



# Application Security

---

Usergroup-Treffen Karlsruhe, 12. Oktober 2023



PRESENTER

**Tania Bayer**  
Security Specialist



PRESENTER

**Robin Whyss**  
Lead Security SE



PRESENTER

**Rob Vissers**  
Regional Director Security

# What is the value?

Value is clear  
for the  
observability  
platform:

But what is the  
value of the  
application  
security option?



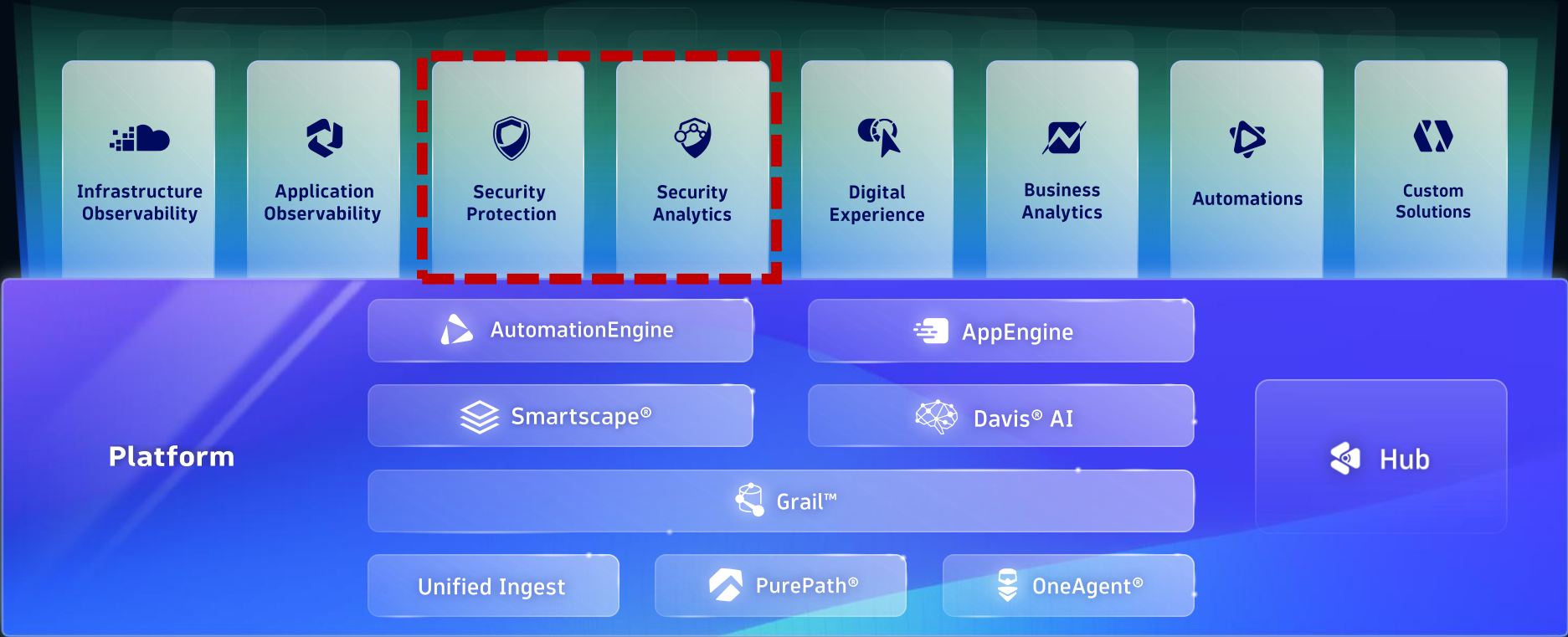
**What is this?**

**Dynatrace: Application  
Security but with  
much more context.**



Analytics and Automation for Unified Observability and Security

CLOUD DONE RIGHT.



