KUBE-HOUND MICROSERVICES SECURITY SMELLS

Alessandro Bocci

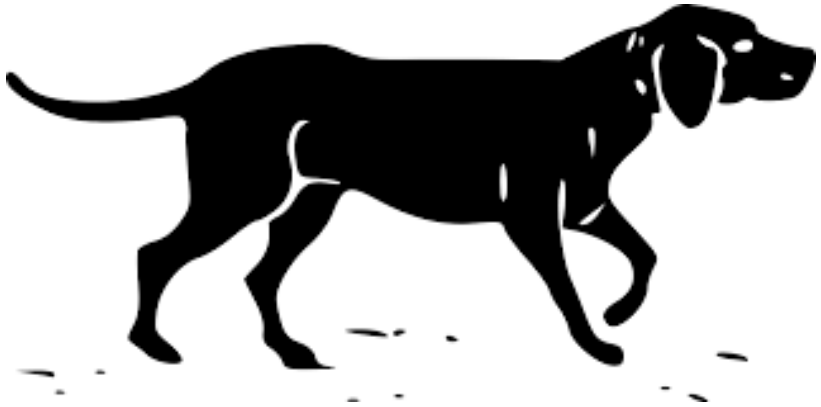
name.surname@phd.unipi.it

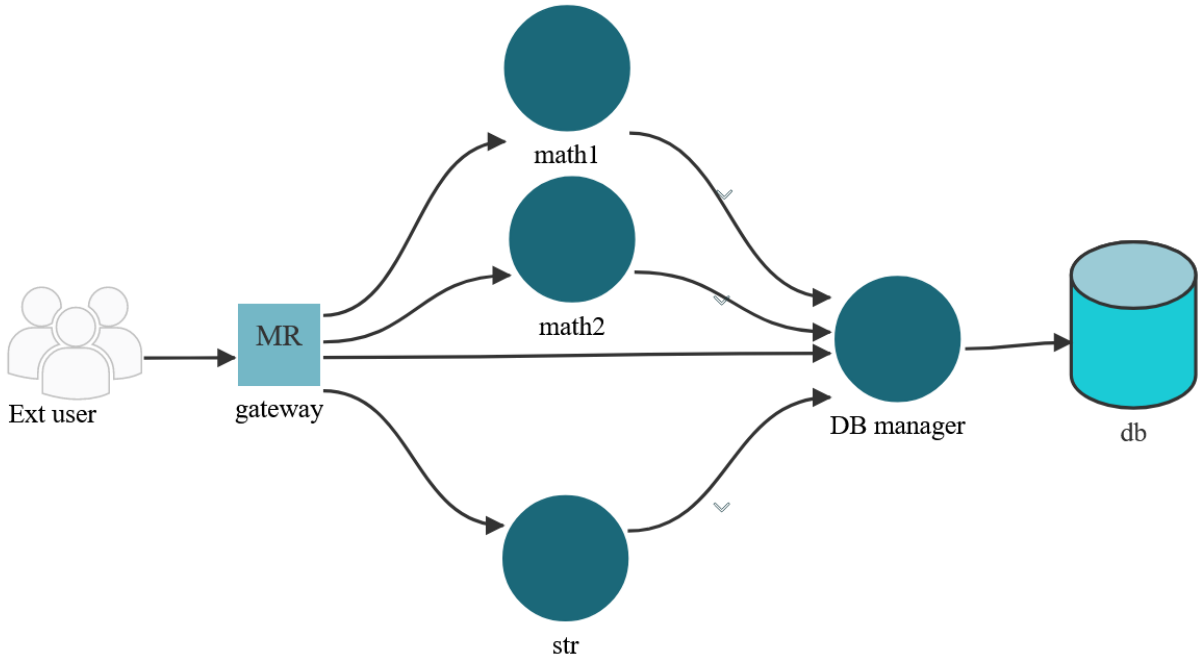
Advanced Software Engineering (Lab)

22/11/2023

Software Prerequisites

- kube-hound (https://github.com/di-unipi-socc/kube-hound)
- New `microase2324` from Moodle

