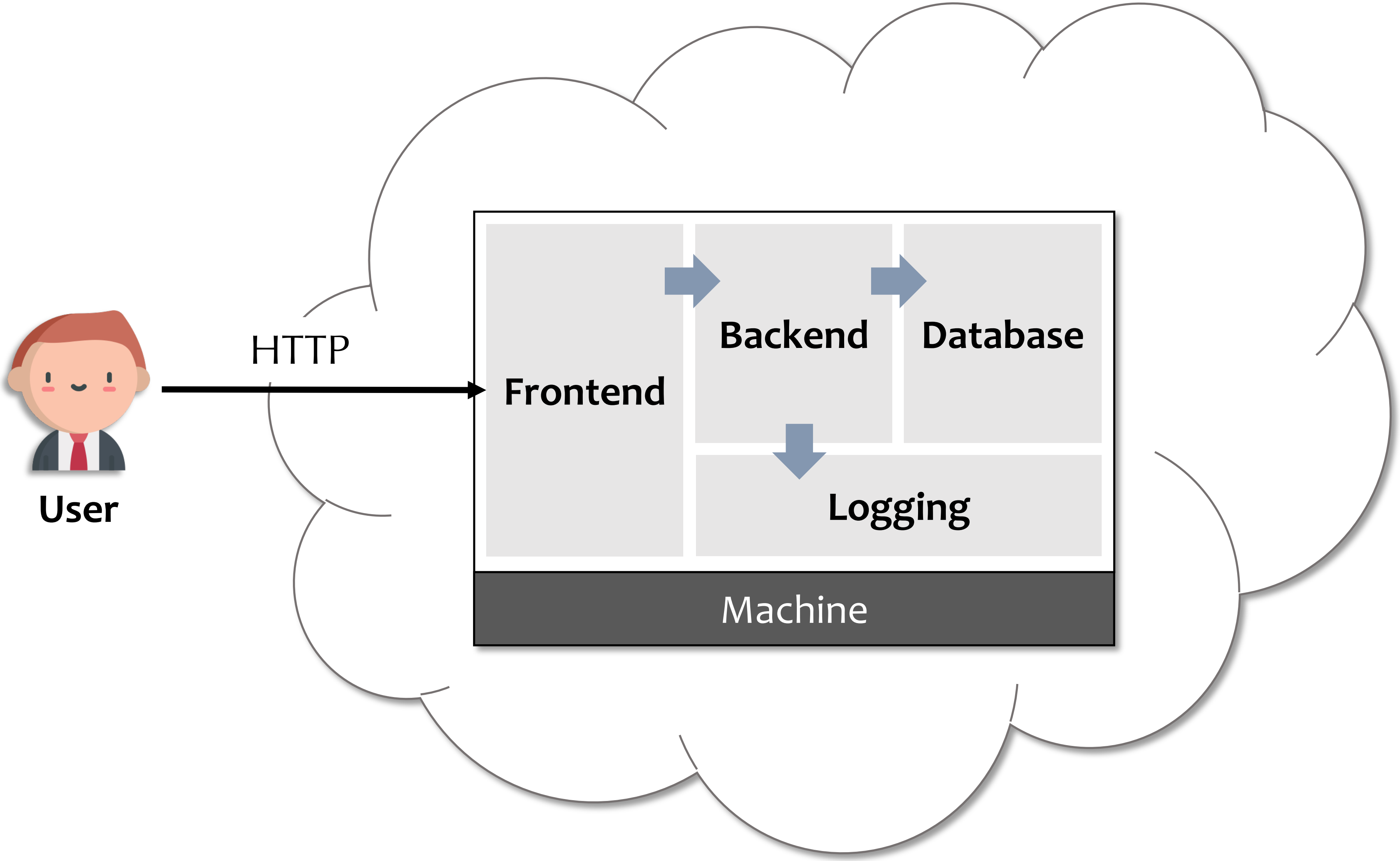
Automatic Policy Generation for Inter-Service Access Control of Microservices

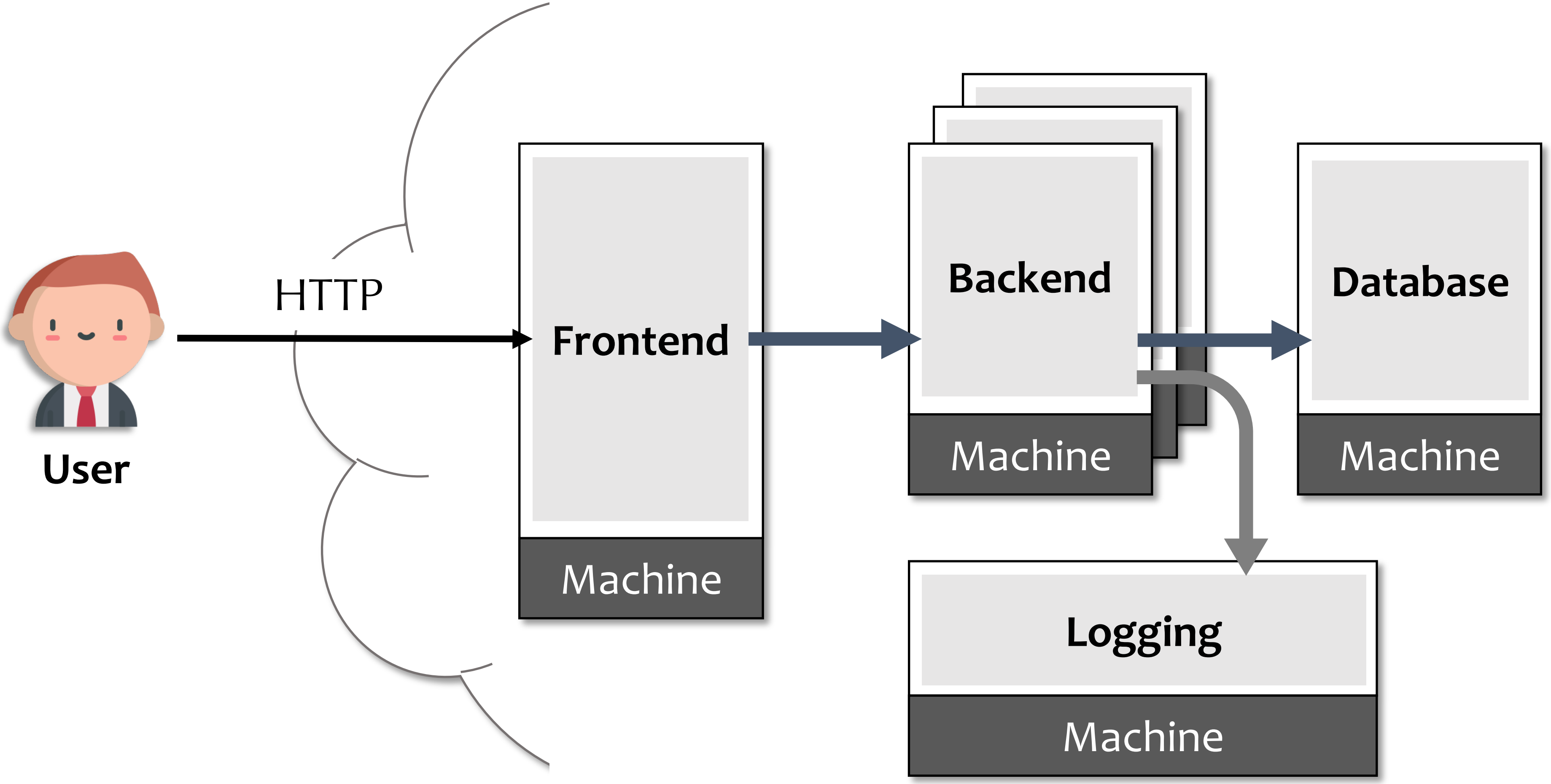
Xing Li[1,2], Yan Chen[2], Zhiqiang Lin[3], Xiao Wang[2], and Jim Hao Chen[2]

