Continuous Delivery

DevSecOps

Agenda

- Continuous Delivery
- CVEs
- Software Composition Analysis (SCA)
- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)

Continuous Integration

Secure DevOps

What is Continuous Delivery

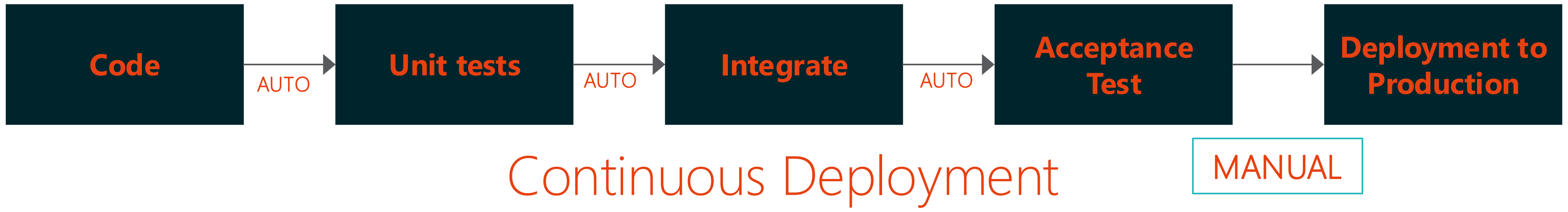
"Continuous delivery is a **software engineering approach** in which teams **produce software in short cycles**, helping to ensure that the software can be **released quickly, reliably, at any time**, following a **repeatable and sustainable process**.

It aims at building, testing, and releasing software with greater speed and frequency.

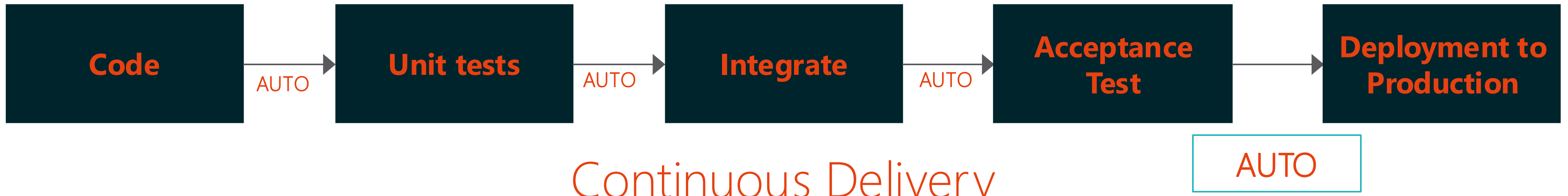
The approach helps **reduce the cost, time, and risk of delivering changes** by allowing for more incremental updates to applications in production.

A straightforward and repeatable deployment process is important for continuous delivery." (Martin Fowler)

