Automated Least Privileges in Cloud-Based Web Services

Matthew Sanders
Colorado School of Mines
Golden, CO
mwsanders@mines.edu

Chuan Yue
Colorado School of Mines
Golden, CO
chuanyue@mines.edu

ABSTRACT

The principle of least privilege is a fundamental guideline for secure computing that restricts privileged entities to only the permissions they need to perform their authorized tasks. Achieving least privileges in an environment composed of many heterogeneous web services provided by a third party is an important but difficult and error prone task for many organizations. This paper explores the challenges that make achieving least privileges uniquely difficult in the cloud environment and the potential benefits of automated methods to assist with creating least privilege policies from audit logs. To accomplish these goals, we implement two frameworks: a Policy Generation Framework for automatically creating policies from audit log data, and an Evaluation Framework to quantify the security provided by generated roles. We apply these frameworks to a real world dataset of audit log data with 4.3 million events from a small company and present results describing the policy generator's effectiveness. Results show that it is possible to significantly reduce over-privilege and administrative burden of permission management.

ACM Reference format:
Matthew Sanders and Chuan Yue. 2017. Automated Least Privileges in Cloud-Based Web Services. In *Proceedings of HotWeb'17, San Jose / Silicon Valley, CA, USA, October 14, 2017,* 6 pages.
DOI: 10.1145/3132465.3132470

1 INTRODUCTION

The commoditization of web services by cloud computing providers enables the outsourcing of IT services on a massive scale. The business model of providing software, platform, and infrastructure components via web services has seen tremendous growth over the last decade and is forecast to continue expanding at a rapid pace [10]. From small startups to large companies such as Netflix, Expedia, and Yelp [3], many organizations rely on services provided by a third party for their mission critical operations. While the adoption of these hosted web services continues, there are significant security and usability concerns yet to be solved. Privilege management is a key issue in managing the operation of the diverse array of web services available.

The principle of least privilege is a design principle where privileged entities operate using the minimal set of privileges necessary to complete their job [6]. Least privileges protect against several threats, primarily among them being the *compromise of privileged entities' credentials and functions* by a malicious party. Other relevant threats mitigated by least privileges include *accidental misuse*, whereby privileged entities may delete or misconfigure resources which they do not require access to. Another threat is *intentional misuse*, where insiders can abuse over-privileges to cause more damage than they would be able to under a least privilege policy.

While implementing the principle of least privilege is a desirable and sometimes mandatory requirement for software systems, proper implementation can be difficult and is often not even attempted. Previous research into the use of least privilege practices in the context of operating systems [5] revealed that the overwhelming majority of study participants did not utilize least privilege policies. This was due to their partial understanding of the security risks, as well as a lack of motivation to create and enforce such policies. In comparison to the operating system environment, the use of third party web services present a much larger number of services, resource types, access control policy languages, and audit mechanisms even within a single service provider making it significantly more difficult to manage access control.

The main contributions of this paper are: (1) an exploration of the challenges and benefits of implementing an automated least privileges approach for third party web services using real world data, (2) a concrete implementation of a framework for generating least privilege policies from audit log data, and (3) metrics and methodology for quantifying the effectiveness of least privilege policies. Related works are described in Section 2. The motivating example of a real world dataset of manually created policies is analyzed in Section 3. Automated least privilege generation and evaluation frameworks used are describe in Section 4, the metrics used to evaluate adherence to PoLP are described in Section 5 and the results of our analysis are described in Section 6.

2 RELATED WORK

Addressing the administrative burden of access control management is a well-studied problem. While many access control models have been researched, Role Based Access Controls (RBAC) remains a common model for implementing access control policies. The fundamental premise of RBAC is to create a set of permissions for each functional role required to perform a job, and then assign privileged entities to these roles [7]. This allows policy creators to reason about access controls in terms of privileges needed to perform a task and the tasks an entity must perform.