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

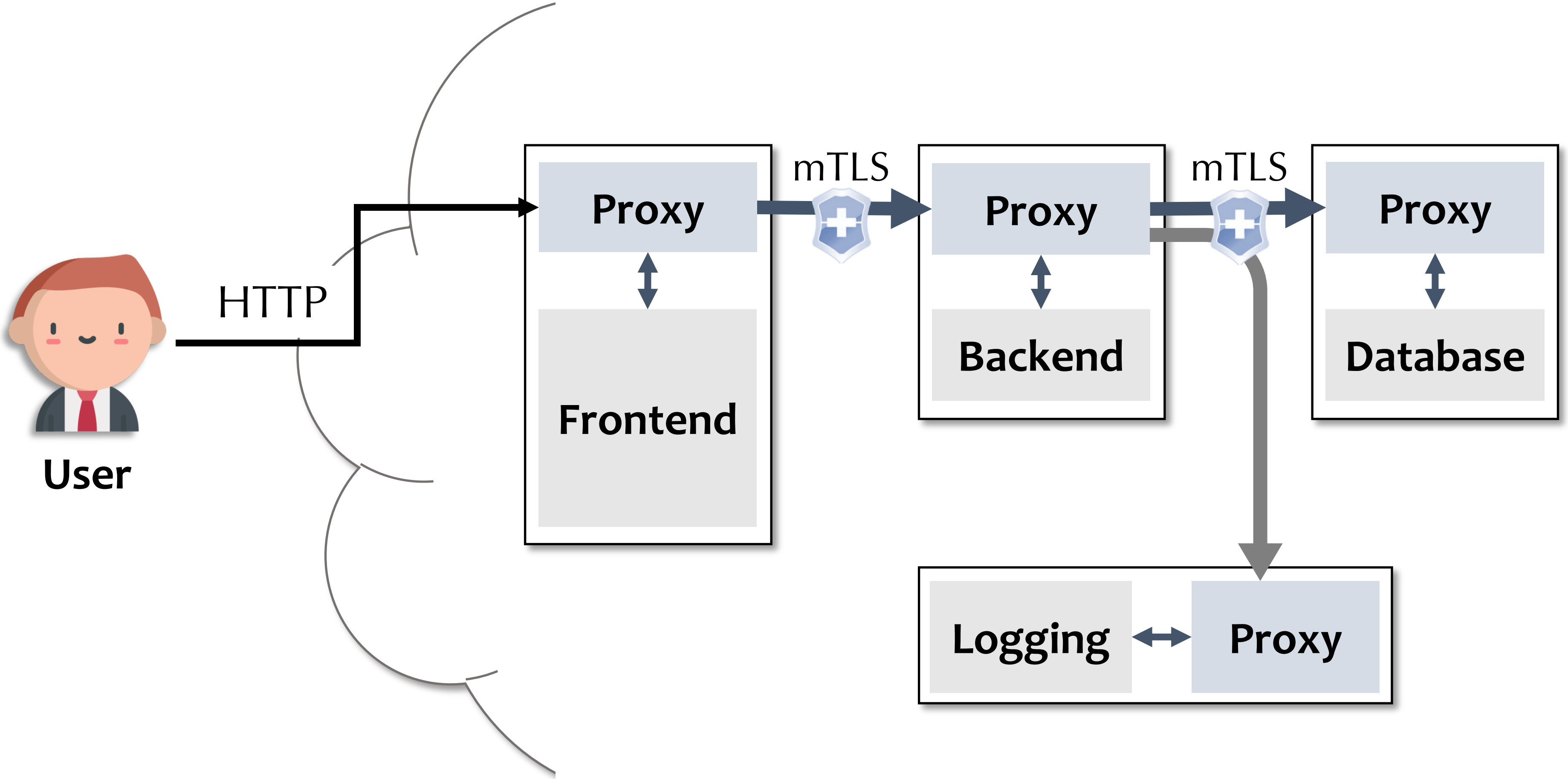
A Cloud Application



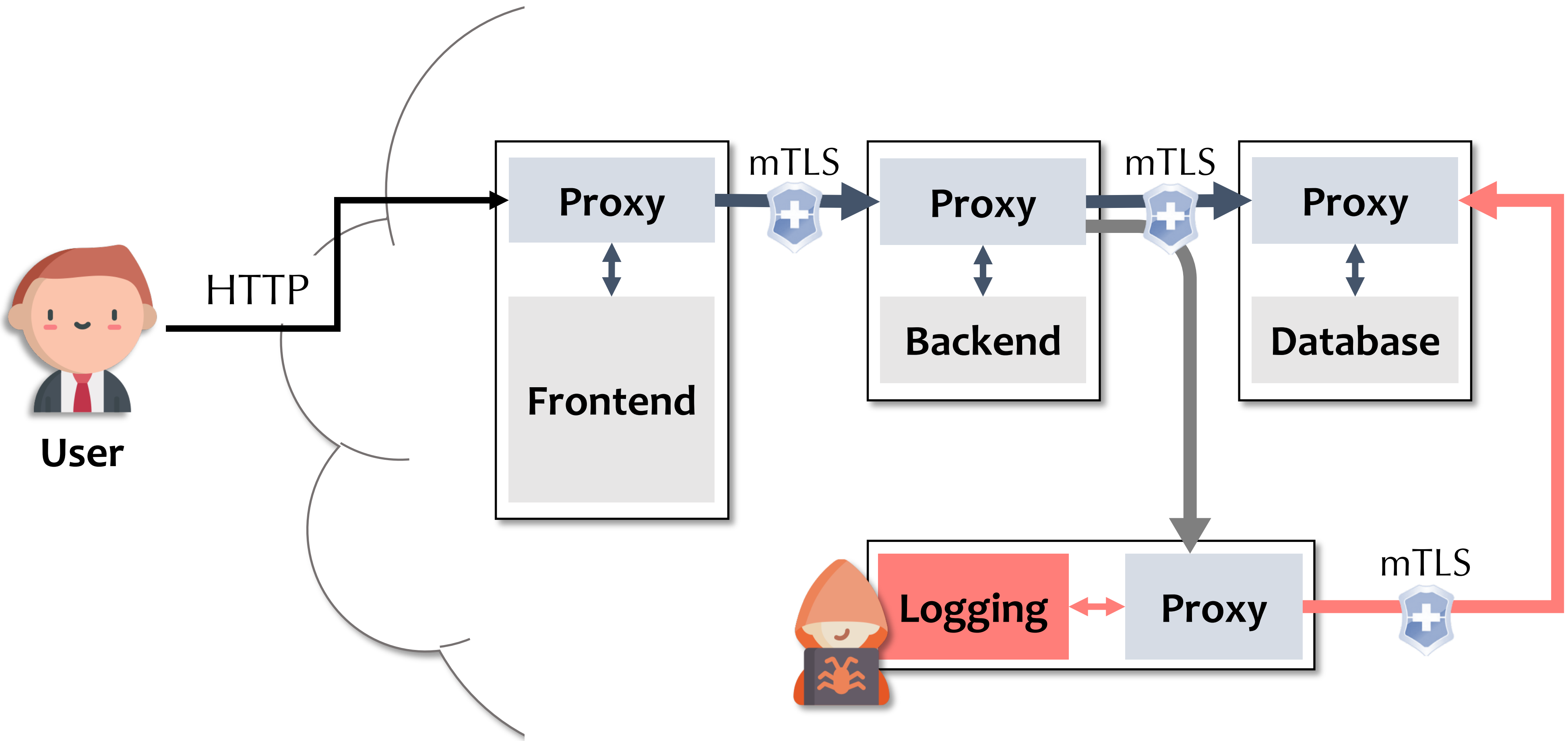
A Cloud Application



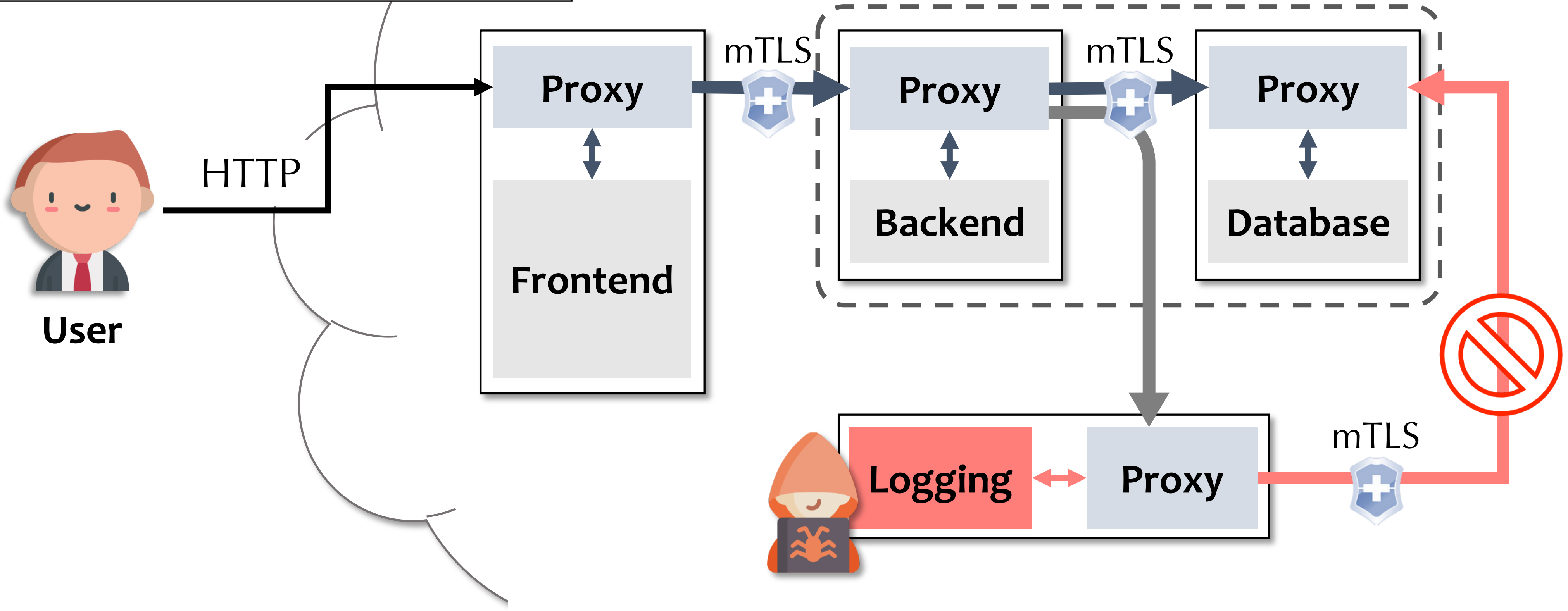
A Cloud Application

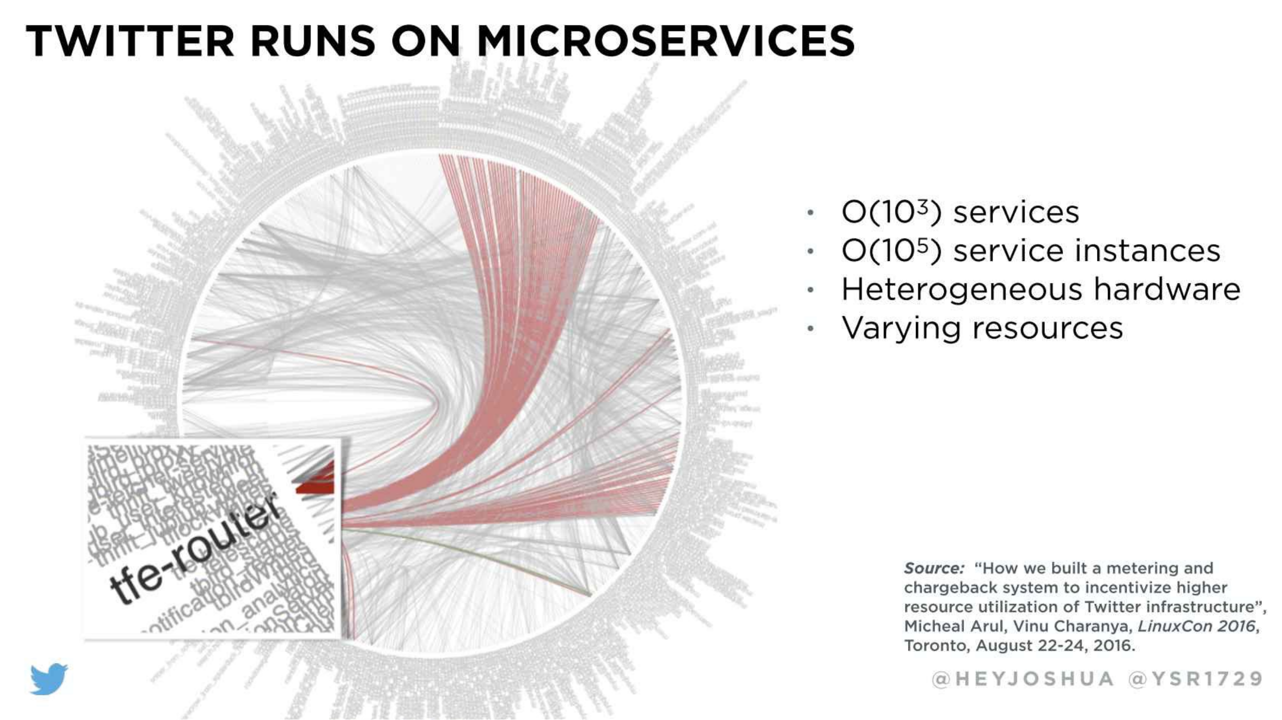


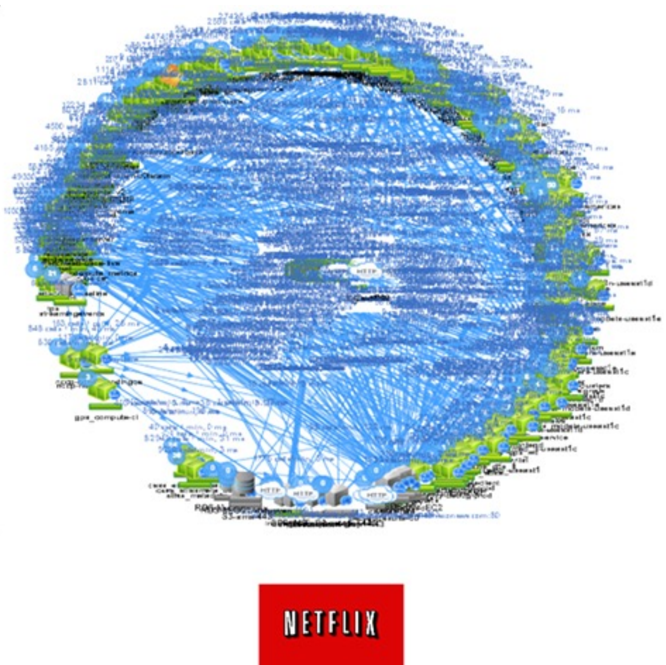
A Cloud Application