Continuous Deployment vs Continuous Delivery



Continuous Deployment

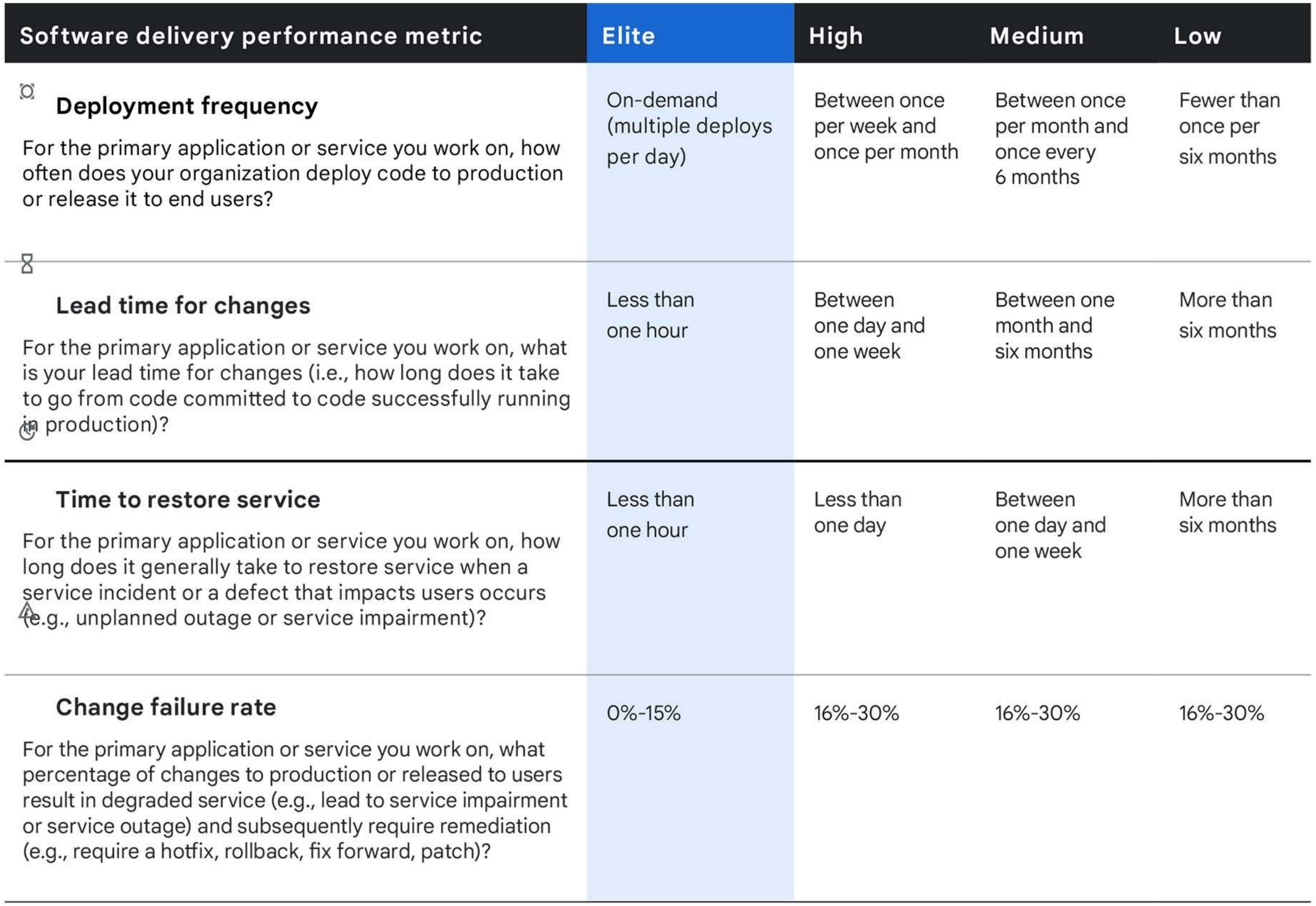


Continuous Delivery

Benefits

- Fast, repeatable, predictable configurable deployments
- Lower Risk and higher quality
- Early feedback
- Faster collaboration, everyone is involved, and anyone can initiate deployments
- Adapt and react a lot quickly
- Deploy during any business hours
- Change delivered without significant delay

DORA Metrics



Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Practices

- Automated Deployments
- One Build Only
- Move forward
- Buildable and Deployable Trunk
- Feature Flags
- Shift-Right with Modern Deployments