A significant amount of work has been published on role mining methods which create more maintainable RBAC policies from existing privilege assignments. The basic RMP uses the minimal set of roles as the measure of goodness for deriving roles [11]. Alternatives to the minimum number of roles as a goodness metric for

HotWeb'17, San Jose / Silicon Valley, CA, USA

DOI: 10.1145/3132465.3132470

role mining algorithms have also been explored. A discussion of these alternative goodness metrics is given in [9], which include measuring similarity with existing roles, minimizing the number of user-role assignment and permission-role assignment relations, metrics that seek to reduce administrative cost, weighted structures that assign adjustable weights to assignment relationships, and minimizing the number of edges in the role hierarchy graph.

Another related area of research uses audit data to create least privilege policies. Privileged entities often already possess the privileges necessary to do their jobs, thus roles can be derived from existing permissions via data mining methods [8]. Notable examples of mining data to create least privilege policies include EASEAndroid [12] for mobile devices, ProgramCutter [13] for desktop applications, and Passe [4] for web applications. However, these approaches do not provide a quantified assessment of how well they achieve the PoLP.

Like role mining, our research aims to reduce the administrative burden of creating access control policies. However, instead of seeking to make roles more easily maintainable, we seek to reduce administrator burden by generating secure and complete policies via easily and frequently repeatable automated methods. The focus of this research is directly on quantifying and improving the security of automatically generated privilege assignments regardless of their size and complexity, thus we are addressing a problem different from the RMP.

3 OVER-PRIVILEGE IN MANUALLY GENERATED POLICIES

To illustrate the challenges of creating least privilege policies and to highlight the potential of using an automated approach to policy generation, we examine **a real world dataset** of policies manually created by administrators. The Amazon Web Services (AWS) CloudTrail [1] logs of a company which provides a Software as a Service (SaaS) product were analyzed (with permission). The audit logs contained 4.3M events collected over a period of 307 days. During this period, 37 unique roles and 15 unique users exercised privileges. Data gathered from the logs were analyzed and compared with the account Identity and Access Management (IAM) [2] policies as they existed at the end of the collection period. To quantify the effectiveness of these manually created policies at limiting over-privilege, we compare the actions and services granted by these policies to those exercised in the audit log data.

The privileged entities considered in this paper are *users* and virtual machine instances which can both be assigned to *roles*. In our dataset, users were granted unconstrained access making their comparison with exercised privileges somewhat uninteresting, but also demonstrating a situation where achieving least privilege policies on users was not even attempted. In contrast to users, virtual machines in our dataset were not granted unrestricted access but were assigned roles manually created by administrators with the intent of constraining the virtual machines to least privilege policies. While data for both users and roles were analyzed, this section focuses on role policies granted to virtual machines to illustrate the over-privilege present in manually created policies. As the results show, over-privilege was common for these roles even though the role creators had the benefit of familiarity with the application and the privileges it required. Services and actions not supported by CloudTrails were excluded from these results.

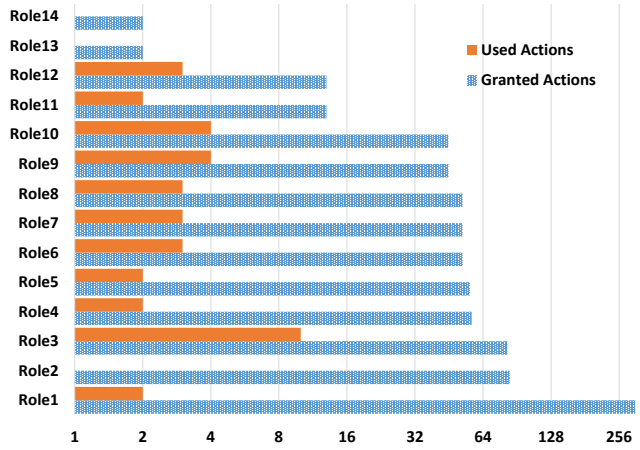

Figure 1: Number of Granted & Used Actions by Role

Of the 37 unique roles identified in the dataset, 14 were present in the AWS IAM data at the end of the collection period (those not found in the IAM policies had been deleted during the collection period). Figure 1 shows a comparison between the actions granted and used by virtual machine roles during the observation period. Even though the policies for each role were intended to approximate least privileges, clearly there is a significant difference between the number of actions granted and number of actions used. The average number of actions granted to these 14 roles was 61.14, while the average number of privileges used was 2.92.