- Topology
- Traces
- Metrics
- Logs
- Behaviour
- Code
- Metadata
- Network





# 100+ BUILT-IN USE-CASES IN 20+ CATEGORIES

Leverage platform engineering in the Dynatrace platform right away

DEVELOP


RELEASE

OPERATE

RESOLVE

PROTECT


IMPROVE


 **OBSERVABILITY AS CODE**


 **QUALITY GATING**

 **MONITORING**


 **ROOT CAUSE**


 **VULNERABILITY ANALYSIS**


 **CARBON IMPACT**


 **TRACING**


 **ANALYZE**

 **FINOPS**

 **PROBLEM REMEDIATION**


 **RUNTIME PROTECTION**

 **APPLICATION PERFORMANCE MONITORING**


 **PROFILING**

 **RELEASES**

 **DIGITAL EXPERIENCE**

 **WORKFLOW AUTOMATION**


 **SECURITY ANALYTICS**

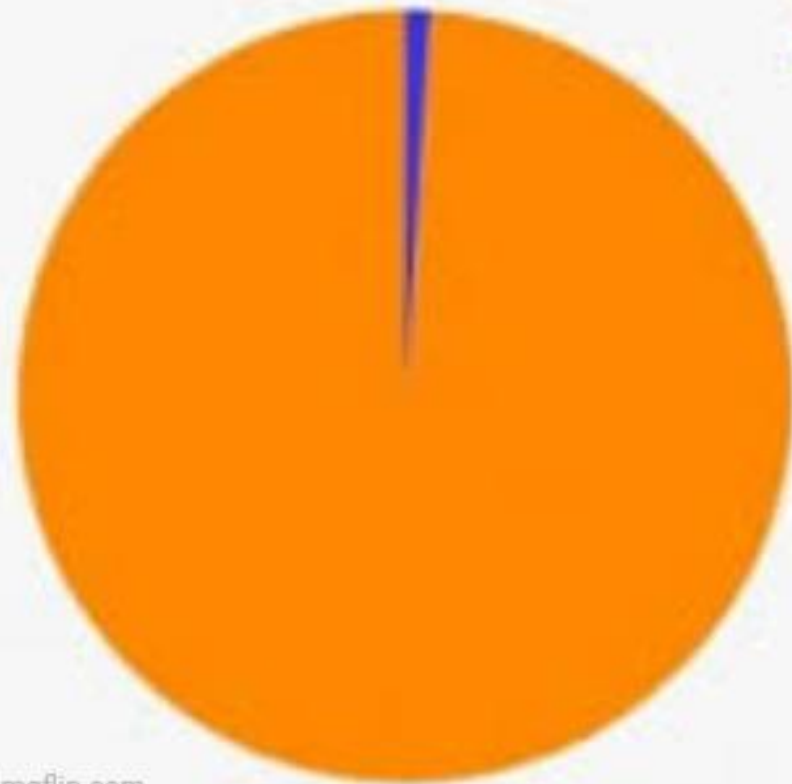
 **SESSION REPLAY**



# WHAT I SPEND MY TIME ON AS A SOFTWARE DEVELOPER

 Actual development work

 Finding creative ways to get around my company's terrible security policies

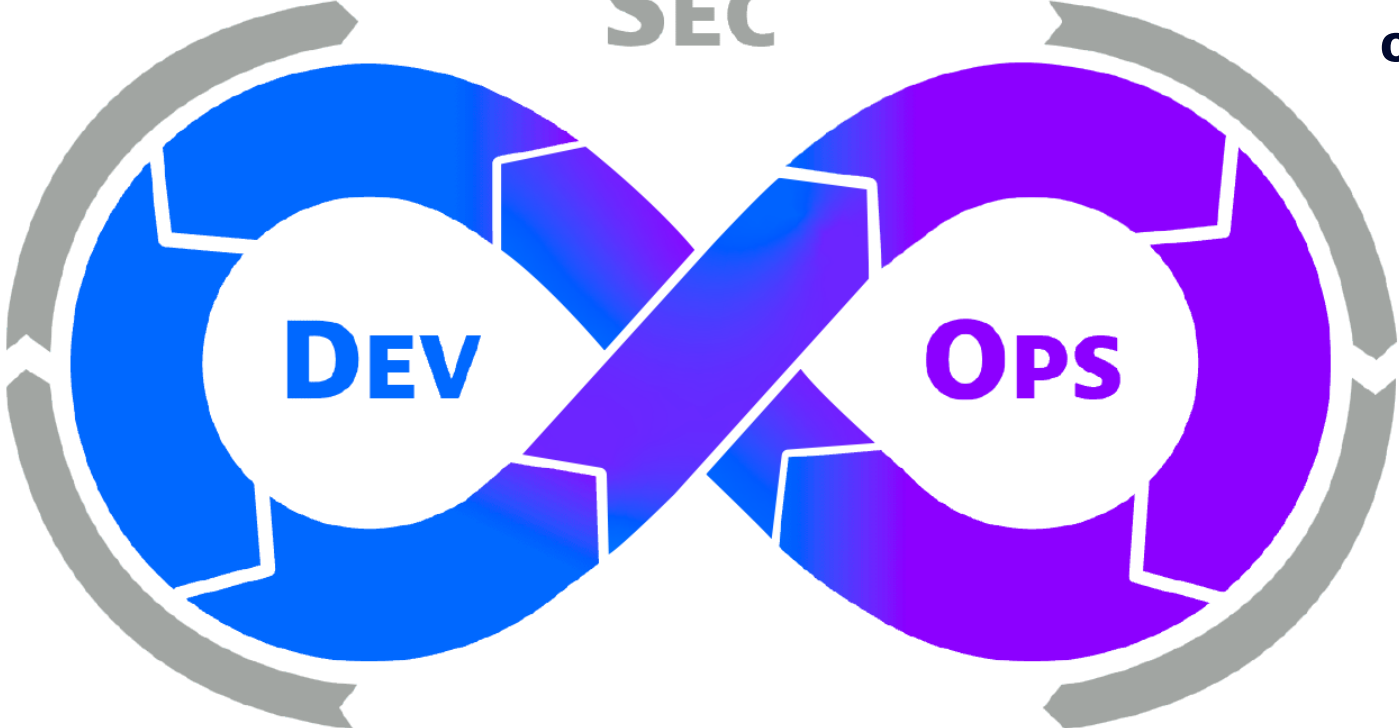


# DevSecOps

Pre-deployment  
security

SEC

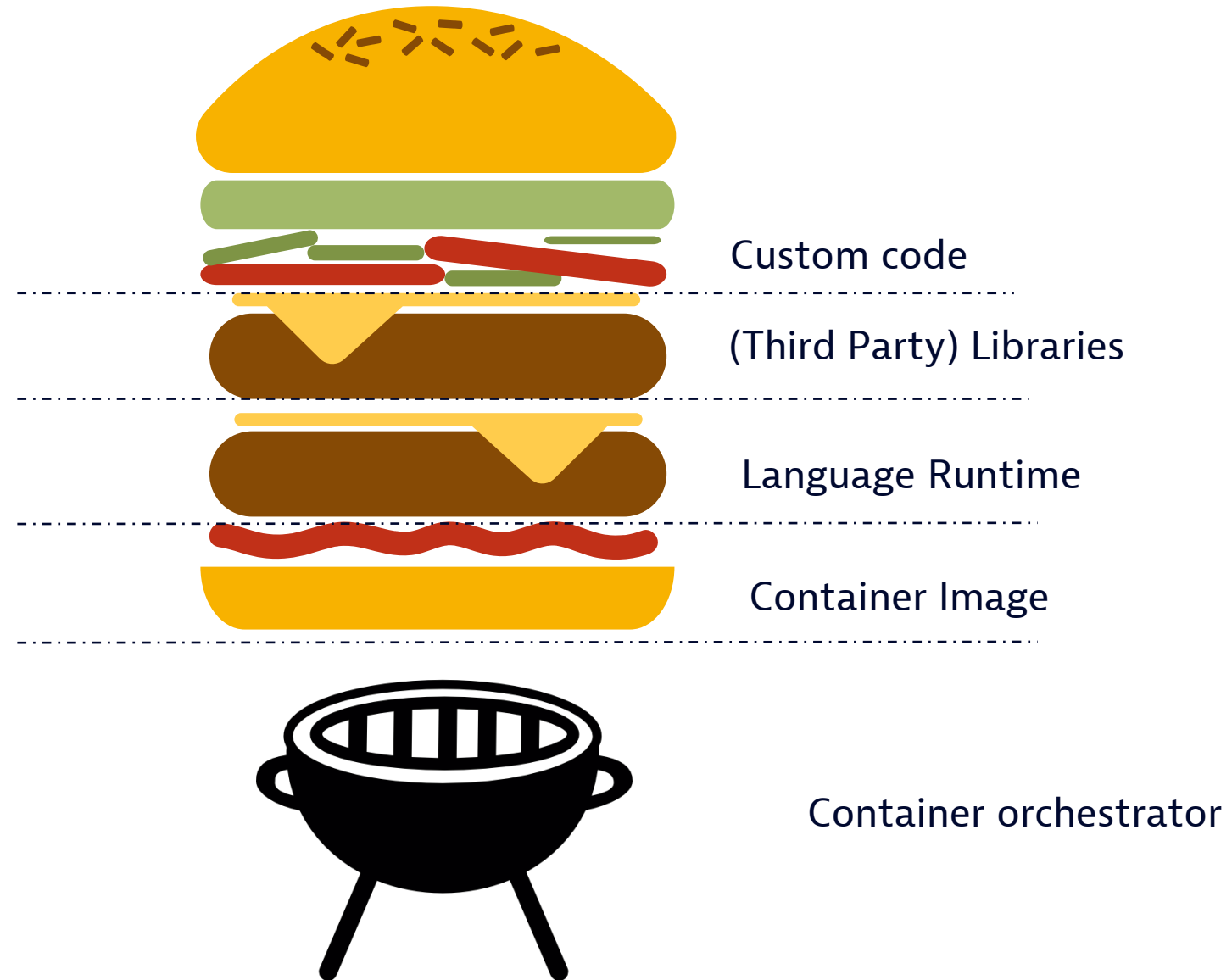
Runtime  
observability &  
security



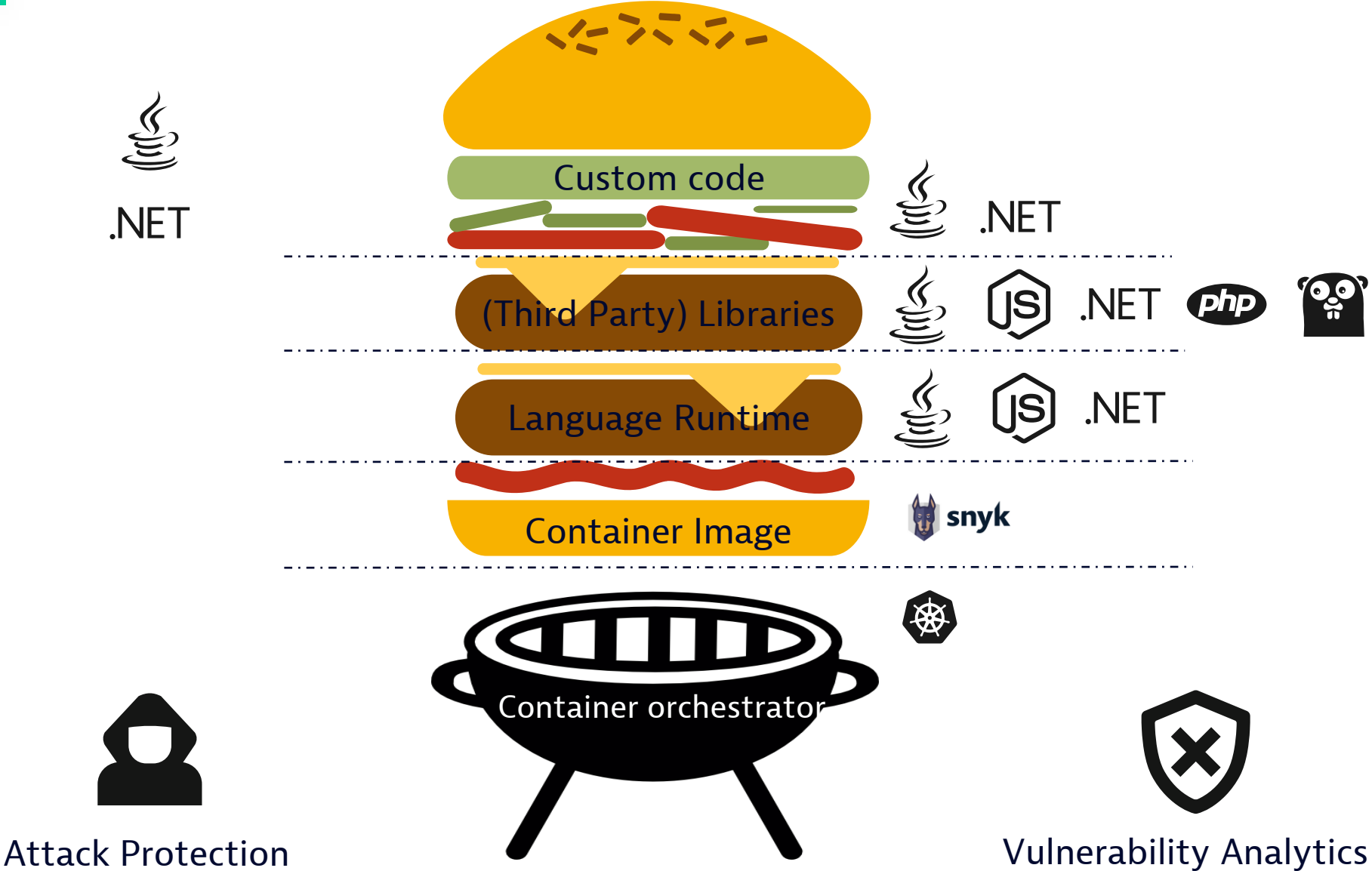
Missing Governance,  
Blindspots &  
Incomplete Coverage