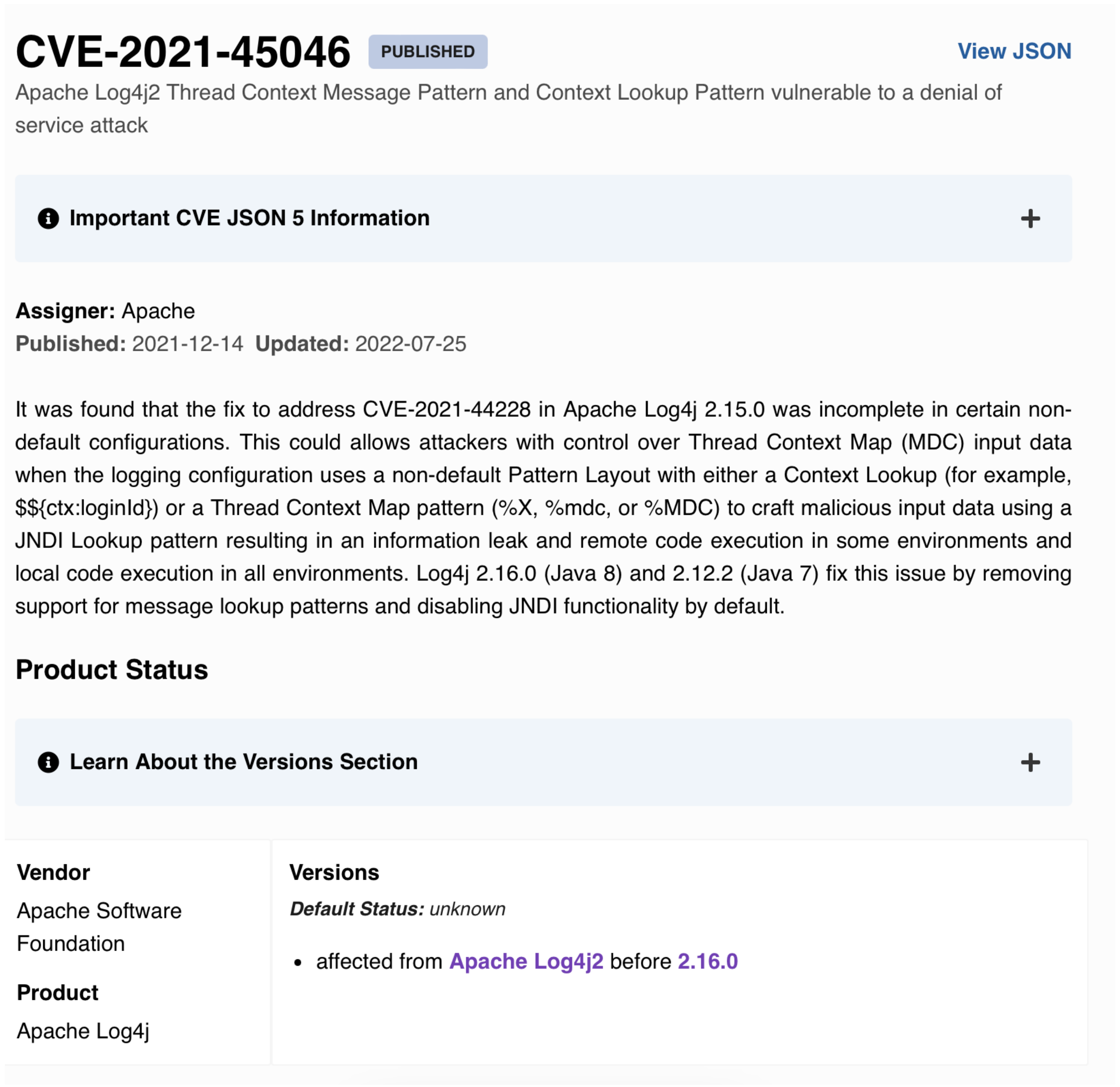
CVEs

Secure DevOps

Common Vulnerability and Exposures (CVE)

- CVE is a dictionary of publicly known information security vulnerabilities and exposures that provides a standardized naming scheme for these issues
- Each CVE entry contains a unique identifier, a description of the vulnerability, and references to advisories and patches
- On this database, each CVE found on open-source software are published with vulnerability description and how to fix
- Allow to clearly identify all known vulnerabilities on open-source code to allow you to have a more secure code

Common Vulnerability and Exposures (CVE)



CVE-2021-45046 PUBLISHED

View JSON

Apache Log4j2 Thread Context Message Pattern and Context Lookup Pattern vulnerable to a denial of service attack

Important CVE JSON 5 Information +

Assigner: Apache
Published: 2021-12-14 **Updated:** 2022-07-25

It was found that the fix to address CVE-2021-44228 in Apache Log4j 2.15.0 was incomplete in certain non-default configurations. This could allows attackers with control over Thread Context Map (MDC) input data when the logging configuration uses a non-default Pattern Layout with either a Context Lookup (for example, $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC) to craft malicious input data using a JNDI Lookup pattern resulting in an information leak and remote code execution in some environments and local code execution in all environments. Log4j 2.16.0 (Java 8) and 2.12.2 (Java 7) fix this issue by removing support for message lookup patterns and disabling JNDI functionality by default.