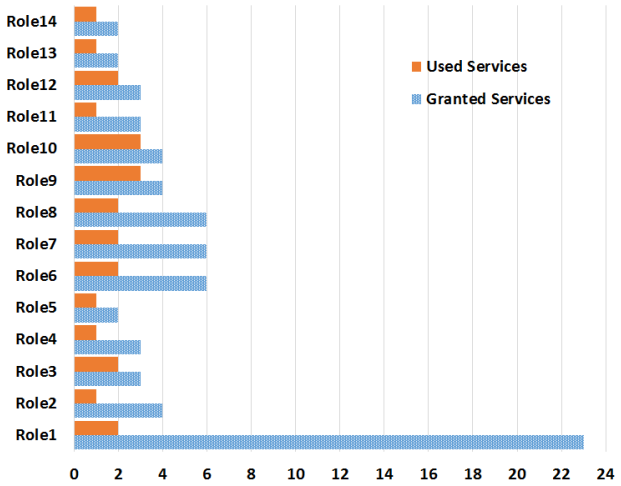

Figure 2: Number of Granted & Used Services by Role

The comparison of privileges granted to those actually used at the service level of granularity is shown in Figure 2. Significant over-privilege is present at the service level, with every role being granted privileges to at least one service for which it did not perform any actions. The average number of services used by roles was 1.71 while the average number of services granted was 5.07.

The results presented in this section demonstrate the over-privilege present in a real world dataset of manually created policies

with significant over-privilege present at both the action and service level for all virtual machine roles. Achieving least privilege policies for users was not even attempted in the dataset. These results underscore the difficulty and administrative burden of achieving least privilege policies in a cloud environment and provide motivation for an automated least privilege policy generation approach.

4 POLICY GENERATION AND EVALUATION

This section describes the frameworks for generating and evaluating least privilege policies. First we present a framework for generating least privilege policies from audit logs. We then present a framework for evaluating the effectiveness of a policy generator.

The process of generating policies begins with ingesting the raw data audit logs for a given observation period into a datastore. Once ingested, the logs are normalized by creating a projection of the events onto each unique privileged entity identified in the audit logs for a specified observation period. Next, the policy generator algorithm is applied to the normalized data. The generator implemented for this paper uses a simple counting based approach which creates policy grants for each action an entity successfully exercised during the observation phase. After policy generation is complete, additional modifications may be made to the policies such as denying access to privileges which can be used to escalate privileges. The policy generation framework is a bottom-up approach to building RBAC policies where exercised permissions are used to create roles. This design can also be applied to audit log data that have been previously collected in an organization's environment, and does not require an active presence in the cloud environment during log collection.

We next implemented a framework for evaluating the generated policies. This evaluation framework simulates the application of an automated least privileges policy generator across varying observation periods and operation periods. The purpose of these simulations is to provide a quantified evaluation of the effectiveness of our current and future policy generators if they were to be adopted in production by an organization. The information obtained from these simulations can help determine how long the observation period should be, how long these generated policies should be used for, and how effective the policy generator is. For these evaluations we chose one day as the finest granularity of time period as this provides enough time for entities to complete tasks requiring related privileges.

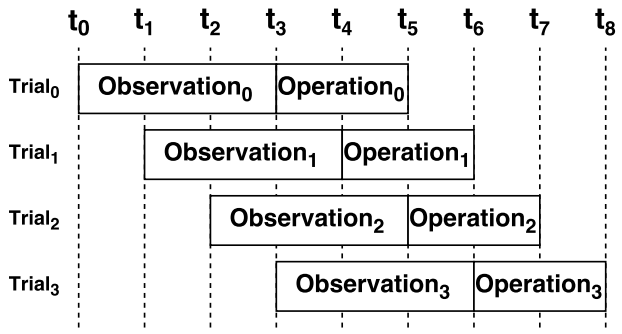

Figure 3: Sliding Window Evaluation

The evaluation framework uses a sliding window approach to perform its duties. It repeatedly generates observation and operation phases of predetermined sizes and compares the policy generated during the observation phase to the privileges exercised during the operation phase. Each of these single evaluations is a *trial* and multiple trials for the same evaluation parameters are achieved by incrementing the dates of the observation phase and operation phase by a fixed amount. Figure 3 provides a visual representation of how the sliding window technique is used to generate evaluation trials using the available audit log data.

