white duck

Hijack a Kubernetes Cluster – a Walkthrough

Azure Meetup Ruhrgebiet, May 2022

Microsoft Partner

Microsoft

Gold Cloud Platform
Gold DevOps
Silver Application Development
Silver Security
Silver Application Integration

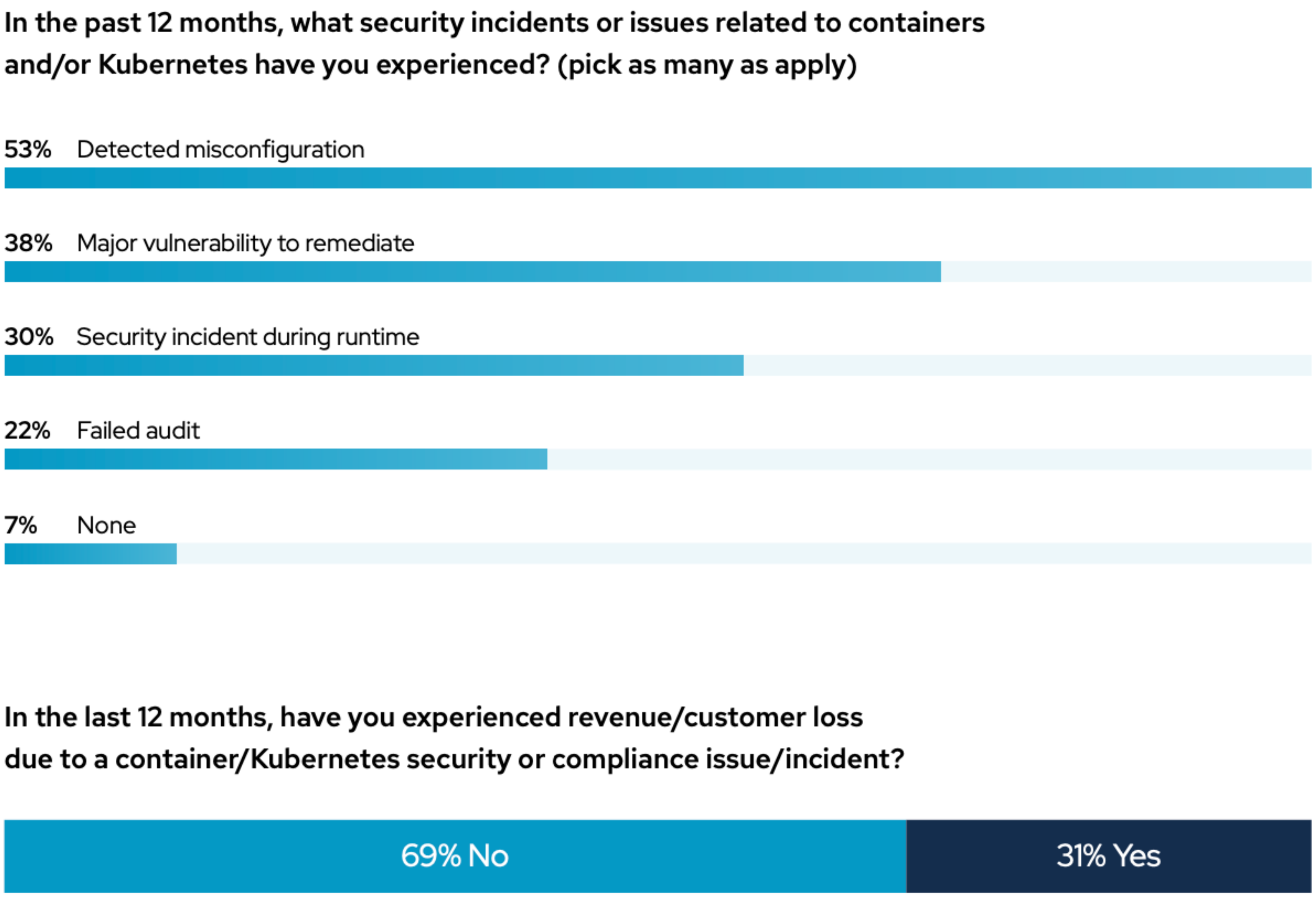
GitHub

Nico Meisenzahl

- Head of DevOps Consulting & Operations at white duck
- Microsoft MVP, GitLab Hero
- Cloud Native, Kubernetes & Azure

Phone:	+49 8031 230159 0
Email:	nico.meisenzahl@whiteduck.de
Twitter:	@nmeisenzahl
LinkedIn:	https://www.linkedin.com/in/nicomeisenzahl
Blog:	https://meisenzahl.org

Why do we need to care about security?