Product Status

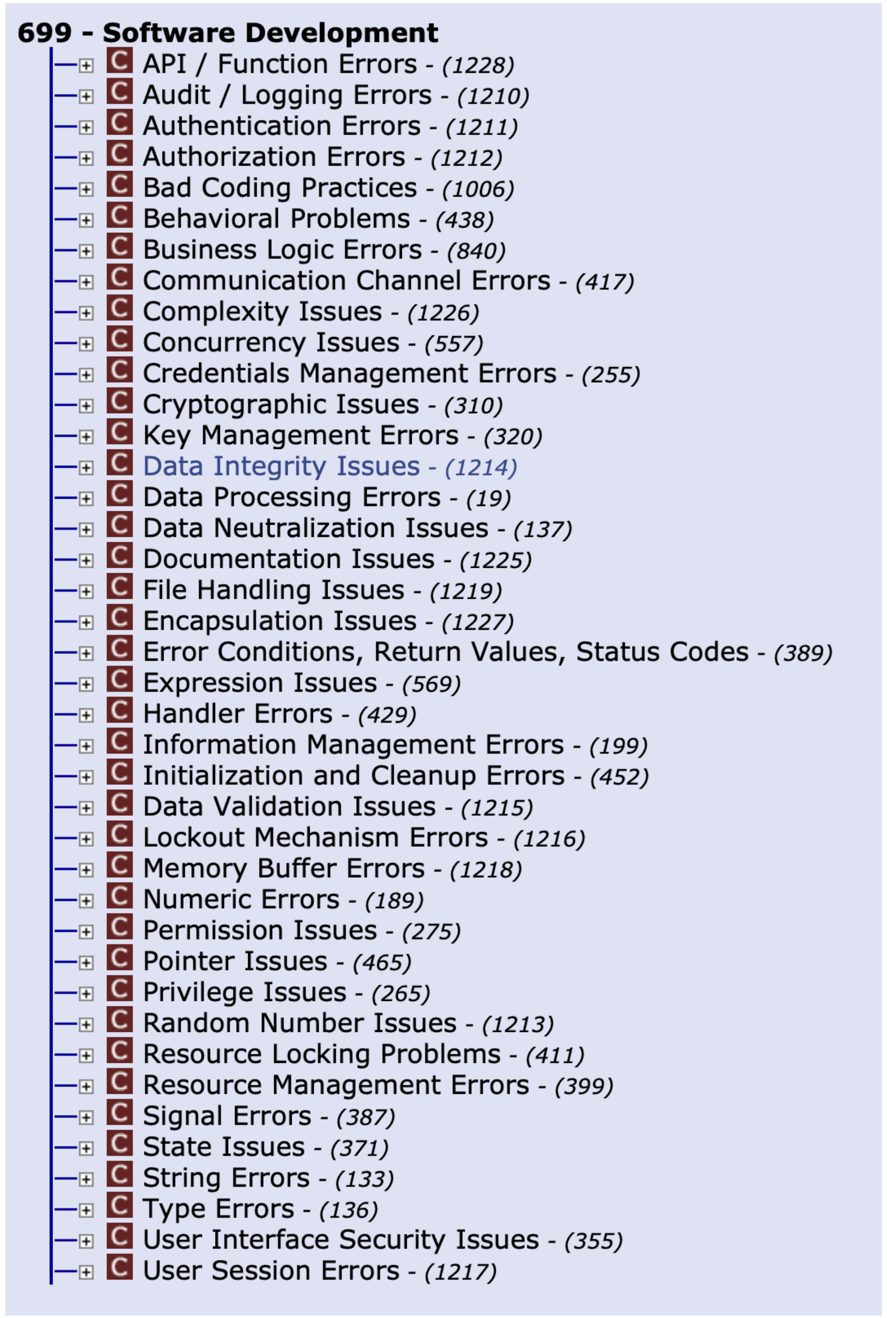
Learn About the Versions Section +

Vendor	Versions
Apache Software Foundation	***Default Status:** unknown*
Product	• affected from Apache Log4j2 before 2.16.0
Apache Log4j	

Common Weakness Enumeration (CWE)

- It is a community-developed list of common software security weaknesses.
- Each CWE entry provides a description of the weakness, examples of its occurrence in real-world software, and guidance on how to mitigate or eliminate it.
- A CVE is defined on top of CWE (Common Weakness Enumeration) definition, being an effective instance of 1+ CWE exploit

Common Weakness Enumeration (CWE)



