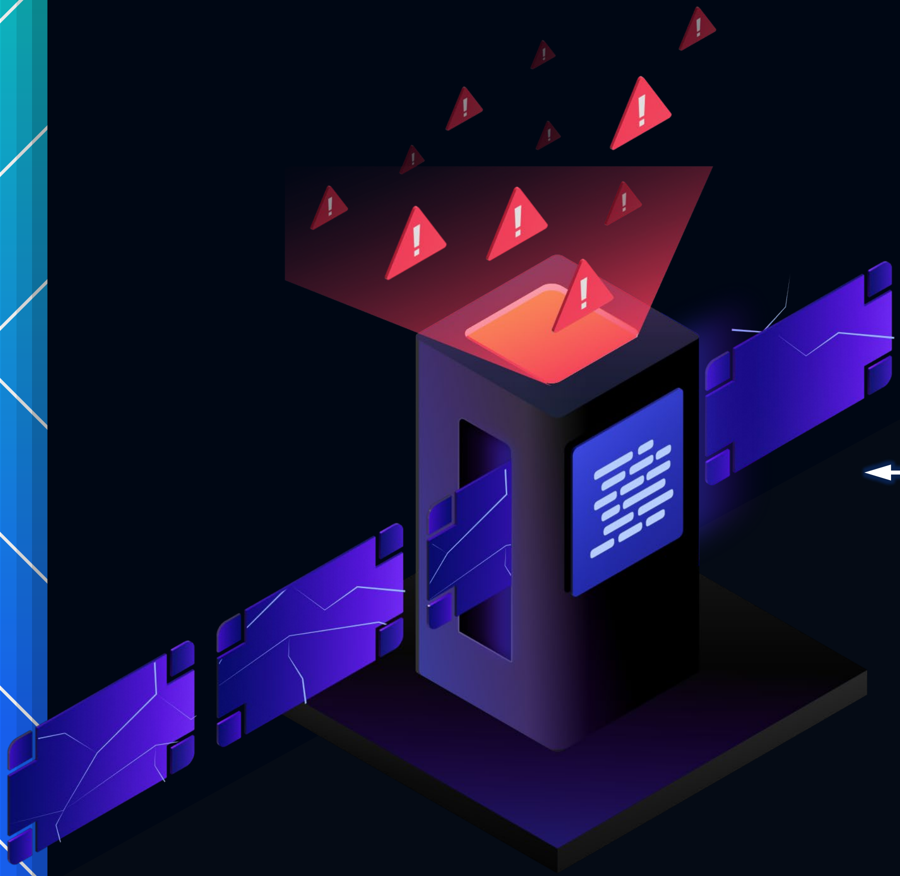
POTENCIA DEVSECOPS A ESCALA CON SEGURIDAD EN TIEMPO REAL

Alberto Dimanuel
SOLUTIONS ENGINEER
DYNATRACE

dynatrace
IMMERSE

TRADITIONAL APPROACHES AREN'T ENOUGH



Static Code Scanners

- Work well for static code early in the pipeline
- Customizable, manual config and updates
- Missing run-time context
- Too many alerts and false positives

