Kubernetes cluster network security

Jacopo Bufalino, Aalto University

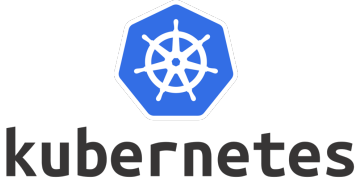
Aalto University
School of Science

12/09/2024

Agenda

- **Kubernetes key concepts**
- **State of the art of kubernetes (network) security**
- **Cluster network misconfigurations**
  - Find
  - Collect and Analyze
  - Mitigate
- **Takeaways and lessons learned**

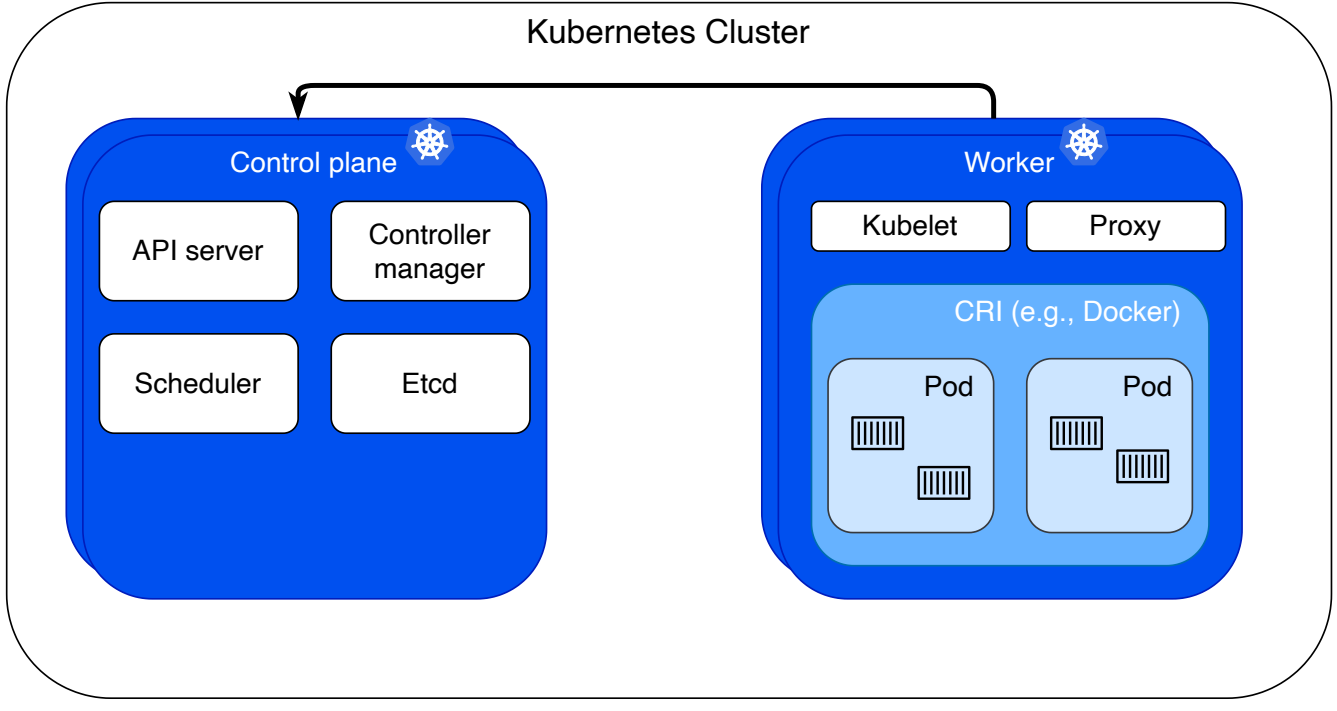
Key concepts



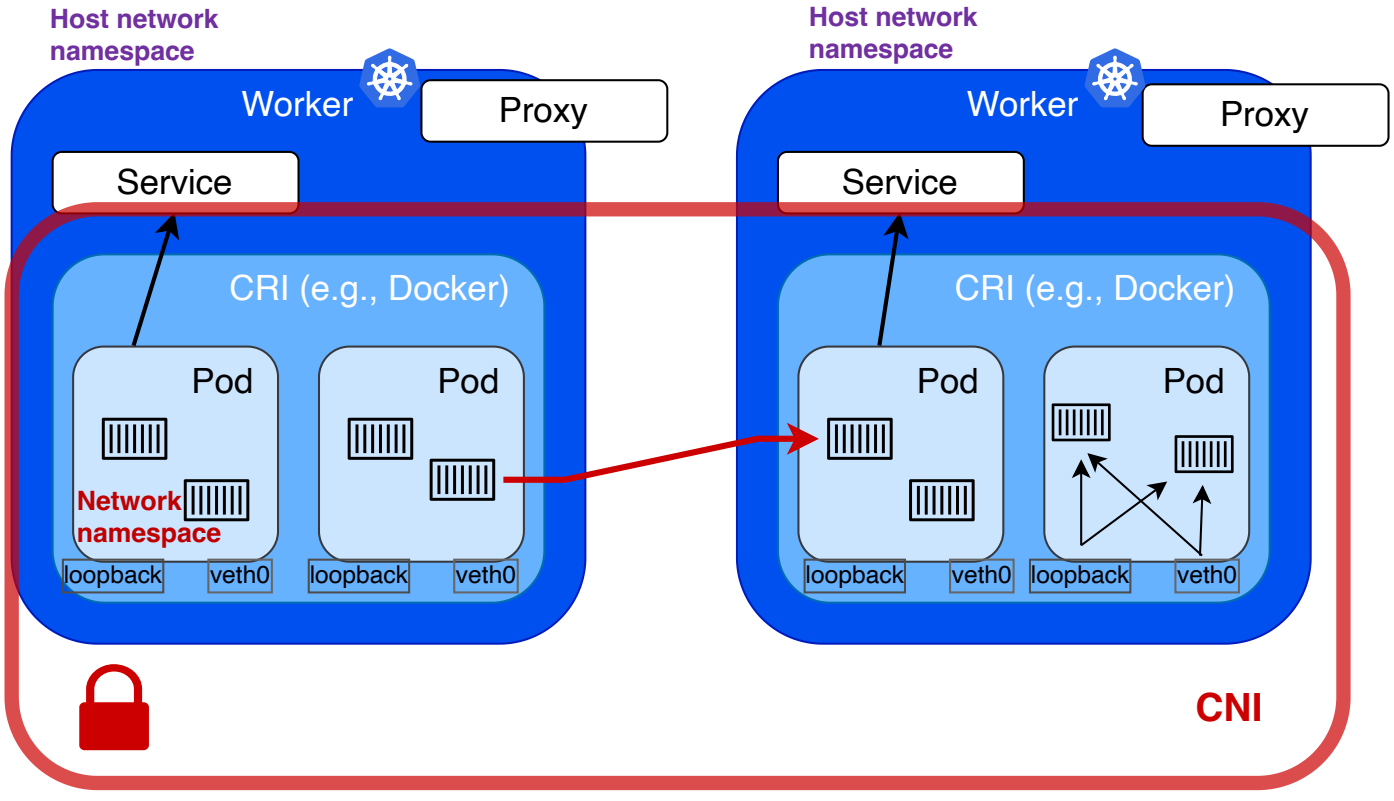
- **Platform** to run containerized workloads
- Configuration as Code (YAML)
- Used on-premises, on hybrid and cloud infrastructures
- Abstracts concepts to favour portability (e.g., CRI)