```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: backend-v1-to-database
  namespace: default
spec:
  selector:
    matchLabels:
      app: database
  rules:
  - from:
    - source:
        principals:["cluster.local/ns/default/sa/backend"]
    to:
    - operation:
        ports: ["9000"]
```







- Manual Policy Configuration?

  Time-consuming, error-prone, inflexible

- Automatic Security Policy Generation Approaches for Distributed Systems ?

  1. Document-based approaches ?

     Low accuracy, poor availability

  2. History-based approaches ?

     Requiring complete historical data

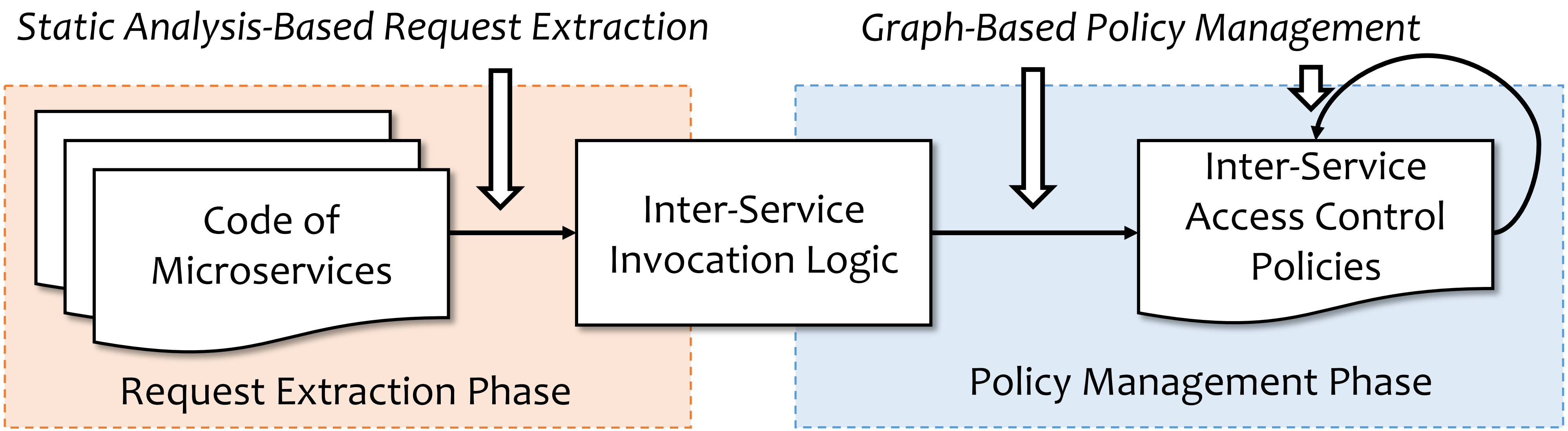
  3. Model-based approaches ?

