CyberRes Fortify

# Application Security Present and Future

Frans van Buul, CISSP, CSSLP
Senior Product Manager, Fortify SAST
27 September 2022

# Introductions

Fortify

## About me…

- Frans van Buul
- Working at Fortify since 2014; as product manager since 2021.
- Background in (Java) software development and security consulting.
- Based in the Netherlands.
- Contact me:
  frans.buul at microfocus.com

## About Fortify…

- One of the world's leading appsec suites.
- Delivers SAST/DAST/SCA, on-prem and as-a-service.
- Originated around 2005.
- Part of the CyberRes portfolio.

# Agenda

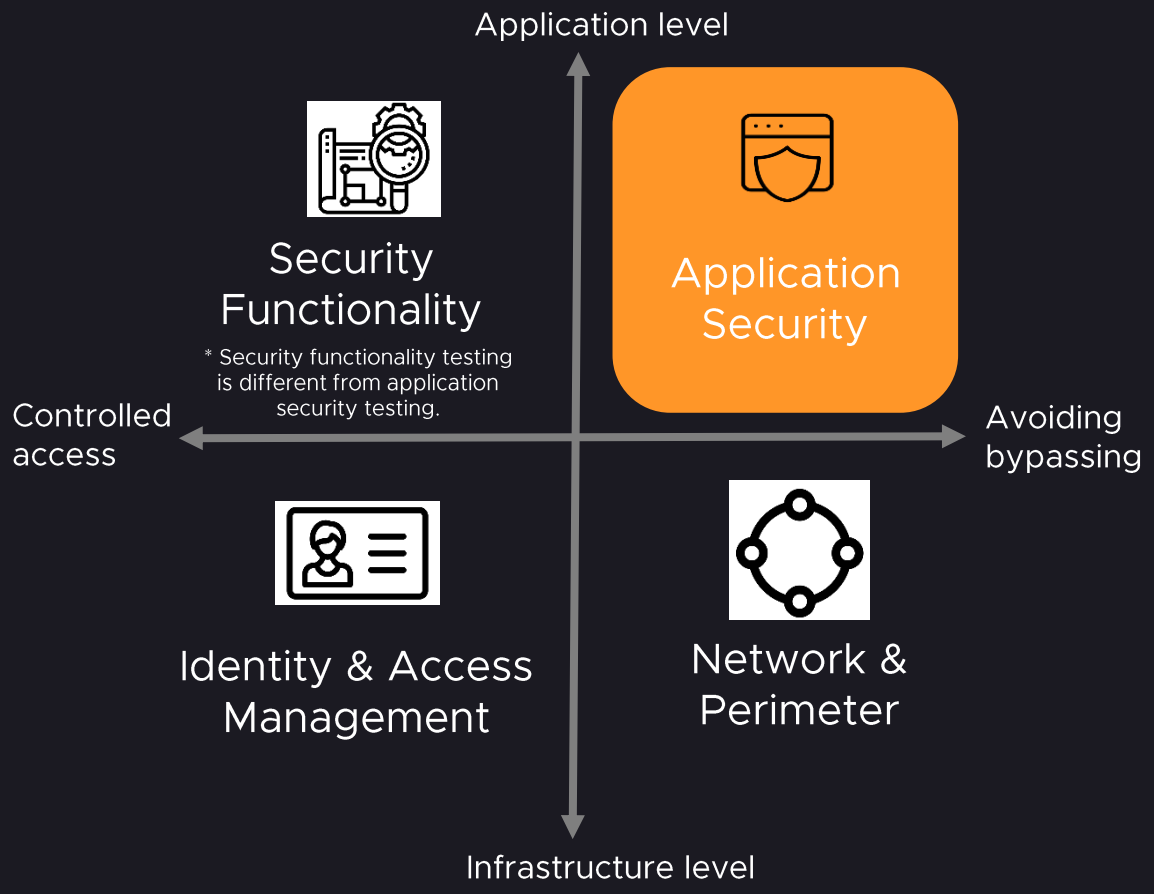
- Current state of AppSec; widely accepted good practices.
- Three important trends for the future of AppSec:
  - Supply chain risks
  - Cloud-Native AppSec
  - API Security


https://www.microfocus.com/en-us/cyberres/application-security

https://www.microfocus.com/media/white-paper/2022-appsec-trend-report-wp.pdf

# AppSec today: attackers moved to the app level

Application layer attacks are perceived as normal traffic and pass-through network, perimeter, data and endpoint security systems.