- Template engine and package manager
- Share and create applications (charts)
- Artifacthub.io has the largest collection of charts

Kubernetes architecture (overview)



Kubernetes networking model



Need cloud computing? Get started now

Partners Resources Contact Sales Start free trial All Microsoft Search

Search the blog

+33-15669

Products Solutions Why Akamai Resources Partners Contact Us

Blog > Security Research >
Can't Be Contained: Finding a Command Injection Vulnerability in Kubernetes

Research Cloud security Microsoft Defender for Cloud Cloud threats · 4 min read

Attackers exploiting new critical OpenMetadata vulnerabilities on Kubernetes clusters

By Microsoft Threat Intelligence

Can't Be Contained: Finding a Command Injection Vulnerability in Kubernetes

Kubernetes Exposed: One Yaml away from Disaster

Threat Alert

utions Why Akamai Resources Partners

What a Cluster: Local Volumes Vulnerabilit

Hackers like Kubernetes too…

Aalto University
School of Science

What a Cluster: Local Volumes Vulnerability in Kubernetes

MITRE® Threat matrix for Kubernetes

Microsoft **Threat Matrix for Kubernetes**

Search

Tactics | Mitigations | About

Tactics

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Impact

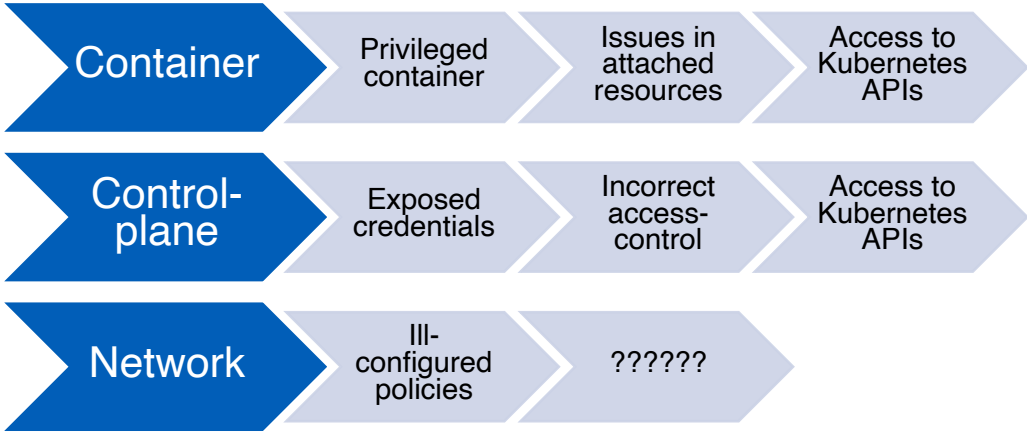
Tactics

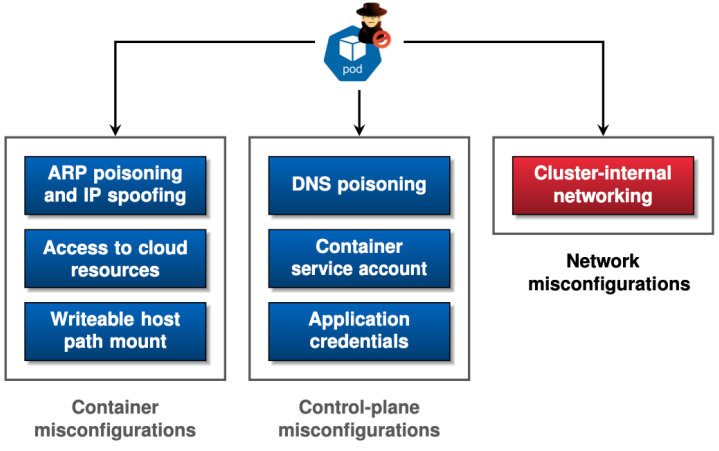
Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access Kubernetes API server	Access cloud resources	Images from a private registry	Data destruction
Compromised image In registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Collecting data from pod	Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Application credentials in configuration files	Exposed sensitive interfaces	Application credentials in configuration files		
Exposed sensitive interfaces	SSH server running inside container	Container service account			Access managed identity credentials	Instance Metadata API	Writable hostPath mount		
	Sidecar injection	Static pods			Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Aalto University
School of Science

MITRE® Threat matrix for Kubernetes

The 7 lateral movement techniques





MITRE® description:
[…]*Attackers who gain access to a single container may use it for network reachability to another container in the cluster.*

HOW?
STILL UNCLEAR !

Kubernetes cluster network security: state of the art

Static analysis of YAML files

checkov by bridgecrew

Analysis using Kubernetes APIs

Kubescape

kube-hunter

Security platforms

NeuVector
Full Lifecycle Container Security

StackRox

Kubernetes cluster network security: state of the art

Misconfigurations reported

- **Lack** of network policies
- Container attached to **host network**

What is missing

- **Connections** between resources
- Report of **potential** connectivity

Example of undetected misconfiguration

NGINX **Pod** is available to the the cluster via a **Service**. A **NetworkPolicy** is in place to avoid unwanted access.

Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
  - name: nginx-port
    containerPort: 80
    protocol: TCP
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    name: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: nginx-port
    - protocol: TCP
      port: 8081
      targetPort: 81
```

NetworkPolicy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: nginx-policy
spec:
  podSelector:
    matchLabels:
      name: nginx
…
ports:
    - protocol: TCP
      port: 6379
    - protocol: TCP
      port: nginx-port
```

Goal #1: explore the internal cluster network

Network policies and services can be malformed

- How to ensure their correct application?

Observability

- Can we monitor the potential connectivity between applications?