Some vulnerabilities slip into production

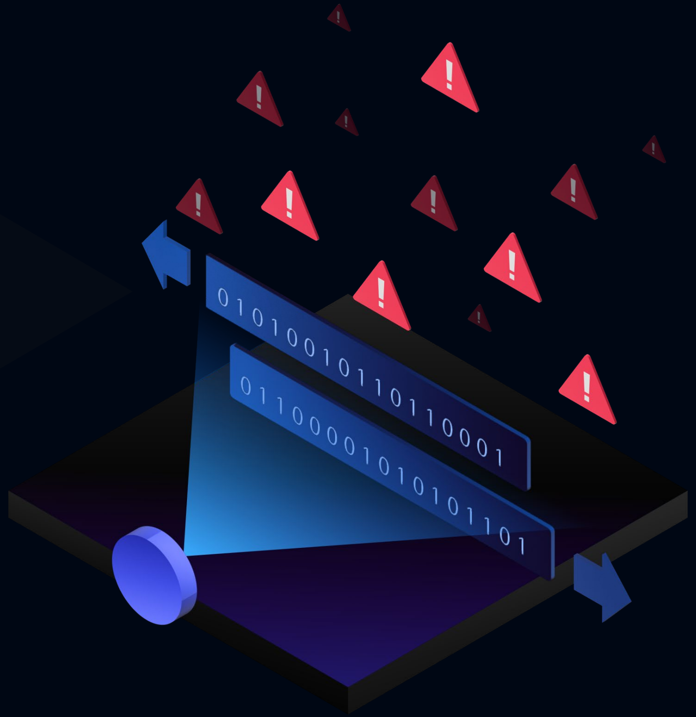
Pre-Production

Production

TRADITIONAL APPROACHES AREN'T ENOUGH

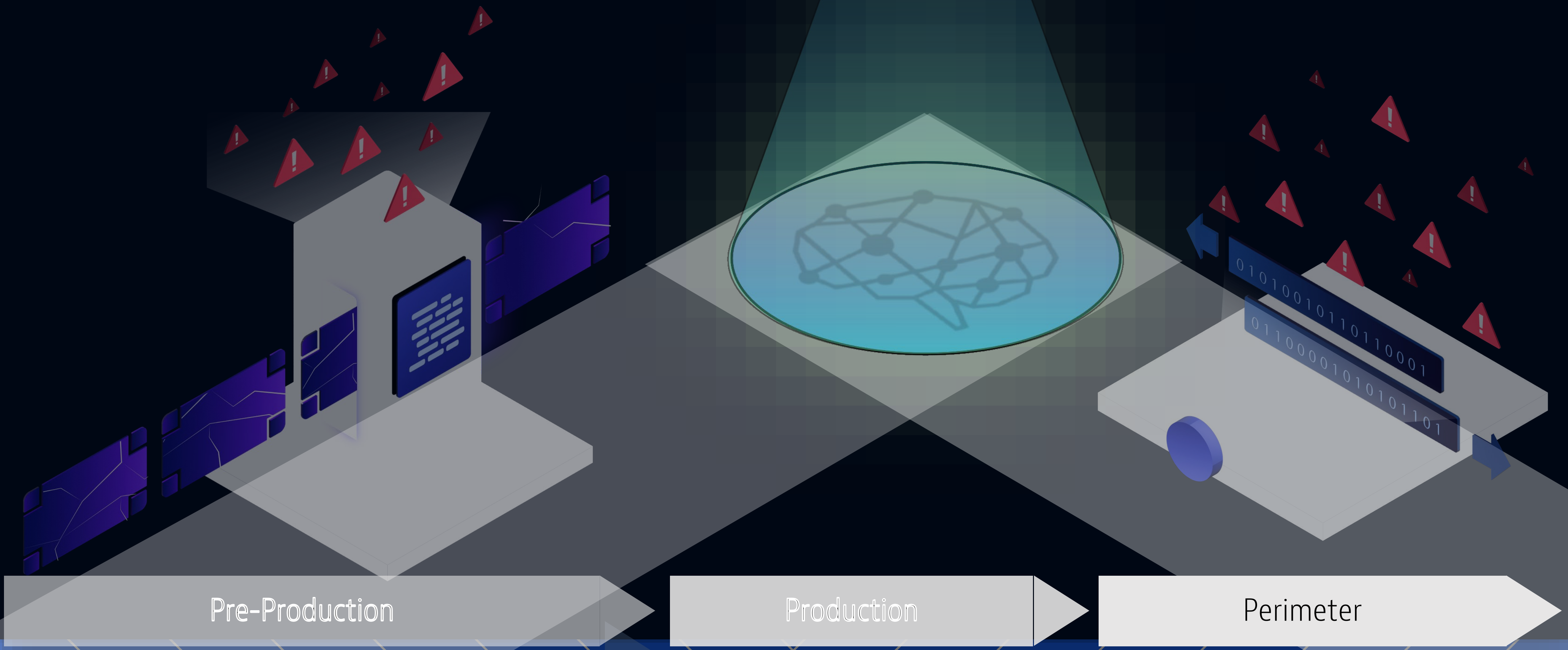
Network Traffic Scanners

- Designed for application agnostic attacks at perimeter
- Needs frequent updates
- Doesn't have application context
- Too many alerts and false positives



