Log2Policy: An Approach to Generate Fine-Grained Access Control Rules for Microservices from Scratch

Shaowen Xu$^{1,2}$, Qihang Zhou$^1$, Heqing Huang$^1$, Xiaoqi Jia$^{1,2}$, Haichao Du$^1$, Yang Chen$^{1,2}$, Yamin Xie$^1$

$^1$Institute of Information Engineering, Chinese Academy of Sciences
$^2$School of Cyber Security, University of the Chinese Academy of Sciences
Microservices

• Applications are divided into services.
• Services communicate with each other by remote procedure calls (RPCs).
• Improve flexibility, maintainability and serviceability

But, new attack surface introduced.
Motivation

Attackers can exploit vulnerabilities to take over a microservice and send arbitrary requests to others!

Can be solved by an **ACCESS CONTROL** mechanism!
Motivation

• It’s hard to apply access control rules manually
  Error-prone, microservice system updates frequently;

• Generate access control rules automatically
  • Documention based approaches
    Low Coverity, coarse-grained;
  • Source code based approaches
    Attributes Limited, depending on the quality of code;
  • Historical data based approaches
    Unable to handle upgrade and generate from scratch;
Threat Model

• Test phase:
  • Microservices are trustworthy.
  • No attackers within the internal development team.
  • Using TPM to secure boot.

• Production phase:
  • Can be compromised.
  • Attacks can send requests with arbitrary parameters to any microservice.
  • Development team may update the microservice application.

• Out of scope
  • Side-channel attacks.
  • DoS attacks.
Log2Policy’s Goal

Provide a tool to generate fine-grained access control rules for microservices applications from scratch and update the rules rapidly.
Log2Policy’s Goal

Provide a tool to generate fine-grained access control rules for microservices applications from scratch and update the rules rapidly.
Rules Generation

**Step-I: Data pre-processing**
Sampling and handling missing values.

**Step-II: Topology generation**
A topological graph consists of service nodes, version nodes, version edges and invocation edges.

<table>
<thead>
<tr>
<th>Roles</th>
<th>Attributes</th>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requester</td>
<td>Source</td>
<td>src</td>
<td>The source service of this traffic.</td>
</tr>
<tr>
<td>Requester</td>
<td>Source Version</td>
<td>src_ver</td>
<td>The source service version of this traffic.</td>
</tr>
<tr>
<td>Object</td>
<td>Destination</td>
<td>dest</td>
<td>The destination service of this traffic.</td>
</tr>
<tr>
<td>Object</td>
<td>Destination Port</td>
<td>port</td>
<td>The port which this traffic accessed.</td>
</tr>
<tr>
<td>Object</td>
<td>Destination Path</td>
<td>path</td>
<td>The path of this traffic.</td>
</tr>
<tr>
<td>Session</td>
<td>Protocol</td>
<td>proto</td>
<td>The protocol of this traffic.</td>
</tr>
<tr>
<td>Session</td>
<td>Method</td>
<td>mthd</td>
<td>The method of this traffic.</td>
</tr>
</tbody>
</table>
Why we need Attributes Mining?

Variables exist in attributes.
Rule Generation

• **Step-III: Attributes mining**

  - **String matching or statistics**
    Lack flexibility, difficult to apply;

  - **Skip-gram + DBSCAN**
    NLP-based methods are widely used to analyze URLs. Instances of variables can be regarded as synonyms.

  The instances of variables have similar word representation, and if words are clustered, they will be clustered into the same cluster.

• **Step-IV: Policy optimization**

  - Istio don’t support regular expressions based exact match.

  - We propose to use Istio’s virtual service mechanism.
Policy Updating Design Goals

• Goal-1: Only analyze logs generated by newly updated services or versions when upgrading access control rules.

• Goal-2: The system needs to be protected by the original authorization policy when generating new logs.
Policy Updating

Version Removal
- Remove all associated versions;
- Update the topological graph;

Service Removal
- Remove all its versions;
- Update the topological graph;

Version Addition
- Istio’s traffic mirror mechanism;
- New version don’t affect the original system;

Service Addition
- A free namespace;
- Collect logs;
Evaluation

• **Microservices applications:**
  
  **Book-info**: a sample example microservice application provided by Istio.
  
  **Online-Boutique**: e-commerce microservices application.
  
  **Sock Shop**: e-commerce microservices application with relatively complex internal logic.
  
  **Pitstop**: an event-driven microservice application with the main function of managing appointments.
  
  **Mesh demo**: an e-commerce application with traffic control provided by Tencent Cloud.

• **Environment:**
  
  **Local environment**: Minikube v1.24.0 with Kubernetes v1.22.3 and Istio v1.13.2;
  
  **Cloud mesh platform**: Tencent Cloud with Kubernetes v1.22.5 and Istio v1.14.5;
  
  **Log analysis**: Eight 4.20-GHz Intel(R) Core (TM) CPUs (i7-7700k) and 16GB of RAM;
  
  **Attributes mining**: Pytorch and Sci-learn to implement algorithm, eight Intel Xeon Cascade Lake CPU (2.50-GHz), 32GB of RAM and an NVIDIA T4 GPU;
Evaluation

1. Log2Policy generates access control rules based on logs, can it cover all the normal behaviors of the microservice application?
2. Can Log2Policy generate access control rules from scratch? How is the quality of the generated rules?
3. Can Log2Policy improve the efficiency when generating rules, especially when access control policy needs to be upgraded?

- Collects the logs generated by microservices during the testing phase
- Use the test scripts provided by the applications to simulate the testing behaviors
- Combine the test scripts of the microservices that interact with users with the Locust framework.
Q1: Log2Policy generates access control rules based on logs, can it cover all the normal behaviors of the microservice application?

- Request identification rate: 100%
- Attributes extraction rate: 100%
- Rules Coverage in large microservice application: 100%
Q2:
Can Log2Policy generate access control rules from scratch? How is the quality of the generated rules?

- All the attacks are blocked.
- All normal access is forwarded.
Q3:
Can Log2Policy improve the efficiency when generating rules, especially when access control policy needs to be upgraded?

- Compared to AUTOARMOR, Log2Policy reduce at least 59% analysis time.
- Log2Policy’s update mechanism increase at least 3.17 times of analysis speed compared to the baseline.
Log2Policy

A tool can generate **fine-grained** access control rules **from scratch** based on access logs for microservices applications.

- Mine attributes from microservice logs using the *word2vec* technique and the *DBSCAN* algorithm.
- A mechanism for updating microservice access control rules based on traffic management.
- Evaluate *Log2Policy* with five real-world microservice applications.
Thanks!

xushaowen@iie.ac.cn