

A black and white photograph of four runners (two men and two women) running through a long, narrow tunnel with vertical wooden slat walls. The runners are in motion, captured from a side-on perspective. The lighting is dramatic, with strong highlights and deep shadows.

ITSECX 2019

The nuts and bolts of Security Automation at Runtastic

Markus Donko-Huber

 **RUNTASTIC**

Runtastic Facts & Figures

We are

4

Founders

We are

10

Years Old

We were profitable
after just

20

Months

We are

235

Employees

We come from

38+

Countries

We have

3

Offices in Linz, Vienna
And Salzburg

Our products are
available in

15

Languages

We are


1

Team with
a single vision

Engineering Facts & Figures

We are
120
engineers

We run
70
micro services

Linux powers
99%
of our services


Virtualization
Hardware with
3600
CPU cores

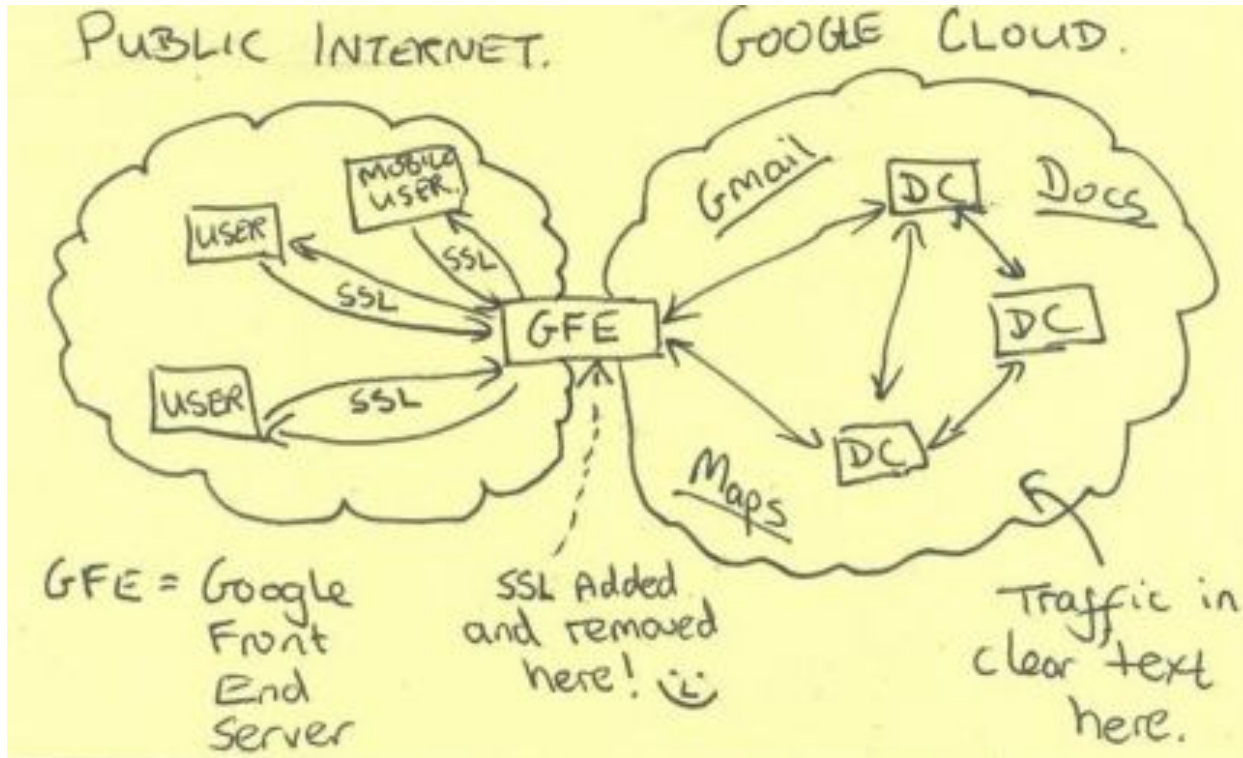
Motivation for this talk

1. **Scalability** of Security == **Automation** of Security
2. Promote Open Source **tools for your daily work**
3. Discuss **automation ideas** based on commercial services

(Hidden Agenda: Promote Runtastic as an IT-company [headhunting bonus \$\$\$])