# The modern application

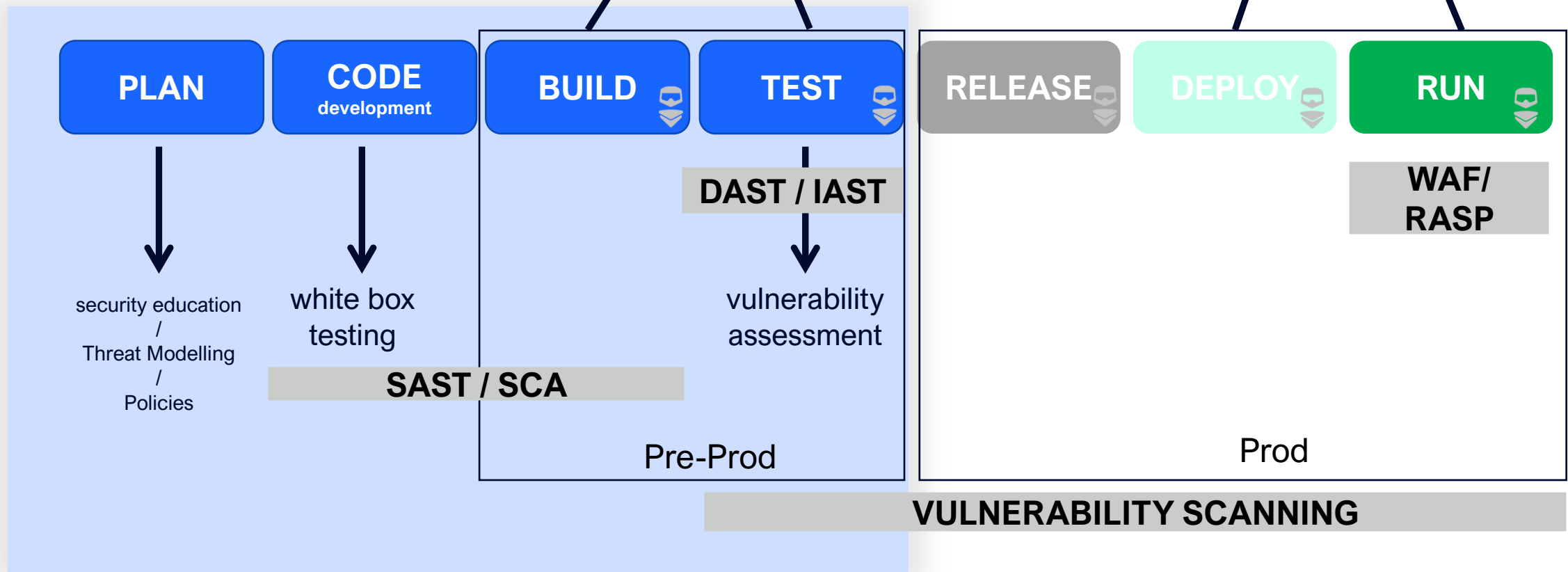


# The modern application





# SDLC



**Shift Left**





Analytics and Automation for Unified Observability and Security **CLOUD DONE RIGHT.**

Infrastructure Observability    Application Observability    **Security Protection**    **Security Analytics**    Digital Experience    Business Analytics    Automations    Custom Solutions