**Application security**

- Not mature; lack of developer training
- Growing attack surface: more applications, more connected to the Internet
- Accelerating releases reduce time available for security

**Infrastructure security**

- Highly mature
- Substantial investments in place
- Systems are more secure out-of-the-box than ever

# AppSec today: automation is key

*Assume as an example...*

- 1000 apps
- 50 releases/app/year
- Security Check before each release

*then you're executing...*

50,000 checks/year

Any manual approach will be entire infeasible!

*which requires...*

- Automation of the check itself (scanning technology).
- Automation, integration and management of the process

# AppSec today: there is no silver bullet

| Tech | What is it? | +Strengths / -Weaknesses |
|---|---|---|
| **SAST** | Static Application Security Testing<br><br>≈ Automated Security Code Review<br>Code doesn't get executed | + Fast<br>+ Clear, early, actionable feedback to developers<br>- Requires source code to be available and supported language<br>- Sometimes high false positive rate |
| **DAST** | Dynamic Application Security Testing<br><br>≈ Automated Penetration Test<br>Code is running | + Doesn't require source code access<br>+ Any programming language<br>- Relatively slow<br>- Higher false negative rate<br>- Unclear feedback for developers |
| **IAST** | Interactive Application Security Testing<br><br>Code is running with agent inspecting it | + Combines many of the strong points of SAST and DAST<br>- Inherently highly limited in supported tech<br>- Needs another tech (DAST or testing) to trigger the detection. |
| **SCA** | Software Composition Analysis<br><br>Checks libraries/components of software for known vulnerabilities. | + Only practical way to detect issues in (open-source) components.<br>- Only checks issues in components. |

# 2022 AppSec Trends

# Software Supply Chain Security

## Beyond Composition Analysis and CVE scanning

**What do we mean *exactly* when we talk about**

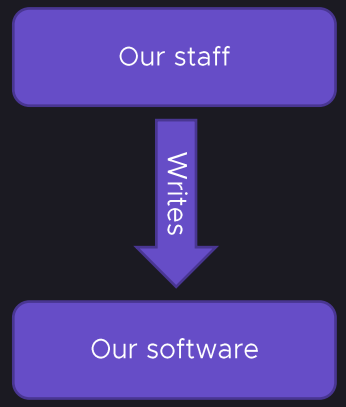
**“in-house developed software” or “custom software”**

**and its security aspects?**

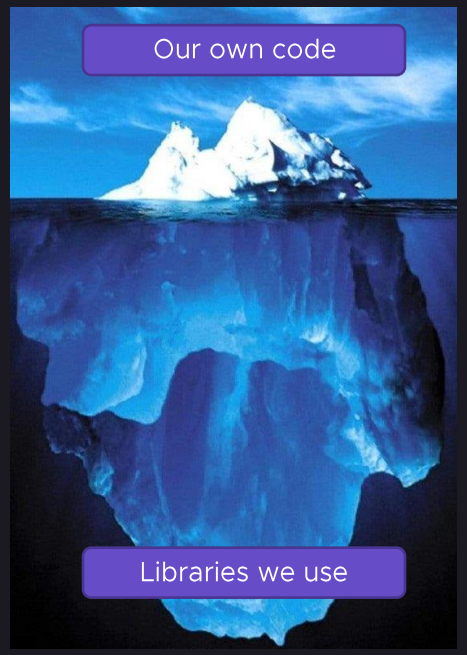
# Software Supply Chain Security

"in-house developed software"