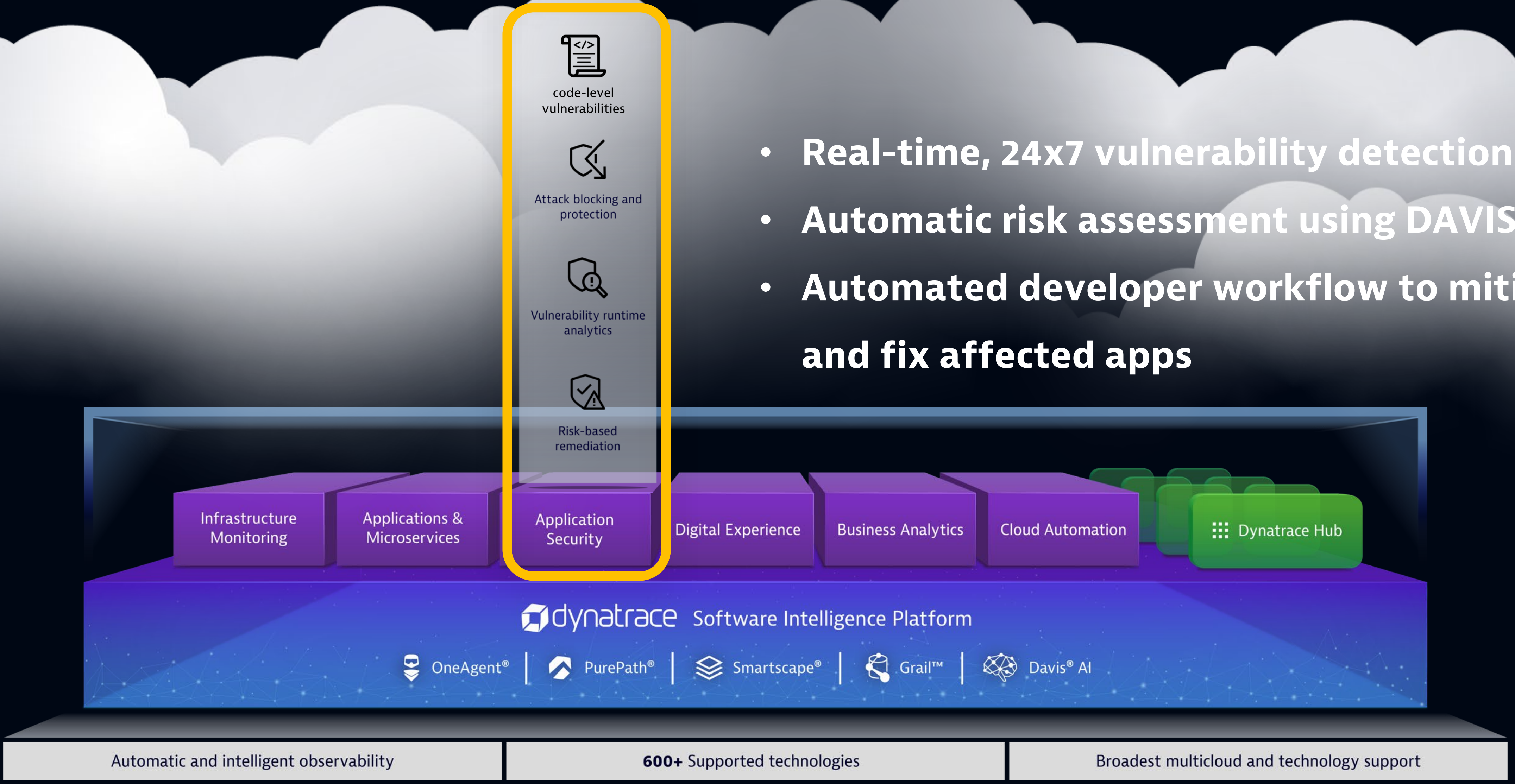
Pre-Production → Production → Perimeter

TRADITIONAL APPROACHES AREN'T ENOUGH



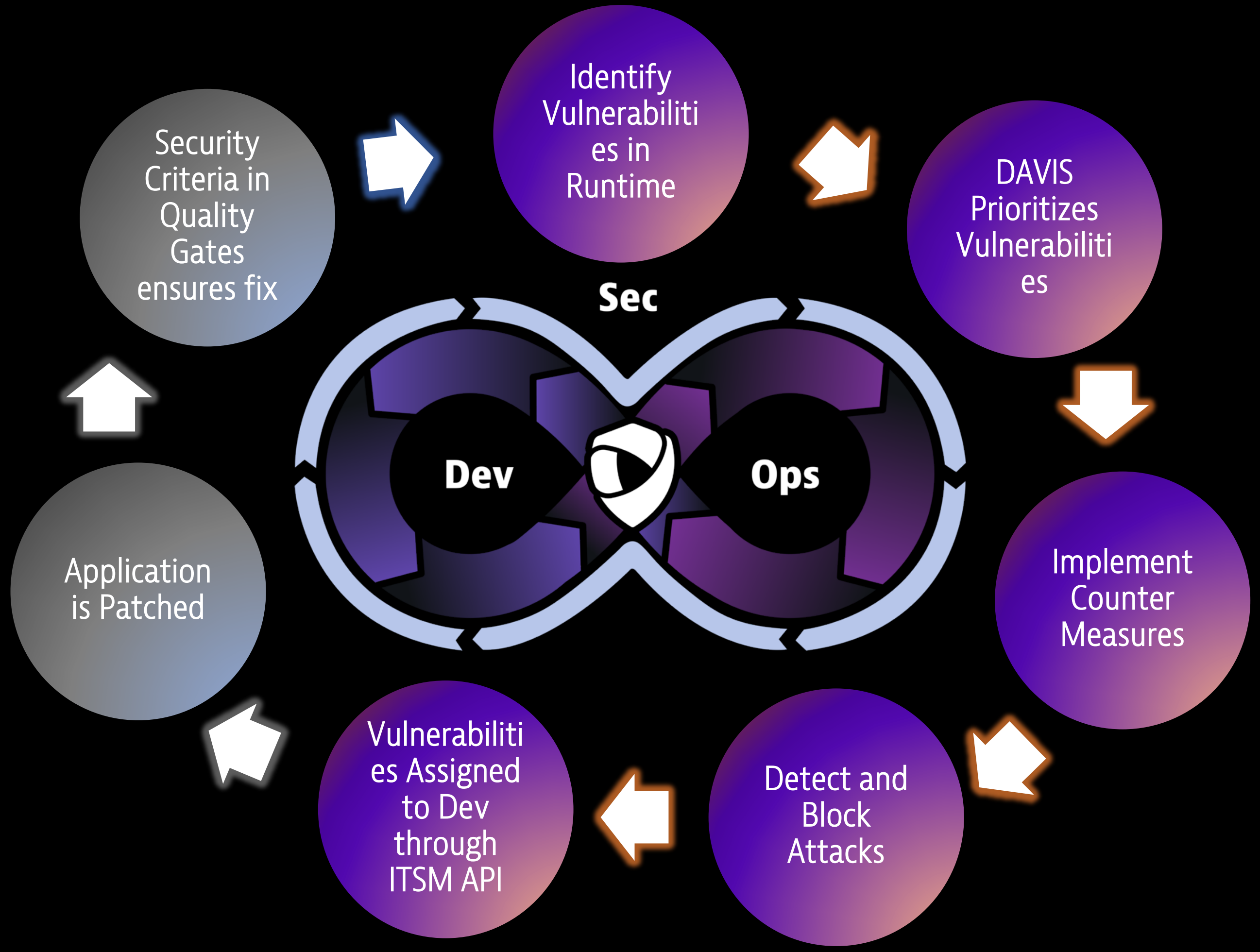
DYNATRACE APPLICATION SECURITY APPROACH IS… DIFFERENT

JUST SWITCH IT ON


code-level vulnerabilities
Attack blocking and protection
Vulnerability runtime analytics
Risk-based remediation
Infrastructure Monitoring
Applications & Microservices
Application Security
Digital Experience
Business Analytics
Cloud Automation
Dynatrace Hub
dynatrace Software Intelligence Platform
OneAgent®
PurePath®
Smartscape®
Grail™
Davis® AI
Automatic and intelligent observability
600+ Supported technologies
Broadest multicloud and technology support

- **Real-time, 24x7 vulnerability detection**
- **Automatic risk assessment using DAVIS AI**
- **Automated developer workflow to mitigate and fix affected apps**

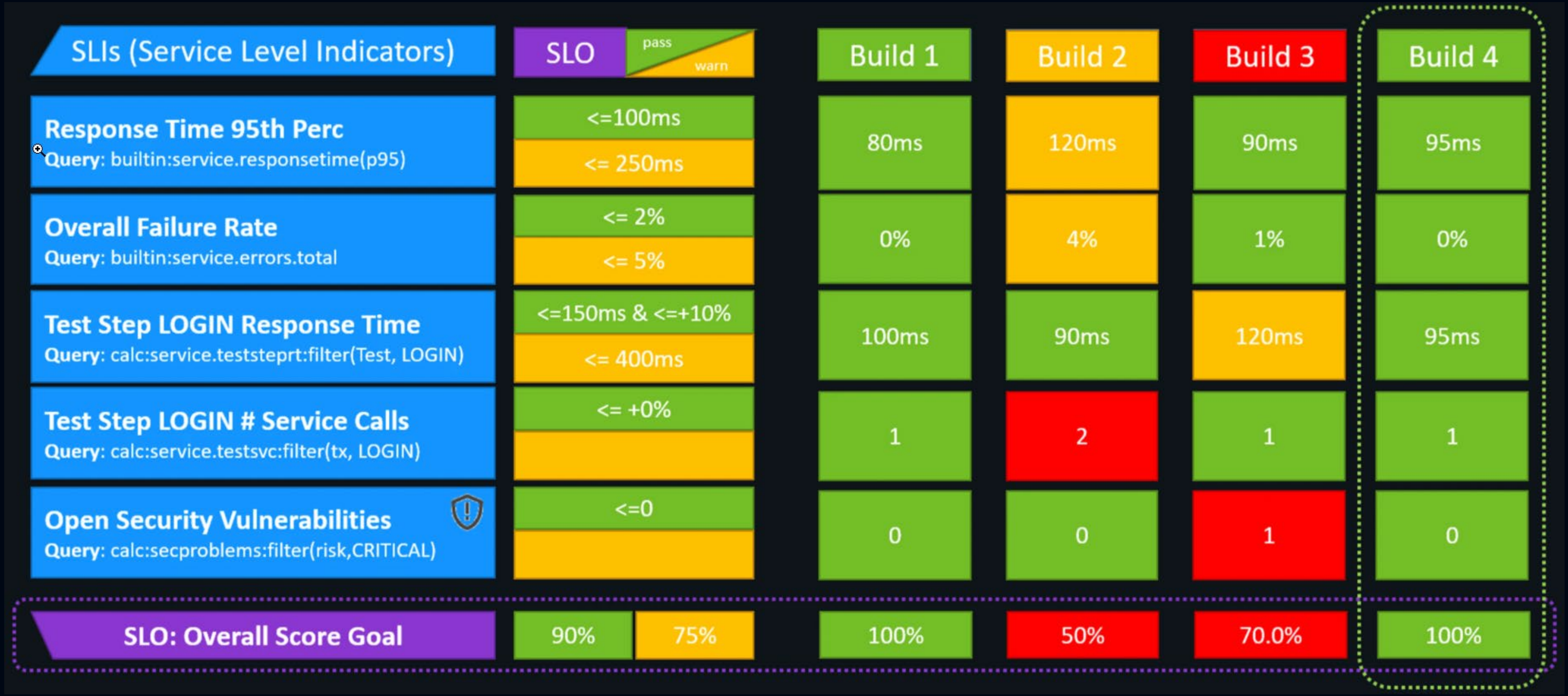
Dynatrace **integrates** security and observability


Identify Vulnerabilities in Runtime
DAVIS Prioritizes Vulnerabilities
Implement Counter Measures
Detect and Block Attacks
Vulnerabilities Assigned to Dev through ITSM API
Application is Patched
Security Criteria in Quality Gates ensures fix
Sec
Dev
Ops

APPSEC + DYNATRACE PLATFORM = **DEVSECOPS ACROSS THE LIFECYCLE**

DYNATRACE APPSEC + WORFLOWS + SITE RELIABILITY GUARDIAN

Integration of Application Security module with DevSecOps to Automate SLOs validation into your Delivery Pipeline with Dynatrace