699 - Software Development

- API / Function Errors - *(1228)*
- Audit / Logging Errors - *(1210)*
- Authentication Errors - *(1211)*
- Authorization Errors - *(1212)*
- Bad Coding Practices - *(1006)*
- Behavioral Problems - *(438)*
- Business Logic Errors - *(840)*
- Communication Channel Errors - *(417)*
- Complexity Issues - *(1226)*
- Concurrency Issues - *(557)*
- Credentials Management Errors - *(255)*
- Cryptographic Issues - *(310)*
- Key Management Errors - *(320)*
- Data Integrity Issues - *(1214)*
- Data Processing Errors - *(19)*
- Data Neutralization Issues - *(137)*
- Documentation Issues - *(1225)*
- File Handling Issues - *(1219)*
- Encapsulation Issues - *(1227)*
- Error Conditions, Return Values, Status Codes - *(389)*
- Expression Issues - *(569)*
- Handler Errors - *(429)*
- Information Management Errors - *(199)*
- Initialization and Cleanup Errors - *(452)*
- Data Validation Issues - *(1215)*
- Lockout Mechanism Errors - *(1216)*
- Memory Buffer Errors - *(1218)*
- Numeric Errors - *(189)*
- Permission Issues - *(275)*
- Pointer Issues - *(465)*
- Privilege Issues - *(265)*
- Random Number Issues - *(1213)*
- Resource Locking Problems - *(411)*
- Resource Management Errors - *(399)*
- Signal Errors - *(387)*
- State Issues - *(371)*
- String Errors - *(133)*
- Type Errors - *(136)*
- User Interface Security Issues - *(355)*
- User Session Errors - *(1217)*

Common Vulnerability Scoring System (CVSS)

- It is a framework for assessing the severity of vulnerabilities in software systems
- CVSS assigns a score to each vulnerability based on its impact on the system's confidentiality, integrity, and availability, as well as other factors such as complexity and exploitability
- For each CVE a CVSS score is calculated granting a potential risk you’re exposed

Common Vulnerability Scoring System (CVSS)

- Uses 3 metrics to make the calculations: Base Metrics, Temporal Metrics and Environmental Metrics

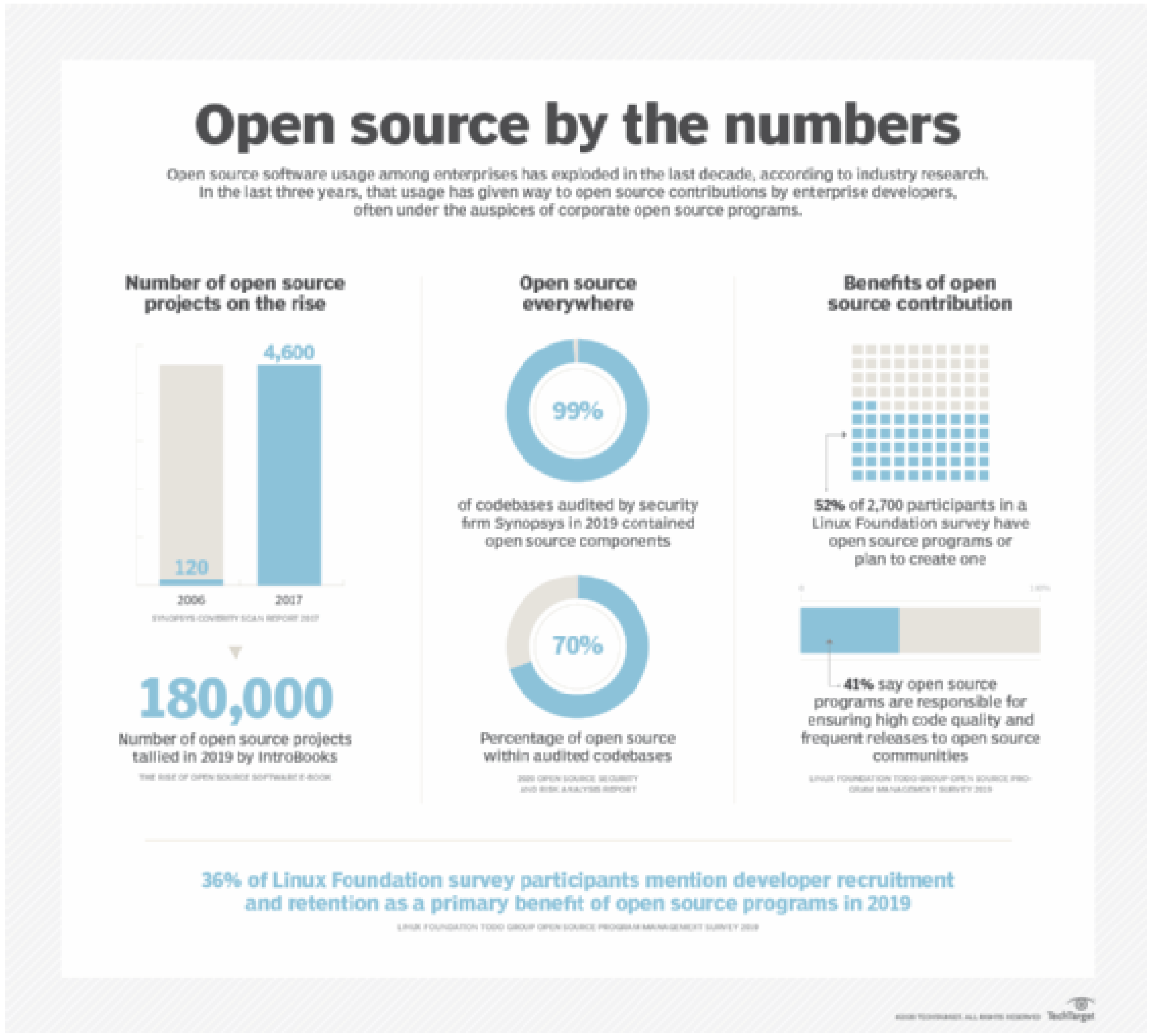
- This metrics produces a values between 0-10 to define severity

