Towards automated inter-service authorization for microservice applications

Xing Li[1], Yan Chen[2] and Zhiqiang Lin[3]

[1]Zhejiang University, [2]Northwestern University, [3]The Ohio State University

Background

- **Microservices need inter-service authorization.**
  - Network-based inter-service communication is a potential attack surface.
  - Services may be compromised due to container image vulnerabilities, etc.
  - Compromised microservices can send malicious requests to other services to initiate attacks or steal data.
- **Current inter-service authorization mechanisms is not practical.**
  - These mechanisms still rely on the administrator's manual configuration.
  - They use complex policies for fine-grained authorization.
  - The large scale and frequent iteration nature of microservice applications make the manual method unrealistic.

➡ **Automated Inter-service authorization**

Objectives

1. Completeness
Automatically gaining complete invocation logics among microservices.

2. Fine Granularity
Mining the detailed attributes of the inter-service invocations.

3. Agility
Dynamically adjusting the policies based on the changes in microservices.

Solutions	Completeness	Fine Granularity	Agility
Document-based approaches	×	×	✓
History-based approaches	×	✓	×
Model-based approaches	✓	✓	×
JARVIS (proposed approach)	✓	✓	✓

Table 1. Comparison of JARVIS with existing security policy automation works.

Assumptions

- The administrator **is trusted.**
- The source code of microservices **is trusted**.
- The behaviors that **violate code logic** are considered malicious.
- The source code of microservices **can be obtained.**

Architecture

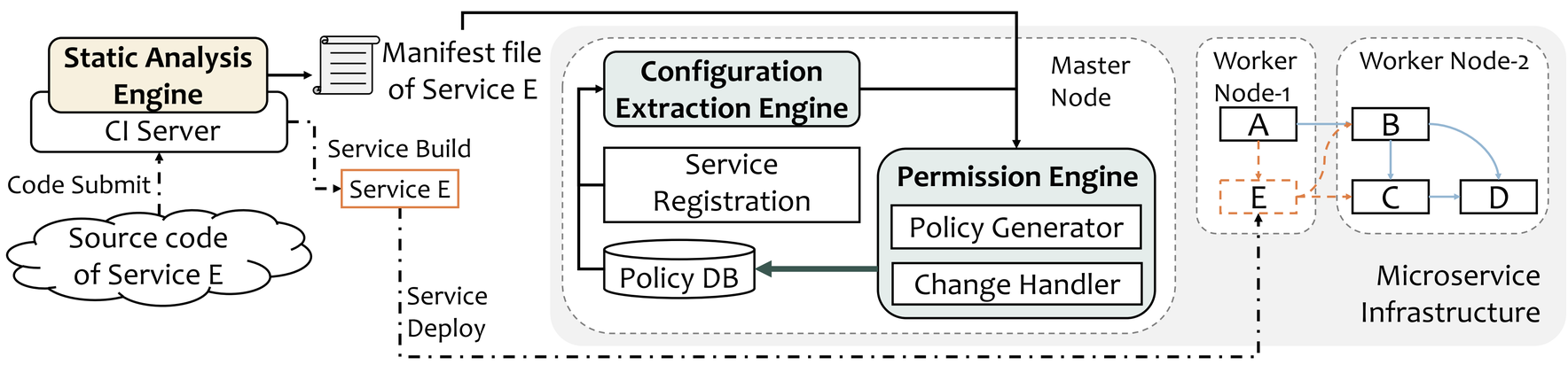
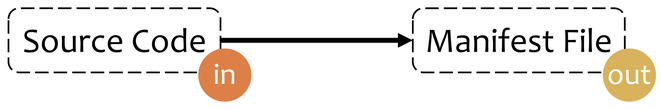


Figure 1. The system architecture of JARVIS.