Platform

AutomationEngine    AppEngine    Hub

Smartscape®    Davis® AI

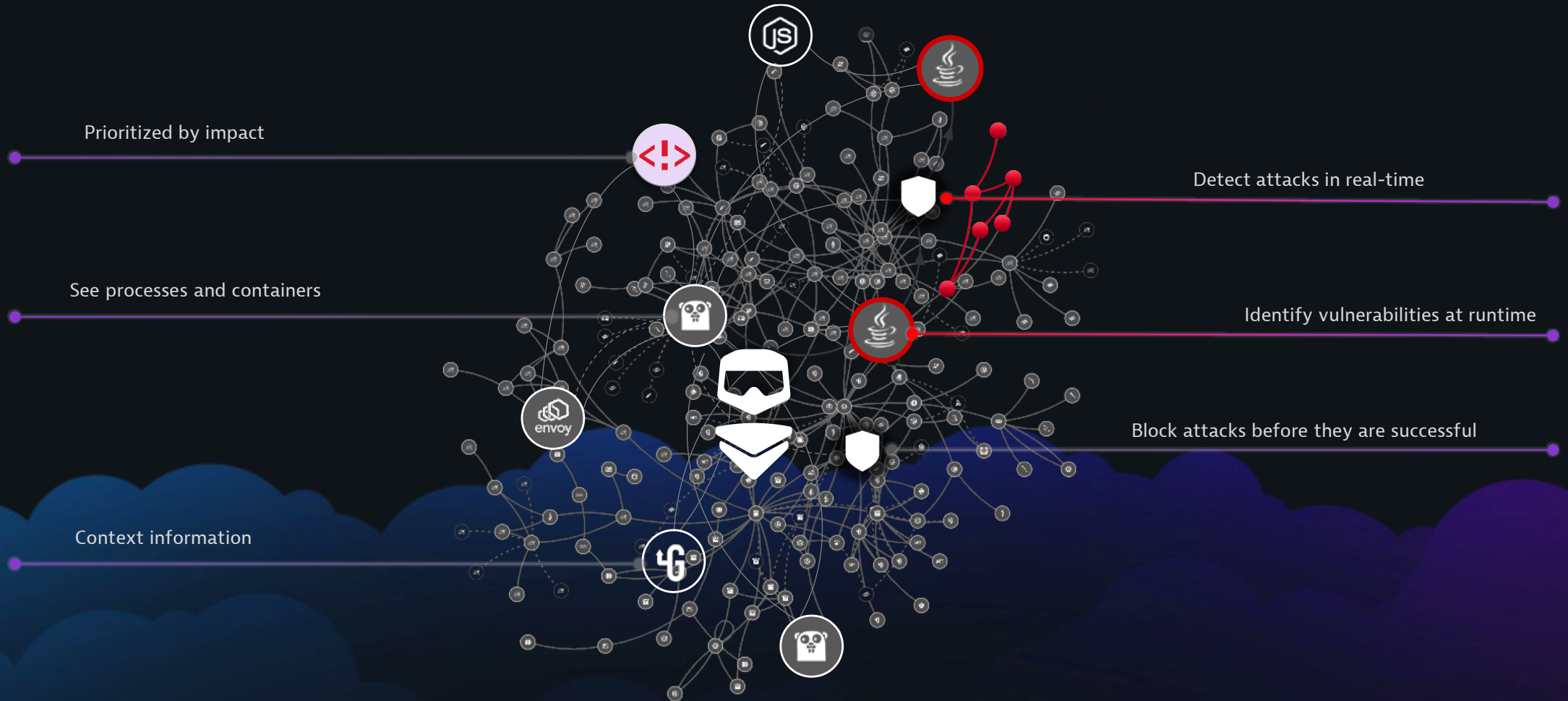
Grail™

Unified Ingest    PurePath®    OneAgent®

Topology    Traces    Metrics    Logs    Behaviour    Code    Metadata    Network


# Observability + Security: Real-time Security


Dynatrace continuously observes, learns and auto-adapts to changes in real-time to detect security problems automatically (even the ones you never anticipated).



# Dynatrace Application Security

Full-stack analysis of the entire environment  
Where are the weak spots?  
Vulnerabilities – assets – exposure  
Overview without scanning lists etc.

 **Third-party vulnerabilities**  
Detect vulnerabilities and assess their risk in real-time

 **Code-level vulnerabilities**  
Find vulnerabilities in custom code



Continuous runtime & forensic analytics  
How is the environment being used?  
Traces – attacks – logs

 **Runtime Application Protection**  
Detect and block attacks in real-time



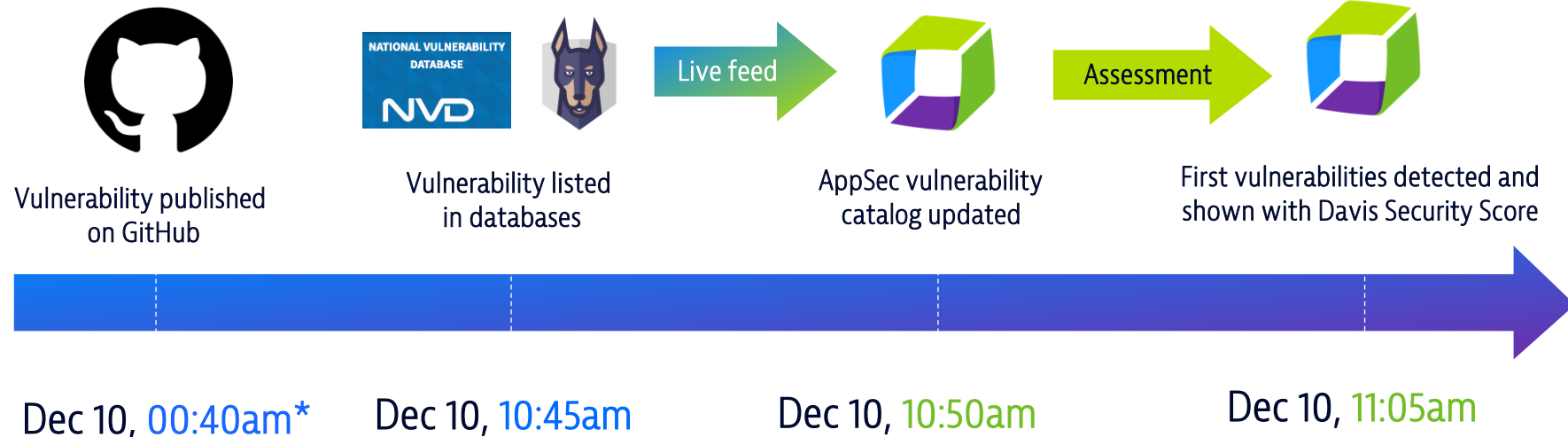
AI-assisted risk assessment in real time

# Use Case: Log4Shell – detect and mitigate new critical vulnerabilities

- **Reduce risk** by identifying vulnerabilities at runtime in real-time

- **Save time** with context-based risk assessment and prioritization

- **Improve DevSecOps collaboration** across teams with a consolidated view and context information



snyk Vulnerability DB

Arbitrary Code Execution

Affecting org.apache.logging.log4j:log4j-core package.