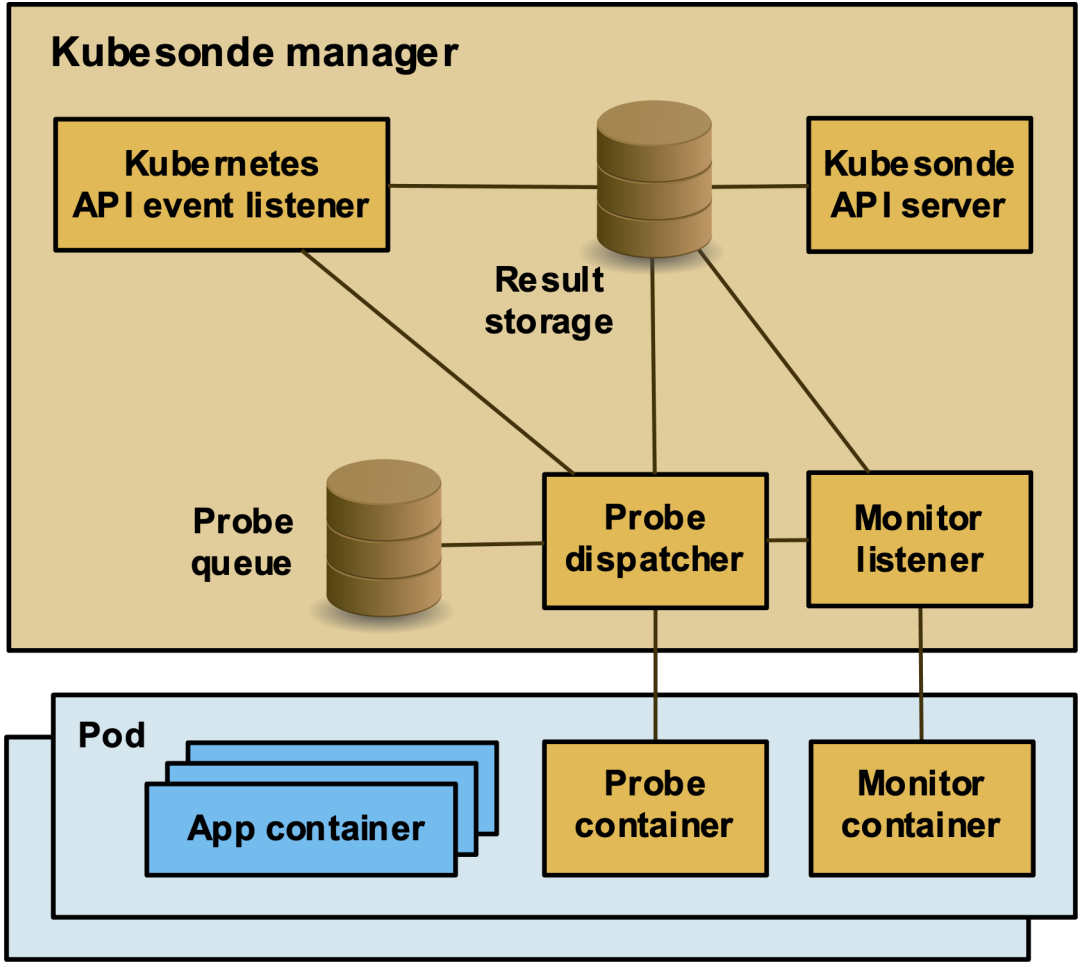
Container ports declarations are only documentation

- Is that really the case?

Kubesonde

Kubesonde

How it works



Kubesonde

How it works

Event

New pod is created

Deploy containers

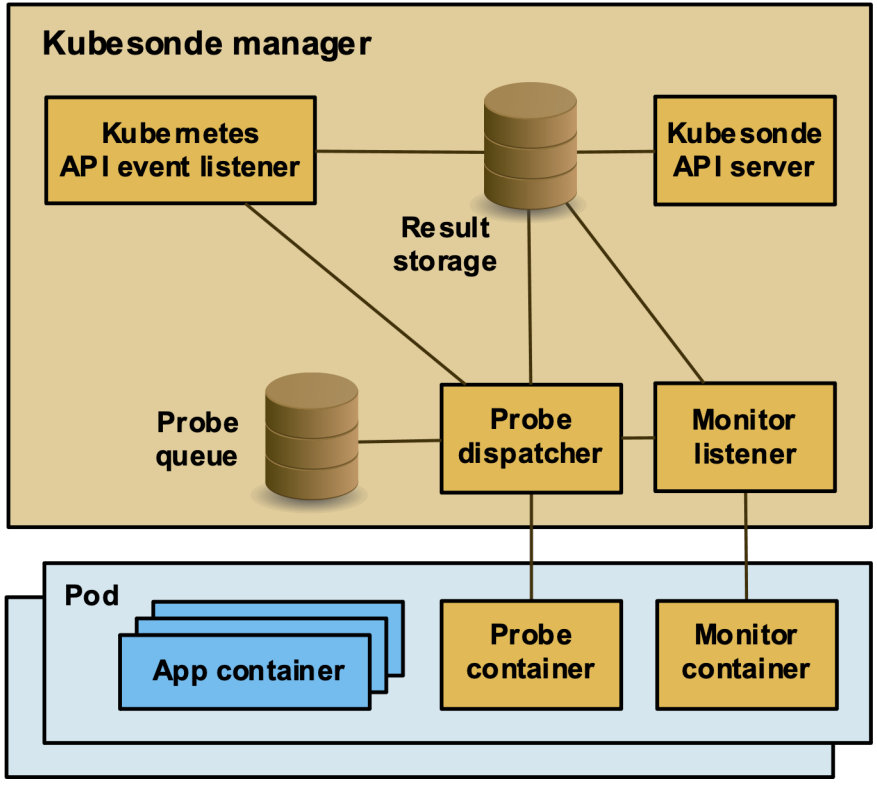
Monitor and probe are injected into new pod

