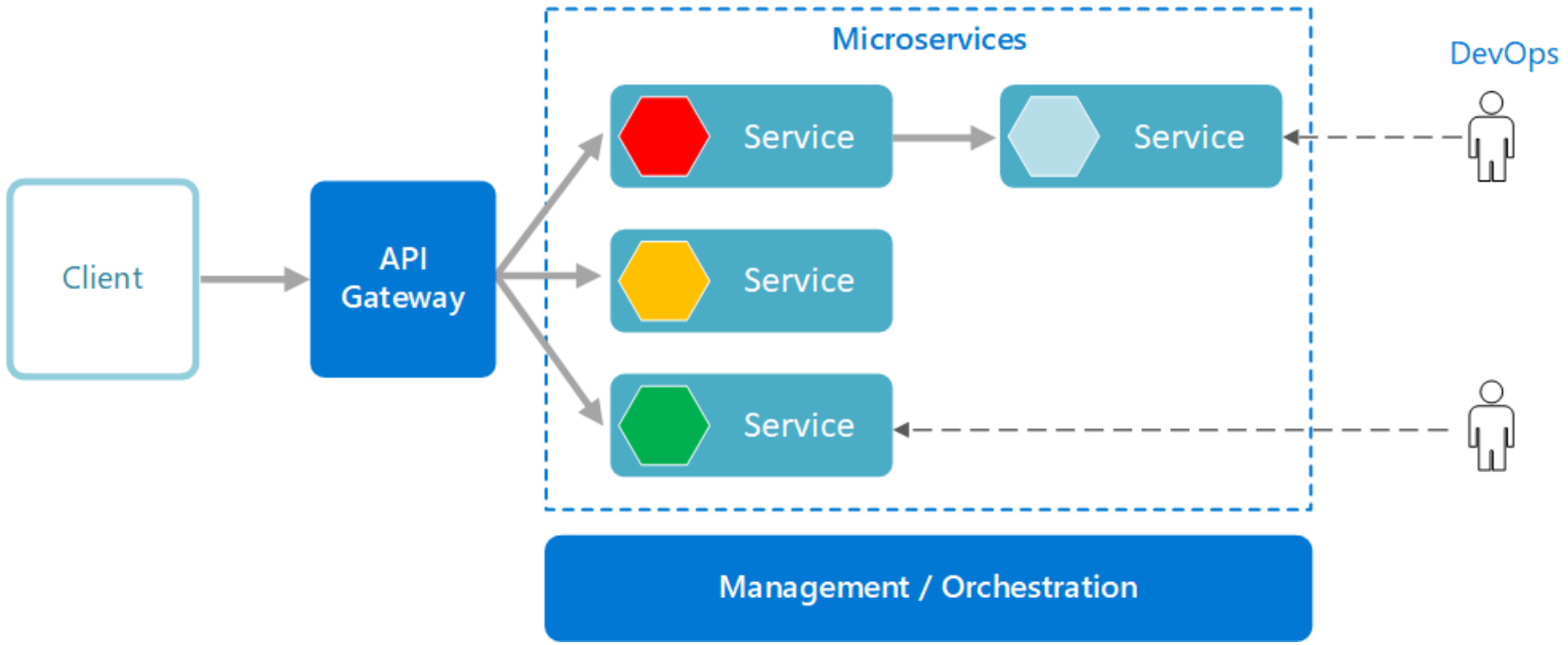
Log2Policy: An Approach to Generate Fine-Grained Access Control Rules for Microservices from Scratch