10.0 CRITICAL

ATTACK COMPLEXITY: Low

SCOPE: Changed

CONFIDENTIALITY: High

INTEGRITY: High

AVAILABILITY: High

Public internet exposure  
Exposure: Public network

Sensitive data assets  
Affected: within range

Vulnerable functions  
Not available

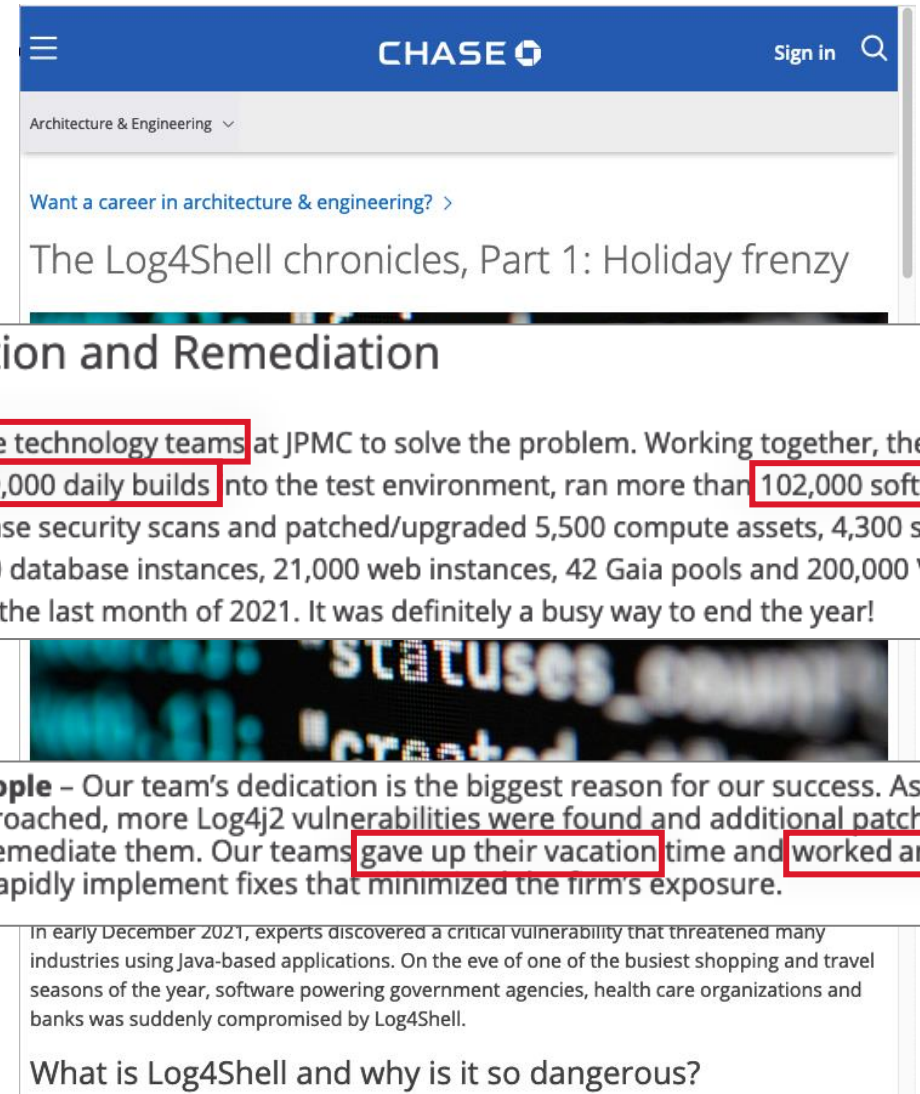
10  
Critical risk

Dynatrace identified Log4Shell in production apps minutes after it became known



# New critical vulnerability

- Static scans / file system search
  - Can take a long time
  - High false-positive rate (files not in use)
  - Might miss some instances (fat-jar, renamed files)
  - Does it cover containers?
- Repository scan / run-pipelines
  - Significant effort
  - Does not cover all applications (e.g. COTS)
  - False positives (libraries not used in production)
- How to prioritize?
- This often ends in war rooms, over-time, excel files being sent around, etc.



The screenshot shows a Chase website page for 'Architecture & Engineering'. The article title is 'The Log4Shell chronicles, Part 1: Holiday frenzy'. The main heading is 'Prevention and Remediation'. The text describes the efforts of JPMC technology teams to address the Log4Shell vulnerability during the holidays. Key statistics and actions are highlighted in red boxes: 'It took all the technology teams at JPMC to solve the problem. Working together, they performed 9,000 daily builds into the test environment, ran more than 102,000 software update release security scans and patched/upgraded 5,500 compute assets, 4,300 storage nodes, 2,300 database instances, 21,000 web instances, 42 Gaia pools and 200,000 VDI instances in the last month of 2021. It was definitely a busy way to end the year!'. A section titled 'Talented people' states: 'Our team's dedication is the biggest reason for our success. As the holidays approached, more Log4j2 vulnerabilities were found and additional patches were released to remediate them. Our teams gave up their vacation time and worked around the clock to rapidly implement fixes that minimized the firm's exposure.' A sub-section mentions: 'In early December 2021, experts discovered a critical vulnerability that threatened many industries using Java-based applications. On the eve of one of the busiest shopping and travel seasons of the year, software powering government agencies, health care organizations and banks was suddenly compromised by Log4Shell.' The article concludes with the question 'What is Log4Shell and why is it so dangerous?'.

