Definitions

• **Access Control**: the process of allowing only authorized privileged entities to observe, modify, or otherwise take possession of the resources of a computer system. It is also a mechanism for limiting the use of resources to authorized users.

• **Principle of Least Privilege**: an access control design principle where privileged entities operate using the minimal set of privileges necessary to complete their job. Least privilege policies protect against:
  • Compromise of privileged entities’ credentials.
  • Accidental misuse.
  • Intentional misuse.
Motivation

All compliance standards have a requirement to implement “Least Privilege”

• DoD Information Assurance Certification and Accreditation Process (DIACAP), Control ECLP-1 Least Privilege: Access procedures enforce the principles of separation of duties and "least privilege."

• Payment Card Industry Data Security Standard (PCI-DSS), Requirement 7: “Restrict access to cardholder data by business need to know.”

• Health Insurance Portability and Accountability (HIPAA), §164.312(a)(3)(ii)(B): “Implement procedures to determine that the access of a workforce member to electronic protected health information is appropriate.”

• NIST SP 800-171 Protecting Controlled Unclassified Information in Nonfederal Systems, 3.1.7: “Employ the principle of least privilege, including for specific security functions and privileged accounts.”
Problem

• Lack of quantified metrics to determine if least privilege is achieved, and to what degree.
• Least privilege is difficult to achieve.
• Privilege assignments and user behavior changes over time.
• ABAC privilege space is massive, difficult to create/manage good policies, overwhelming flexibility.
ABAC Privilege Error Minimization Problem

• OBP observation period, the time-period during which exercised permissions are observed and used for creating an access control policy.

• OPP operation period, the time-period during which the access control policy is to be considered in operation.

• ABAC Privilege Error Minimization Problem (PEMP\textsubscript{ABAC}). Given the universe of all valid attribute:value combinations, find the set of attribute:value constraints that minimizes the over-privilege and under-privilege errors for a given operation period OPP.
General Approach

• Mine audit logs during an Observation Period (OBP) to create policies.
• Evaluate policies during an Operation Period (OPP) to score them.
• Privileged entities often already possess the privileges necessary to do their jobs, so policies can be derived from existing permissions.
Security policies are predictions about privilege usage:

- **True Positives (TP):** Granted privileges which were needed.
- **False Positives (FP):** Granted but unused privileges.
- **False Negatives (FN):** Denied privileges which were needed.
- **True Negatives (TN):** Denied and unused privileges.
Attribute Based Access Control (ABAC)

1. Subject requests access to object
2. Access Control Mechanism evaluates:
   a) Rules
   b) Subject Attributes
   c) Object Attributes
   d) Environment Conditions
3. Subject is given access to object if authorized

ABAC Advantages: Flexibility and granularity.
ABAC Challenges: Large privilege space. Flexibility can be overwhelming.
Example AWS CloudTrail Audit Log Entry

{"Records": [{
    "eventVersion": "1.0",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "accountId": "123456789012",
        "userName": "Alice"
    },
    "eventTime": "2014-03-06T21:22:54Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "StartInstances",
    "sourceIPAddress": "205.251.233.176",
    "userAgent": "ec2-api-tools 1.6.12.2",

....

<table>
<thead>
<tr>
<th>Audit Log Entries</th>
<th>4.7M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Span</td>
<td>16 Months</td>
</tr>
<tr>
<td>Users</td>
<td>38</td>
</tr>
</tbody>
</table>
# Policy Generation using Frequent Itemset Mining

## Features

<table>
<thead>
<tr>
<th>Entry</th>
<th>User</th>
<th>Service</th>
<th>Action</th>
<th>Resource Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry1</td>
<td>User1</td>
<td>IAM</td>
<td>Create</td>
<td>User</td>
</tr>
<tr>
<td>Entry2</td>
<td>User2</td>
<td>EC2</td>
<td>Update</td>
<td>Instance</td>
</tr>
<tr>
<td>Entry3</td>
<td>User3</td>
<td>S3</td>
<td>Delete</td>
<td>Bucket</td>
</tr>
<tr>
<td>Entry4</td>
<td>User3</td>
<td>EC2</td>
<td>Update</td>
<td>Volume</td>
</tr>
<tr>
<td>Entry5</td>
<td>User1</td>
<td>EC2</td>
<td>Create</td>
<td>Instance</td>
</tr>
<tr>
<td>Entry6</td>
<td>User2</td>
<td>S3</td>
<td>Read</td>
<td>Object</td>
</tr>
<tr>
<td>Entry7</td>
<td>User1</td>
<td>EC2</td>
<td>Read</td>
<td>Instance</td>
</tr>
<tr>
<td>Entry8</td>
<td>User3</td>
<td>IAM</td>
<td>Read</td>
<td>User</td>
</tr>
</tbody>
</table>

