al.'s ReBAC mining algorithm [21] using modified versions of the eWorkforce datasets described in Section 7.1. We use Iyer et al.'s translation of a subset of the eWorkforce rules (used in experiments in [21]) as a starting point, and update it retain more of the original ORAL2 rules. We also modify the ORAL2 rules to exactly match (in meaning and structure, not syntax) the translated rules. We end up with 17 rules in each framework. Note that the original eWorkforce rules cannot be used directly: they need to be simplified, because Iyer et al.'s framework in [21] is less expressive than ORAL2. In particular, we eliminate Boolean attributes, and set comparison operators other than equality. We also simplify the object models in the eWorkforce_10 dataset by eliminating fields and classes not used in the modified rules. We implemented a translator that converts the simplified object models into Iyer et al.'s "system graph" representation. This enables us to run their system on significantly larger system graphs than used in any of the experiments (with any policy, not just eWorkforce) in [21]. We then compare the results of running our algorithms and their implementation of their algorithms.

When run on the modified eWorkforce_10 dataset, their system does not finish in a reasonable time (we used a timeout of 30+ minutes, since DTRM and DTRM$^-$ take less than a minute for this dataset) for some object models, and it returns errors, such as "MemoryError" and "IndexError: pop from empty list", for others. We reported these

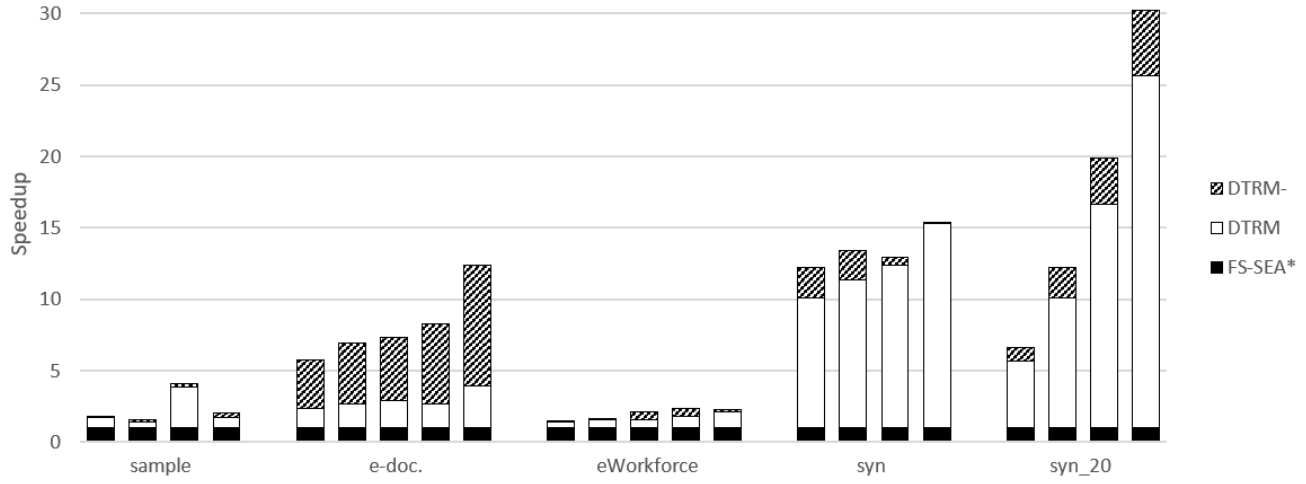

Figure 3: Speedups of DTRM and DTRM⁻ relative to FS-SEA*. There are 5 clusters, corresponding to 5 groups of policies. The "sample" cluster contains bars for the following policies (from left to right): EMR_15, healthcare_5, project-management_5 and university_5; "e-doc" cluster for e-document_75, e-document_100, e-document_125, e-document_150, e-document_175; "eWorkforce" cluster for eWorkforce_10, eWorkforce_15, eWorkforce_20, eWorkforce_25, eWorkforce_30; "syn" cluster for syn_20_20, syn_25_20, syn_30_20, syn_35_20; "syn_20" cluster for syn_20_10, syn_20_20, syn_20_30, syn_20_40.