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

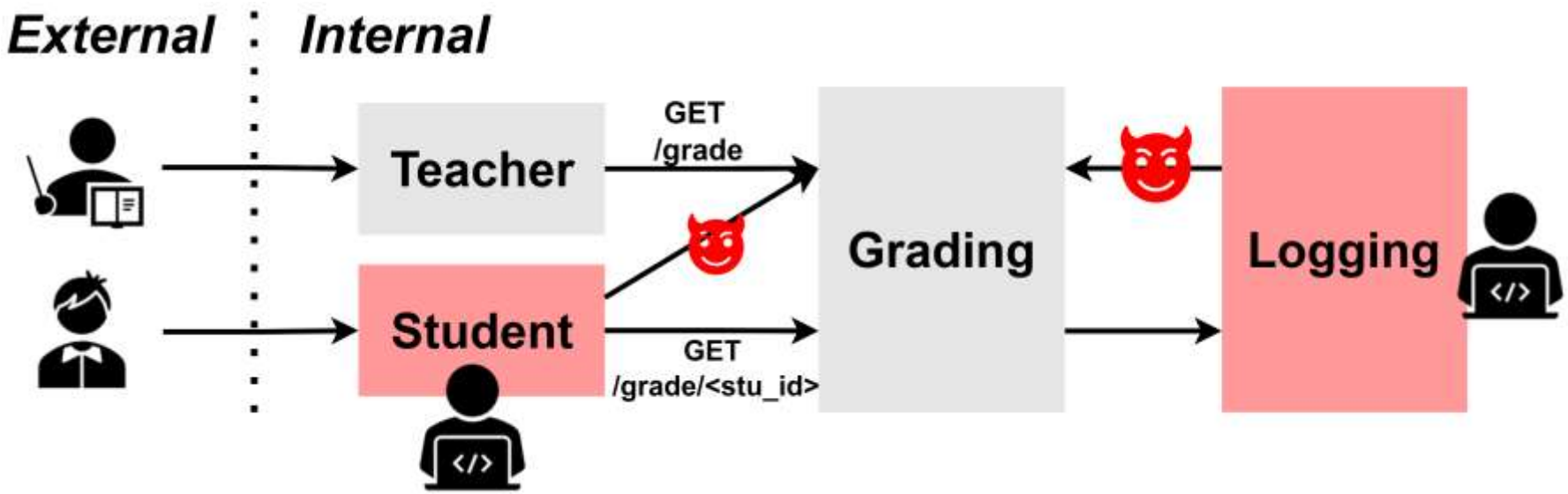
Microservices



- 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

Rules Generation

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

Policy Updating

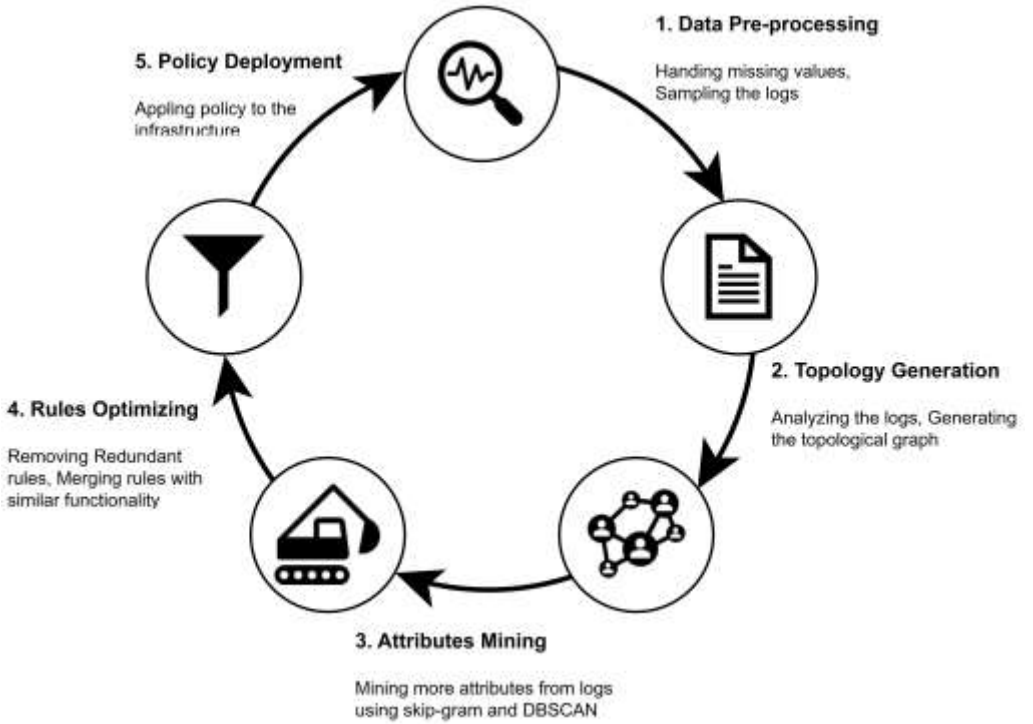
Rules Generation

Step-I: Data pre-processing

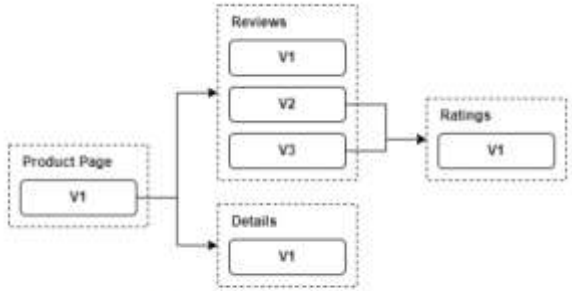
Sampling and handling missing values.

Step-II: Topology generation

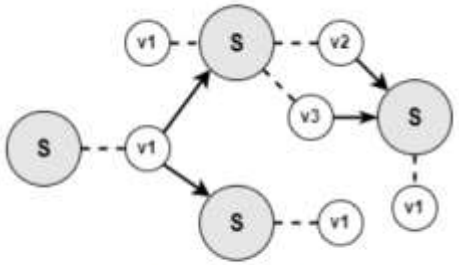
A topological graph consists of service nodes, version nodes, version edges and invocation edges.