## Itemset

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service=EC2</td>
<td>4</td>
<td>50%</td>
</tr>
<tr>
<td>Service=EC2, Type=Instance</td>
<td>3</td>
<td>37.5%</td>
</tr>
<tr>
<td>User=User1</td>
<td>2</td>
<td>37.5%</td>
</tr>
<tr>
<td>Service=EC2, Action=Update</td>
<td>2</td>
<td>25%</td>
</tr>
</tbody>
</table>

- **Itemset**: A collection of one or more `Attribute:Value` Conditions
- **\(\varepsilon\)**: threshold value for minimum itemset frequency.
- **Support**: Frequency of occurrence of an itemset.
- **Coverage**: The % of transactions an itemset is present in.
Evaluating Candidate Rules

Definitions:
• $r$ is a rule consisting of one or more ABAC attribute:value constraints.
• $p$ is an ABAC policy consisting of one or more rules.
• $\mathbb{L}_{OBP}$ The set of log entries representing user actions during the observation period OBP.
• $\mathfrak{e}$ is the set of all possible attribute:value combinations.
• $\mathfrak{e}^\prime$ is the set of all valid attribute:value combinations when considering the dependency relationships between all attributes and values.

Metrics:
• $\text{CoverageRate}(r, p, \mathbb{L}_{OBP}) = \frac{|\mathbb{L}_{OBP}(r) \setminus \mathbb{L}_{OBP}(p)|}{|\mathbb{L}_{OBP} \setminus \mathbb{L}_{OBP}(p)|}$
• $\text{OverPrivilegeRate}(r, p, \mathbb{L}_{OBP}, \mathfrak{e}^\prime) = \frac{|\mathfrak{e}^\prime(r) \setminus (\mathbb{L}_{OBP}(p) \setminus \mathbb{L}_{OBP}(r))|}{|\mathfrak{e}^\prime|}$
• $C_{\text{score}}(\text{rule}, \text{policy}, \mathbb{L}_{OBP}, \mathfrak{e}^\prime, \omega) = \text{CoverageRate}(r, p, \mathbb{L}_{OBP}) + \omega \times (1 - \text{OverPrivilegeRate}(r, p, \mathbb{L}_{OBP}, \mathfrak{e}^\prime))$
Algorithm 1 ABAC Policy Generator

Input: $\mathbb{L}_{O BP}$ The set of log entries representing user actions during the observation period $O BP$.
Input: $\omega$ under-privilege vs. over-privilege weighting variable.
Input: $\epsilon$ Threshold value for minimum itemset frequency.
Input: $\xi$ The set of all valid attribute:value combinations that comprise the privilege universe.
Output: $policy$ The set of ABAC rules that make up the policy to be applied during the operation period $O PP$.

1. $policy \leftarrow \emptyset$;
2. $\mathbb{L}_{uncov} \leftarrow \mathbb{L}_{O BP}$;
3. while $|\mathbb{L}_{uncov}| > 0$ do
4.   $itemsets \leftarrow FP\text{-}growth\_frequentItemsets(\mathbb{L}_{uncov}, \epsilon)$;
5.   $candidateRules \leftarrow \emptyset$;
6.   for $itemset \in itemsets$ do
7.     $rule = createRule(itemset)$;
8.     $coverageRate = \frac{|\mathbb{L}_{uncov}(rule)|}{|\mathbb{L}_{uncov}|}$;
9.     $overAssignmentRate = \frac{|\xi^c(rule)| - |\mathbb{L}_{uncov}(rule)|}{|\xi^c|}$;
10.    $rule.Score = coverageRate + \omega \times (1 - overAssignmentRate)$;
11.    $candidateRules \leftarrow candidateRules \cup rule$;
6.   end
7. end
8. $bestRule = sortDescending(candidateRules, C_{score})[0]$;
9. $policy \leftarrow policy \cup bestRule$;
10. $\mathbb{L}_{uncov} \leftarrow \mathbb{L}_{uncov} \setminus \mathbb{L}_{uncov}(bestRule)$;
11. end
12. return $policy$
Algorithm 2  ABAC Policy Evaluator

Input: $L_{OPP}$ The set of log entries representing user actions during the operation period $OPP$.
Input: $\xi^i$ The set of all valid attribute:value combinations that comprise the privilege universe.
Input: $policy$ The set of ABAC rules that make up the policy to be applied during the operation period $OPP$.
Output: $TPR, FPR$ The true positive and false positive rates of the policy evaluated against the operation period $OPP$.