5 METRICS

The PoLP implies two competing fundamental requirements. ***Minimize Over-Privilege***: Privileged entities should not be granted more permissions than necessary to complete their tasks. ***Minimize Under-Privilege***: Privileged entities should be granted all of the permissions that are necessary to complete their assigned tasks. Balancing between these requirements presents a trade-off between accepting risk and administrative overhead. To assess the effectiveness of automatically generated policies, we quantify their adherence to these requirements for meeting PoLP. We provide a variable weight to balance between these requirements so that organizations of automated least privilege policy tools can determine how to vary the observation length, operation length, and resource level restrictions depending on how much they value over-privilege vs. under-privilege.

To provide a quantitative evaluation we adopt terminology common to statistical hypothesis testing. The granting of a privilege by the policy generator is a positive prediction and the denial of a privilege is a negative prediction. For each evaluation trial, if the policy generated from the events of the observation phase granted a privilege which was exercised during the operation phase it is a **True Positive (TP)**, while a granted privilege that was not exercised during the operation phase is a **False Positive (FP)**. Similarly, if the automatically generated policy denied a privilege which was exercised during the operation phase it is a **False Negative (FN)**. Privileges which were denied by the policy and not exercised during the operation phase are a **True Negative (TN)**.

Precision and recall are metrics commonly used in hypothesis testing. **Precision** is the fraction of granted privileges that are exercised, high precision values indicate low over-privilege. **Recall** is the fraction of exercised privileges that are granted, high recall values indicate low under-privilege. The case where all privileges are denied is redefined to be $Precision = 1$ because there is no possibility of over-privilege, and the case where all privileges are granted is redefined to be $Recall = 1$ because there is no possibility of under-privilege. To present more intuitive metrics, we take the compliment of precision and recall to create metrics where lower values are more favorable: the *Over Privilege Rate (OPR)* in Equation 1 and *Under Privilege Rate (UPR)* in Equation 2, respectively.

$$OPR = 1 - Precision = \frac{UnexercisedGranted}{AllGranted} \tag{1}$$

$$UPR = 1 - Recall = \frac{ExercisedDenied}{AllExercised} \tag{2}$$

It is important to consider the amount of time which over-privilege exists. While the cost of under-privilege is a decreased ability for privileged entities to perform their tasks, high over-privilege can result in compromises of confidentiality, integrity, and availability if the over-privilege is exploited by an attacker. The longer that over-privilege exists the greater the possibility of it being exploited, thus we introduce an additional weight on the *OPR* to account for the amount of time which unused privilege grants existed. The *Temporal Over Privilege Rate (TOPR)* in Equation 3 is the *OPR* multiplied by the number of days the privileges went unused (the length of the operation period).

$$TOPR = OPR \cdot OperationPeriodLength \tag{3}$$

OPR and *UPR* are two individual metrics for measuring the generated least privilege policies. To provide a single metric that weights minimal over-privilege vs. minimal under-privilege, we use the F-score metric (Equation 4). Higher β values for the F-score indicate a higher weight for recall, which indicates a higher weight for minimal under-privilege. Lower β values for the F-score weight minimal over-privilege higher. We use a temporally weighted version of the F-score, TF_β (Equation 5), that accounts for the length of time which an over-privilege was granted. To incorporate a temporal weighting of over-privilege in TF_β, we divide the precision by the operation period length because precision is the compliment of *OPR* and thus is directly tied to how we score over-privilege. Note that F_β and TF_β are equivalent for the finest granularity of the operation period which is one day in our simulations.

$$F_\beta = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{4}$$

$$TF_\beta = (1+\beta^2) \cdot \frac{\frac{Precision}{OperationPeriodLength} \cdot Recall}{(\beta^2 \cdot \frac{Precision}{OperationPeriodLength}) + Recall} \tag{5}$$