Roles	Attributes	Abbreviation	Explanation
Requester	Source	src	The source service of this traffic.
Requester	Source Version	src_ver	The source service version of this traffic.
Object	Destination	dest	The destination service of this traffic.
Object	Destination Port	port	The port which this traffic accessed.
Object	Destination Path	path	The path of this traffic.
Session	Protocol	proto	The protocol of this traffic.
Session	Method	mthd	The method of this traffic.



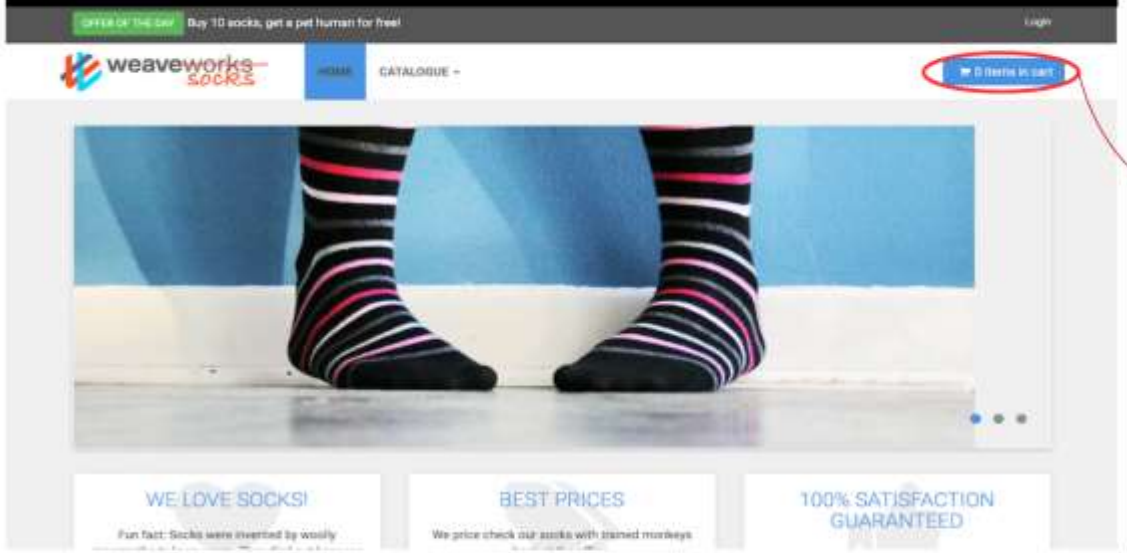
(a) Book-info Application



(b) Topological Graph of Book-info

Why we need Attributes Mining?

