Detecting Anomalous Misconfigurations in AWS Identity and Access Management Policies

Thijs van Ede, Niek Khasuntsev, Bas Steen & Andrea Continella

Contact: t.s.vanede@utwente.nl
Misconfigurations

Capital One Attacker Exploited Misconfigured AWS Databases

After bragging in underground forums, the woman who stole 100 million credit applications from Capital One has been found guilty.
Misconfigurations

Capital One Attacker Exploited Misconfs

Three million senior citizens’ info exposed by SeniorAdvisor

A security breach at SeniorAdvisor, a review website, compromised over three million elderly adults’ personal information in August. WizCase researchers observed that a misconfigured Amazon S3 bucket exposed details including individuals’ names, numbers, and email addresses. The information pertained to
Misconfigurations

Capital One Attacker Exploited Misconfiguration

Three million senior citizens’ info exposed by SeniorAdvisor

A security breach at SeniorAdvisor, a review website, compromised over three

Amazon Web Services Misconfiguration Exposes Half a Million Cosmetics Customers

se researchers
tails including
ation pertained to
Misconfigurations

Capital One Attacked

Misconfigurations

In May 2022, a security firm discovered an unprotected AWS S3 bucket containing 6.5 terabytes of “Electronic Flight Bag” information, including navigation information, proprietary software, and personal information pertaining to Pegasus Airlines crew members. Once notified of the exposed information, Pegasus Airlines promptly secured the unprotected S3 bucket.

May 2022: 23 Million Files Exposed in Pegasus Airlines Breach

A security breach at SeniorAdvisor, a review website, compromised over three million customers’ details, including personal information pertained to Amazon Web Services Misconfiguration Exposes Half a Million Cosmetics Customers

se researchers
Misconfigurations

May 2022: 23 Million Files Exposed in Pegasus Airlines Breach
In May 2022, a security firm discovered an unprotected AWS S3 bucket containing 6.5 terabytes of “Electronic Flight Bag” information, including navigation information, proprietary software, and personal information pertaining to Pegasus Airlines crew members. Once notified of the exposed information, Pegasus Airlines promptly secured the unprotected S3 bucket.

July 2021: PeopleGIS Exposes Sensitive Data for Over 80 Municipalities
In July 2021, a group of ethical hackers at WizCase discovered a vulnerability affecting at least 80 municipalities in the United States. This breach resulted from misconfigured Amazon S3 buckets related to MapsOnline, a service run by the software company PeopleGIS. It's unclear whether the misconfiguration was made by PeopleGIS or by the municipalities in question.
Misconfigurations

Capital One Attack

Misconfiguration

Three Million

by Seven

A security

July 2021

On

In July 2021, a group of ethical hackers at WizCase discovered a vulnerability affecting at least 80 municipalities in the United States. This breach resulted from misconfigured Amazon S3 buckets related to MapsOnline, a service run by the software company PeopleGIS. It's unclear whether the misconfiguration was made by PeopleGIS or by the municipalities in question.

Misconfigured cloud environments are a problem!
Identity and Access Management

Entities

Action

Resources

S3 bucket

S3 bucket

EC2

EC2
Identity and Access Management

We want to control which entity can perform which action to which resource.
Identity and Access Management

Entities

Request Action

IAM

Policy

Resources

S3 bucket

S3 bucket

EC2

EC2

Perform Action

UNIVERSITY OF TWENTE

Detecting Anomalous Misconfigurations in AWS Identity and Access Management Policies
Identity and Access Management

IAM controls resource access
Identity and Access Management

Entities

Resources

S3 bucket

S3 bucket

EC2

EC2

Request Action

IAM

Policy

Perform Action

Detecting Anomalous Misconfigurations in AWS Identity and Access Management Policies
Identity and Access Management

```
{
    "Version": "2012-10-17",
    "Name": "AdminAccess",
    "Statement": {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
    }
}
```
Identity and Access Management

Do these configured policy allow **too much** access?