The F-score is the harmonic mean of precision and recall. The advantage of using the harmonic mean F-score over arithmetic mean is that low scores for either precision or recall will result in an overall low F-score which avoids allowing extreme policies to achieve favorable scores. Consider an example policy which grants all privileges to an entity. This would result in a perfect score in terms of precision (1), but the worst possible score in terms of recall (0). The resulting F-score in this example would be 0 while arithmetic mean score would be 0.5, the same as if precision and recall were both 0.5. This equal scoring between an extreme policy and a balanced policy is not desirable in applications which values both precision and recall.

6 RESULTS

This section presents the results of our analysis tying together all of the work described thus far. We consider the behavior of users and roles granted to virtual machines separately when evaluating the effectiveness of their policies because they have different usage patterns which produce significantly different scores. The behavior of virtual machines is fairly consistent in both the actions and resources used while users are less predictable.

6.1 Impact of Varying the Operation Period

The results of evaluating the least privilege policy generator for observation periods of 7 and 28 days as the operation phase varies from 1 to 7 days are shown for users in Figure 4 and for virtual machine roles in Figure 5. The results for both entity types show that as the length of the operation phase increases, the *UPR* also increases which is to be expected as privileged entities use privileges that were not observed during shorter operation phases. For virtual machine roles, there is very little difference between the *UPR* for 7 days of operation vs. 28 days of operation. As we will see later in the metrics, the most variability in virtual machine permissions exercised occurs during the first few days of the observation phase.

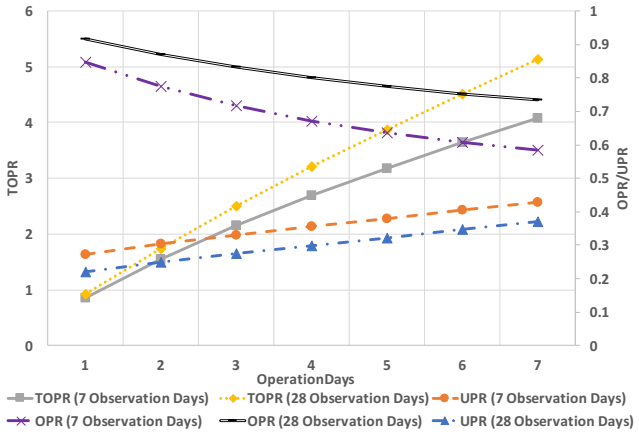

Figure 4: User Evaluation as Opr Days Vary (Obs Days=7,28)

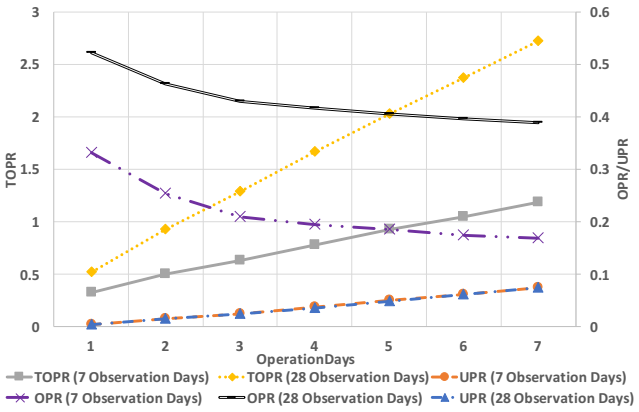

Figure 5: Role Evaluation as Opr Days Vary (Obs Days=7,28)

As the operation phase increases entities are more likely to use privileges they may not have exercised previously during shorter periods. Thus the unweighted *OPR* decreases for both entity types as the operation period increases. However, the *TOPR* in Figure 4 increases as the operation phase increases, indicating that the new privileges exercised during each additional day of the operation phase do not reduce over-privilege enough to offset the over-privilege caused by leaving the unexercised privileges granted

to the entities longer. The effect is more pronounced users than virtual machine roles - the virtual machine roles have lower *TOPR* scores for all operation and observation periods.

