SECURITY IN SOFTWARE DEVELOPMENT

ABOUT TEKKAMAKI

- Security Assessment
- Security Awareness
- ISO 27001 certification
- Consultancy

https://www.tekkamaki.nl

THE CONTEXT

- Scrum-based development
- Online app or service
- Microservice architecture
- Containers running in a scheduler (k8s / Nomad / Mesos)
- CI/CD
- Highly agile and lots of releases per week

TYPICAL WORKFLOW

- Source code uploaded to repository (git)
- CI is triggered by web hook
- Check out source code
- Download dependencies (from internal repositories or externally)
- Build code
- Unit test code
- Build artifacts (packages, containers, images)
- Upload artifacts to repository

TYPICAL WORKFLOW - CONT

- Deploy to non-production environment
- Integration tests
- Regression tests
- Performance tests
- Stress tests
- Security tests?
- Tag artifact as production
- Deploy in production

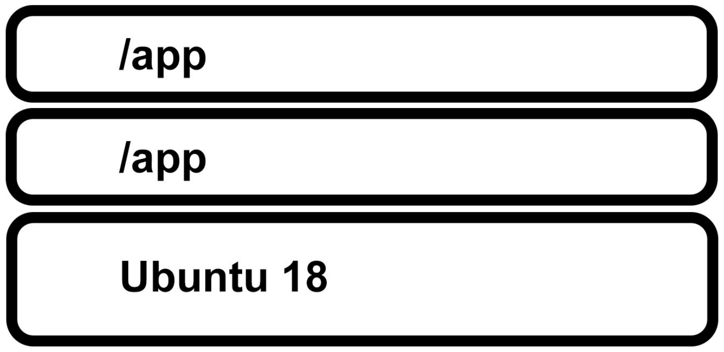
ARTIFACTS TO WATCH

- OS packages
- Local repositories:
  - Source code
  - Mirrored OS package repositories
  - Dependencies in code (external libraries)
  - Packages built from own source
  - Container images
  - Machine images

CONTAINER LAYOUT

- Dockerfile:

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```



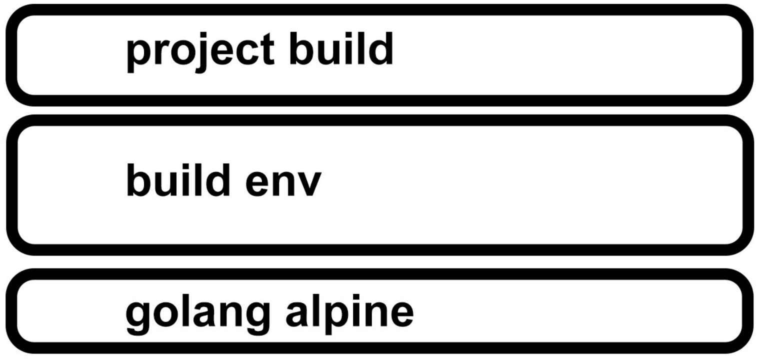
/app

/app

Ubuntu 18

- Multi stage Dockerfile:

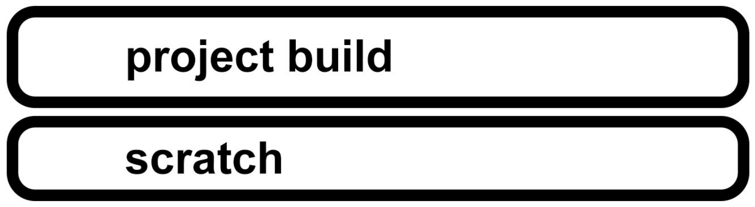
```
FROM golang:1.11-alpine AS build
RUN go get github.com/golang/dep/cmd/dep
[...]
RUN go build -o /bin/project

FROM scratch
COPY --from=build /bin/project /bin/project
```



project build

build env

golang alpine



project build

scratch

CONTAINER BEST PRACTICES

- Leave everything out that's not needed at runtime
  - More secure - less to abuse
  - Faster upload
  - Faster boot
  - Less memory / disk usage
- No sensitive data
- Build everything automatically in CI
- Tag containers with metadata to find versions in production

CLOUD IMAGE BUILDS

- packer: build cloud OS-images:

```
{
  "variables": {
    "region": "us-east-1"
  },
  "builders": [{
    "ami_name": "gruntwork-packer-training-rails-{{isotime | clean_ami_name}}",
    "source_ami_filter": {
      "filters": {
        "name": "*ubuntu-xenial-16.04-amd64-server-*",
      },
  }],
  "provisioners": [{
    "type": "shell",
    "script": "{{template_dir}}/install-rails.sh"
  },{
    "type": "file",
    "source": "{{template_dir}}/../example-rails-app",
    "destination": "/home/ubuntu"
  }]
}
```

CLOUD IMAGES

- Just as container images: based on OS version, with possible vulnerabilities built-in
- Again: make them as small as possible
- Do not store sensitive data in images
- Build automatically
- Test security automatically (tripwire for devops / Lynis / Nessus)

COMMON VULNERABILITIES AND EXPOSURES (CVE)

- Make sure you receive them (mail, rss, other), filter where appropriate
- Make sure CI/CD is automated so a rebuild can be triggered with a keystroke
- Problem: what exactly do we run and include?

SOLUTIONS

- Reproducible-builds.org - have verifiable assets
- Sqreen.com - application scanning
- Trivy - Vulnerability Scanner for Containers
- Whitesourcesoftware.com - monitor and alert OS components
- Stackrox.com - scan containers
- Threatstack.com - monitor cloud behaviour

CONCLUSIONS

- It’s easy to have vulnerabilities in your infrastructure or code
- Make sure no unneeded code is deployed
- Automate build and deploy to make incident response easy, reproducible, fast
- Look for tools that support your workflow

QUESTIONS