In the past 12 months, what security incidents or issues related to containers and/or Kubernetes have you experienced? (pick as many as apply)

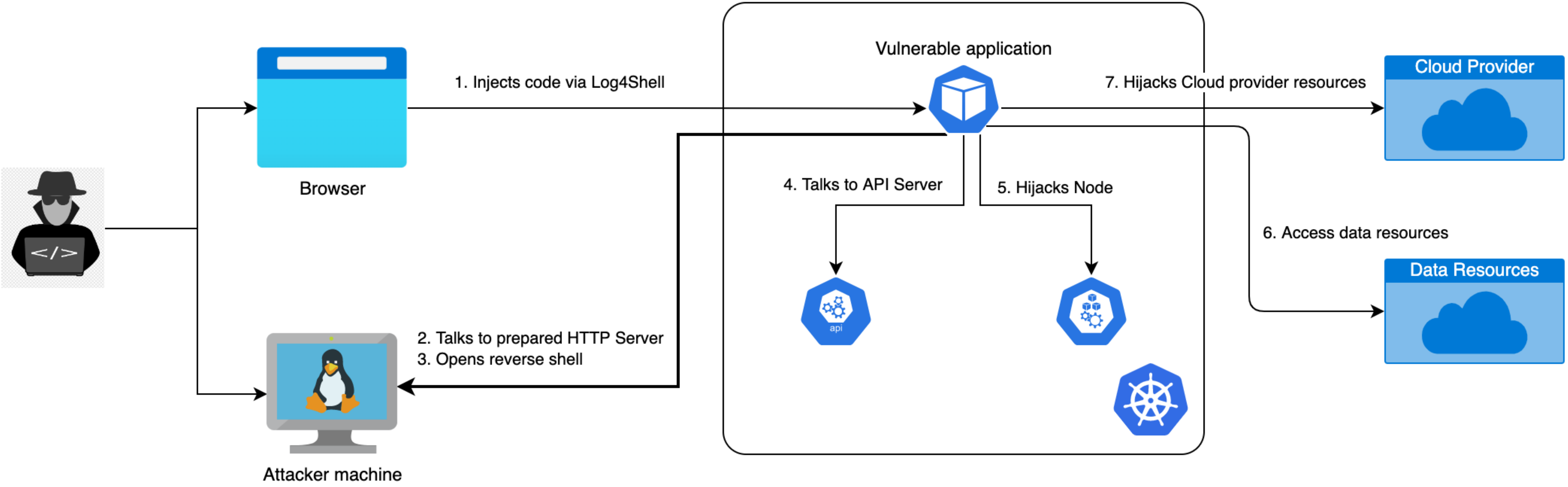
In the last 12 months, have you experienced revenue/customer loss due to a container/Kubernetes security or compliance issue/incident?