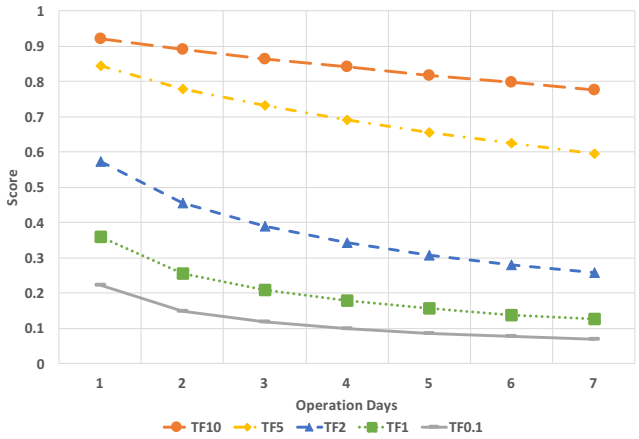

Figure 6: TF_β score as Opr Days Vary (Obs Days=7)

To determine a recommended operation period based on how much one values minimal over-privilege vs. minimal under-privilege, we use the TF_β metric (Formula 5). Figure 6 shows the combined TF_β score for both user and virtual machine role data for varying operation period lengths and β values. In these charts $\beta = 10$ represents that minimal under-privilege is considered to be 10 times more important than minimal over-privilege while $\beta = 0.1$ represents that minimal over-privilege is 10 times more important than minimal under-privilege. All of the calculated TF_β scores constantly decrease as the operation period increases indicating the smallest operation period of one day is the optimal choice for minimizing temporal weighted over-privilege and under-privilege. The higher β values show generally higher scores which decrease less as the operation period increases, indicating that increasing the operation period would have a less negative impact for those that value minimal under-privilege.

6.2 Impact of Varying the Observation Period

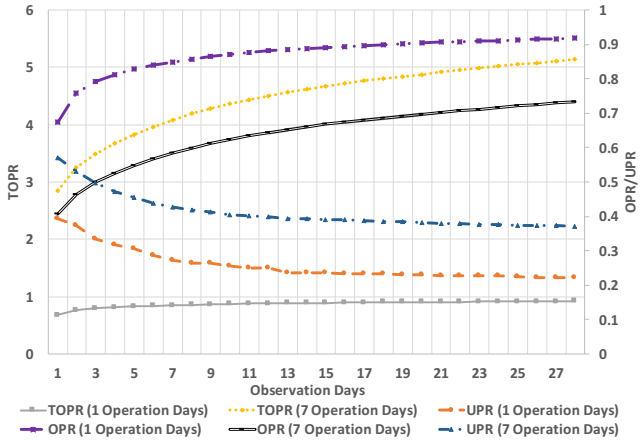

Figure 7: User Evaluation as Obs Days Vary (Opr Days=1,7)

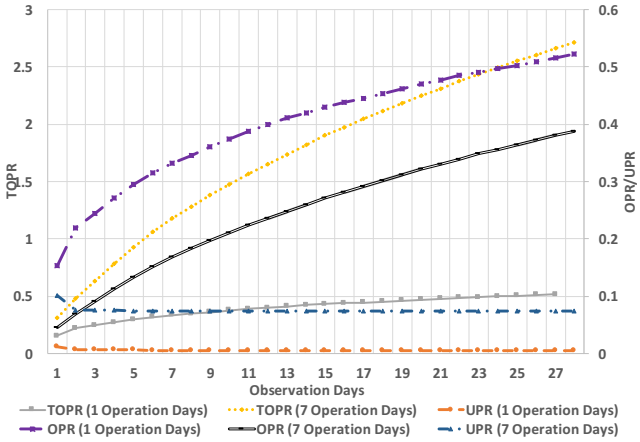

Figure 8: Role Evaluation as Obs Days Vary (Opr Days=1,7)

Next we evaluate the impact of varying the observation period. The results of evaluating the automated least privilege policy generator for operation phases of lengths 1 and 7 days as the observation phase varies from 1 to 28 days are shown for users in Figure 7 and for virtual machine roles in Figure 8. As the observation period increases the *UPR* decreases for users at a logarithmic rate because more privileges exercised by users are captured during longer observation phases. For virtual machine roles however there is little benefit in increasing the observation period beyond two days as these virtual machines are unlikely to exercise additional privileges that have not been exercised after the first day of observation. For both entity types the *UPR* is again lower for the 1 day operation period vs. the 7 day operation period.