Policy	Input Policies				Avg. # of Mined Rules			Avg. Running Time (sec)				
	#obj	#field	#FtVec	#rules	[21]	DTRM	DTRM⁻	[21]	DTRM	SpdUp	DTRM⁻	SpdUp
eWorkforce_10	354	530	8662	7	8	7	7	14	3	4.67	3	4.67
eWorkforce_15	505	751	20170	7	8	7	7	34	10	3.40	10	3.40
eWorkforce_20	601	897	29158	7	8	7	7	78	19	4.11	19	4.11
eWorkforce_25	755	1121	48253	7	8	7	7	101	42	2.40	41	2.46
eWorkforce_30	888	1304	67653	7	8.3	7	7	346	76	4.55	74	4.68

Table 2: Comparison of DTRM, DTRM⁻ and Iyer et al.'s algorithm on the simplified eWorkforce_10 dataset. #obj, #field, #FtVec and #rules have the same meanings as in Figure 2. SpdUp is the speedup of DTRM and DTRM⁻ relative to Iyer et al.'s algorithm.

issues to Iyer et al. Until they provide a fix, we circumvented these issues by removing the rules that trigger these issues, and removing parts of the object models unused by the remaining rules, until their system ran successfully for the remaining rules and at least one of the simplified object models for each object model size. In the end, we removed 10 rules that their system has trouble with, leaving 7 rules. The majority of the problematic rules are syntactically more complicated than the remaining ones. Specifically, 8 out of 10 of the problematic rules contain more than two atomic conditions/constraints (in ORAL2) or relationship patterns (in [21]'s policy language). In contrast, most (specifically, 5 out of 7) of the remaining rules contain only one atomic condition/constraint or relationship pattern (the other two remaining rules contain 3 atomic conditions/constraints). Results of these experiments are reported in Table 2. We set the path length limits for DTRM and DTRM⁻ to smaller values suitable for these simplified policies: MCSE = 5, MSPL = 2, MRPL = 1, SPED = 0, RPED = 0, and MTPL = 4. Even using these 7 remaining rules and significantly simplified object models, their system does not finish in a reasonable time (30 minutes) for some of the 5 object models for each policy size. Although we do not know for certain whether this is due to inefficiency of their algorithm or bugs in their implementation, we make the more generous assumption (i.e., assume the latter) and therefore omit those object models from the reported results. Consequently, the results in Table 2 are averages over 4 object models for eWorkforce_10, 1 for eWorkforce_15, 1 for eWorkforce_20, 2 for eWorkforce_25, and 3 for eWorkforce_30.

All three algorithms mine policies that grant the same authorizations as the input policies. For DTRM, the mined policies are identical to the input policies. For DTRM⁻, the mined policies are almost identical to the input policies: the only difference is replacement of the condition tenant.id = PP in one input rule with the negative condition tenant.id ≠ Telco, which is equivalent in context of these simplified object models. For Iyer et al.'s algorithm, the mined policy contain one more rule than the original policy (8 instead of 7) for all object models, except it contains two more rules for one object model of eWorkforce_30, because their algorithm fails to mine some of the

desired relationship patterns, generating instead multiple rules containing longer relationship patterns. We do not report WSC for these experiments, because the algorithms use different policy languages, and WSC is language-dependent.

DTRM and DTRM$^-$ are faster than Iyer et al.'s algorithm for all policies. Averaged over all policies, DTRM is 3.83 times faster, and DTRM$^-$ is 3.86 times faster. DTRM and DTRM$^-$ have very similar running times in these experiments, because very few negative features appear in the rules extracted from the decision trees.

8 FUTURE WORK

Directions for future work include: extending our algorithms to handle incompleteness and noise in the ACLs, perhaps using decision tree pruning methods, which are designed to avoid overfitting; extending our algorithms to identify errors in attribute values, and possibly suggest corrections; and developing incremental algorithms that efficiently handle updates to the object model or authorizations.