https://www.redhat.com/en/resources/state-kubernetes-security-report

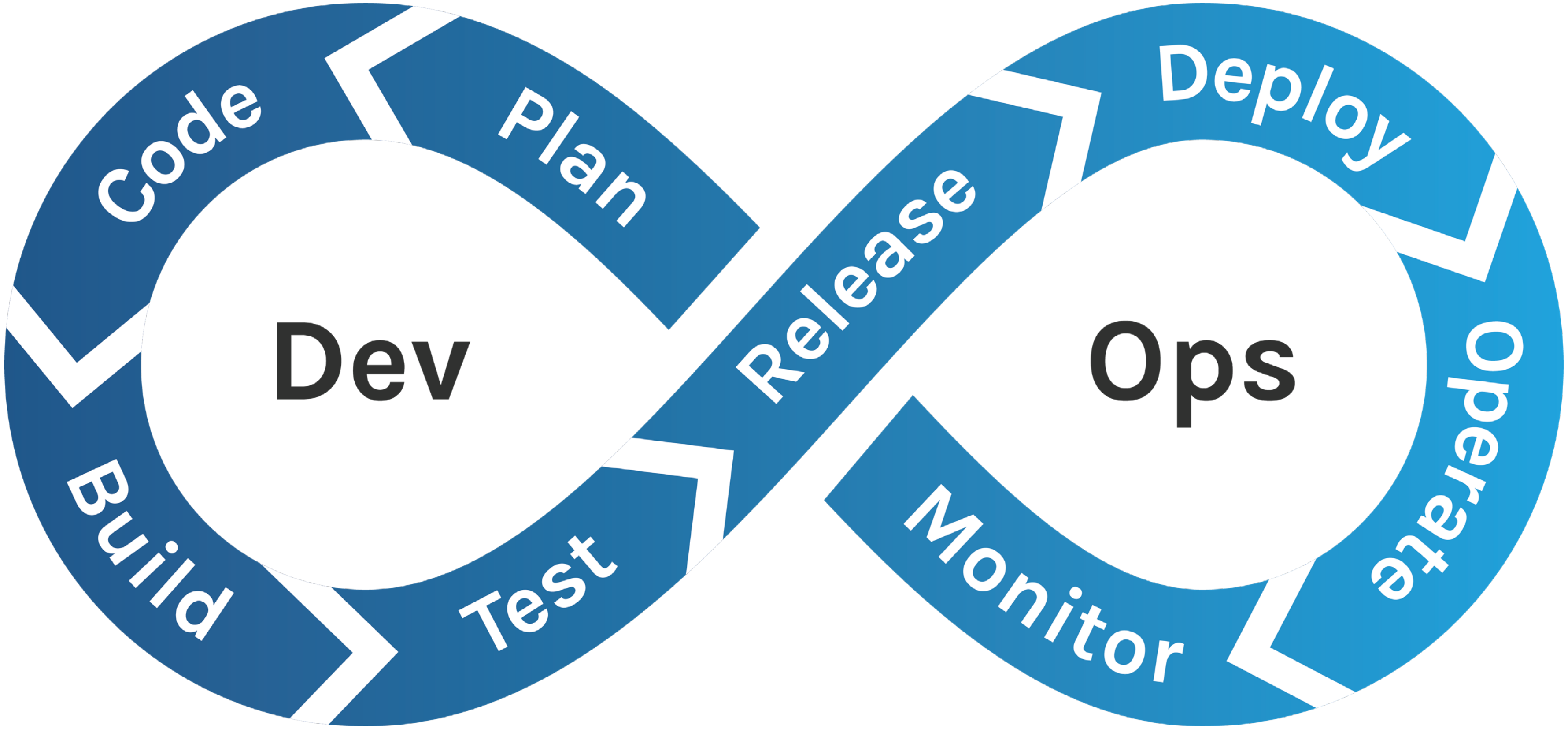
About this talk

- this is not an in-depth security talk
- it should make you aware of common attack vectors and how to prevent them
  - you will see demos on how to hijack a cluster
  - you will learn how to prevent those with common best practices
- one more slide, then we will start hijacking
  - https://github.com/nmeisenzahl/hijack-kubernetes

What we will do



Security quick wins through the DevOps cycle



Ensure secure application code

- automate and enforce code checks
- schedule dependency scanning
  - e.g. Dependabot
- enforce Static Application Security Testing (SAST) in PRs
  - scans your code to identify potential security vulnerabilities
  - more details: https://owasp.org/www-community/Source_Code_Analysis_Tools

Build secure container images

- build secure/small container images – less
  - do only include required dependencies (no de
  - use self-contained binaries or “distroless” if po
    - https://github.com/GoogleContainerTools/distroless
  - otherwise, use a small and secure Linux distro
- use and enforce SAST for validating your Dockerfiles
- scan your container images (on build an

Would have made it much harder to hijack the container and further expend

Would have shown the possibility of code injection

Ensure secure deployment code

- as important as secure application code and Dockerfiles
- validate your deployment manifests using SAST
  - and enforce them via PRs
- can help you to implement best practices like denying
  - containers running as root
  - mounting hostPath
  - …

Would have made it much harder to hijack the node

SAST Tooling

- Source code
  - https://codeql.github.com
  - https://security-code-scan.github.io
  - https://securego.io
- Kubernetes manifests
  - https://kubesec.io
  - https://github.com/aquasecurity/trivy
- Dockerfiles
  - https://github.com/aquasecurity/trivy
- Terraform
  - https://github.com/tfsec/tfsec
  - https://github.com/aquasecurity/trivy

Kubernetes policies

- enforce compliance and governance within clusters
  - verifying manifests is not enough!
- examples include enforcement of
  - read-only filesystems
  - denying hostPath mounts
  - denying containers running as root
  - …

Would have made it much harder to further hijack the nodes and cloud resources

Kubernetes policy tooling

- Open Policy Agent Gatekeeper
  - https://github.com/open-policy-agent/gatekeeper
- Kyverno
  - https://kyverno.io

Network Policies

Would have denied network connections (reverse shell, Redis, Internet, metadata service)

- granular deny or explicitly allow b and ingress/egress of the cluster
  - limit egress access to the internet
  - limit access between applications/namespaces
  - deny access to the Cloud provider metadata service
- https://kubernetes.io/docs/concepts/services-networking/network-policies

Container Runtime Security

Would have detect all our “work” within the containers

- helps to detect malicious threads and v
  - untrusted process within container
  - a shell is running inside a container
  - container process mounting a sensitive path
  - a process making outbound network connections
- container runtime security tools like Falco can help
  - https://github.com/falcosecurity/falco

Further best practises

Wouldn’t have allowed us to talk to the API server

- do not
  - share service accounts between applications
  - enable higher access levels for the default service account if not required
  - mount service account token if not required
    - https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/#use-the-default-service-account-to-access-the-api-server
- review all third-party snippets before applying them
- implement a Web Application Firewall (WAF) to further secure your application

Would have denied our code injection

Questions?

- Slides: https://www.slideshare.net/nmeisenzahl
- Demo: https://github.com/nmeisenzahl/hijack-kubernetes

Phone:	+49 8031 230159 0
Email:	nico.meisenzahl@whiteduck.de
Twitter:	@nmeisenzahl
LinkedIn:	https://www.linkedin.com/in/nicomeisenzahl
Blog:	https://meisenzahl.org