     Poor agility and scalability

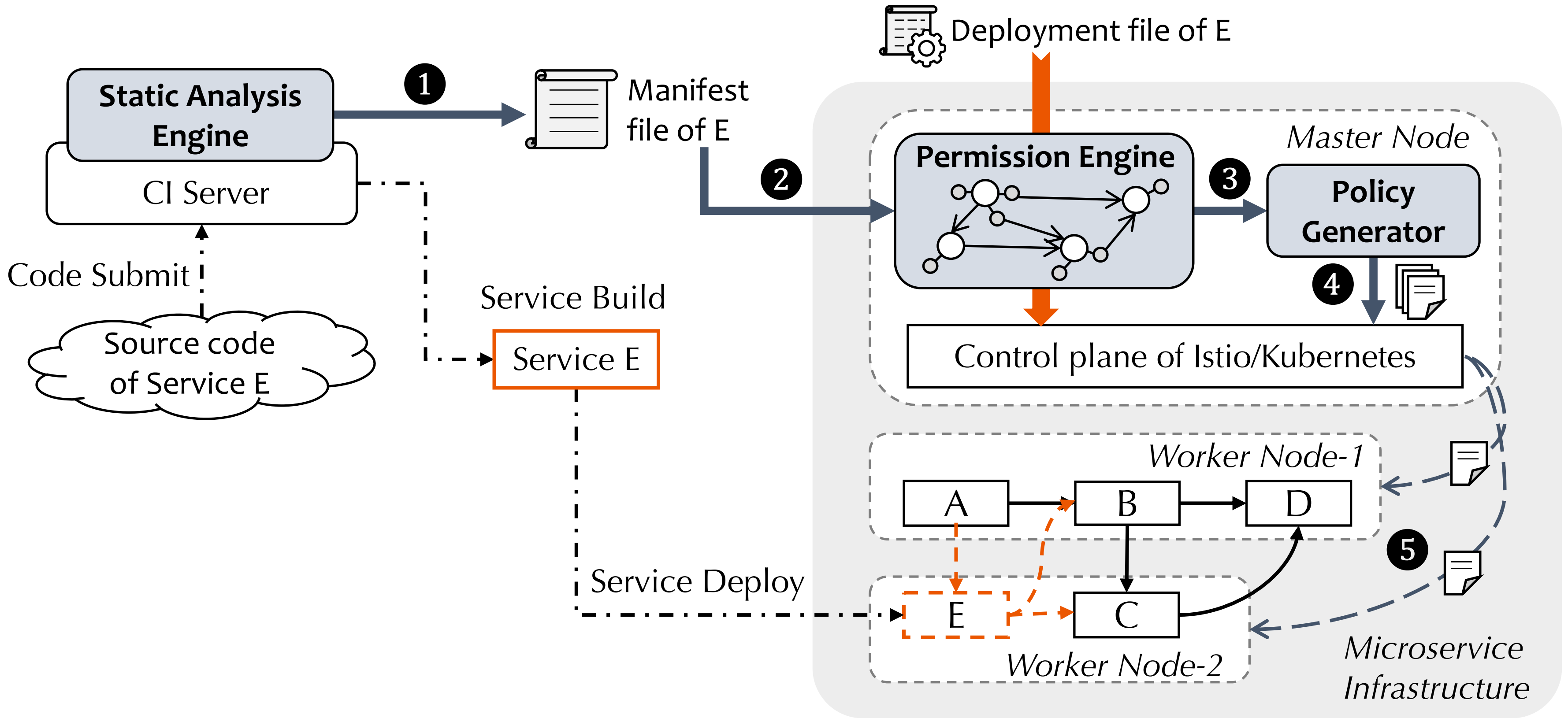
What's new in microservices?

- Microservices are small: a single service has low internal complexity.
- The inter-service invocation manner in the same application is relatively uniform.
- The amounts of involved protocols and libraries are limited.

Extract the normal system behavior with static analysis



The deployment of microservice E



AUTOARMOR

- Static Analysis Engine
- Permission Engine
- Policy Generator

Phase 1: Request Extraction

Semantic Model

```
Library: requests
    Method: get(url, params=None, **kwargs)
            Semantics: HTTP-GET
            Key parameters: url (Semantics: HTTP-URL)
```

Step-I:

Identifying the statements that initiate inter-service invocations

Source Code

```
import requests
from flask import request, session
…
reviews = {
      "name" : "http://reviews:9080",
      "endpoint" : "reviews"
}
…
@app.route('/api/v1/products/<product_id>/reviews')
def reviewsRoute(product_id):
      headers = getForwardHeaders(request)
      user = session.get('user', ")
      status, reviews = getProductReviews(product_id, headers)
…
def getProductReviews(product_id, headers):
      try:
      url = reviews['name'] + "/" + reviews['endpoint'] + "/" + str(product_id)
    res = requests.get(url, headers=headers, timeout=3.0)
…
```

Phase 1: Request Extraction

Semantic Model

```
Library: requests
    Method: get(url, params=None, **kwargs)
            Semantics: HTTP-GET
            Key parameters: url (Semantics: HTTP-URL)
```

Source Code

```
import requests
from flask import request, session
…
reviews = {
      "name" : "http://reviews:9080",
      "endpoint" : "reviews"
}
…
@app.route('/api/v1/products/<product_id>/reviews')
def reviewsRoute(product_id):
      headers = getForwardHeaders(request)
      user = session.get('user', ")
      status, reviews = getProductReviews(product_id, headers)
…
def getProductReviews(product_id, headers):
      try:
      url = reviews['name'] + "/" + reviews['endpoint'] + "/" + str(product_id)
    res = requests.get(url, headers=headers, timeout=3.0)
…
```

Step-II:

Performing backward taint propagation to get the program slices associated with each invocation

Phase 1: Request Extraction

Program Slice

```
reviews = {
      "name" : "http://reviews:9080",
      "endpoint" : "reviews"
}
@app.route('/api/v1/products/<product_id>/reviews')
url = reviews['name'] + "/" + reviews['endpoint'] + "/" + str(product_id)
res = requests.get(url, headers=headers, timeout=3.0)
```