SLIs (Service Level Indicators)	SLO		Build 1	Build 2	Build 3	Build 4
	pass	warn				
Response Time 95th Perc **Query:** builtin:service.responsetime(p95)	<=100ms	<= 250ms	80ms	120ms	90ms	95ms
Overall Failure Rate **Query:** builtin:service.errors.total	<= 2%	<= 5%	0%	4%	1%	0%
Test Step LOGIN Response Time **Query:** calc:service.teststeprt:filter(Test, LOGIN)	<=150ms & <=+10%	<= 400ms	100ms	90ms	120ms	95ms
Test Step LOGIN # Service Calls **Query:** calc:service.testsvc:filter(tx, LOGIN)	<= +0%		1	2	1	1
Open Security Vulnerabilities **Query:** calc:secproblems:filter(risk,CRITICAL)	<=0		0	0	1	0
SLO: Overall Score Goal	90%	75%	100%	50%	70.0%	100%

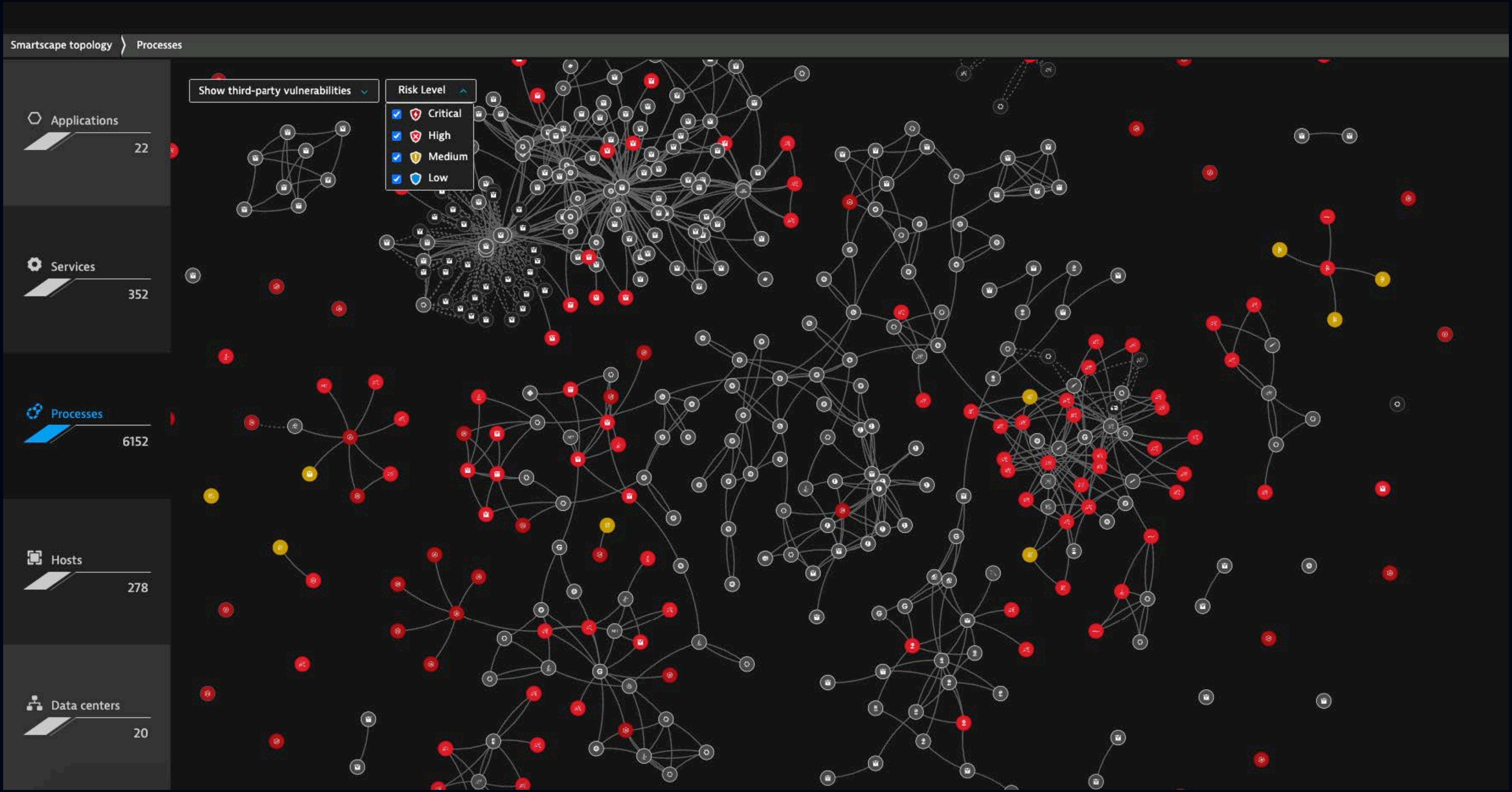
EFFORTLESSLY SIFT THROUGH THE NOISE WITH RUNTIME VULNERABILITY ANALYSIS (RVA)

Observability
Automation
Intelligence
Security

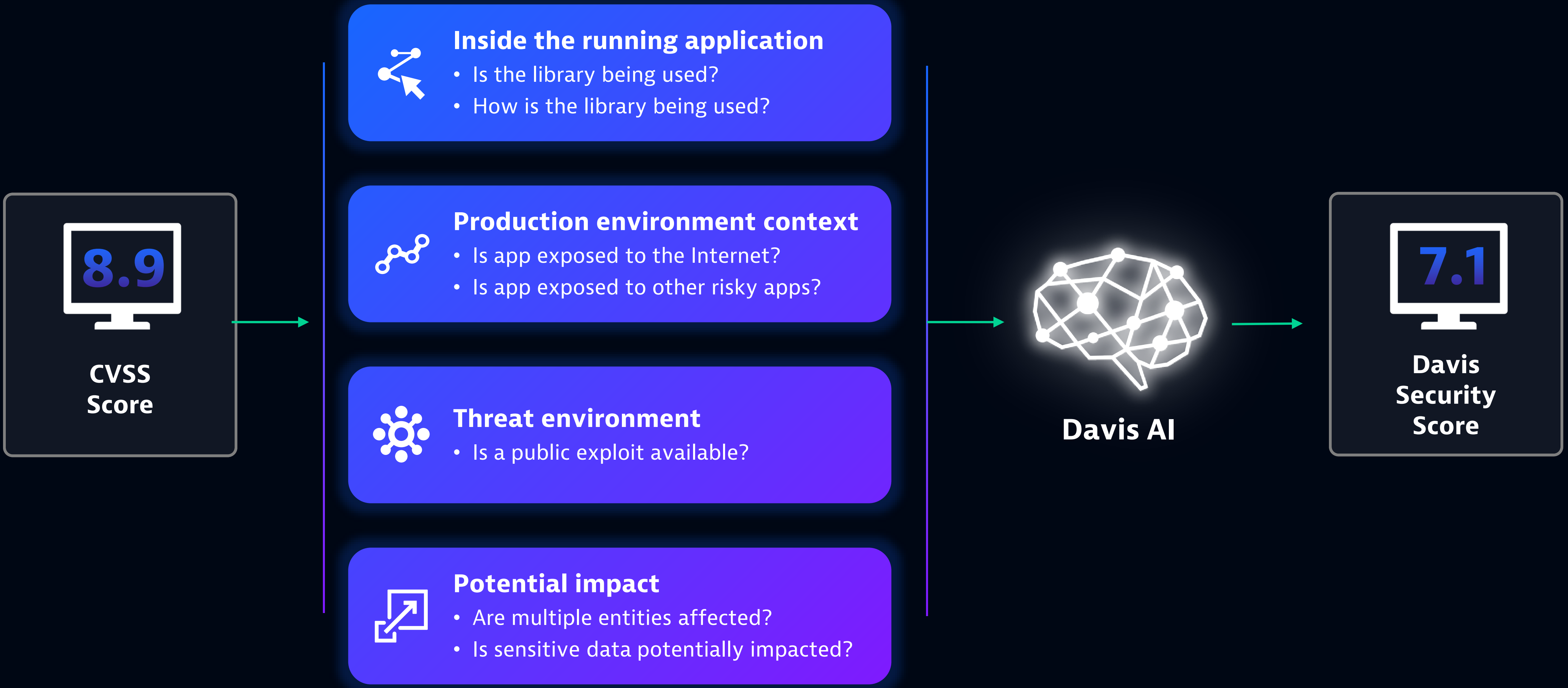
Find and fix vulnerabilities that leaked into runtime