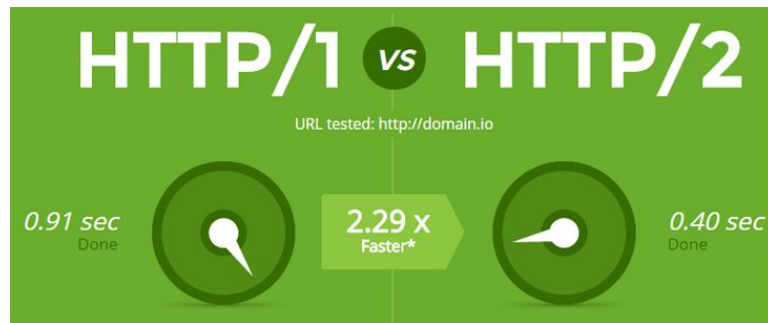
Automate Transport Layer Security

Still true for most companies (5 years later)



TLS for internal (DC) traffic

- TLS is about **integrity, speed, security**
- TLS **is a nightmare to configure**
- Manual TLS configuration does not scale



TLS automation == Public Key Infrastructure (PKI)

PKI Candidates

1. [Let's encrypt](#)
2. [Boulder \(ACME CA\)](#)
3. [Dogtag Certificate System](#)
4. [CFSSL](#)

TLS automation == Public Key Infrastructure (PKI)

PKI Candidates

1. ~~Let's encrypt~~

- No client certificates + potential information leaks (certificate transparency)

2. ~~Boulder (ACME CA)~~

- High number of modifications for internal use

3. ~~Dogtag Certificate System~~

- No active development, Fedora focus

4. CFSSL

CFSSL: Cloudflare's PKI and TLS toolkit

<https://github.com/cloudflare/cfssl>

Written in Go (precompiled binaries available)

APT package for Ubuntu ≥ 18.04

- CFSSL simplifies **CA creation**
- CFSSL comes with an HTTP-based **API server**

Two binaries + three commands => simple PKI

```
cfssl gencert -initca ca.json | cfssljson -bare ca  
-> ca.pem, ca-key.pem
```

```
cfssl serve -ca ca.pem -ca-key ca-key.pem  
-> API server on "127.0.0.1:8888"
```

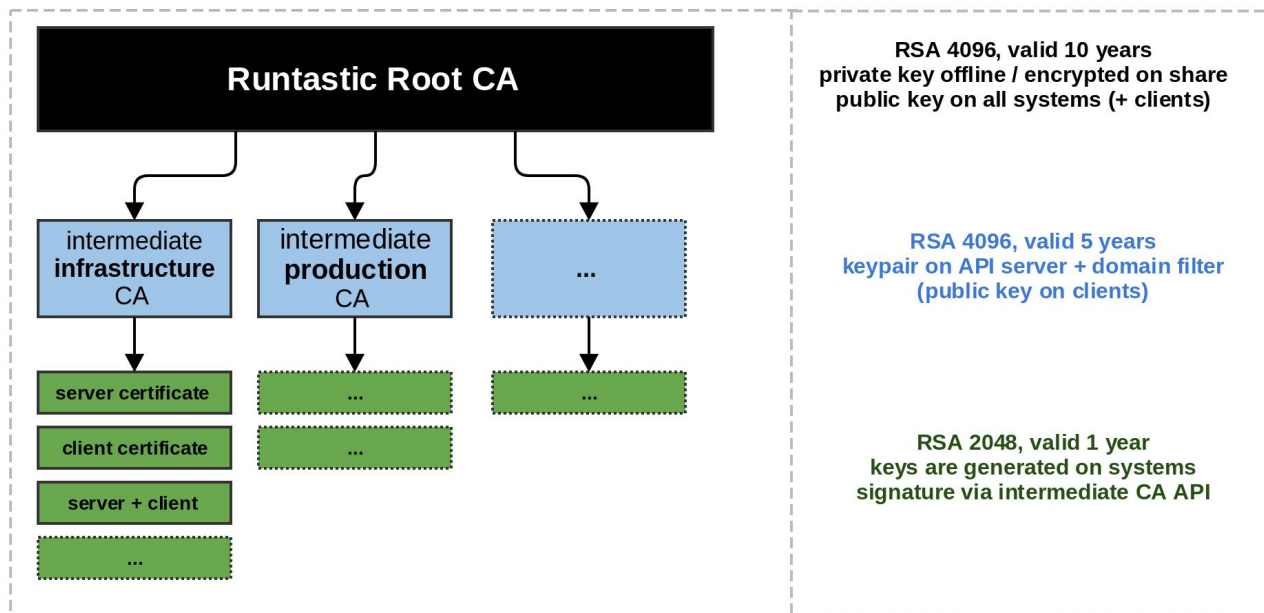
```
cfssl gencert -remote=127.0.0.1:8888 csr.json | cfssljson -bare server  
-> server.pem, server-key.pem
```