For both entity types the *OPR* and *TOPR* increase as the observation phase increases because longer observation phases result in entities being granted more privileges. This is intuitively obvious for users as they are likely to use some privileges periodically which are captured during the observation phase, and then not use them again for extended periods of time or at all during the operation phase. Although the virtual machine roles are unlikely to spontaneously use new privileges like users, not all privileges are exercised on a daily basis.

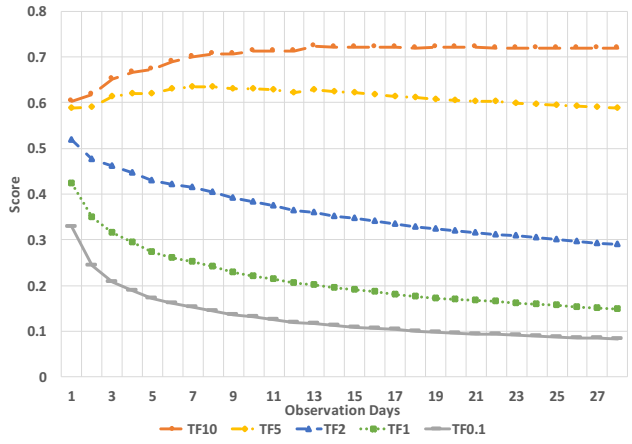

Figure 9: User TF_β scores as Obs Days Vary (Opr Days=1)

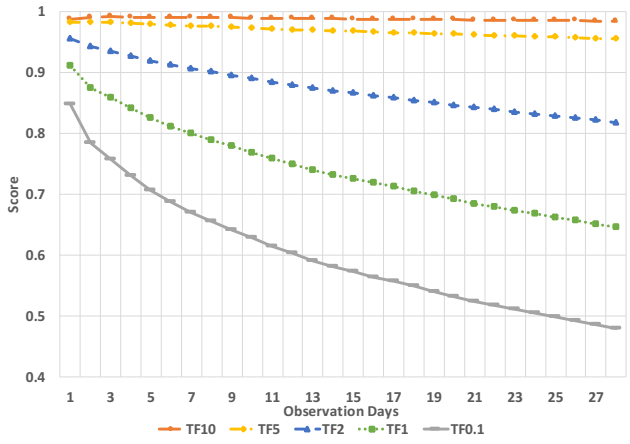

Figure 10: Role TF_β scores as Obs Days Vary (Opr Days=1)

To determine a recommended observation period based on how much one values minimal over-privilege vs. minimal under-privilege, we again use the TF_β metric. For this evaluation the user and virtual machine role scores are presented separately because (unlike varying the operation phase in Figure 6) the dissimilar behavior patterns of users and virtual machines produce different recommended observation periods. Figure 9 displays the TF_β scores for user entities as the observation phase varies and the operation phase remains fixed at one day. The decreasing scores for $\beta = 0.1, 1, 2$ imply that organizations which value minimal over-privilege should choose a lower observation period. Even if minimal under-privilege is valued twice as much as minimal over-privilege as indicated by $\beta = 2$, the *OPR* rises significantly faster than the under-privilege rate decreases as the observation period increases (as shown in Figure 7). For $\beta = 5, 10$ the TF_β increases as the observation period increases before eventually decreasing at 8 days for $\beta = 5$ and stabilizing at 13 days for $\beta = 10$ as the increasing *OPR* outweighs the more heavily rated but slower to decline *UPR*. The TF_β scores for virtual machine roles are presented in Figure 10. The role based scores for low β again show that organizations which value minimal over-privilege should use small observation periods, while organizations which value minimal under-privilege will see little or no benefit in extending the observation period for these roles as the under-privilege rate showed little decline for observation periods over two days (as shown in Figure 8).