CVSS Score	Qualitative Rating
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

Software Composition Analysis (SCA)

Secure DevOps

Open Source



Open source by the numbers

Open source software usage among enterprises has exploded in the last decade, according to industry research. In the last three years, that usage has given way to open source contributions by enterprise developers, often under the auspices of corporate open source programs.

Number of open source projects on the rise

SYNOPSYS COVERITY SCAN REPORT 2017

180,000

Number of open source projects tallied in 2019 by IntroBooks

THE RISE OF OPEN SOURCE SOFTWARE E-BOOK

Open source everywhere

of codebases audited by security firm Synopsys in 2019 contained open source components

Percentage of open source within audited codebases

2020 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT

Benefits of open source contribution

52% of 2,700 participants in a Linux Foundation survey have open source programs or plan to create one

41% say open source programs are responsible for ensuring high code quality and frequent releases to open source communities

LINUX FOUNDATION TODO GROUP OPEN SOURCE PROGRAM MANAGEMENT SURVEY 2019

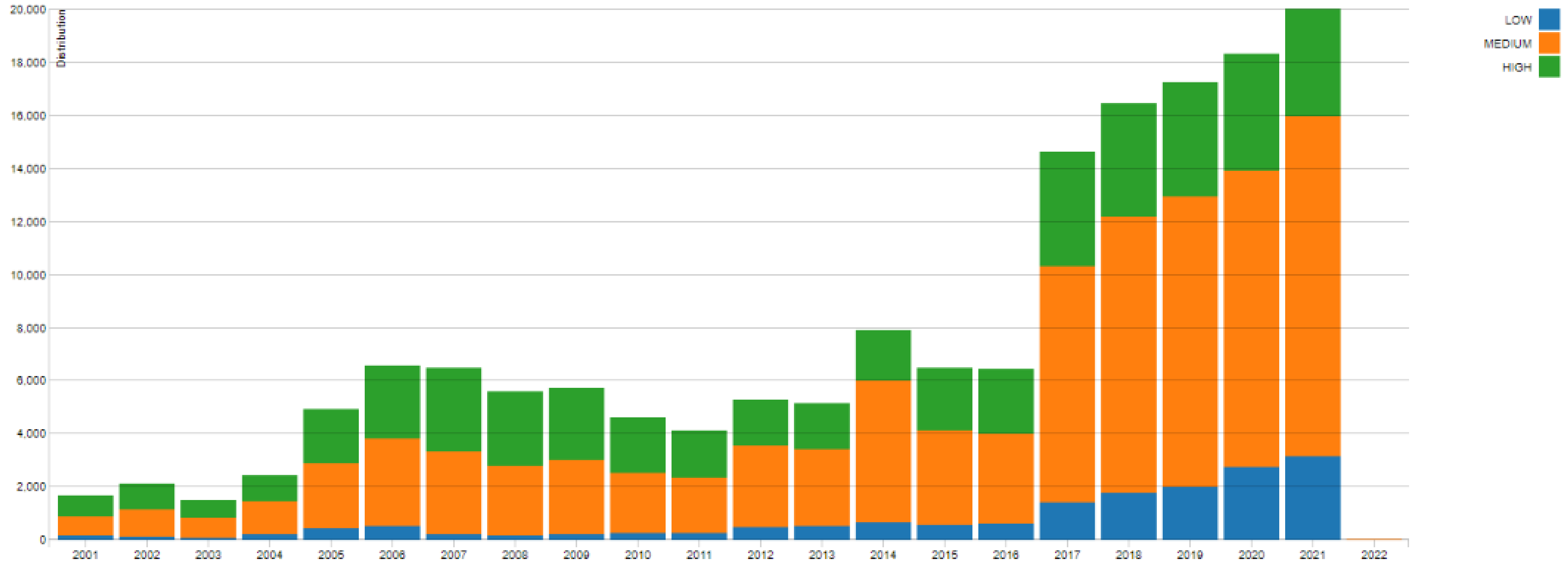
36% of Linux Foundation survey participants mention developer recruitment and retention as a primary benefit of open source programs in 2019