- Continuously identify vulnerabilities in real-time
- Analyze runtime context using Smartscape topology
- Prioritize vulnerabilities with AI and security intelligence
- Precisely implement remediation and countermeasures

Analyze runtime context using Smartscape topology



HOW DAVIS SECURITY SCORE WORKS



Inside the running application

- Is the library being used?
- How is the library being used?

Production environment context

- Is app exposed to the Internet?
- Is app exposed to other risky apps?

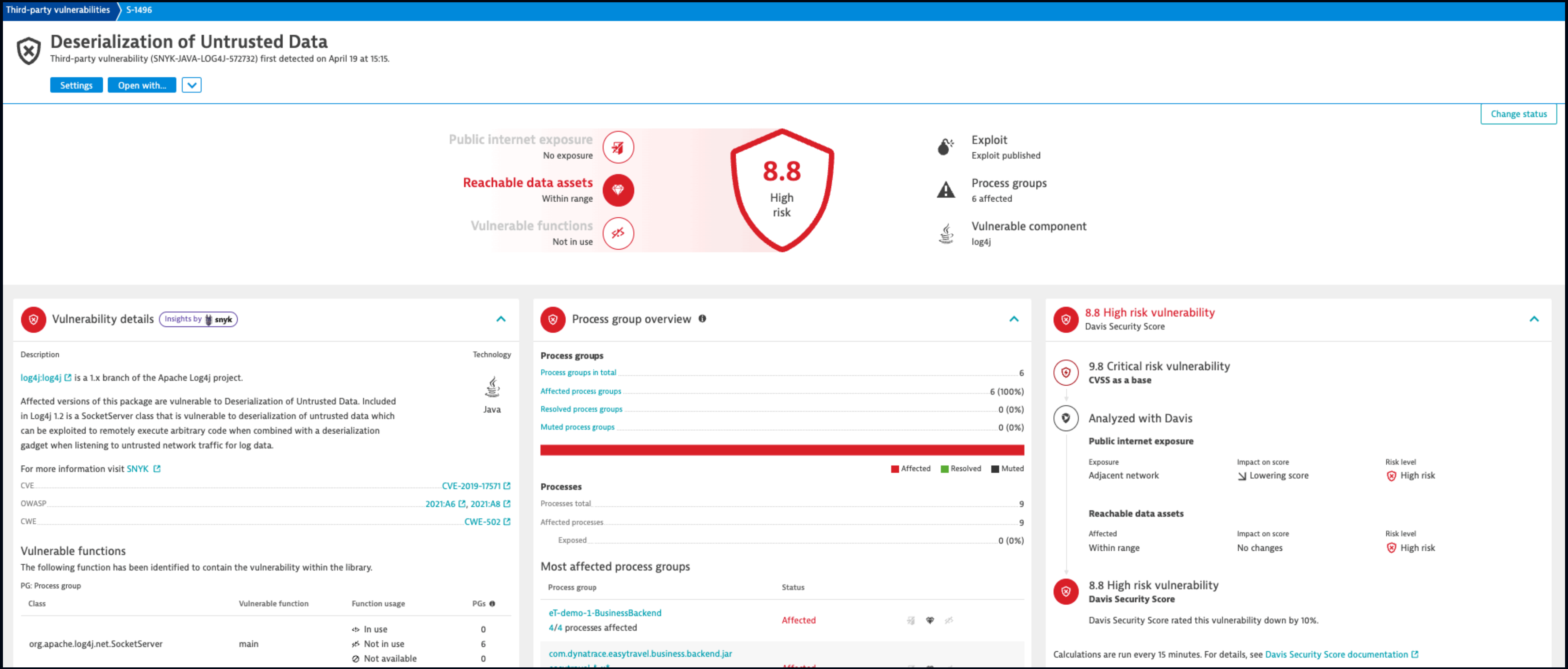
Threat environment

- Is a public exploit available?

Potential impact

- Are multiple entities affected?
- Is sensitive data potentially impacted?

Automatic vulnerabilities detection and assessment with **real-time risk calculation**



Third-party vulnerabilities › S-1496

Deserialization of Untrusted Data

Third-party vulnerability (SNYK-JAVA-LOG4J-572732) first detected on April 19 at 15:15.

Settings | Open with... | Change status

Public internet exposure — No exposure
Reachable data assets — Within range
Vulnerable functions — Not in use

8.8 High risk

Exploit — Exploit published
Process groups — 6 affected
Vulnerable component — log4j

Vulnerability details (Insights by snyk)

Description

log4j:log4j is a 1.x branch of the Apache Log4j project.

Affected versions of this package are vulnerable to Deserialization of Untrusted Data. Included in Log4j 1.2 is a SocketServer class that is vulnerable to deserialization of untrusted data which can be exploited to remotely execute arbitrary code when combined with a deserialization gadget when listening to untrusted network traffic for log data.

Technology: Java

For more information visit SNYK

CVE	CVE-2019-17571
OWASP	2021:A6, 2021:A8
CWE	CWE-502

Vulnerable functions

The following function has been identified to contain the vulnerability within the library.

PG: Process group

Class	Vulnerable function	Function usage	PGs
org.apache.log4j.net.SocketServer	main	In use	0
		Not in use	6
		Not available	0

Process group overview

Process groups

Process groups in total	6
Affected process groups	6 (100%)
Resolved process groups	0 (0%)
Muted process groups	0 (0%)

🟥 Affected 🟩 Resolved ■ Muted

Processes

Processes total	9
Affected processes	9
Exposed	0 (0%)

Most affected process groups

Process group	Status
eT-demo-1-BusinessBackend 4/4 processes affected	Affected
com.dynatrace.easytravel.business.backend.jar	

8.8 High risk vulnerability

Davis Security Score

9.8 Critical risk vulnerability
CVSS as a base

Analyzed with Davis

Public internet exposure

Exposure	Impact on score	Risk level
Adjacent network	Lowering score	High risk

Reachable data assets

Affected	Impact on score	Risk level
Within range	No changes	High risk

8.8 High risk vulnerability
Davis Security Score

Davis Security Score rated this vulnerability down by 10%.