Probe

Launch probes and wait for results

Process result

Save result of probe

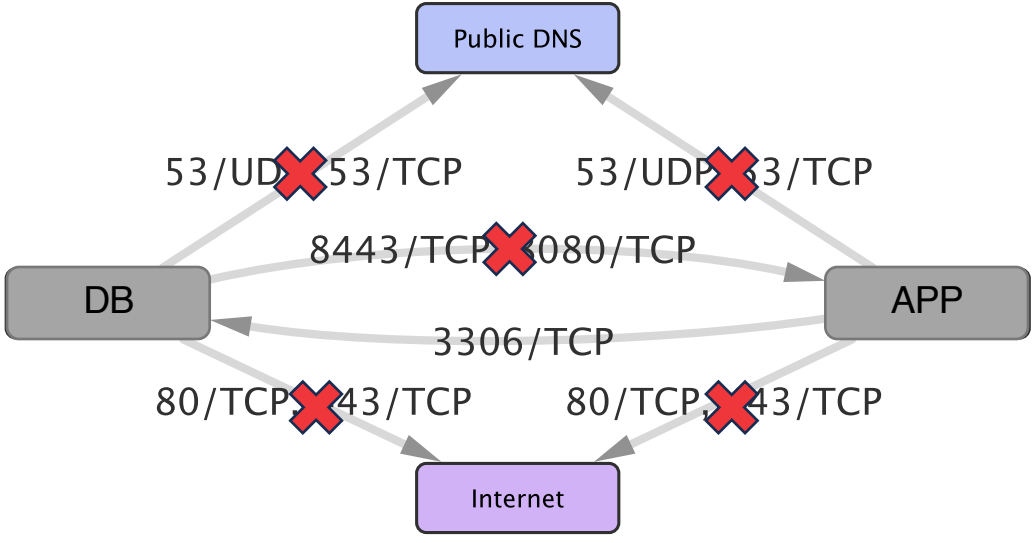


Testbed and results

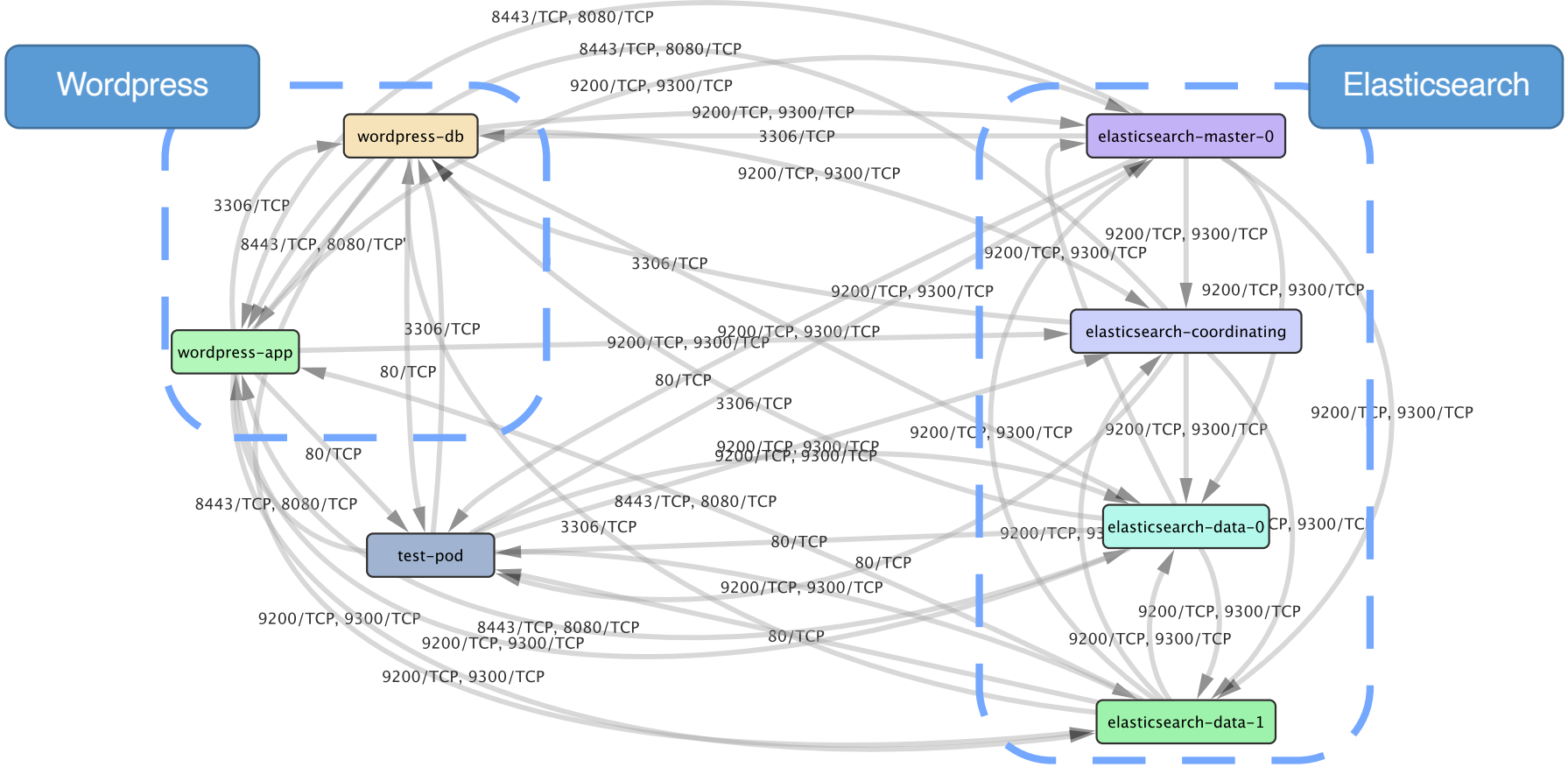
Run Kubesonde on the 20 most popular Helm charts

- Open ports not matching specifications
- Containers listening on all interfaces
- Lack of access control between pods
- Access to public DNS and Internet
- Dynamic ports

Results - simple application



Results - service composition



Where we are so far…

Many types of misconfigurations

- More precise mapping is needed

Network policies are usually not available

- Developers must have knowledge of the applications

Mapping misconfigurations

Goal #2: map misconfigurations in the internal cluster network

- Study their impact on cluster security
- Investigate how frequently they arise

The misconfigurations

M1 Port open on container is not declared

```
containers :
- name : flink
image : bitnami/flink
ports :
- containerPort : 8081
```

```
# netstat -a
Active Internet connections ( servers and established )
Proto Recv-Q    Send-Q  Local Address       Foreign Address     State
tcp         0         0  0.0.0.0:8081        0.0.0.0:*           LISTEN
tcp         0         0  0.0.0.0:8080        0.0.0.0:*           LISTEN
```

M2 Container allocates dynamic ports

```
containers :
- name : clickhouse
image :
bitnami/clickhouse
ports :
- containerPort : 8080
```

```
# netstat -a
Active Internet connections ( servers and established )
Proto Recv-Q    Send-Q  Local Address       Foreign Address     State
tcp         0         0  0.0.0.0:8080        0.0.0.0:*           LISTEN
tcp         0         0  0.0.0.0:47123       0.0.0.0:*           LISTEN
```

The misconfigurations

M3 Port declared on container is not open

```
containers :
- name : consul
image : bitnami/consul
ports :
-  containerPort : 8080
- containerPort : 8300
- containerPort : 8360
```

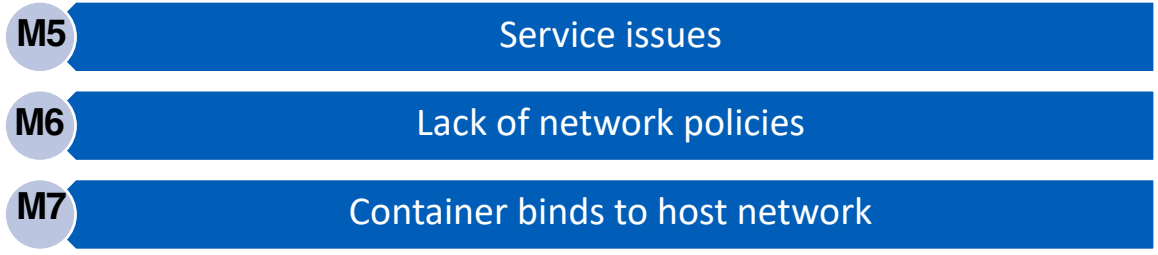
```
# netstat -a
Active Internet connections ( servers and established )
Proto Recv-Q      Send-Q   Local Address      Foreign Address    State
tcp        0           0   0.0.0.0:8080       0.0.0.0:*          LISTEN
```

M4 Label collision

```
# Pod 1
…
labels:
role: frontend
…
```

```
# Pod 2
…
labels:
role: frontend
…
```

The misconfigurations




```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    name: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: nginx-port
    - protocol: TCP
      port: 8081
      targetPort: 81
```