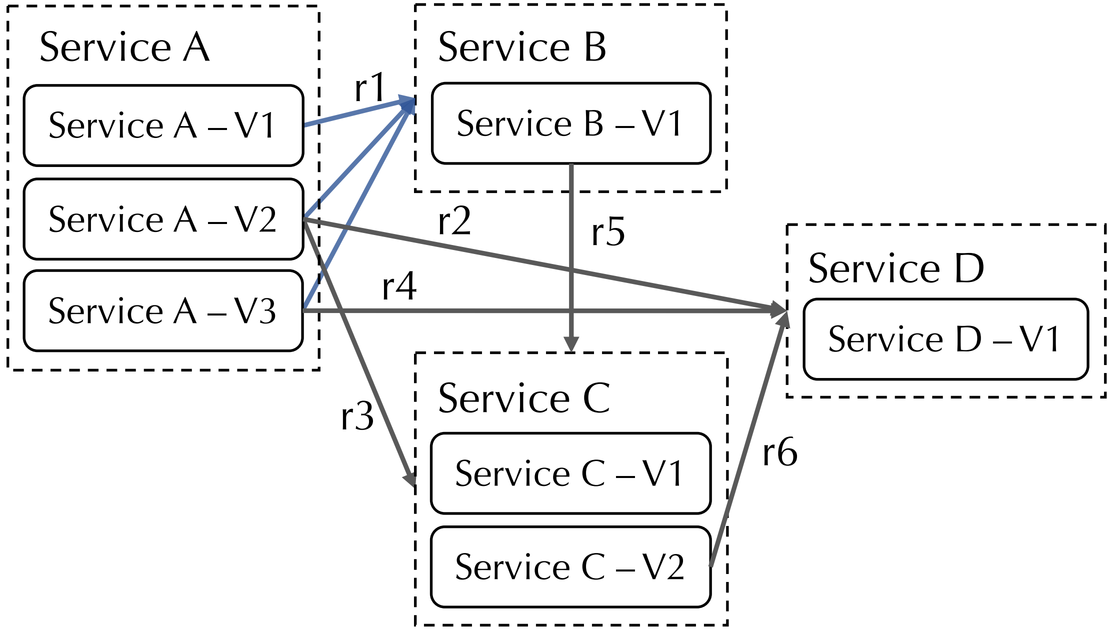
Extracted Request

```
{
    "type": "HTTP",
    "url": "http://reviews:9080/reviews/*",
    "path": "/reviews/*",
    "method": "GET"
}
```

Step-III:

Extracting the detailed attributes of invocations

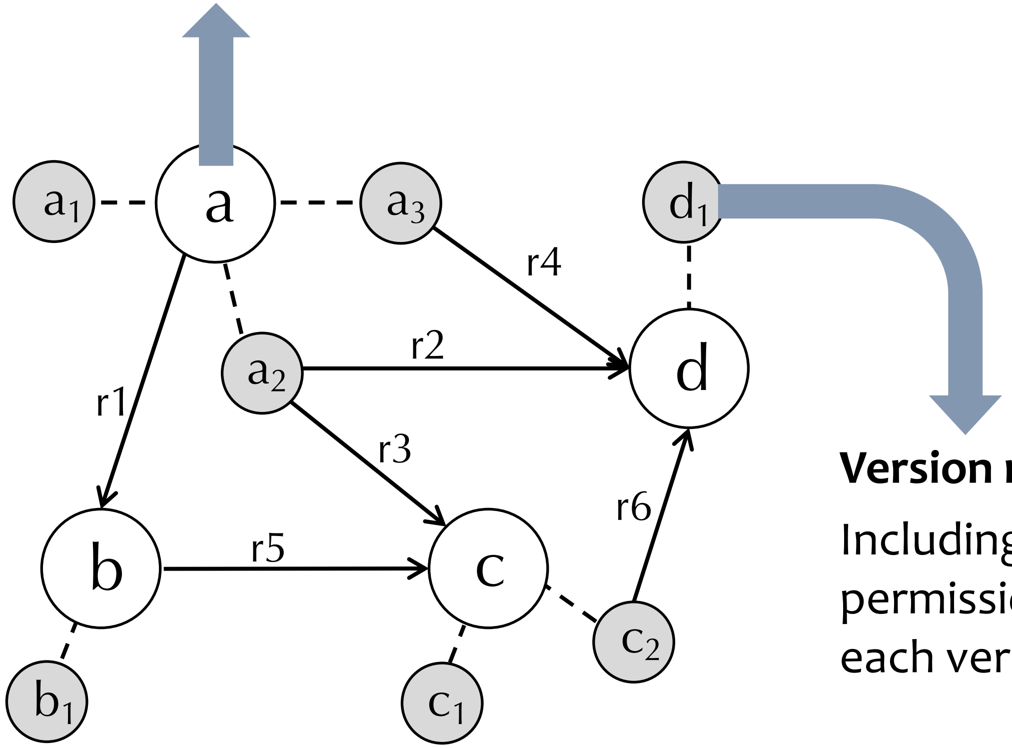
Phase 2: Policy Management



System Behavior

Service node:

Including permissions shared by all versions



Version node:

Including unique permissions for each version

Permission Graph

Integrate the permissions shared by all versions of the same service

- Eliminate redundant policies
- Eliminate unnecessary policy updates

Evaluation

- **Materials:** 5 popular open-source microservice applications

Name	# of Services	LoCs	Type	Multiple Languages	★ on GitHub
Bookinfo	6	2,702	Demo	✓	24.7k
Online Boutique	11	23,219	Demo	✓	8.8k
Sock Shop	13	20,150	Demo	✓	2.5k
Pitstop	13	45,028	Demo	✗	630
Sitewhere	21	53,751	Industrial	✗	717

- **Hardware:** a 3-node Kubernetes cluster (v.1.18.6) with Istio (v1.6.8); Each node is equipped with eight 2.30-GHz Intel(R) Core(TM) CPUs (i5-8259U) and 32 GB of RAM