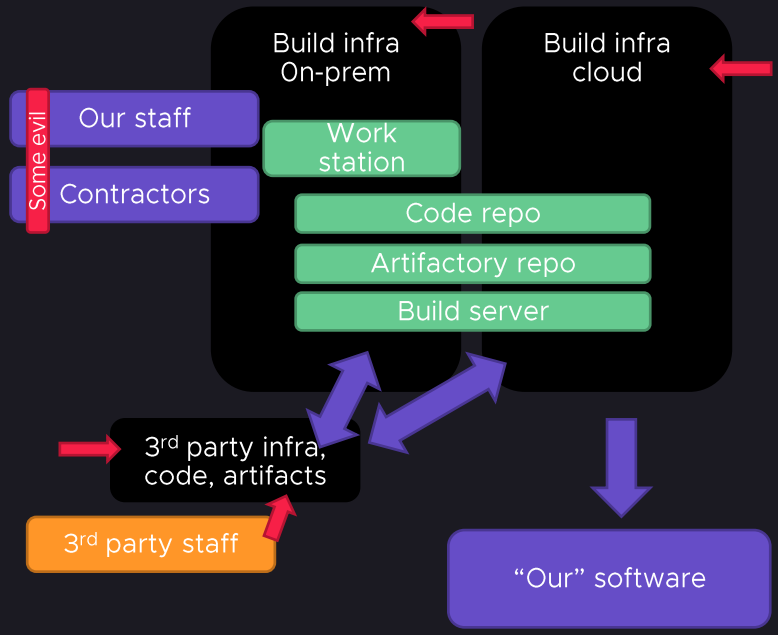
## Simplest picture




## Somewhat less naive




## More realistic (but still very incomplete)




- **This requires a holistic view on risk. It's not *just* SAST+SCA.**
- **Since much infra is code, SAST can play a role beyond appsec.**

# Software Supply Chain Security

Two illustrative examples of how SAST can help in supply chain security

## "Trojan Source"

- Attacking allowing malicious developers to make code look different to reviewer as it does to a compiler, bypassing peer review.
- Based on Unicode bidirectionality.
- Detectable using SAST, looking for specific Unicode patterns.

https://trojansource.codes/

## GitHub Script Injection

- GitHub workflows are YAML files defining the build.
- Provides access to variables containing user input.
- Variants on classic injection attacks are possible.
- Vulnerability detectable using SAST, parsing YAML/Bash

https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions

# Cloud-Native Application Security

**When we move our application workloads to the cloud,**

**does that impact application security?**

# Cloud-Native Application Security

**When we move our application workloads to the cloud,**

**does that impact application security?**

## In theory, maybe “no”

If you bring your applications to the cloud as-is (a.k.a. “**lift and shift**”), then: no.

## In theory, maybe “yes”

If you bring you change your apps to leverage cloud tech (a.k.a. “**cloud-native**”), then: yes. Your appsec program and tools may or may not be effective in the face this new technology, and you need to evaluate that.

## Practically, always “yes”

Majority of organization doesn’t choose one model or the other, but make pragmatic choices and have cloud-native tech in some cases.

# Cloud-Native Application Security

**Many cloud-native technologies can be covered very well by SAST, but only if the tool has support for the right languages, SDKs, vulnerability categories, etc.**

**Infrastructure as Code**

CloudFormation, Terraform, Ansible, etc.

(JSON, YAML, HCL)

**Cloud-specific SDKs**

E.g. AWS DynamoDB, Azure CosmosDB

(8+ programming languages)