Calculations are run every 15 minutes. For details, see Davis Security Score documentation

DYNATRACE IDENTIFIED LOG4SHELL IN PRODUCTION APPS **MINUTES** AFTER IT BECAME KNOWN

THIS IS THE VALUE OF CONTINUOUS MONITORING

NATIONAL VULNERABILITY DATABASE NVD

Vulnerability listed on GitHub — Dec 10 00:40am*

Vulnerability listed on NVD — Dec 10 10:15am

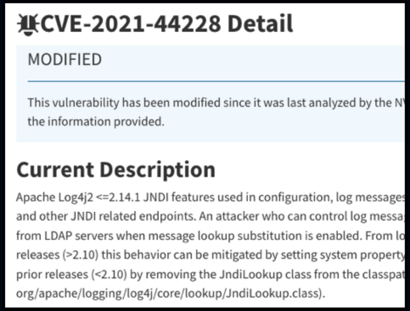
Vulnerability listed on Snyk — Dec 10 10:45am

Live feed

AppSec vulnerability catalog updated — Dec 10 10:50am

Assessment

First vulnerabilities detected and shown with Davis Security Score — Dec 10 11:05am



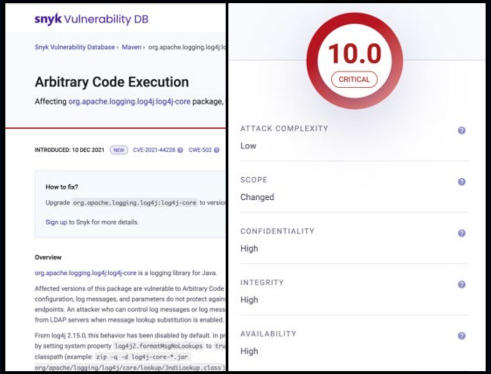
CVE-2021-44228 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the N
the information provided.

Current Description

Apache Log4j2 <=2.14.1 JNDI features used in configuration, log messages
and other JNDI related endpoints. An attacker who can control log messa
from LDAP servers when message lookup substitution is enabled. From lo
releases (>2.10) this behavior can be mitigated by setting system property
prior releases (<2.10) by removing the JndiLookup class from the classpat
org/apache/logging/log4j/core/lookup/JndiLookup.class).



snyk Vulnerability DB

Snyk Vulnerability Database › Maven › org.apache.logging.log4j:lo

Arbitrary Code Execution

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

INTRODUCED: 10 DEC 2021 NEW CVE-2021-44228 CWE-502

How to fix?

Upgrade `org.apache.logging.log4j:log4j-core` to versio

Sign up to Snyk for more details.

Overview

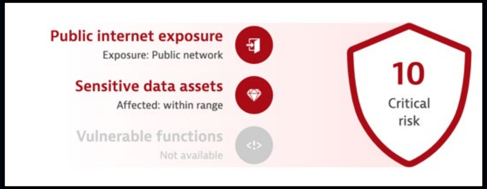
org.apache.logging.log4j:log4j-core is a logging library for Java.

Affected versions of this package are vulnerable to Arbitrary Code
configuration, log messages, and parameters do not protect again
endpoints. An attacker who can control log messages or log mess
from LDAP servers when message lookup substitution is enabled.

From log4j 2.15.0, this behavior has been disabled by default. In pr
by setting system property `log4j2.formatMsgNoLookups` to `tru`
classpath (example: `zip -q -d log4j-core-*.jar`
`org/apache/logging/log4j/core/lookup/JndiLookup.class`)

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

IDENTIFY VULNERABILITIES IN YOUR CODE AND PROTECT YOUR APP

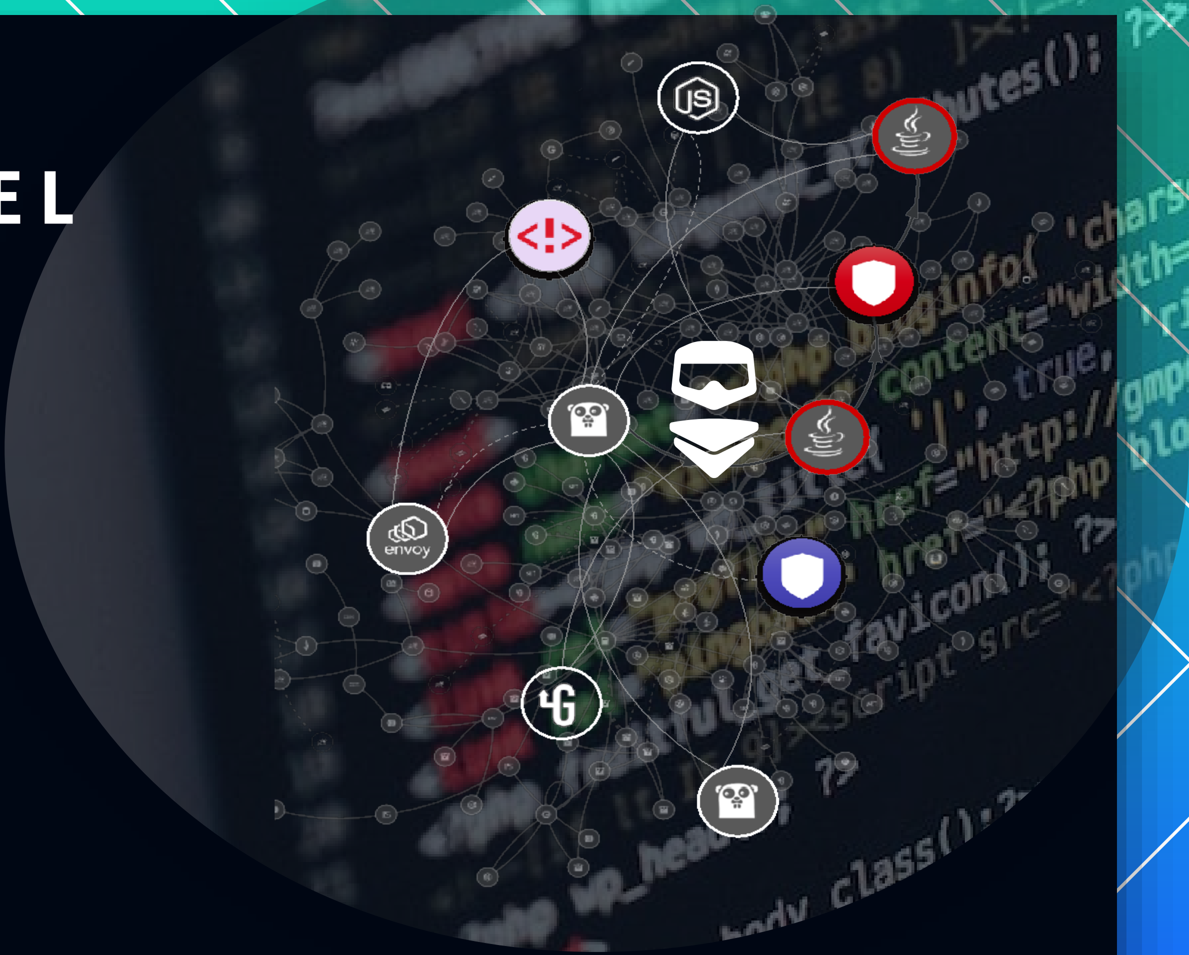
Code-level vulnerabilities (CLV) &

Runtime Application Protection (RAP)