Application	Microservice	Language	LoCs	Identified Requests			Extracted Attributes			Time
				HTTP	gRPC	TCP	URL	Method	Port	
Bookinfo	productpage	Python	2,061	3/3	-	-	3/3	3/3	N/A	21s
	details	Ruby	122	1/1	-	-	1/1	1/1	N/A	4s
	reviews	Java	301	1/1	-	-	1/1	1/1	N/A	27s
	ratings	JavaScript	218	-	-	2/2	2/2	N/A	2/2	27s
Online Boutique	frontend	Go	3,666	-	11/11	-	11/11	N/A	N/A	35s
	cartservice	C#	5,941	-	-	7/7	7/7	N/A	7/7	38s
	productcatalogservice	Go	2,460	-	-	-	N/A	N/A	N/A	18s
	currencyservice	JavaScript	359	-	-	-	N/A	N/A	N/A	25s
	paymentservice	JavaScript	343	-	-	-	N/A	N/A	N/A	26s
	shippingservice	Go	2,458	-	-	-	N/A	N/A	N/A	18s
	emailservice	Python	2,146	-	-	-	N/A	N/A	N/A	20s
	checkoutservice	Go	2,816	-	8/8	-	8/8	N/A	N/A	21s
	recommendationservice	Python	2,112	-	1/1	-	1/1	N/A	N/A	28s
	adservice	Java	918	-	-	-	N/A	N/A	N/A	29s
Sock Shop	front-end	JavaScript	9,922	33/33	-	-	33/33	33/33	N/A	125s
	orders	Java	2,187	6/6	-	2/2	4/8	6/6	2/2	55s
	payment	Go	863	-	-	-	N/A	N/A	N/A	11s
	user	Go	2,515	-	-	24/24	24/24	N/A	24/24	33s
	catalogue	Go	1,439	-	-	8/8	8/8	N/A	8/8	23s
	carts	Java	1,840	-	-	7/7	7/7	N/A	7/7	48s
	shipping	Java	929	-	-	3/3	3/3	N/A	3/3	34s
	queue-master	Java	926	-	-	3/3	3/3	N/A	3/3	31s
Pitstop	webapp	C#	40,461	16/16	-	-	16/16	16/16	N/A	52s
	customermanagementapi	C#	423	-	-	5/5	5/5	N/A	5/5	19s
	vehiclemanagementapi	C#	451	-	-	5/5	5/5	N/A	5/5	18s
	workshopmanagementapi	C#	1,563	4/4	-	20/20	24/24	4/4	20/20	46s
	workshopmanagementeventhandler	C#	685	10/10	-	14/14	24/24	10/10	14/14	30s
	auditlogservice	C#	136	1/1	-	2/2	3/3	1/1	2/2	7s
	notificationservice	C#	511	7/7	-	12/12	19/19	7/7	12/12	42s
	invoiceservice	C#	641	9/9	-	14/14	23/23	9/9	14/14	45s
	timeservice	C#	157	1/1	-	1/1	2/2	1/1	1/1	7s
Sitewhere	web-rest	Java	6,648	-	215/215	-	215/215	N/A	N/A	242s
	instance-management	Java	4,069	-	-	35/35	35/35	N/A	35/35	99s
	event-sources	Java	6,619	-	1/1	3/3	4/4	N/A	3/3	130s
	inbound-processing	Java	825	-	2/2	4/4	6/6	N/A	4/4	49s
	device-management	Java	6,381	-	-	74/74	74/74	N/A	74/74	156s
	event-management	Java	4,799	-	4/4	60/60	64/64	N/A	60/60	204s
	asset-management	Java	5,993	-	-	10/10	10/10	N/A	10/10	142s
	schedule-management	Java	1,964	-	-	10/10	10/10	N/A	10/10	77s
	batch-operations	Java	2,122	-	6/6	16/16	22/22	N/A	16/16	105s
	device-registration	Java	1,075	-	10/10	4/4	14/14	N/A	4/4	57s
	device-state	Java	1,739	-	1/1	7/7	8/8	N/A	7/7	61s
	event-search	Java	769	4/4	-	-	4/4	4/4	N/A	34s
	label-generation	Java	1,379	-	10/10	-	10/10	N/A	N/A	66s
	rule-processing	Java	1,091	-	2/2	2/2	4/4	N/A	2/2	50s
	command-delivery	Java	3,417	-	6/6	3/3	9/9	N/A	3/3	123s
	streaming-media	Java	736	-	-	10/10	10/10	N/A	10/10	49s
	outbound-connectors	Java	4,125	-	13/13	2/2	15/15	N/A	2/2	145s
Total	48 unique services	6 languages	-	96/96	290/290	369/369	751/755	96/96	369/369	-

Q1:

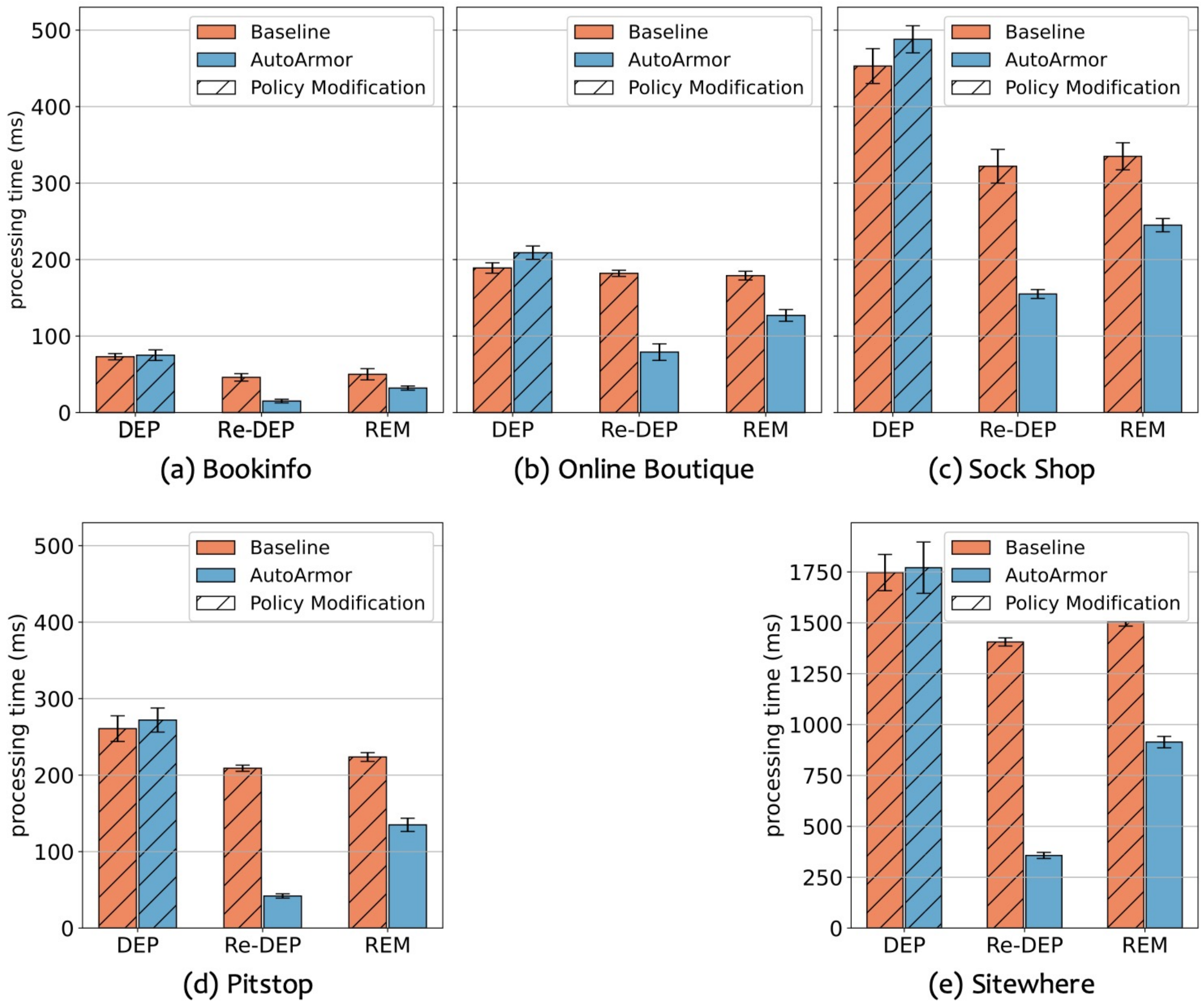
Can AutoArmor extract the inter-service invocation logic?