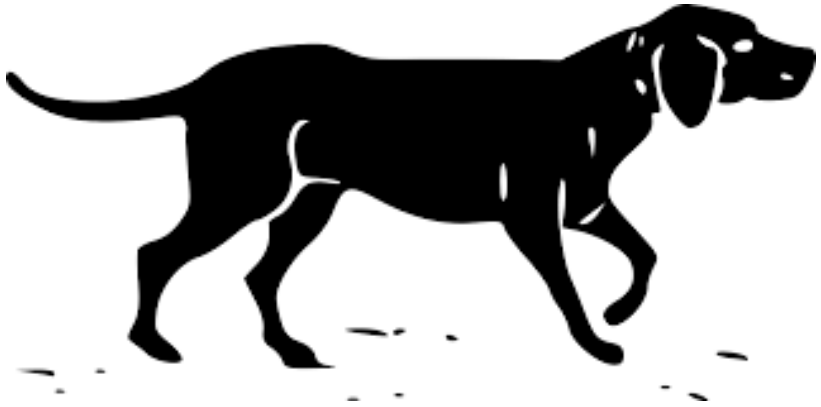


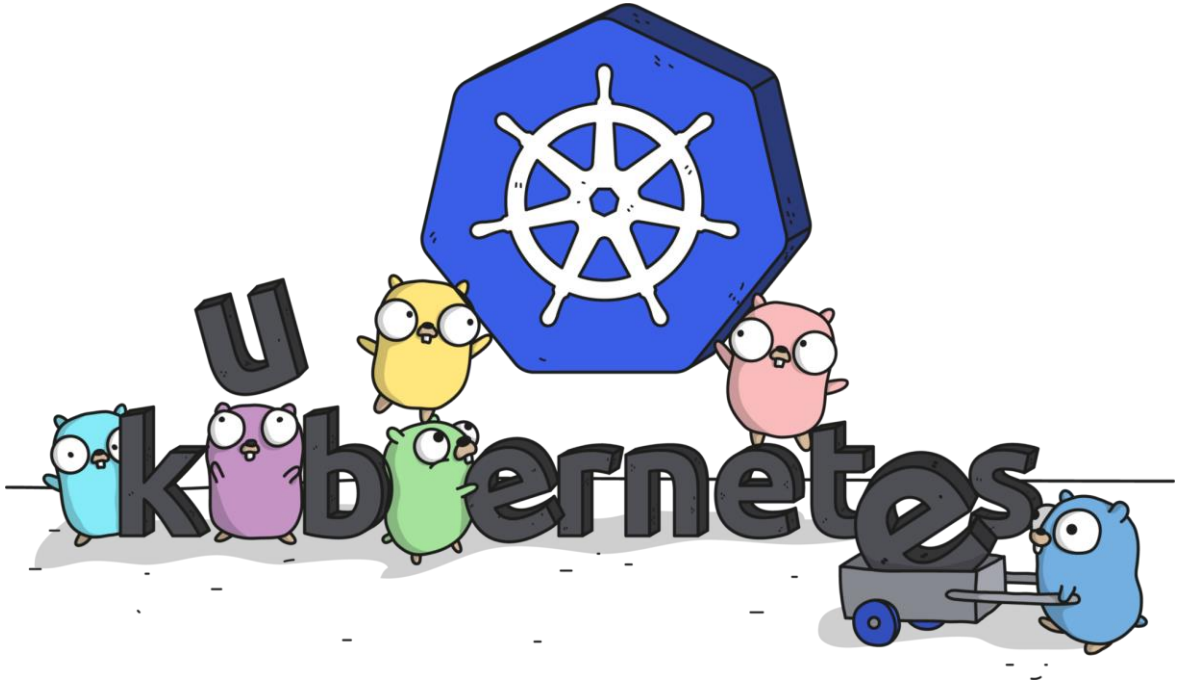


kube-hound

Tool to detect security smells in Kubernetes-based microservice applications exploiting:

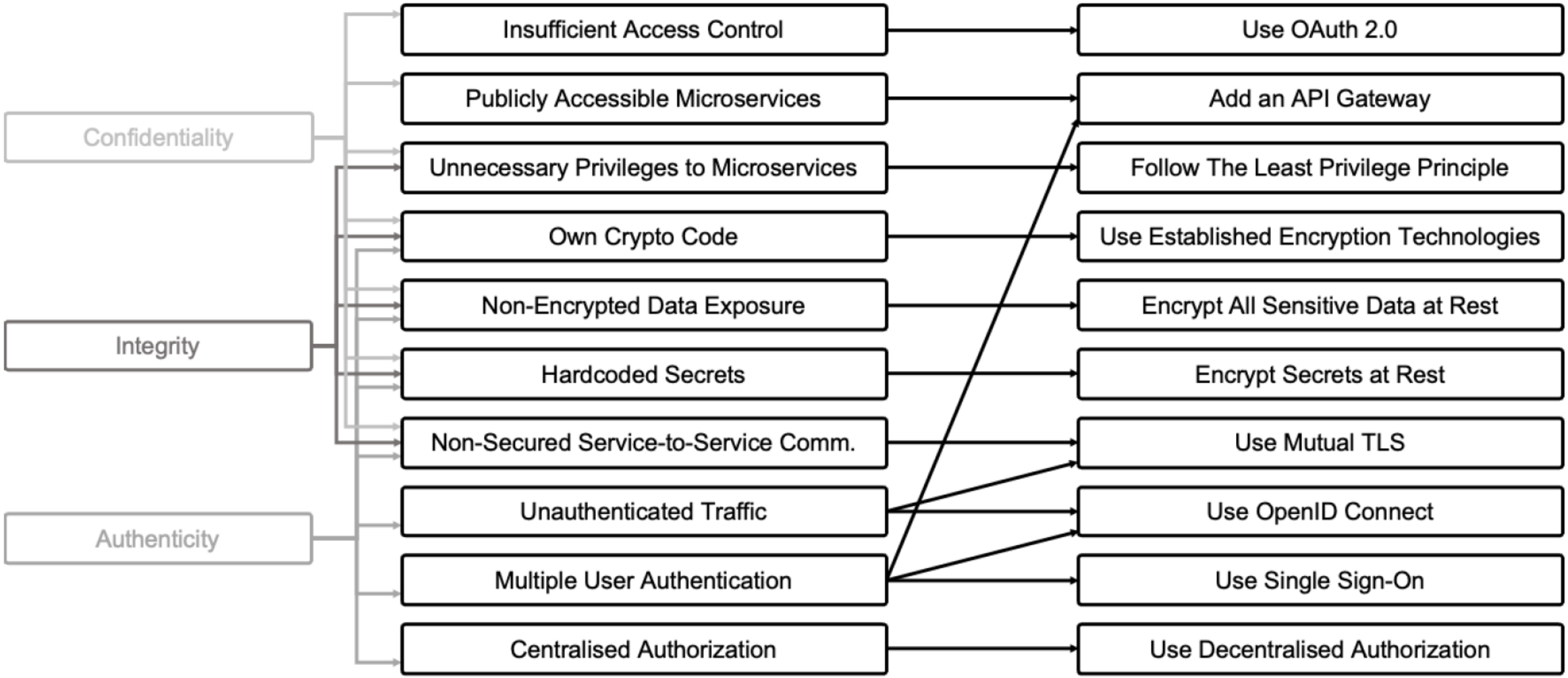
- Static analyses
- Dynamic analyses



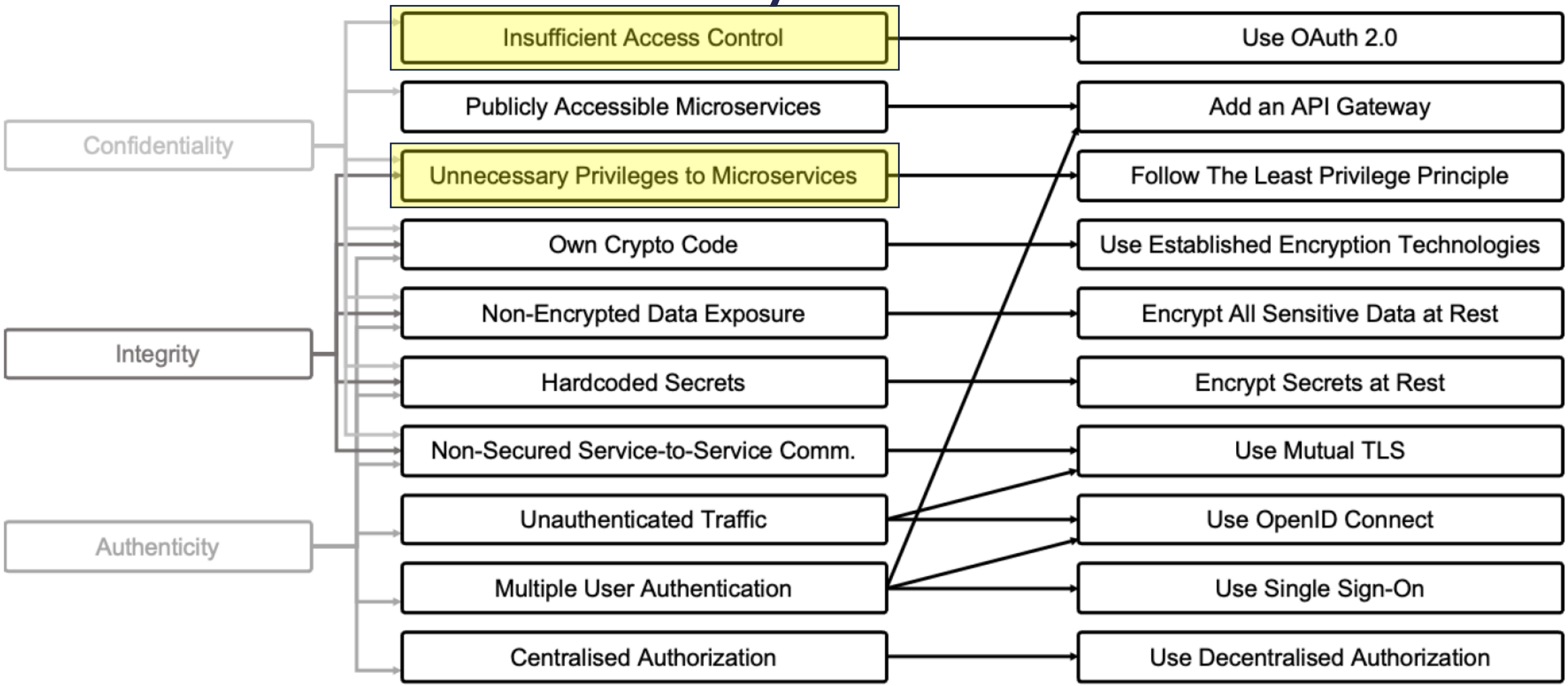


Dell'Immagine, G.; Soldani, J.; Brogi, A. *KubeHound: Detecting Microservices' Security Smells in Kubernetes Deployments.