**Container technology**

Docker
Kubernetes

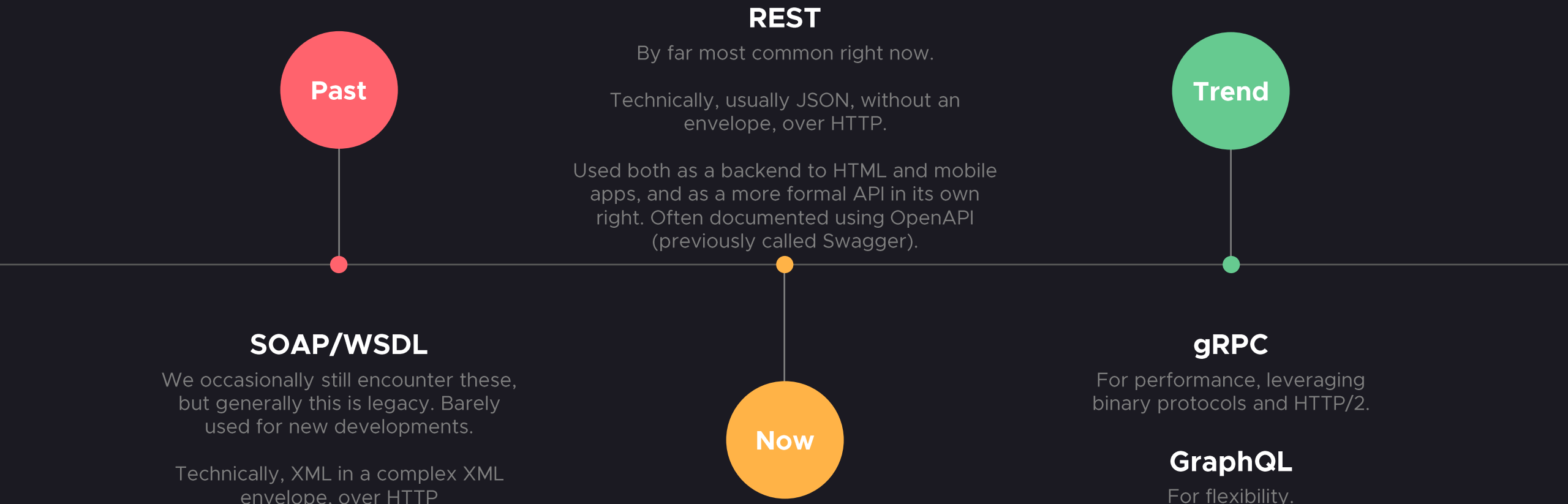
**Cloud Secrets**

Detection of hardcoded API keys/secrets etc. using known regular expression patterns

This space is huge ( {cloud providers} x {services} x {languages + config mechanisms} ).
No vendor covers it all. Ensure that your tool covers what you need.

# API Security

## A rapidly growing and evolving landscape



# API Security

What should we detect?


- OWASP Top 10 forms a good point of orientation.
  - OWASP Top 10 has been created for web apps (as an awareness tool).
  - OWASP **API Security** Top 10 is the equivalent for APIs.
- On a high level, **the differences are small**. Many Top 10 elements occur in both, e.g.,
  - Injection
  - Broken Authentication
  - Broken Authorization (Object, Function level)
  - Security Misconfiguration
- Some more API-specific things are lack of resource/rate-limiting, and improper asset management (including inappropriate debug endpoints in production).

# API Security

## **SAST** Challenges and Solutions

**Now**

- From a SAST point of view, the current **REST** APIs aren't that special. They are technically implemented in the same way between interactive webapps and APIs. Covering this is the heart of any SAST tool.
  - a small exception would be Cloud "serverless" architecture (AWS Lambda, etc.)
- A novel application of SAST in the REST space is to scan OpenAPI definition files for API problems that are detectable at that level.

**Trend**

- **GraphQL** and **gRPC** have new libraries and programming interfaces to implement these in Java/.NET/Python, etc. These require specific support by a SAST tool, otherwise it will be blind to potential problems.
  - This doesn't require fundamentally new SAST functionality, but it does require new rule content.

# API Security

## **DAST** Challenges and Solutions

**Now**

- Traditional DAST works by “crawling” HTML. That doesn’t work for **REST** APIs.
  → DAST solutions should read OpenAPI definitions or use Postman collections.
- Traditional DAST uses browser-based authentication methods. REST APIs may use others, like JWT. Specific challenges like short-livedness of tokens occur.
  → DAST solutions should support these alternative authentication methods as well.

**Trend**

- **GraphQL** and **gRPC** share with REST that they can’t be discovered using crawling. Definitions need to be read, but these aren’t OpenAPI/Swagger definitions.
- Both styles of APIs support introspection that allows a definition to be generated dynamically, which can then be picked up by a DAST tool.

# Some other trends...

- *AppSec Is Evolving from Shift-Left to Shift Everywhere.*
- *AppSec Orchestration and Correlation.*
- *Next-Generation DAST.*
- *Machine Learning and AI Are Key to the Next Evolution of Automation.*

**Come see us at our booth, visit the website and/or read the full trend report to learn more.**


Fortify
CyberRes
A Micro Focus line of business