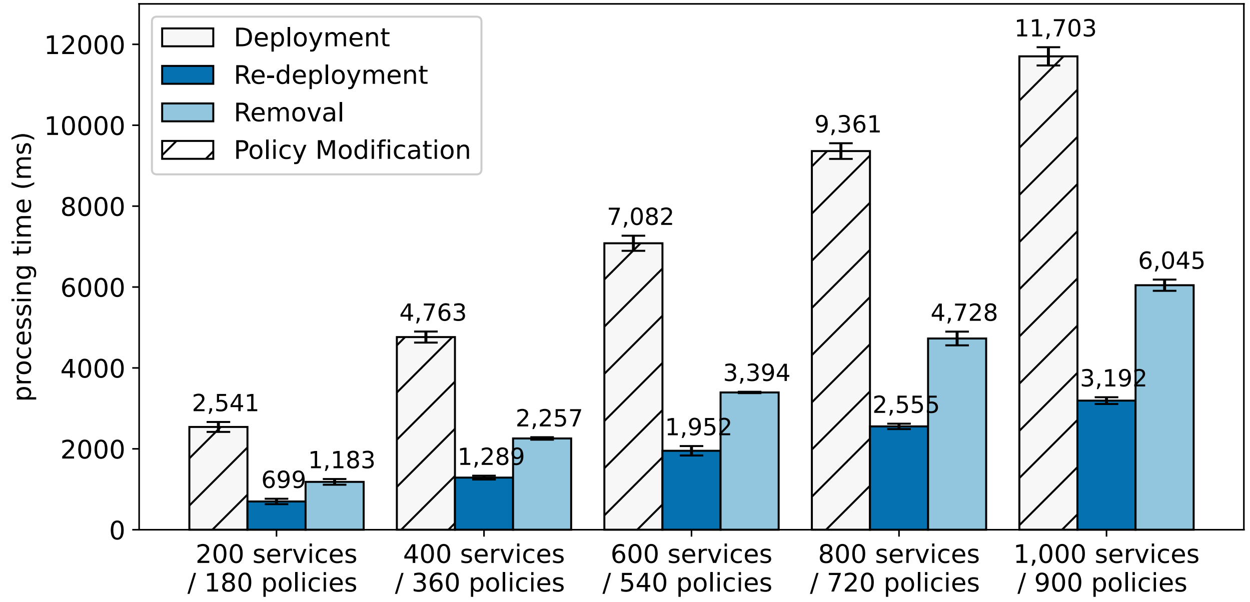
- Request identification rate: 100%
- Request attribute extraction rate: 99.5%
- Average static analysis time: 57 s/svc

Q2:

Can AutoArmor efficiently generate, manage, and update access control policies?

- The policy generation time for each evaluation application is less than 2 s.
- It takes less than 12 s to generate 900 policies for a large application with 1,000 services.





Q3:

Can AutoArmor improve the application's performance via the optimized policy set?

- By eliminating redundant policies, it enables microservice applications to achieve better end-to-end performance.

Application	External Requests			Average End-to-End Latency				
	URL	Method	Quantity	w/o Policies	w/ Baseline Policies		w/ AutoArmor Policies	
Bookinfo	http://<bookinfo-url>/productpage	GET	100000	319ms	323ms	▲ 4ms(1.25%)	321ms	▼ 2ms (0.62%)
Online Boutique	http://<boutique-url>/	GET	4,400	82ms	86ms	▲ 4ms (4.88%)	84ms	▼ 2ms (2.33%)
	http://<boutique-url>/cart	GET	13,000	80ms	91ms	▲ 11ms (13.75%)	86ms	▼ 5ms (5.81%)
	http://<boutique-url>/cart	POST	13,000	139ms	158ms	▲ 19ms (13.67%)	144ms	▼ 14ms (8.86%)
	http://<boutique-url>/cart/checkout	POST	4,400	112ms	129ms	▲ 17ms (15.18%)	121ms	▼ 8ms (6.20%)
	http://<boutique-url>/product/*	GET	56,000	76ms	85ms	▲ 9ms (11.84%)	82ms	▼ 3ms (3.53%)
	http://<boutique-url>/setCurrency	POST	9,000	91ms	93ms	▲ 2ms (2.20%)	94ms	▲ 1ms (1.08%)
Sock Shop	http://<sockshop-url>/	GET	11,000	95ms	104ms	▲ 9ms (9.47%)	98ms	▼ 6ms (5.77%)
	http://<sockshop-url>/basket.html	GET	11,000	101ms	111ms	▲ 10ms (9.90%)	105ms	▼ 6ms (5.41%)
	http://<sockshop-url>/cart	DELETE	11,000	190ms	204ms	▲ 14ms (7.37%)	197ms	▼ 7ms (3.43%)
	http://<sockshop-url>/cart	POST	10,000	364ms	436ms	▲ 72ms (19.78%)	401ms	▼ 35ms (8.03%)
	http://<sockshop-url>/catalogue	GET	11,000	168ms	177ms	▲ 9ms (5.36%)	169ms	▼ 8ms (4.52%)
	http://<sockshop-url>/category.html	GET	11,000	96ms	105ms	▲ 9ms (9.38%)	98ms	▼ 7ms (6.67%)
	http://<sockshop-url>/detail.html?id=*	GET	11,000	95ms	105ms	▲ 10ms (10.53%)	98ms	▼ 7ms (6.67%)
	http://<sockshop-url>/login	GET	11,000	350ms	373ms	▲ 23ms (6.57%)	367ms	▼ 6ms (1.61%)
	http://<sockshop-url>/orders	POST	9,500	392ms	476ms	▲ 84ms (21.42%)	468ms	▼ 8ms (1.68%)

AutoArmor

The **first** automatic policy generation tool for inter-service access control of microservices

- A static analysis-based request extraction mechanism
- A graph-based policy management mechanism
- effectively bridge the policy generation gap with only a minor overhead

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

30TH USENIX SECURITY SYMPOSIUM

Thanks!

xing.li@zju.edu.cn