$TP = FN = 0$;

for $event \in L_{OPP}$ do

if $policyAllowsEvent(policy, event)$ then

$TP = TP + 1$;

exercisedGrantedEvents $\leftarrow$ exercisedGrantedEvents $\cup$ event;

else

$FN = FN + 1$;

end

end

$eventsAllowedByPolicy \leftarrow \emptyset$;

for $r \in policy$ do

$eventsAllowedByPolicy \leftarrow eventsAllowedByPolicy \cup \xi^i(rule)$;

end

$FP = |eventsAllowedByPolicy \setminus exercisedGrantedEvents|$

$TN = |privUniverse| - (TP + FN + FP)$;

if $TP + FN == 0$ then

$TPR = 1$;

else

$TPR = TP/(TP + FN)$;

end

$FPR = FP/(FP + TN)$;

return $TPR, FPR$
Optimizations for large ABAC privilege spaces

Feature Selection:
• Use attributes that occur frequently with high uniqueness, but not unique values with every request (ex. RequestId). Remove redundant attributes (1:1 correlation).
• Apply binning to highly unique attributes like IP addresses and client library versions.

Policy Mining Optimizations:
• Divide privilege space into independent partitions for 1) user attributes, 2) environment attributes, and 3) operation & resource attributes. Use inverted indexes to calculate over-privilege rate of a rule quickly.
• Algorithm 1 only counts the number of privileges covered by a rule, does not enumerate full privilege space.

Policy Scoring Optimizations:
• Algorithm 2 must enumerate all privilege combinations covered by a rule to find if they are also covered by other rules in the policy.
• Partition $\xi$ by attribute values to create one partition per processor core, distribute OPP evaluation on AuN (“Golden”) with 144 compute nodes, 2,304 cores.
C-Score Analysis

Candidate evaluation metrics should:
1. Have a high Area Under The ROC Curve (AUC) value.
2. Exhibit Smoothness, TPR values should increase monotonically as FPR increases.
3. Be Interpretable, changing the weighting variable show a proportionate change in TPR & FPR and easy to understand for policy creators.
Frequency and Observation Period Analysis

**Itemset Frequency (ε) Varies**

- ε=0.3  AUC=0.9640
- ε=0.2  AUC=0.9940
- ε=0.1  AUC=0.9993
- ε=0.05  AUC=0.9996

**Observation Period Size Varies**

- 7 Days   AUC=0.9963
- 15 Days  AUC=0.9985
- 30 Days  AUC=0.9993
- 45 Days  AUC=0.9991
- 60 Days  AUC=0.9987
ABAC vs. RBAC Performance

Comparison of ABAC mining and naïve RBAC algorithm for RBAC policy generation.

- ABAC Fixed $|L_{OBP}| = 30$ days
- ABAC varied $\omega = \left[\frac{1}{8192}, \ldots, 16\right]$ days
- RBAC varied $|L_{OBP}| = [3, \ldots, 120]$ days
- Used ABAC policy scoring (Algorithm 2)
Contributions / Conclusion

• Our optimization methods for feature selection, policy generation and policy evaluation enabled us to work on a real world dataset of millions of audit events with a privilege space spanning billions of possible entries.

• Our candidate evaluation metric ($C_{\text{score}}$) provides better AUC, smoothness, and predictability than metrics from related works.

• Our policy generation algorithm provides significant flexibility by varying itemset frequency ($\varepsilon$) and under- vs. over-privilege weighting ($\omega$).

• Our methodology addresses changes in user behavior over time.

• The granularity of the ABAC model allows for mining policies that have much fewer assignment errors than RBAC policies.
email: mwsanders@mines.edu

source code: https://github.com/mwsanders/AssociationAbacMiner

Questions?