M5

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: influxdb
spec:
  hostNetwork: true
  containers:
    - name: influxdb
      image: influxdb
```

M7

The data

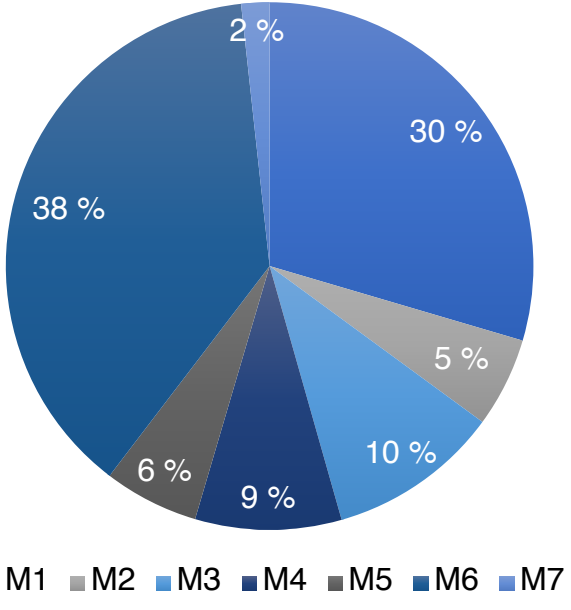

- Selected **6** datasets from Helm providers including:
  - **Bitnami**
  - **Wikimedia**
  - **CNCF**
  - **Prometheus Community**
  - **European Environmental Agency**
  - **Banzaicloud**
- **287** charts in total

Findings

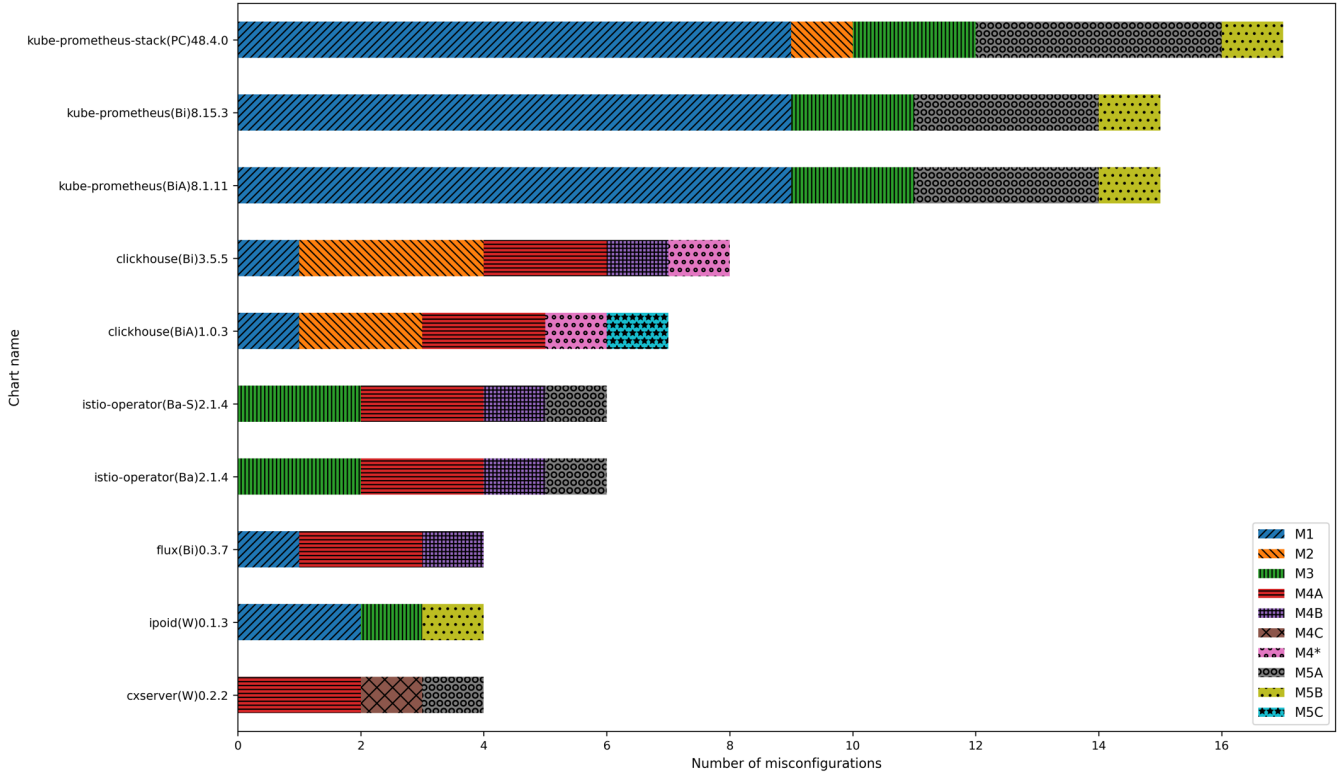
Over 90% of the charts have at least one misconfiguration (259 / 287)

- **More than 80%** (241/287) of the apps do not include NetworkPolicy
- **30%** of the total are type M1 (Port not declared)
- **60%** not reported by state-of-the-art tools
- **8%** of the charts with NetworkPolicy have errors

Misconfigurations (634 in total)



The top 10

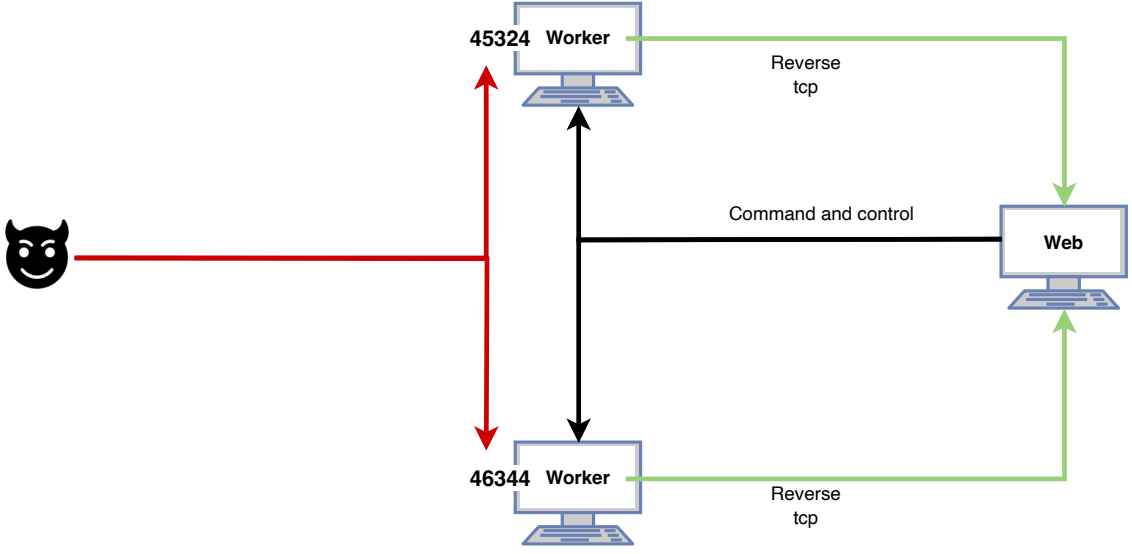

kube-prometheus-stack(PC)48.4.0
kube-prometheus(Bi)8.15.3
kube-prometheus(BiA)8.1.11
clickhouse(Bi)3.5.5
clickhouse(BiA)1.0.3
istio-operator(Ba-S)2.1.4
istio-operator(Ba)2.1.4
flux(Bi)0.3.7
ipoid(W)0.1.3
cxserver(W)0.2.2
Chart name
0
2
4
6
8
10
12
14
16
Number of misconfigurations
M1
M2
M3
M4A
M4B
M4C
M4*
M5A
M5B
M5C

Successful attack

Concourse

Concourse is a CI/CD software part of the CNCF landscape

Master web node and a number of different workers



[bitnami/flux] fix: 🐛 🔒 Add missing notification controller ports #25

Merged javsalgar merged 2 commits into `main` from `fix/flux-add-missing-ports` 2 weeks ago

Conversation 7 Commits 2 Checks 11 Files changed 7

javsalgar commented 3 weeks ago Contributor

Reviewers

[bitnami/jaeger] fix: 🐛 🔒 Expose missing ports in deployment spec

Merged javsalgar merged 1 commit into `main` from `fix/jaeger-add-missing-ports` last week

Conversation 0 Commits 1 Checks 20 Files changed 8

javsalgar commented 2 weeks ago

Signed-off-by: Javier Salmeron Garcia js

Description of the change

This PR adds to the deployment spec th
values.

Many organizations fixed the misconfigurations

Wikimedia Code Review CHANGES DOCUMENTATION BROWSE

Merged 992899 ipoid: Fix chart default ports