* *Future Internet* **2023**, *15*, 228. https://doi.org/10.3390/fi15070228

Microservices security smells

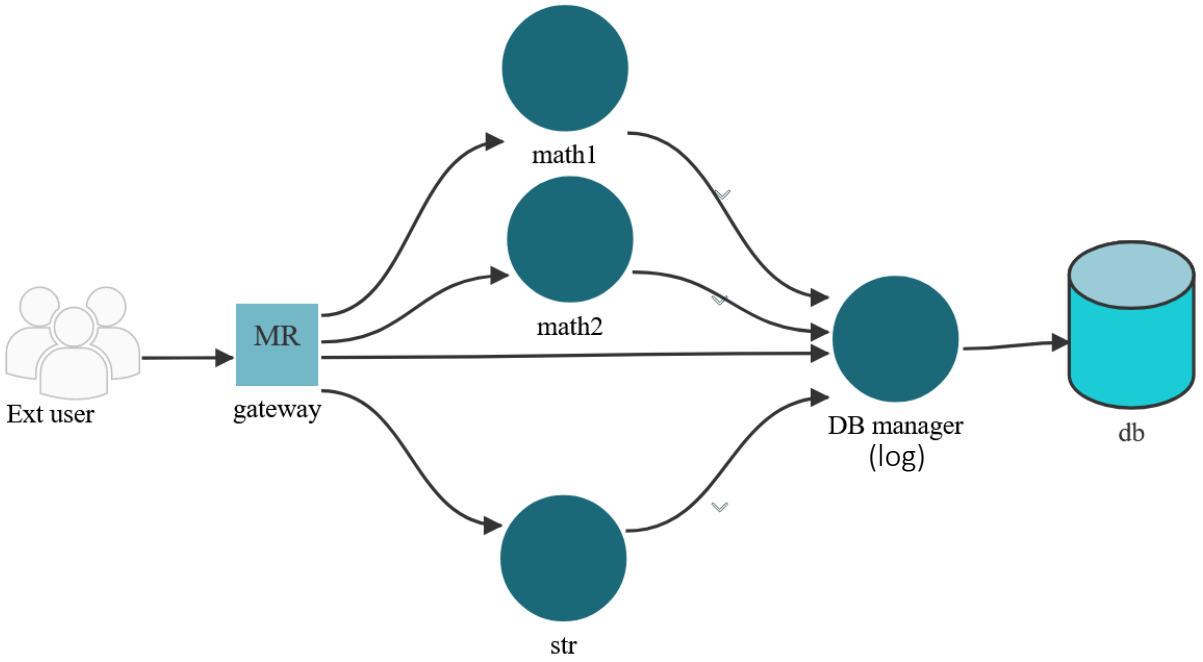


Microservices security smells



What will you do?