<https://www.chase.com/digital/resources/next-at-chase/architecture-engineering/log4shell-chronicles-part-1>



# Customer example: CVE-2022-42889 (Text4Shell)

- Repository scan alerted on CVE-2022-42889
  - Critical vulnerability (CVSS 9.8) affecting apache-commons-text
  - “a library focused on algorithms working on strings”
- Application that uses spring-data-solr
  - Depends on apache-solr
    - Depends on apache-commons-text
- Runtime analysis showed that the library was not loaded (not used)
- Spring Data Solr is end of life
  - To be replaced by Spring Data Elasticsearch

open / source / insights commons-text Maven

Maven artifact  
org.apache.commons:commons-text 1.9

Overview Dependencies Dependents Compare Versions

Direct 1352  
Indirect 7125

Note: Due to the large number of dependents, we show only a sample in the list below.

Artifact	Version	Relation
ca.uhn.hapi.fhir:hapi-fhir-base	6.0.1	Direct
cn.herodotus.engine:assistant-core	2.7.2.3	Direct
com.centit.support:centit-utils	5.3.2302	Direct
com.gitee.zodiacstack:zodiac-commons	1.5.18	Direct
com.github.liaomengge:base-common-utils	3.0.0.RELEASE	Direct
com.github.spotbugs:spotbugs	4.7.2	Direct

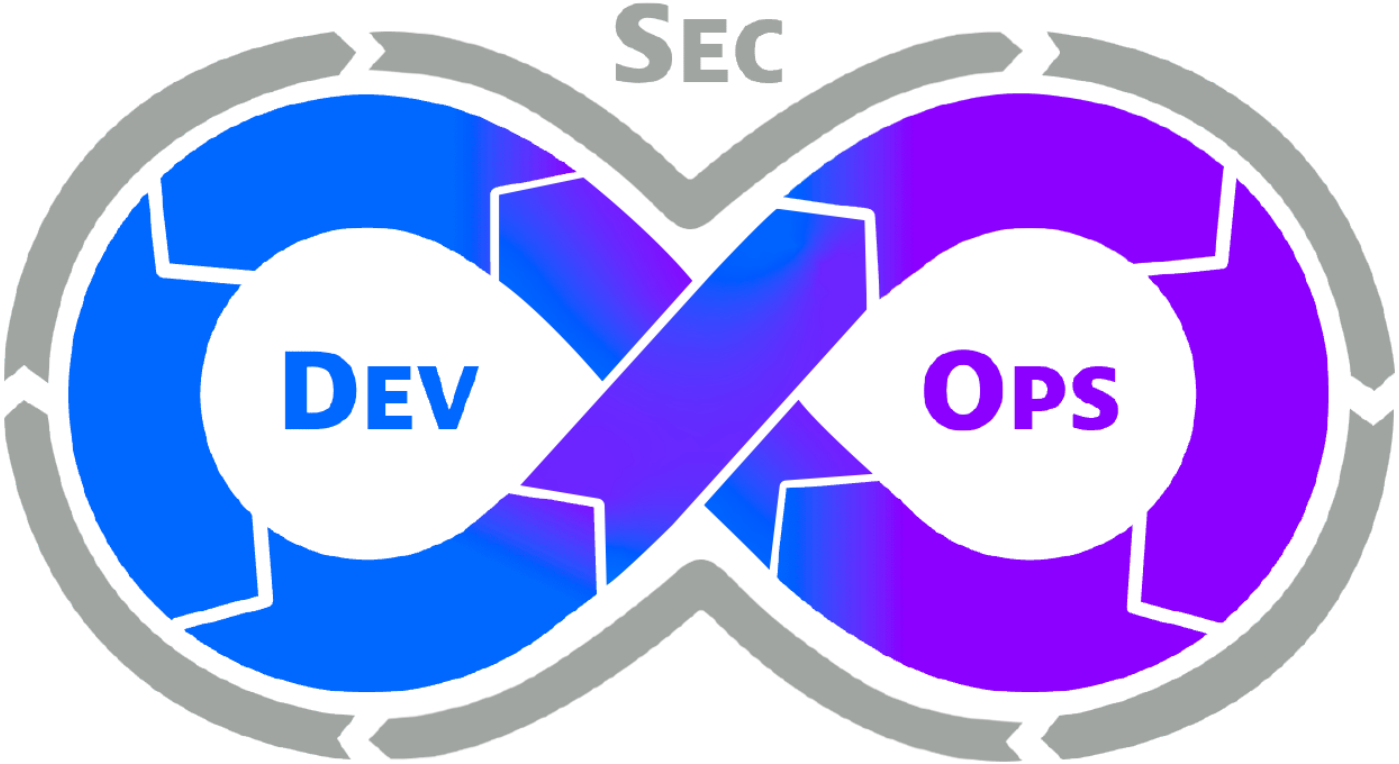
<https://deps.dev/maven/org.apache.commons%3Acommons-text/1.9/dependents>



# DevSecOps



Integrated pre-deployment security with  
Runtime observability and security



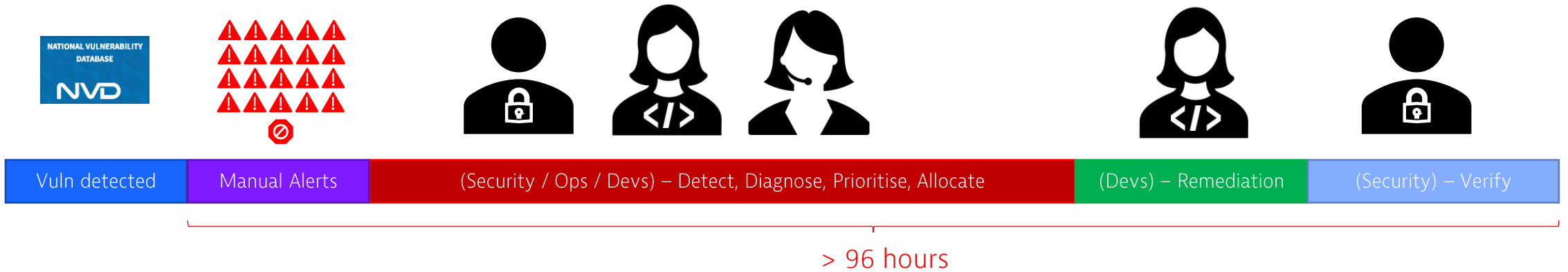
Full Governance  
No Blindspots &  
Complete Coverage