{  
  "Version": "2012-10-17",
  "Name": "AdminAccess",
  "Statement": {
    "Effect": "Allow",
    "Action": "*",
    "Resource": "*"
  }
}
Identity and Access Management

Entities
- Users
- Admin

Resources
- S3 bucket
- S3 bucket
- EC2
- EC2

Request Action

IAM
- Policy

Perform Action
Identity and Access Management

How do we identify misconfigurations?
Existing solutions

- Cloud Custodian
Existing solutions

- Cloud Custodian
  - Rule-based
Existing solutions

- Cloud Custodian
  - Rule-based
  - Requires manual tweaking of rules
Existing solutions

- Cloud Custodian
  - Rule-based
  - Requires manual tweaking of rules
- P-Diff
Existing solutions

- Cloud Custodian
  - Rule-based
  - Requires manual tweaking of rules
- P-Diff
  - Learns control policies from access logs
Existing solutions

- Cloud Custodian
  - Rule-based
  - Requires manual tweaking of rules
- P-Diff
  - Learns control policies from access logs
  - Reactive approach
Idea

- Most policies are properly configured
Idea

- Most policies are properly configured
- Use **anomaly detection** to learn properly configured policies
Idea

- Most policies are properly configured
- Use anomaly detection to learn properly configured policies
- Any found anomalies will likely be misconfigurations
Challenges

- Policies are **specific** to the **context** of the organization
Challenges

- Policies are **specific** to the **context** of the organization
- Checks must be **proactive** to ensure policies are not abused
Challenges

- Policies are **specific** to the **context** of the organization
- Checks must be **proactive** to ensure policies are not abused
- Checks must be **low maintenance** to ensure adoption
Approach

- Model policies as a graph
Approach

- Model policies as a graph
Approach

- Model policies as a graph
Approach

- Model policies as a graph
  - A policy can have multiple statements

```json
{
  "Version": "2012-10-17",
  "Name": "AdminAccess",
  "Statement": {
    "Effect": "Allow",
    "Action": "*",
    "Resource": "*"
  }
}
```
Approach

- Model policies as a graph
  - A policy can have multiple statements
Approach

- Model policies as a graph
  - A policy can have multiple statements
Approach

- Model policies as a graph
  - A policy can have multiple statements
Approach

- Model policies as a graph
  - A policy can have multiple statements

How do we find misconfigurations?
Approach

- Model policies as a graph
  - A policy can have multiple statements
- Policies are **specific** to the **context** of the organization
Approach

- Model policies as a graph
  - A policy can have multiple statements
- Policies are specific to the context of the organization
- Model the context of policies using **Node2vec**
Approach - Node2vec

- Select a starting **policy node**
Approach - Node2vec

- Select a starting **policy node**
- Perform random walks
Approach - Node2vec

- Select a starting policy node
- Perform random walks
  - Collect information about visited nodes and edges
Approach - Node2vec

- Select a starting policy node
- Perform random walks
  - Collect information about visited nodes and edges
Approach - Node2vec

- Select a starting policy node
- Perform random walks
  - Collect information about visited nodes and edges
- Store information in a fixed length vector
Approach - Node2vec

- Select a starting policy node
- Perform random walks
  - Collect information about visited nodes and edges
- Store information in a fixed length vector

We still haven’t detected misconfigurations…
Approach - Anomaly detection

- Each policy node is represented by a vector
Approach - Anomaly detection

- Each policy node is represented by a vector
- We can train an anomaly detection model to find anomalous policies
Approach - Anomaly detection

- Each policy node is represented by a vector
- We can train an anomaly detection model to find anomalous policies
  - One-Class SVM
  - Local Outlier Factor
  - Isolation Forest
  - Robust Covariance
How does this work in practice?

- Security operators manually verify a set of policies
How does this work in practice?

- Security operators manually verify a set of policies
- We run our approach and train the anomaly detector
How does this work in practice?

- Security operators manually verify a set of policies
- We run our approach and train the anomaly detector
- When new policies are added, we run our pipeline to check if we find an anomaly
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>employees</th>
<th>Total number of</th>
<th>roles</th>
<th>Number of collections</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>policies</td>
<td>users</td>
<td>groups</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12,000</td>
<td>842</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>130</td>
<td>812</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>826</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets
  - Dataset 1 & 2 are SSO users