```
{{- end }}
serviceAccountName: {{ include "app1.serviceAccountName" . }}
containers:
- name: {{ .Release.Name }}
  image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
  stdin: true
  tty: true
  # Port 8834 - User Interface, TSC communication, and API calls
  ports:
  - name: nessus
    containerPort: 8834
    protocol: TCP
{{- with .Values.nodeSelector }}
nodeSelector:
  {{- toYaml . | nindent 8 }}
```

```
    - name: serflan-udp
      containerPort: {{ .Values.containerPorts.serfLAN }}
      protocol: "UDP"
    - name: serfwan-tcp
      protocol: "TCP"
      containerPort: {{ .Values.containerPorts.serfWAN }}
    - name: serfwan-udp
      containerPort: {{ .Values.containerPorts.serfWAN }}
      protocol: "UDP"
    - name: rpc-server
      containerPort: {{ .Values.containerPorts.rpcServer }}
    - name: dns-tcp
      value: {{ .Values.containerPorts.rpc | quote }}
    - name: CONSUL_SERF_LAN_PORT_NUMBER
      value: {{ .Values.containerPorts.serfLAN | quote }}
    - name: CONSUL_SERF_WAN_PORT_NUMBER
      value: {{ .Values.containerPorts.serfWAN | quote }}
{{- if .Values.extraEnvVars }}
{{- include "common.tplvalues.render" (dict "value" .Values.extraEnvVars "context" $) | nindent 12 }}
{{- end }}
```

Aalto University
School of Science

bitnami/charts #25106
[bitnami/kafka] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25141
[bitnami/zookeeper] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25069
[bitnami/contour] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25078
[bitnami/grafana-operator] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25156
[bitnami/jaeger] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25097
[bitnami/grafana-tempo] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25110