LINUX FOUNDATION TODO GROUP OPEN SOURCE PROGRAM MANAGEMENT SURVEY 2019

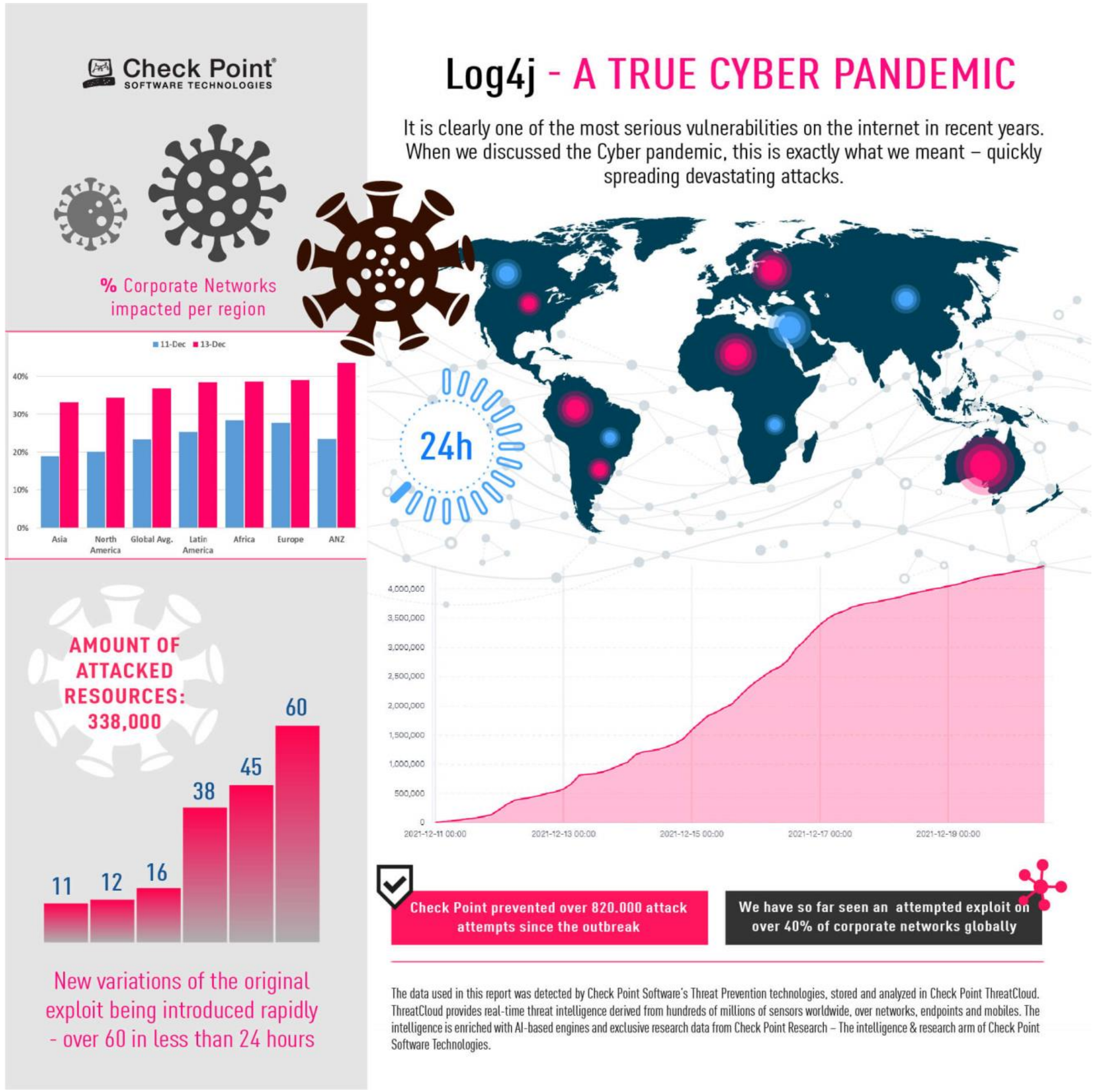
Identified Vulnerabilities

CVSS Severity Distribution Over Time

This visualization is a simple graph which shows the distribution of vulnerabilities by severity over time. The choice of LOW, MEDIUM and HIGH is based upon the CVSS V2 Base score. For more information on how this data was constructed please see the NVD CVSS page .



Log4J Vulnerability



It is clearly one of the most serious vulnerabilities on the internet in recent years. When we discussed the Cyber pandemic, this is exactly what we meant – quickly spreading devastating attacks.