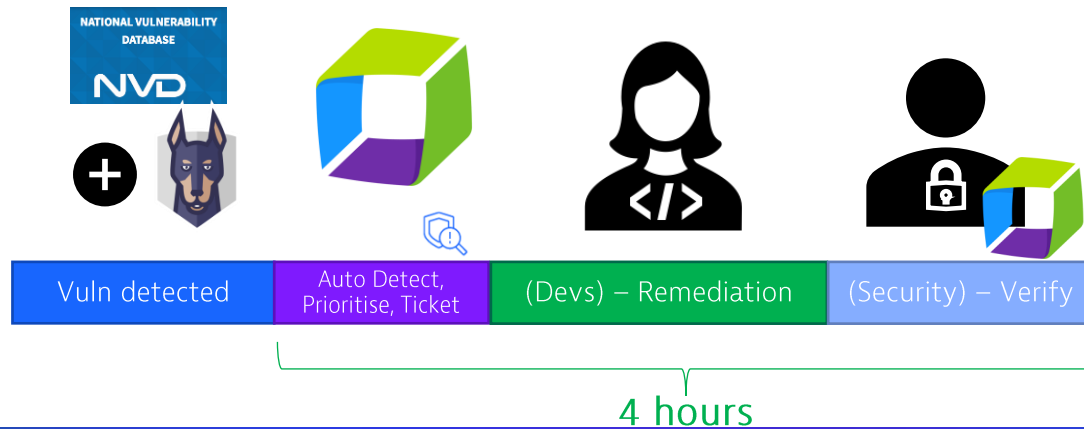


# Large Insurance Customer example

Before AppSec



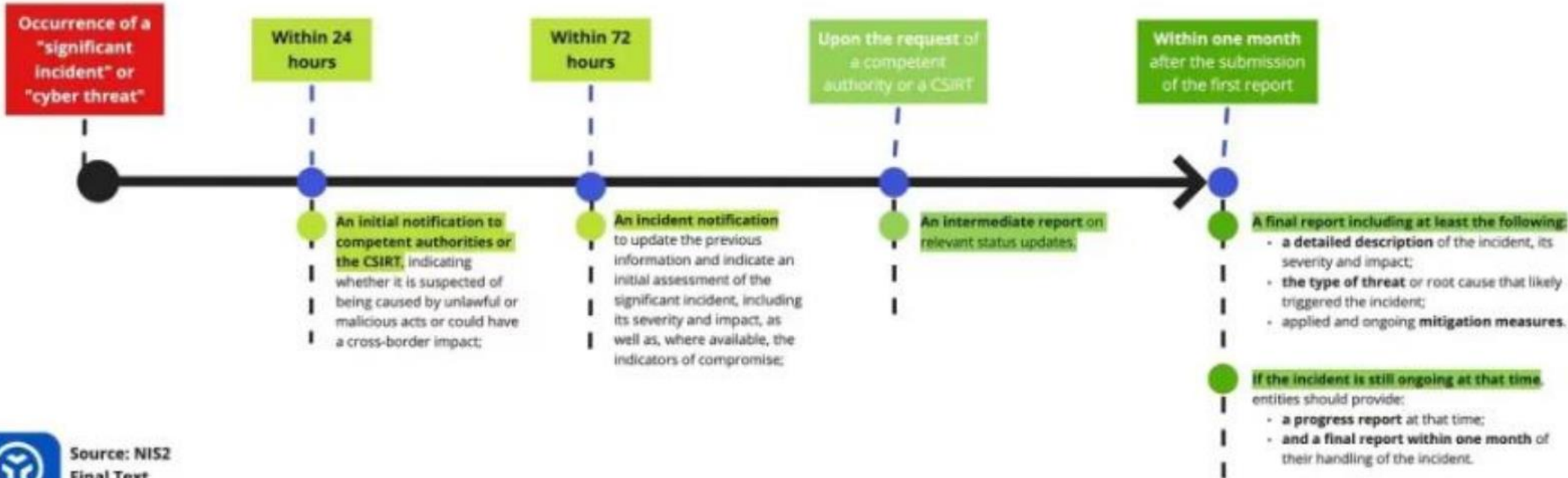
After AppSec



95% reduction in vuln risk remediation time

*“With Dynatrace’s AppSec Solution, we’ve improved critical vulnerability remediation from 96hrs to 4hrs”*

# NIS 2 – Response Obligations Timeline in the event of a significant incident or cyberthreat



Source: NIS2 Final Text, Article 6 & 23

# Example for ROI



## Improve Productivity

### Reduce Developer Time Spent Identifying and Remediating Known Vulnerabilities in OSS Libraries

After a new critical or high vulnerability has been alerted, developers spend time trying to detect not only which parts of the application and code contain the vulnerability, but also need to:

1. Document vulnerability locations (process, applications, containers, hosts)
2. Diagnose appropriate remediations
3. Prioritise application of remediations
4. Allocate remediation of vulnerabilities to fix
5. Verify remediation and rollout

Security and developers rarely know where vulnerabilities are, with environment technologies and versions constantly changing, manual searching, scanning, documenting is very time consuming.

*Our observability context and built-in AI & Automation address a gap in runtime vulnerability assessment that most companies struggle to address with existing security tools. With our approach you can:*

- Identify vulnerabilities better with continuous surveillance in runtime production environments
- Prioritize vulnerabilities using AI with better production environment context
- Operationalize & remediate with automation



# Q&A