Front-end



front-end/api/cart/index.js

```
1  app.get('/cart', function (req, res, next) {
2      console.log('Request received: ' + req.url + ',' + req.query.custId)
3      var custId = helpers.getCustomerId(req, app.get('env'))
4      console.log('Customer ID: ' + custId)
5      request(
6          endpoints.cartsUrl + '/' + custId + '/items',
7          function (error, response, body){
8              if (error) {
9                  return next(error)
10             }
11             helpers.respondStatusBody(res, response.statusCode, body)
12         }
13     );
14 });
```

Legend

path

variable

HTTP Request

GET /carts/<custId>/items

LOG

(front-end, v1, carts, 80, /carts/**76d8e**/items)
(front-end, v1, carts, 80, /carts/**76e3d**/items)

Variables exist in attributes.

Rule Generation

- ***Step-III: Attributes mining***

~~✗ *Stirng 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***

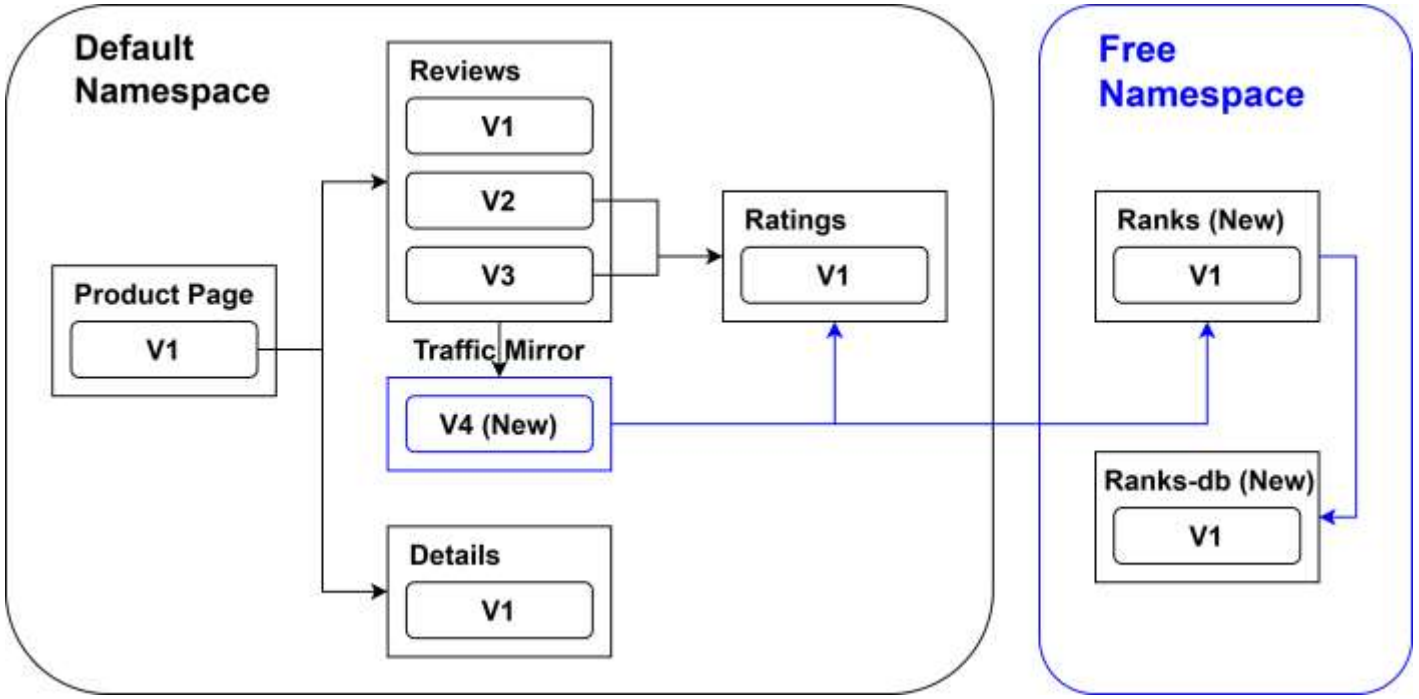
- Istio don't support regular expressions based exact match.