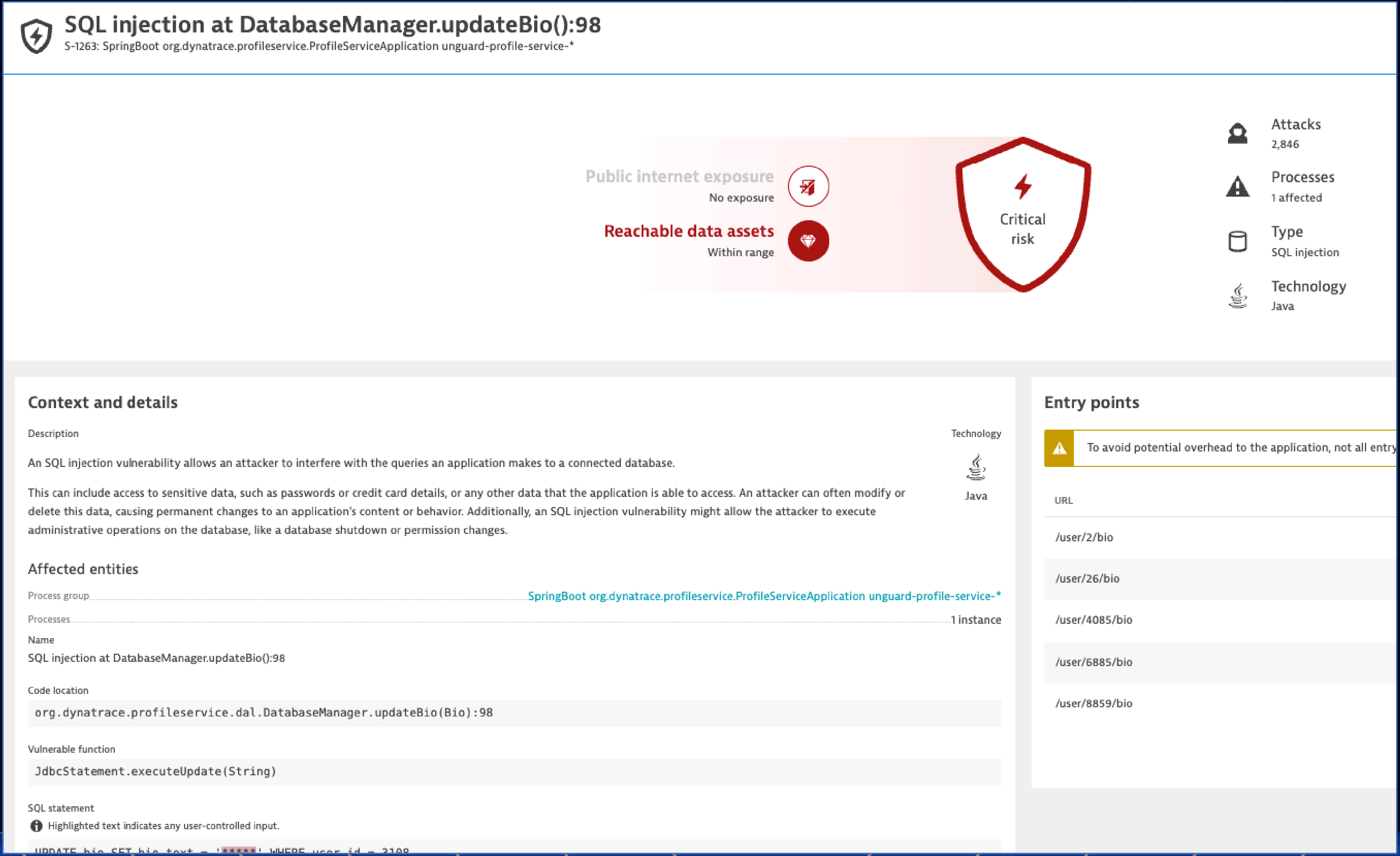
GO BEYOND VULNERABILITY DETECTION WITH **CODE-LEVEL VULNERABILITIES (CLV)**

Reduce risk from missed and zero-day vulnerabilities

- Detect common injection attacks
- No need to "attack" the application, just use legit traffic
- Leverage OneAgent, turn on with flip of switch
- No impact to user experience or operational costs



CODE-LEVEL VULNERABILITIES (CLV)



SQL injection at DatabaseManager.updateBio():98

S-1263: SpringBoot org.dynatrace.profileservice.ProfileServiceApplication unguard-profile-service-*

Public internet exposure
No exposure

Reachable data assets
Within range

Critical risk

Attacks	2,846
Processes	1 affected
Type	SQL injection
Technology	Java

Context and details

Description

An SQL injection vulnerability allows an attacker to interfere with the queries an application makes to a connected database.

This can include access to sensitive data, such as passwords or credit card details, or any other data that the application is able to access. An attacker can often modify or delete this data, causing permanent changes to an application's content or behavior. Additionally, an SQL injection vulnerability might allow the attacker to execute administrative operations on the database, like a database shutdown or permission changes.

Technology
Java

Affected entities

Process group	SpringBoot org.dynatrace.profileservice.ProfileServiceApplication unguard-profile-service-*
Processes	1 instance

Name
SQL injection at DatabaseManager.updateBio():98

Code location

```
org.dynatrace.profileservice.dal.DatabaseManager.updateBio(Bio):98
```

Vulnerable function

```
JdbcStatement.executeUpdate(String)
```

SQL statement

Highlighted text indicates any user-controlled input.

```
UPDATE bio SET bio_text = '*****' WHERE user_id = 3108
```