6.3 Summary of Results

The results of this section quantify the effectiveness of our policy generator applied to real world hosted web service audit log dataset. They describe how the performance of the policy generator is affected by varying the observation period and operation period. Based on this evaluation, we found that the actions of users were relatively difficult to predict compared to virtual machine roles with incidents of under-privilege being much higher for users. Virtual machines could be constrained to their actions used during their first couple days operation to significantly reduce over-privilege present in their policies. For both types of privileged entities, increasing the operation period increased under-privilege while increasing the observation period increased over-privilege.

The conclusions drawn from these results are valuable because they quantify the performance that can be expected by adopting an automated least privilege approach and they provide a benchmark by which to judge future policy generation algorithms. The generation of these results also demonstrates the application of the policy generation and evaluation frameworks which can be used for evaluating future algorithms.

7 CONCLUSION

This paper explored the challenges and benefits of automating least privilege policies in a cloud computing environment. Previous research in role mining approaches in other environments were examined and unique aspects of automated role mining in a cloud computing environment were identified. A bottom-up design to generate least privilege policies was implemented to illustrate the potential of an automated least privilege approach and the results of evaluation on real world audit log data were presented. The results showed that even when administrators attempt to manually create least privilege policies there is significant room for improvement upon these policies. Metrics for evaluating the effectiveness of least privilege policy generators were presented for the same data set. These results showed the trade-offs between over-privilege and under-privilege that can be achieved by varying the observation period, operation period, and resource constraints for the presented policy generator and these results provide benchmarks for future policy generators to be evaluated against.

ACKNOWLEDGMENTS

This research was supported in part by the NSF grant DGE-1619841.

REFERENCES

[1] Amazon Web Services. 2017. AWS CloudTrail. https://aws.amazon.com/cloudtrail/. (2017). Accessed: 2017-02-20.
[2] Amazon Web Services. 2017. AWS Identity and Access Management (IAM). https://aws.amazon.com/iam/. (2017). Accessed: 2017-02-20.
[3] Amazon Web Services. 2017. Case Studies. https://aws.amazon.com/solutions/case-studies. (2017). Accessed: 2017-03-20.
[4] Aaron Blankstein and Freedman J. Michael. 2014. Automating isolation and least privilege in web services.. In *IEEE Symposium on Security and Privacy*. IEEE, 133–148.
[5] Sara Motiee, Kirstie Hawkey, and Konstantin Beznosov. 2010. Do windows users follow the principle of least privilege?: investigating user account control practices.. In *Symposium on Usable Privacy and Security (SOUPS)*.
[6] Jerome H Saltzer and Michael D Schroeder. 1975. The protection of information in computer systems. *IEEE* 63, 9 (1975), 1278–1308.
[7] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. 2000. The NIST model for role-based access control: towards a unified standard.. In *ACM workshop on Role-based access control*.
[8] Jrgen Schlegelmilch and Ulrike Steffens. 2005. Role mining with ORCA.. In *ACM Symposium on Access control models and technologies (SACMAT)*.
[9] Hassan Takabi and James BD Joshi. 2010. StateMiner: an efficient similarity-based approach for optimal mining of role hierarchy. In *Proceedings of the 15th ACM symposium on Access control models and technologies*. ACM, 55–64.
[10] U.S. Department of Commerce. 2016. 2016 Top Markets Report Cloud Computing. http://trade.gov/topmarkets/pdf/Cloud_Computing_Top_Markets_Report.pdf. (2016). Accessed: 2017-03-23.
[11] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. 2006. RoleMiner: mining roles using subset enumeration. In *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 144–153.
[12] Ruowen Wang, William Enck, Douglas Reeves, Xinwen Zhang, Peng Ning, Dingbang Xu, Wu Zhou, and Ahmed M. Azab. 2015. EASEAndroid: Automatic Policy Analysis and refinement for security enhanced android via large-scale semi-supervised learning.. In *USENIX Security Symposium*.
[13] Yongzheng Wu, Jun Sun, Yang Liu, and Jin Song Dong. 2013. Automatically partition software into least privilege components using dynamic data dependency analysis.. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*.