CFSSL PoC to Production Use

- Protect CFSSL endpoints with **API secrets**
- **Whitelist** for allowed hostnames (e.g. *.runtastic.dev)
- Define **cryptographic algorithms** + certificate **lifetime**
- Setup OCSP responder / **revocation** process

CFSSL at Runtastic

- Automated CFSSL setup with `ansible`
- **root CA** is kept **encrypted offline**
- Different **intermediate CAs for different** network **segments**



CFSSL at Runtastic == Automated TLS setup

- Client setup with simple Chef recipe
 - `server-certificate => true, client-certificate => true`
 - `/etc/cfssl/server.pem, /etc/cfssl/client.pem`
 - Applications use the key-pairs without caring about TLS implementation details
- Example use case
 - **authenticated and protected logging** with elasticsearch

Automate Secret Management

Secret Management Challenge: How to protect credentials of server applications?

credentials == db username and password, API tokens, etc.

~~Hardcoded in source code and committed to Git~~ 🤖💣 (see [gitrob](#))

1. Environment variables (*via dotfiles*)

- .env -> contains secrets and not tracked in git
- challenge: deployment and management (hard to automate)

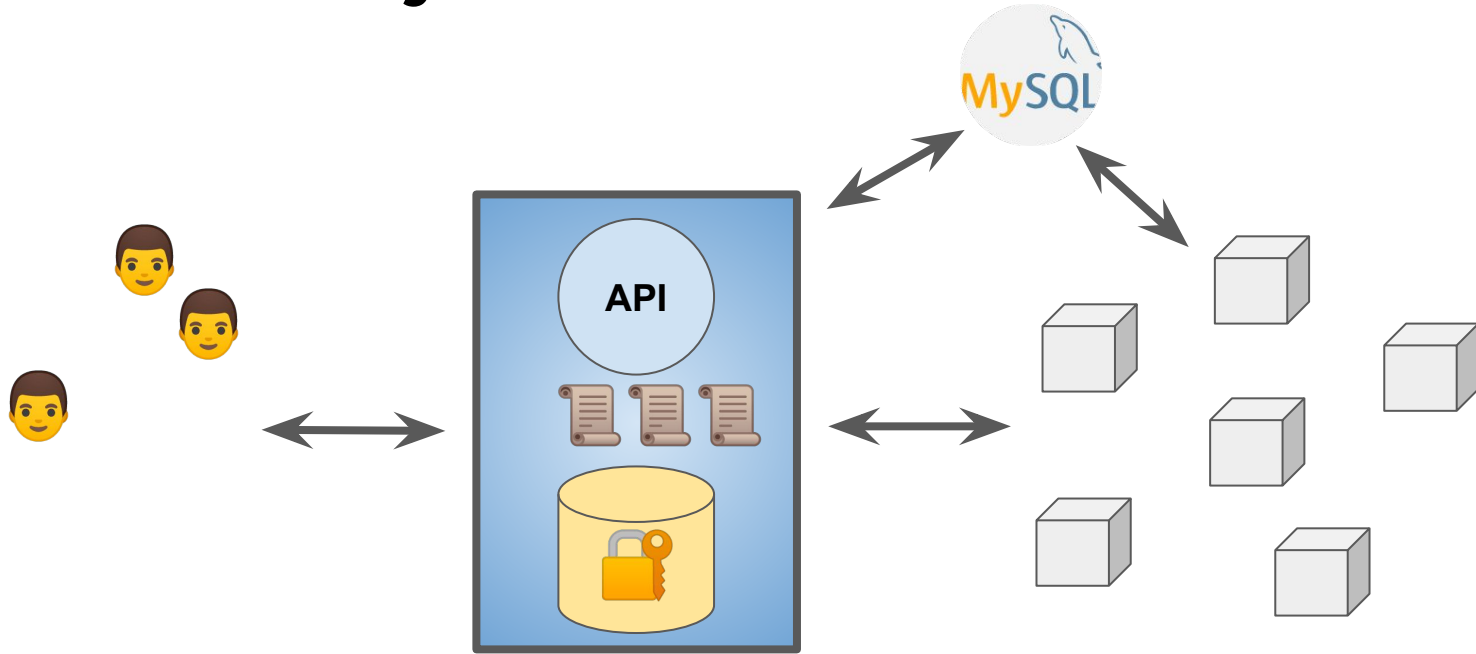
2. Centralized Key Vaults

- Available on AWS, Azure, Google Cloud
- on-premise vault**

```
DXe: 2014-04-21 18:46:21
Branch: master
Commit: Removing aws keys

@@ -57,8 +57,8 @@ public class EurekaEVCacheTest extends Abs
//
- props.setProperty("datacenter", "cloud")
- props.setProperty("awsAccessId", "<aws ac
+ props.setProperty("awsSecretId", "<aws s
+ props.setProperty("awsAccessId", "AKIAJCI