- Use kube-hound to analyse **`microase2324`**
- Examine the output of kube-hound
- Resolve the security smells



Pay attention to:

- Docker images name:
  - microase2324-gateway
  - microase2324-math-service1
  - microase2324-math-service2
  - microase2324-string-service
  - microase2324-log-service1
- Running containers:

If you stop kube-hound during its analyses it could leave some containers running and it fails the next executions.

Today's Lab

a. Download the new version of **`microase2324`** from the Moodle.

b. Check docker images of gateway, maths, string and log.

If you have different names, erase them and from **`microase2324`** folder run: **`docker compose build`**

LAB TODO

1. Run kube-hound to analyse **`microase2324`**

```
poetry run python -m kube_hound -s -l openapi_iac,kubesec_io -c ../ ../microase_config.yml
```

- `poetry run python -m kube_hound` — kube-hound execution
- `-s` — static analysis
- `-l openapi_iac,kubesec_io` — list of analyses
- `-c ../` — Context: **`microase2324`** parent folder
- `../microase_config.yml` — microase2324 configuration .yml

2. Inspect the output and resolve the security smells

- 1 x IAC smell: work on the openapi file of the gateway (**`gatewayAPI.yaml`**).
- 12 x 5 UPM smells: work on k8s configuration files (folder **`deployment/`**).

Resolve the 12 of the **`gateway`** (every service as the same UPM smells)

Example – UPM

```
KubeSec analysis - detected smells {UPM}
        Kubesec.io found potential problems in gateway.yaml
        selector: .metadata .annotations ."container.apparmor.security.beta.kubernetes.io/nginx"
        reason: Well defined AppArmor policies may provide greater protection from unknown threats. WARNING: NOT PRODUCTION READY
```

Example – UPM

```
KubeSec analysis - detected smells {UPM}
        Kubesec.io found potential problems in gateway.yaml
        selector: .metadata .annotations ."container.apparmor.security.beta.kubernetes.io/nginx"
        reason: Well defined AppArmor policies may provide greater protection from unknown threats. WARNING: NOT PRODUCTION READY
```

Detected UPM: Unnecessary Privileges to Microservices

Example – UPM

```
KubeSec analysis - detected smells {UPM}
        Kubesec.io found potential problems in gateway.yaml
        selector: .metadata .annotations ."container.apparmor.security.beta.kubernetes.io/nginx"
        reason: Well defined AppArmor policies may provide greater protection from unknown threats. WARNING: NOT PRODUCTION READY
```

Analyser (`Kubesec.io`) and location (`gateway.yaml`)

Example – UPM

```
KubeSec analysis - detected smells {UPM}
        Kubesec.io found potential problems in gateway.yaml
        selector: .metadata .annotations ."container.apparmor.security.beta.kubernetes.io/nginx"
        reason: Well defined AppArmor policies may provide greater protection from unknown threats. WARNING: NOT PRODUCTION READY
```

Suggested refactoring: adding a specific annotation

(it should be also reflected in the code, but for us it is enough fixing the configuration)

Exploit the suggestion and K8s online documentation.

Example – UPM

```
KubeSec analysis - detected smells {UPM}
        Kubesec.io found potential problems in gateway.yaml
        selector: .metadata .annotations ."container.apparmor.security.beta.kubernetes.io/nginx"
        reason: Well defined AppArmor policies may provide greater protection from unknown threats. WARNING: NOT PRODUCTION READY
```

Smell reason:

AppArmor’s policies are not configured

Example – UPM, How to resolve it?

Add in the **`gateway.yaml`** the needed annotation:

```yaml
metadata:
  name: gateway
  annotations:
    container.apparmor.security.beta.kubernetes.io/nginx: runtime/default
```

Be careful!

If you made typos or use wrong configurations, **kube-hound** fails to parse the .yaml and you will not see smells anymore.

OpenAPI

Specification for a machine-readable interface oriented to RESTful API.

If you open the **`gatewayAPI.yaml`** file with a Swagger editor (https://editor.swagger.io/) you will see the API of **`microase2324`**.

To resolve the IAC smell, you have to modify this **`.yaml`** file

BONUS STAGE!

Bonus stage

Use kube-hound dynamic analyses!

- Install `kubectl` (with Docker Desktop is enough to enalbe it in the settings).
- `kubectl apply -f <yaml file>` for every file in /deployment.
- `poetry run python -m kube_hound -v  -c ../ ../microase_config.yml`
- Resolve all the smells, but pay attention to false positives (e.g. all the OCCs).

Lab take away

- ❑ Familiarise with microservice application security smells.
- ❑ Resolve them at configuration level.
- ❑ Improve your K8s configuration knowledge.