[bitnami/keycloak] fix: 🐛 🔒 Expose missing ports in deployment spec and fix headless service

bitnami/charts #25113
[bitnami/kube-state-metrics] fix: 🐛 🔒 Expose missing ports in deployment spec

[bitnami/charts #25135]
ose missing ports in deployment spec

se missing ports in deployment spec

ssing notification controller ports

r] fix: 🐛 🔒 Expose missing ports in deployment spec

[bitnami/mysql] feat: Add support for mysqlx port

bitnami/charts #25077
[bitnami/grafana-loki] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25043
[bitnami/argo-cd] fix: 🐛 🔒 Expose metrics port in deployment definition

bitnami/charts #25042
[bitnami/appsmith] fix: 🐛 🔒 Add ambassador container to appsmith-backend to contact appsmith-rts

bitnami/charts #25072
[bitnami/ejbca] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25066
[bitnami/consul] fix: 🐛 🔒 Expose missing ports in deployment spec

bitnami/charts #25048
[bitnami/clickhouse] fix: 🐛 🔒 Add shard label to avoid Compute Unit collision

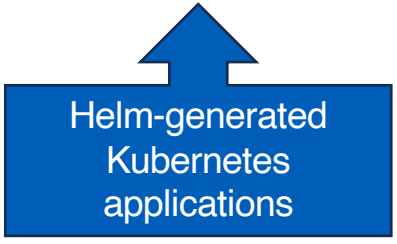
bitnami/charts #25045
[bitnami/cassandra] fix: 🐛 🔒 Do not expose tls internode port unless encryption is set

bitnami/charts #25041
[bitnami/apisix] fix: 🐛 🔒 Do not expose http-metrics unless metrics.enabled=true

bitnami/charts #25047
[bitnami/cert-manager] fix: 🐛 🔒 Expose missing ports in deployment spec

Mitigate misconfigurations (HelmET)

Goal #3: create secure network policies for unknown charts

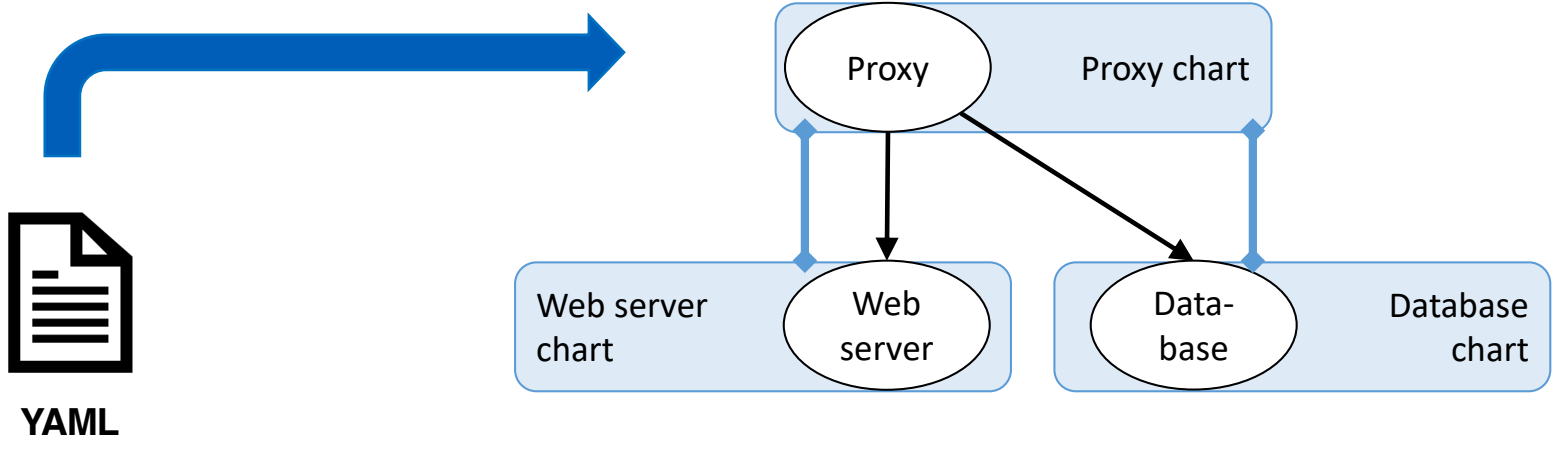


Challenges

- Not all applications are open-source
- How to create policies before deployment?

Idea

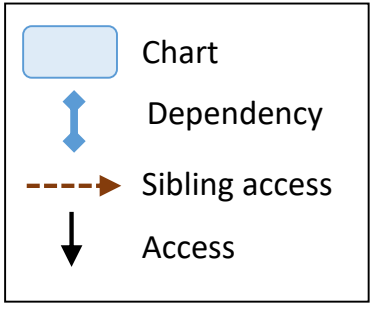
Generate policies using the chart structure and dependencies

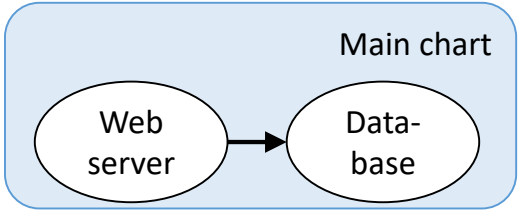


Dataset and rule generation

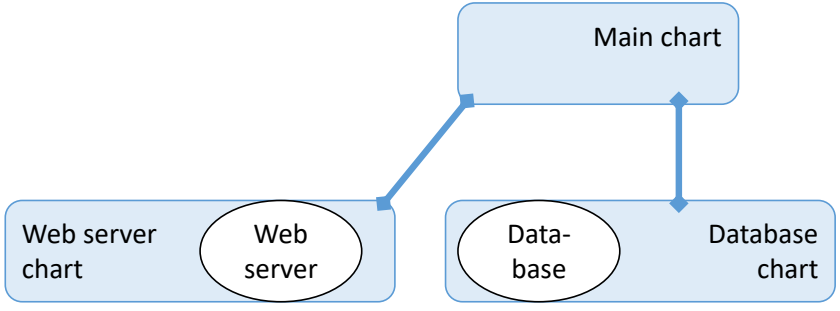
- We analyzed **465** charts