+ props.setProperty("awsSecretKey", "7JyrN
```


Centralized Key Vaults



Engineers

Edit Secrets, Configure Key Rotation

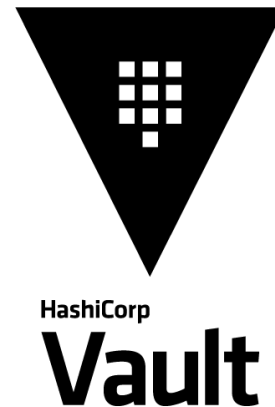
Vault

encrypted password file
(cmp. password manager)
API, Policies, Auditing

Services

Fetch secrets via Vault API

On premise vault: Vault by HashiCorp



1. **Open Source** and Enterprise versions
 - a. <https://www.vaultproject.io>
 - b. Natural fit because we already use Terraform
2. Written in Go (single binary that is both server & client)
3. Support for different “**secret-engines**”
 - a. Simple key-values
 - b. Databases such as MySQL, PostgreSQL, MongoDB, etc.
 - c. PKI engine for creating certificates
 - d. ...

Vault at Runtastic

- High-availability setup
- Auto-Unseal with Cloud Provider (HSM)

Example Use Case: **Vault for Kubernetes**

- Developers can **edit secrets for their services** in Vault
- Secrets are injected into Kubernetes containers
 - a. Policy defines which namespaces can access certain secrets (e.g. "prd")
 - b. Containers are assigned specific secrets (e.g. "websecrets")
 - c. Secrets are injected into Container environment

**Automate the
~~bo~~ring critical stuff**

Automated build pipeline checks

- Build pipeline checks for effective and automated security (audits)
 - Static and dynamic code analysis for potential issues
 - Checks if secrets have been committed
 - ...
- Example at Runtastic: Checks for **outdated/vulnerable libraries**
 - Developers need to keep their backend applications up-to-date
 - **Benefit:** transition from “Heise-driven security” to continuous attack surface reduction