ACKNOWLEDGMENTS

This material is based on work supported in part by NSF grants CNS-1421893, CCF-1414078, and CCF-1954837 and ONR grant N00014-20-1-2751. We thank Hieu Le and R. Sekar for suggesting decision tree learning as an approach to policy mining. We thank Madison Ramos for valuable contributions to the initial stages of this work and the authors of [21] for sharing their code.

REFERENCES

[1] Manar Alohaly, Hassan Takabi, and Eduardo Blanco. 2018. A Deep Learning Approach for Extracting Attributes of ABAC Policies. In *Proc. 23rd ACM on Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 137–148.

[2] Matthias Beckerle and Leonardo A. Martucci. 2013. Formal Definitions for Usable Access Control Rule Sets—From Goals to Metrics. In *Proceedings of the Ninth Symposium on Usable Privacy and Security (SOUPS)*. ACM, Article 2, 11 pages.

[3] Jasper Bogaerts, Maarten Decat, Bert Lagaisse, and Wouter Joosen. 2015. Entity-Based Access Control: supporting more expressive access control policies. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015)*. ACM, 291–300. https://lirias.kuleuven.be/handle/123456789/521795

[4] Jasper Bogaerts, Maarten Decat, Bert Lagaisse, and Wouter Joosen. 2015. Entity-Based Access Control: supporting more expressive access control policies. In *Proc. 31st Annual Computer Security Applications Conference (ACSAC)*. ACM, 291–300.

[5] Leo Breiman, Jerome Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Wadsworth.

[6] Thang Bui and Scott D. Stoller. 2019. A Decision Tree Learning Approach for Mining Relationship-Based Access Control Policies. arXiv preprint arXiv:1909.12095 [cs.CR] (Sept. 2019).

[7] Thang Bui, Scott D. Stoller, and Hieu Le. 2019. Efficient and Extensible Policy Mining for Relationship-Based Access Control. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies (SACMAT 2019)*. ACM, 161–172.

[8] Thang Bui, Scott D. Stoller, and Jiajie Li. 2017. Mining Relationship-Based Access Control Policies. In *Proc. 22nd ACM Symposium on Access Control Models and Technologies (SACMAT)*. 239–246.

[9] Thang Bui, Scott D. Stoller, and Jiajie Li. 2018. Mining Relationship-Based Access Control Policies from Incomplete and Noisy Data. In *Proceedings of the 11th International Symposium on Foundations & Practice of Security (FPS 2018) (Lecture Notes in Computer Science)*, Vol. 11358. Springer-Verlag.

[10] Thang Bui, Scott D. Stoller, and Jiajie Li. 2019. Greedy and Evolutionary Algorithms for Mining Relationship-Based Access Control Policies. *Computers & Security* 80 (jan 2019), 317–333. Preprint available at http://arxiv.org/abs/1708.04749. An earlier version appeared as a short paper in ACM SACMAT 2017.

[11] Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. 2012. A User-to-User Relationship-Based Access Control Model for Online Social Networks. In *Proc. 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec) (Lecture Notes in Computer Science)*, Vol. 7371. Springer, 8–24.

[12] Carlos Cotrini, Luca Corinzia, Thilo Weghorn, and David Basin. 2019. The Next 700 Policy Miners: A Universal Method for Building Policy Miners. In *Proc. 2019 ACM Conference on Computer and Communications Security (CCS 2019)*. 95–112. https://doi.org/10.1145/3319535.3354196

[13] Carlos Cotrini, Thilo Weghorn, and David Basin. 2018. Mining ABAC Rules from Sparse Logs. In *Proc. 3rd IEEE European Symposium on Security and Privacy (EuroS&P)*. 2141–2148.

[14] Saptarshi Das, Barsha Mitra, Vijayalakshmi Atluri, Jaideep Vaidya, and Shamik Sural. 2018. Policy Engineering in RBAC and ABAC. In *From Database to Cyber Security*. Lecture Notes in Computer Science, Vol. 11170. Springer Verlag, 24–54.