Methodology

1. Request Extraction

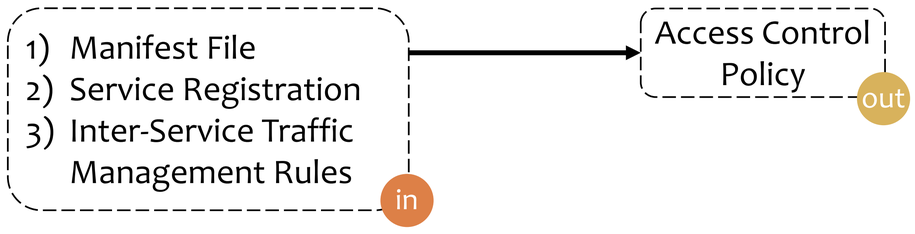


I. **Identifying the statements** that initiate network API invocations.
II. **Performing program slicing** form these statements.
III. **Extracting the details** of the invocations from program slices.

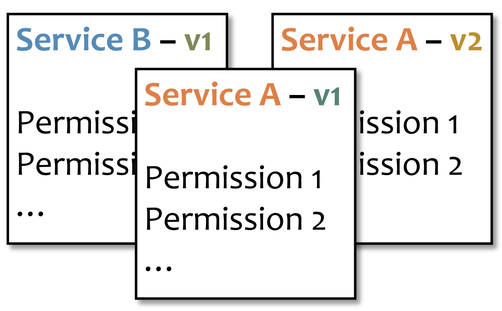
```
...
id = 20                                   Backward Taint
...                                       Propagation
endpoint  = "http://example:9080"
...
url = endpoint + "/" + str(id)
res = requests.get(url)
...
```

Figure 2. Program slicing.

2. Policy Generation

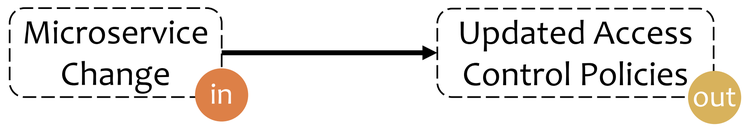


1) **Manifest File:** What requests a microservice may initiate.
2) **Service Registration:** What APIs a microservice provides.
3) **Inter-Service Traffic Management Rules:** Where the requests will eventually arrive.



- Service-based policy aggregation
- Version-aware policy management

3. Policy Updating



Category	Object	Operation
Microservice Update	New Version	Deploy
	Old Version	Delete
Inter-Service Traffic Management Rule Change	New Rule	Apply
	Old Rule	Delete
	Old Rule	Update

Table 2. Changes in microservice applications.

Two-stage change handling

I. Quickly determining if the change will affect the access control policies.
II. Incrementally adjusting the access control policies if they need to be modified.

Preliminary Result

- **Benchmarks:** Bookinfo, Hipster Shop, and Sock Shop
- **Ground Truth:** Manual analysis

Language	# of Microservices	Extracted Requests		
		HTTP	gRPC	TCP
Java	6	5 (100%)	-	2 (100%)
Python	3	3 (100%)	1 (100%)	-
Go	7	-	19 (100%)	28 (100%)
JavaScript	4	28 (100%)	-	2 (100%)
Ruby	1	1 (100%)	-	-
C#	1	-	-	7 (100%)

Table 3. The coverage of request extraction for 22 microservices developed in 6 languages.

Discussion

- **Unable to obtain the source code of microservices. (not common)**
  - Requesting the manifest files from the service providers.
  - Manually configuring access control policies.
  - Reverse engineering.
- **The source code can not be trusted.**
  - Involving the administrator in the review of manifest files.
- **Incomplete request extraction.**
  - The number of invocation protocols and corresponding libraries is limited.
  - The administrator can add semantic models for their dedicated libraries.

Conclusion

✓ **JARVIS**: The **first** automated inter-service authorization mechanism
- A static-analysis based request extraction mechanism
- A fine-grained policy generation mechanism
- A two-stage change handling mechanism

✓ The preliminary result shows the completeness of request extraction

Acknowledgement

1.National Key R&D Program of China (2017YFB0801703)
2.The Key Research and Development Program of Zhejiang Province (2018C01088)

Contact Information: xing.li@zju.edu.cn