- Identified **5** composition patterns
- Generate policies using software engineering concepts
  - Information hiding
  - Modularity
  - Module hierarchy

Dataset and rule generation



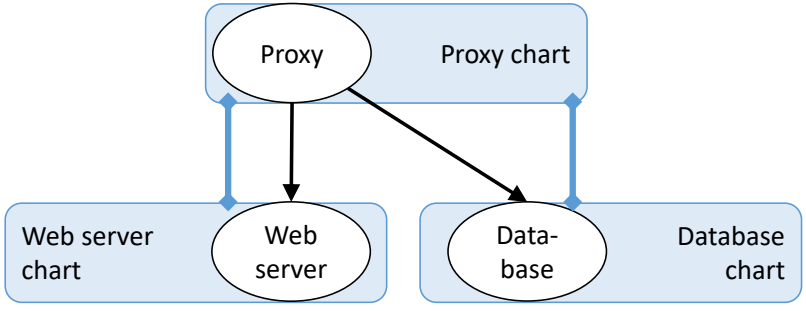


Standalone application

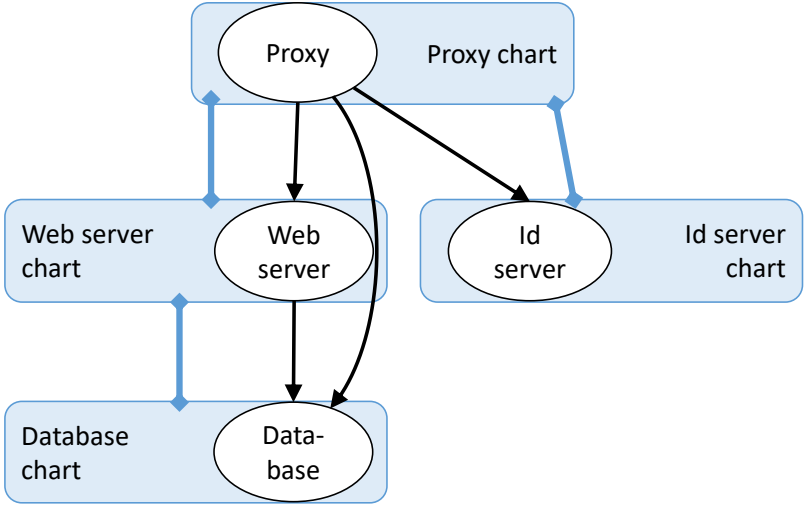


Disjoint applications

Dataset and rule generation

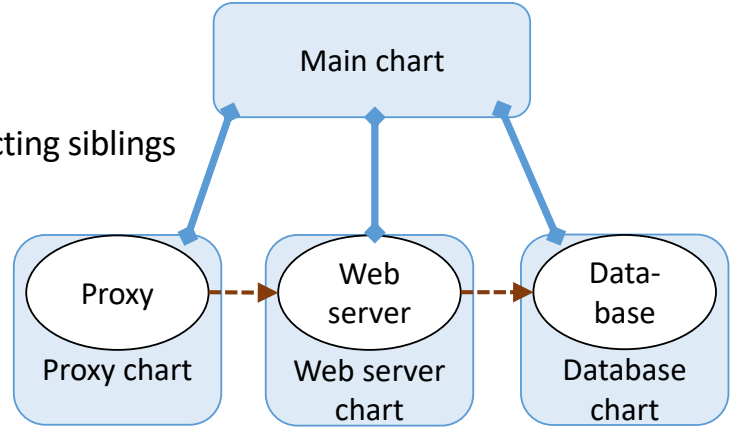


Dependency list



Recursive dependencies

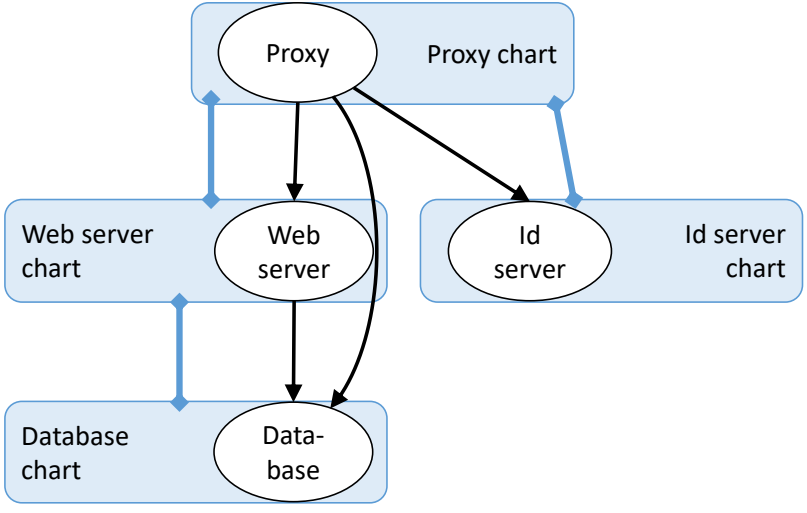
Interacting siblings



Dataset and rule generation

Three simple rules:

1. Deny access between siblings
2. Allow access only from parent to children
3. Default-deny all other connections

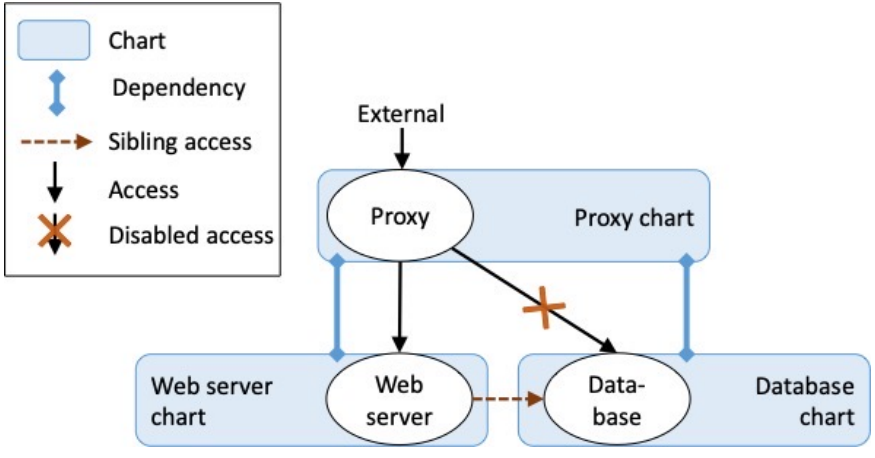


Recursive dependencies

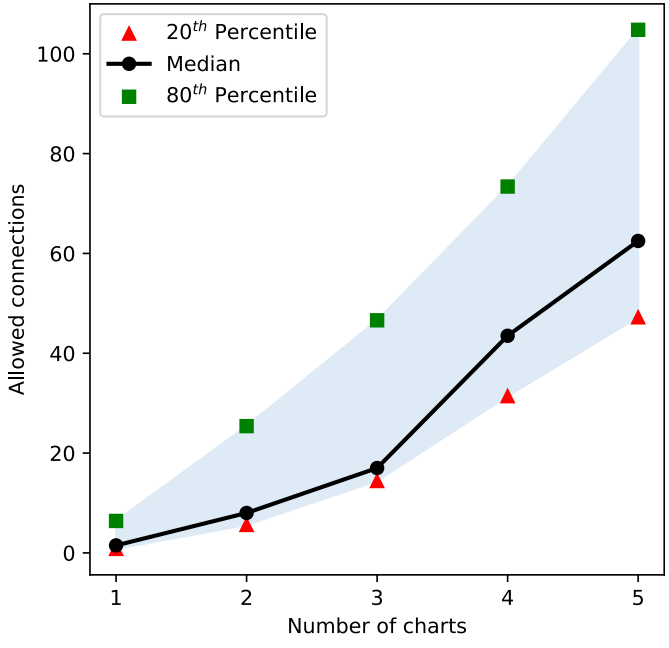
Results

- Over **90%** of the applications can be automatically secured
- **High level policy language** for the remaining ones

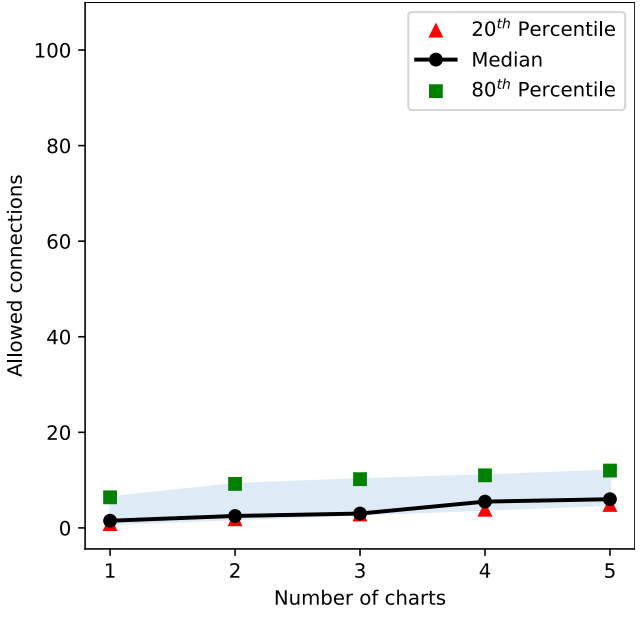
```
Name: "proxy-policy"
ComponentSelector:
...
Interactions:
  - From:
      chart: "proxy"
    To:
      chart: "webserver"
  # Enable communication between siblings:
  - From:
      chart: "webserver"
    To:
      chart: "database"
  # Disable direct access from proxy to database:
  # - From:
  #     chart: "proxy"
  #   To:
  #     chart: "database"
...
```



Results

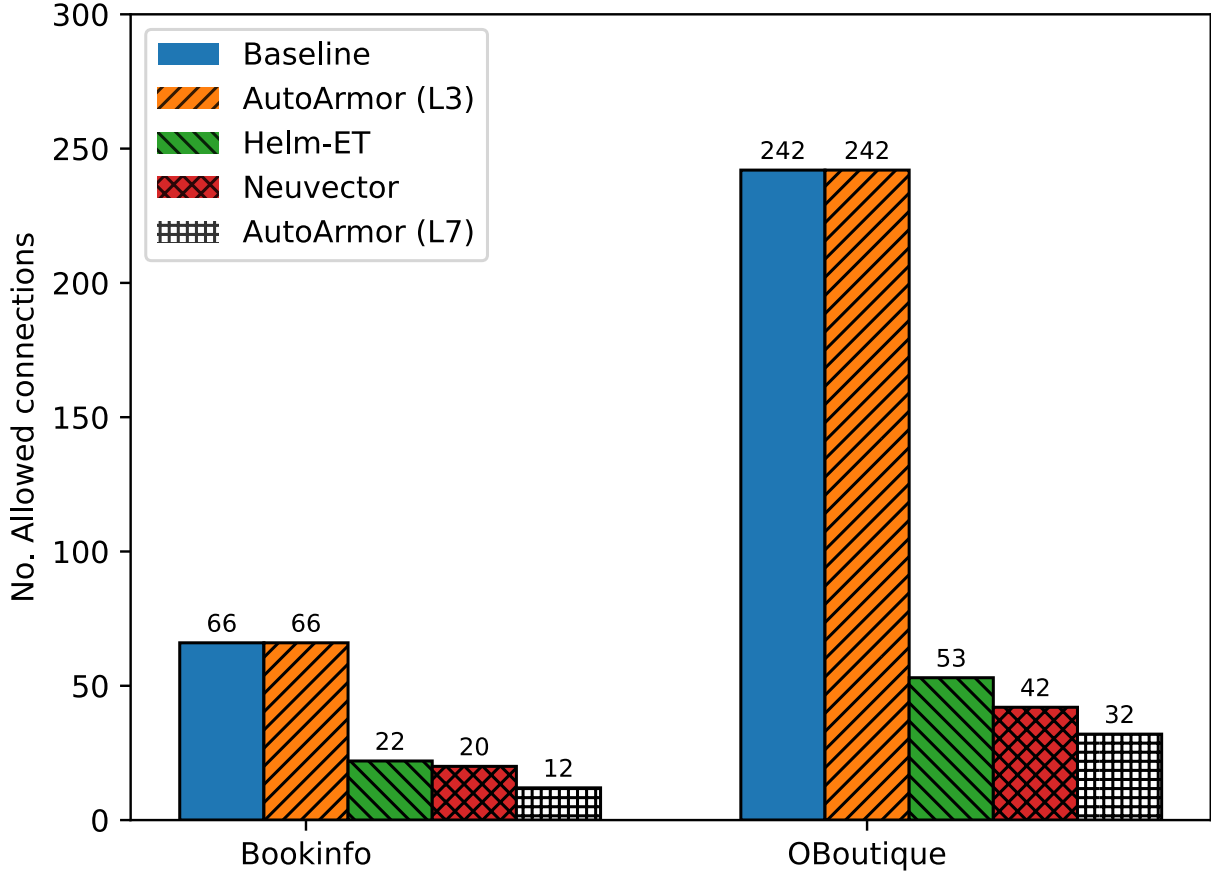


Before



After

Comparison with other work



Takeaways

- Still long way for lateral movement protection/mitigation
- Do not assume third party applications to be hardened in terms of internal networking
- Much depends on the developers to ensure proper access control

The end

Jacopo Bufalino

@JacopoBufalino

github.com/jackap

Cluster network Security survey

https://github.com/jackap/kubesonde

Aalto University
School of Science