Entry points

To avoid potential overhead to the application, not all entry

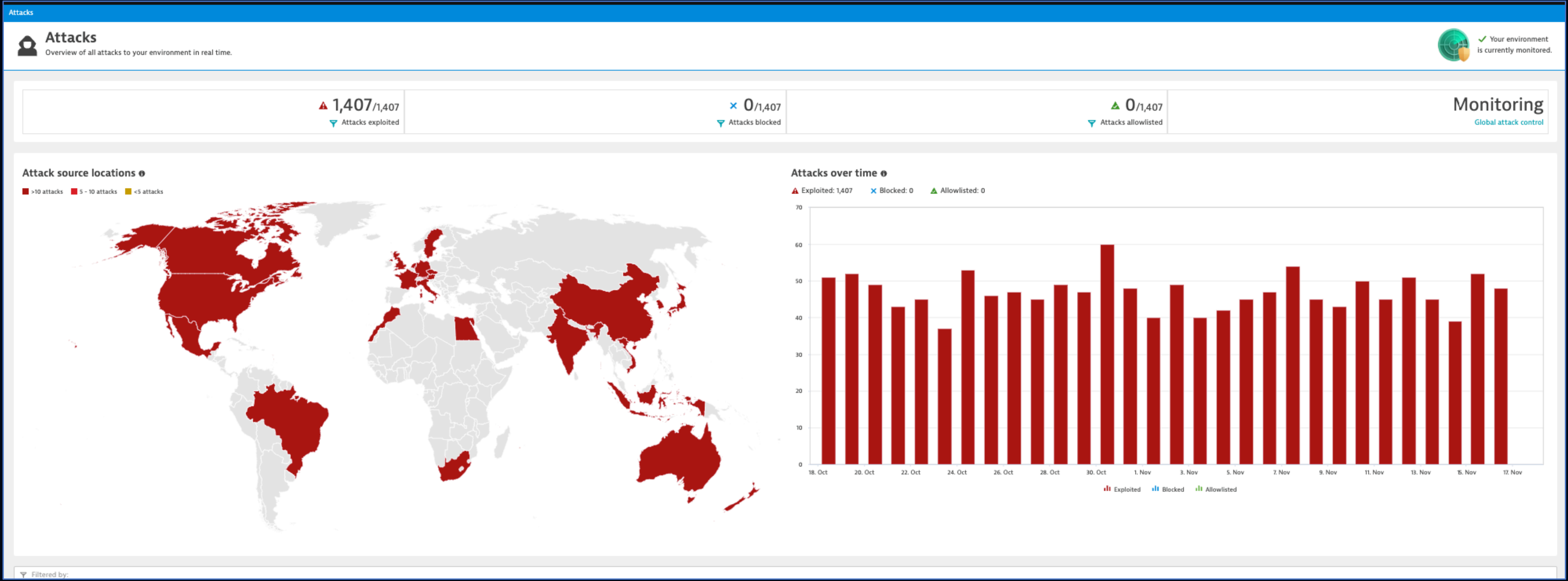
URL
/user/2/bio
/user/26/bio
/user/4085/bio
/user/6885/bio
/user/8859/bio

GO BEYOND VULNERABILITY DETECTION WITH **RUNTIME APPLICATION PROTECTION (RAP)**

Reduce risk from missed and zero-day vulnerabilities

- Detect & block common injection attacks
- No alert storms with high precision, low false positive rates
- Leverage OneAgent, turn on with flip of switch
- No impact to user experience or operational costs

RUNTIME APPLICATION PROTECTION (RAP)



Attacks

Attacks

Overview of all attacks to your environment in real time.

✓ Your environment is currently monitored.

▲ 1,407/1,407	× 0/1,407	▲ 0/1,407	Monitoring
Attacks exploited	Attacks blocked	Attacks allowlisted	Global attack control

Attack source locations

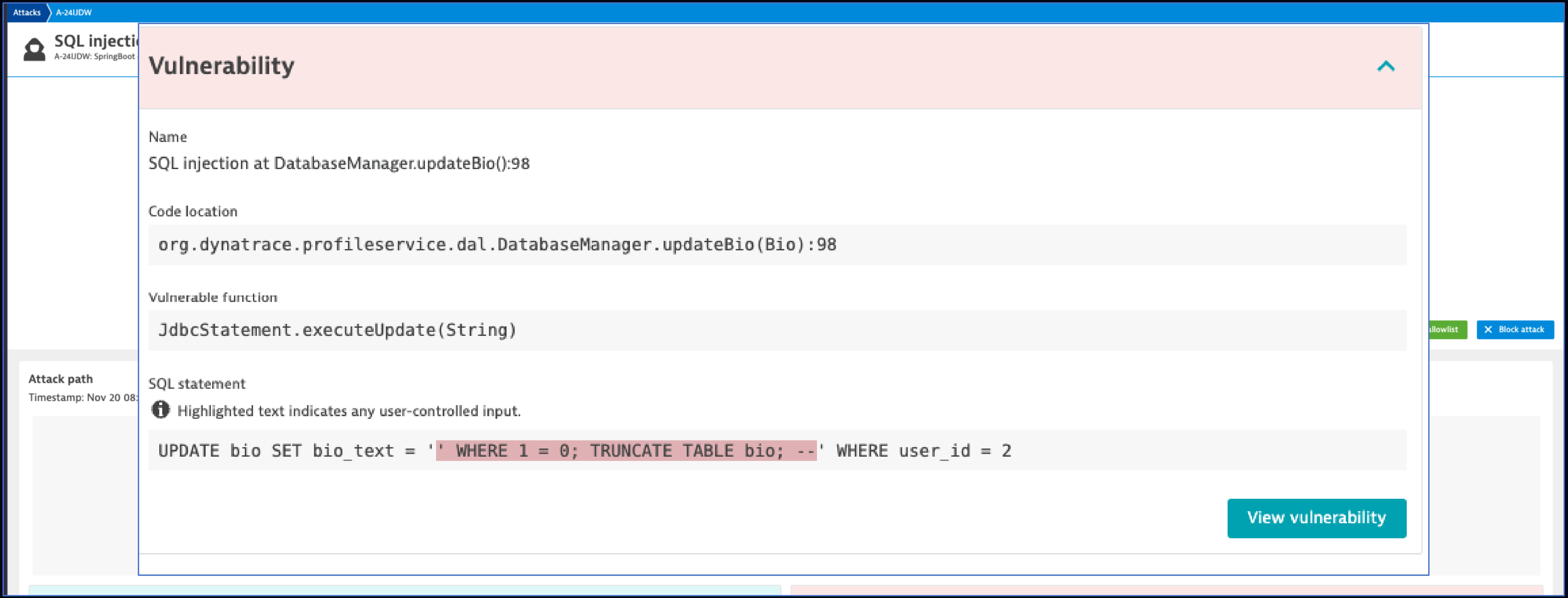
🟥 >10 attacks 🟥 5 - 10 attacks 🟨 <5 attacks

Attacks over time

▲ Exploited: 1,407 × Blocked: 0 ▲ Allowlisted: 0

Filtered by:

RUNTIME APPLICATION PROTECTION (RAP)



Vulnerability

Name

SQL injection at DatabaseManager.updateBio():98

Code location

```
org.dynatrace.profileservice.dal.DatabaseManager.updateBio(Bio):98
```

Vulnerable function

```
JdbcStatement.executeUpdate(String)
```

SQL statement

Highlighted text indicates any user-controlled input.

```
UPDATE bio SET bio_text = '' WHERE 1 = 0; TRUNCATE TABLE bio; --' WHERE user_id = 2
```

View vulnerability

dynatrace

IMMERSE

MADRID

21.11.23