<table>
<thead>
<tr>
<th>Dataset</th>
<th>employees</th>
<th>Total number of</th>
<th>Number of collections</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>policies</td>
<td>users</td>
<td>groups</td>
</tr>
<tr>
<td>1</td>
<td>12,000</td>
<td>842</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>130</td>
<td>812</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>826</td>
<td>2</td>
</tr>
</tbody>
</table>
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets
  - Dataset 1 & 2 are SSO users
  - Data was periodically collected using our tool

<table>
<thead>
<tr>
<th>Dataset</th>
<th>employees</th>
<th>Total number of</th>
<th>Number of collections</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>policies</td>
<td>users</td>
<td>groups</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets
- Compared with rule-based Cloud Custodian

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total number of</th>
<th>Number of collections</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>employees</td>
<td>policies</td>
</tr>
<tr>
<td>1</td>
<td>12,000</td>
<td>842</td>
</tr>
<tr>
<td>2</td>
<td>130</td>
<td>812</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>826</td>
</tr>
</tbody>
</table>
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets
- Compared with rule-based Cloud Custodian
- Increased detection of misconfigurations

<table>
<thead>
<tr>
<th>DS</th>
<th>Misconf.</th>
<th>Our approach</th>
<th>Cloud Custodian</th>
<th>Cloud Custodian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>F1-score</td>
</tr>
<tr>
<td>1</td>
<td>66.67%</td>
<td>66.67%</td>
<td>66.67%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>70.00%</td>
<td>63.34%</td>
<td>66.67%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>75.00%</td>
<td>50.00%</td>
<td>60.00%</td>
<td></td>
</tr>
</tbody>
</table>

Overall:
- Our approach: 91.58% | 91.58% | 91.58%
- Cloud Custodian (All rules): 97.93% | 97.60% | 97.76%
- Cloud Custodian (Selected rules): 98.99% | 98.98% | 98.57%

Increasing detection of misconfigurations.
Evaluation - Are anomalies misconfigurations?

- Evaluated on 3 real-world datasets
- Compared with rule-based Cloud Custodian
- Increased detection of **misconfigurations** but more **FPs**

<table>
<thead>
<tr>
<th>Misconf.</th>
<th>DS</th>
<th>Our approach</th>
<th>Cloud Custodian All rules</th>
<th>Cloud Custodian Selected rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>F1-score</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>66.67%</td>
<td>66.67%</td>
<td>66.67%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>70.00%</td>
<td>63.34%</td>
<td>66.67%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>75.00%</td>
<td>50.00%</td>
<td>60.00%</td>
</tr>
<tr>
<td>Overall</td>
<td>1</td>
<td>91.58%</td>
<td>91.58%</td>
<td>91.58%</td>
</tr>
<tr>
<td>Overall</td>
<td>2</td>
<td>92.03%</td>
<td>92.31%</td>
<td>92.15%</td>
</tr>
<tr>
<td>Overall</td>
<td>3</td>
<td>94.97%</td>
<td>95.45%</td>
<td>95.03%</td>
</tr>
</tbody>
</table>
**Evaluation - Are anomalies misconfigurations?**

- Evaluated on 3 real-world datasets.
- Compared with rule-based Cloud Custodian.
- Increased detection of misconfigurations but more FPs.

<table>
<thead>
<tr>
<th></th>
<th>Our Approach</th>
<th>Cloud Custodian All rules</th>
<th>Cloud Custodian Selected rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DS</td>
<td>Prec.</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>66.67%</td>
<td>66.67%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>70.00%</td>
<td>63.34%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>75.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Misconf.</td>
<td>1</td>
<td>91.58%</td>
<td>91.58%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>92.03%</td>
<td>92.31%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>94.97%</td>
<td>95.45%</td>
</tr>
<tr>
<td>Overall</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

Using anomaly detection in IAM policies:

- **Increases** the number of detected **misconfigurations**
- **Incorrectly** flags **slightly more** policies than rule-based solutions
- Requires **fewer** manual steps than rule-based solutions

https://github.com/utwente-scs/misdet-code
Questions?

Using anomaly detection in IAM policies:

- **Increases** the number of detected **misconfigurations**
- **Incorrectly flags slightly more** policies than rule-based solutions
- Requires **fewer** manual steps than rule-based solutions

[https://github.com/utwente-scs/misdet-code](https://github.com/utwente-scs/misdet-code)

Thijs van Ede
✉️ t.s.vanede@utwente.nl
🐦 @EdeThijs