[15] Maarten Decat, Jasper Bogaerts, Bert Lagaisse, and Wouter Joosen. 2014. *The e-document case study: functional analysis and access control requirements*. CW Reports CW654. Department of Computer Science, KU Leuven. https://lirias.kuleuven.be/handle/123456789/440202

[16] Maarten Decat, Jasper Bogaerts, Bert Lagaisse, and Wouter Joosen. 2014. *The e-document case study: functional analysis and access control requirements*. CW Reports CW654. Department of Computer Science, KU Leuven.

[17] Maarten Decat, Jasper Bogaerts, Bert Lagaisse, and Wouter Joosen. 2014. *The workforce management case study: functional analysis and access control requirements*. CW Reports CW655. Department of Computer Science, KU Leuven. https://lirias.kuleuven.be/handle/123456789/440203

[18] DT 2019. Decision Trees in scikit-learn v0.21.3. https://scikit-learn.org/stable/modules/tree.html.

[19] Philip W. L. Fong. 2011. Relationship-based access control: protection model and policy language. In *Proc. First ACM Conference on Data and Application Security and Privacy (CODASPY)*. ACM, 191–202.

[20] Padmavathi Iyer and Amirreza Masoumzadeh. 2018. Mining Positive and Negative Attribute-Based Access Control Policy Rules. In *Proc. 23rd ACM on Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 161–172.

[21] Padmavathi Iyer and Amirreza Masoumzadeh. 2019. Generalized Mining of Relationship-Based Access Control Policies in Evolving Systems. In *Proc. 24th ACM on Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 135–140.

[22] L. Karimi and J. Joshi. 2018. An Unsupervised Learning Based Approach for Mining Attribute Based Access Control Policies. In *2018 IEEE International Conference on Big Data (Big Data)*. 1427–1436. https://doi.org/10.1109/BigData.2018.8622037

[23] Alex X. Liu, Fei Chen, JeeHyun Hwang, and Tao Xie. 2008. XEngine: a fast and scalable XACML policy evaluation engine. In *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2008)*. ACM, 265–276.

[24] Eric Medvet, Alberto Bartoli, Barbara Carminati, and Elena Ferrari. 2015. Evolutionary Inference of Attribute-based Access Control Policies. In *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization (EMO): Part I (Lecture Notes in Computer Science)*, Vol. 9018. Springer, 351–365.

[25] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. 2016. A Survey of Role Mining. *Comput. Surveys* 48, 4 (2016), 50:1–50:37. https://doi.org/10.1145/2871148

[26] Decebal C. Mocanu, Faith Turkmen, and Antonio Liotta. 2015. Towards ABAC policy mining from logs with deep learning. In *Proc. 18th International Information Society Multiconference (IS 2015), Intelligent Systems*. Institut Jozef Stefan, Ljubljana, Slovenia.

[27] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin B. Calo, and Jorge Lobo. 2010. Mining Roles with Multiple Objectives. *ACM Trans. Inf. Syst. Secur.* 13, 4 (2010), 36:1–36:35.

[28] Masoud Narouei, Hassan Takabi, and Rodney Nielsen. 2018. Automatic Extraction of Access Control Policies from Natural Language Documents. *IEEE Transactions on Dependable and Secure Computing* (2018).

[29] Ronit Nath, Saptarshi Das, Shamik Sural, Jaideep Vaidya, and Vijay Atluri. 2019. PolTree: A Data Structure for Making Efficient Access Decisions in ABAC. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies (SACMAT 2019)*. ACM, 25–35.

[30] SSS 2019. Software from Scott Stoller's Research Group. https://www.cs.stonybrook.edu/~stoller/software/.

[31] Zhongyuan Xu and Scott D. Stoller. 2014. Mining Attribute-Based Access Control Policies from Logs. In *Proc. 28th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec)*. Springer, 276–291. Extended version available at http://arxiv.org/abs/1403.5715.

[32] Zhongyuan Xu and Scott D. Stoller. 2015. Mining Attribute-based Access Control Policies. *IEEE Transactions on Dependable and Secure Computing* 12, 5 (September–October 2015), 533–545.