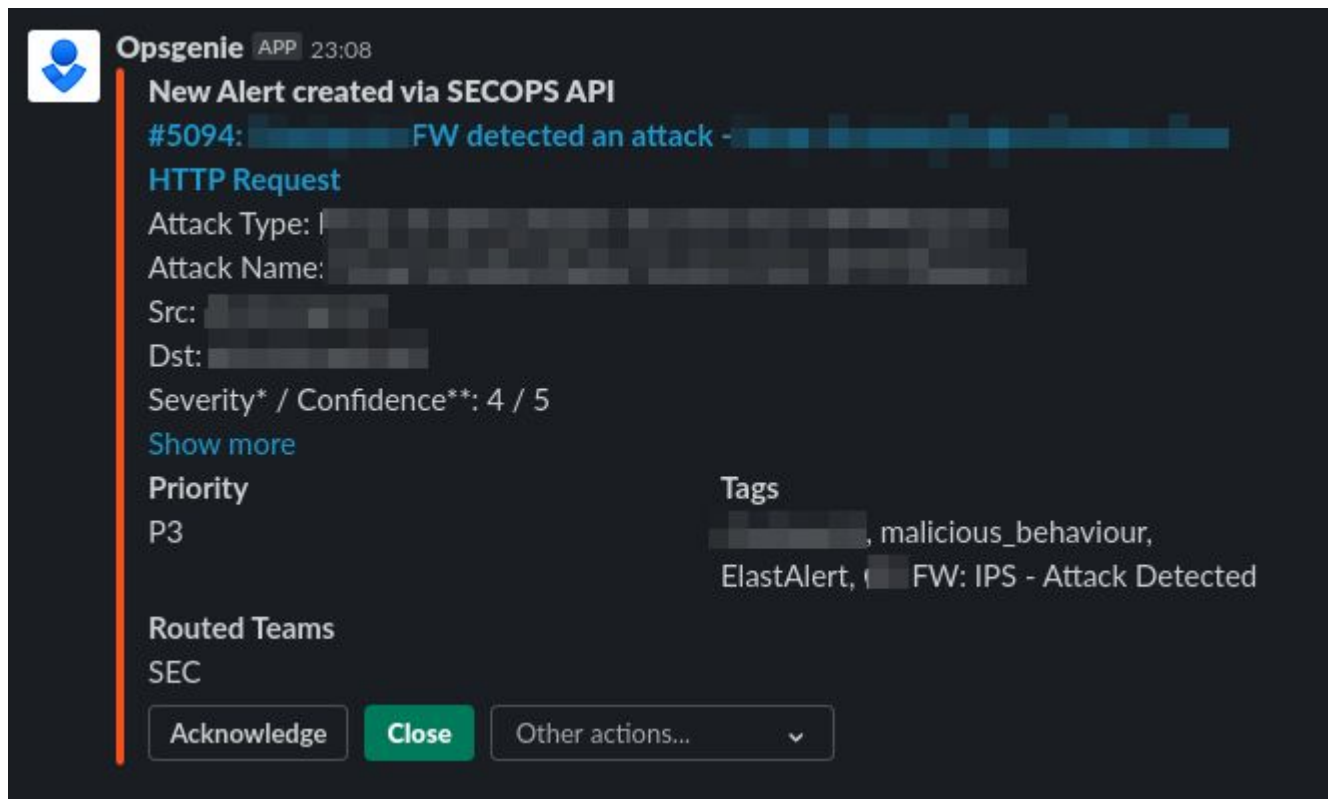


Handling of Alerts

- Organizational challenges
 - High amount of alerts
 - “Bystander-effect”
(somebody else is taking care)
- Spreadsheets / Tickets do not scale
- Runtastic opted for Opsgenie
 - Incident management workflow
 - Escalations and schedules
 - Email, Slack, Rest API, mobile Apps ...



Opsgenie ❤️ Slack



The screenshot shows a Slack message from the Opsgenie app. The message content is as follows:

Opsgenie APP 23:08

New Alert created via SECOPS API

#5094: [redacted] FW detected an attack - [redacted]

HTTP Request

Attack Type: [redacted]

Attack Name: [redacted]

Src: [redacted]

Dst: [redacted]

Severity* / Confidence**: 4 / 5

[Show more](#)

Priority P3

Tags [redacted], malicious_behaviour, ElastAlert, [redacted] FW: IPS - Attack Detected

Routed Teams SEC

Buttons: Acknowledge, **Close**, Other actions... (dropdown)

Automated follow-ups with Slackbots

E.g. If VPN users fail to login / roam country within short time:

1. Alert forwarded to Opsgenie
2. Security Chatbot picks up Opsgenie alert
3. Security Chatbot contacts employee on Slack to clarify issue (and closes ticket)



Security Chatbot APP 15:56

Hi mad! There have been **multiple failed attempts to log in to your Runtastic VPN account** on May 22nd at 3:56 PM.

Do you recognize this activity?

That was me

That was NOT me



Security Chatbot APP 14:57

Hi mad! There was a **login to your Runtastic VPN account from Austria** on May 22nd at 2:57 PM, you previously connected from Germany.

Do you recognize this activity?

That was me

That was NOT me

Automate TLS, use CFSSL

Setup once benefit “forever”

Automate Credential Handling, use Vault

Scales and reduces implementation effort

Build pipelines FTW

External Services for Alert Handling



Questions?

mad@runtastic.com



THANK YOU



runtastic.com

<https://www.runtastic.com/career>
mad@runtastic.com