- 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.

Application	Microservice	RPCs	RPCs Identified by Log2Policy	
			Amount	Attributes
Book-info	Reviews	1	1(100%)	1(100%)
	Ratings	2	2(100%)	2(100%)
	Details	1	1(100%)	1(100%)
Online Boutique	EmailService	1	1(100%)	1(100%)
	PaymentService	1	1(100%)	1(100%)
	ShippingService	3	3(100%)	3(100%)
	CurrencyService	3	3(100%)	3(100%)
	AdService	1	1(100%)	1(100%)
	CheckoutService	1	1(100%)	1(100%)
	ProductCatalogService	4	4(100%)	4(100%)
	CartService	4	4(100%)	4(100%)
	RecomendationService	1	1(100%)	1(100%)
Sock Shop	carts	6	6(100%)	6(100%)
	carts-db	1	1(100%)	1(100%)
	catalogue	4	4(100%)	4(100%)
	catalogue-db	1	1(100%)	1(100%)
	orders	2	2(100%)	2(100%)
	orders-db	1	1(100%)	1(100%)
	payment	1	1(100%)	1(100%)
	shipping	1	1(100%)	1(100%)
	user	10	10(100%)	10(100%)
	user-db	1	1(100%)	1(100%)
Pitstop	logserver	9	9(100%)	9(100%)
	rabbitmq	8	8(100%)	8(100%)
	sqlserver	6	6(100%)	6(100%)
	mailservice	1	1(100%)	1(100%)
	vehiclemanagementapi	2	2(100%)	2(100%)
	customermanagementapi	2	2(100%)	2(100%)
	workshopmanagementapi	7	7(100%)	7(100%)
Tencent Mesh demo	product	1	1(100%)	1(100%)
	user	1	1(100%)	1(100%)
	stock	2	2(100%)	2(100%)
	cart	3	3(100%)	3(100%)
	order	1	1(100%)	1(100%)
Total		97	97(100%)	97(100%)

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%

# of microservices	# of instances	Sampling Rate	# of Rules	Rules Coverage
20	5	10%	18	100%

Microservice	Request Type	Policy Generate		Policy Update	
		Total	# of Blocks	Total	# of Blocks
Book-info	Normal	129798	0	13575	0
	A1 Attack	30	30	33	33
	A2 Attack	12	12	3	3
Online boutique	Normal	164778	0	16547	0
	A1 Attack	288	288	90	90
	A2 Attack	80	80	12	12
Sock shop	Normal	109847	0	11356	0
	A1 Attack	294	294	90	90
	A2 Attack	128	128	12	12
Pitstop	Normal	156678	0	12114	0
	A1 Attack	330	330	108	108
	A2 Attack	93	93	21	21
Mesh-demo	Normal	115819	0	13695	0
	A1 Attack	72	72	45	45
	A2 Attack	32	32	12	12
Total	Normal	676920	0(0%)	67287	0(0%)
	Attack	1359	1359(100%)	324	324(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.

Microservice	*AutoArmor*	*Log2Policy*				Decrease Percent
		Topology Generation	Attributes Mining	Rules Optimization	Total	
Book-info	79s	5.46s	0	0.01s	5.47s	93%
Online Boutique	258s	9.01s	0	0.01s	9.02s	96%
Sock Shop	360s	30.17s	117s	0.01s	147.18s	59%
Pitstop	266s	12.69s	31s	0.01s	43.70s	84%

Microservice	Baseline	Log2Policy	Speed Increase
Book-info	5.85s	0.78s	7.5x
Online Boutique	18.70s	2.58s	7.2x
Sock Shop	190.32s	0.58s	328.1x
Pitstop	48.79s	1.97s	24.7x
Mesh Demo	4.41s	1.39s	3.17x

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