New variations of the original exploit being introduced rapidly - over 60 in less than 24 hours

Check Point prevented over 820.000 attack attempts since the outbreak

We have so far seen an attempted exploit on over 40% of corporate networks globally

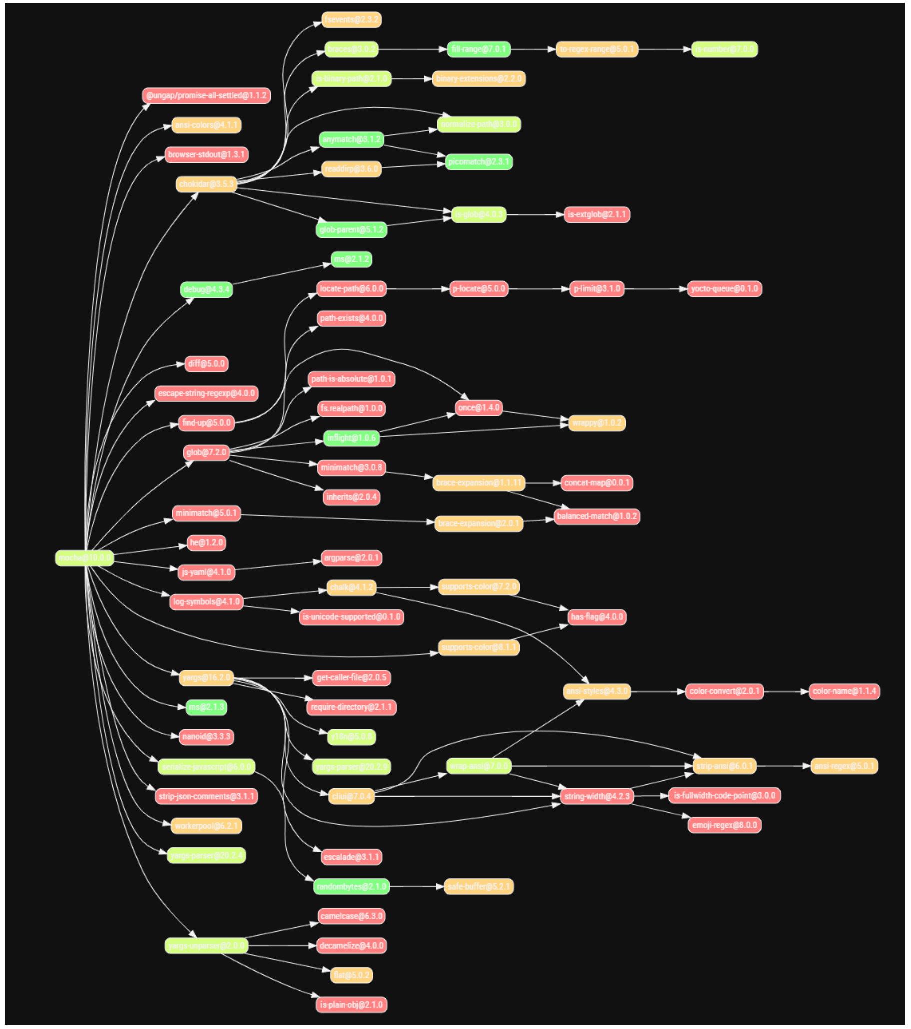
The data used in this report was detected by Check Point Software's Threat Prevention technologies, stored and analyzed in Check Point ThreatCloud. ThreatCloud provides real-time threat intelligence derived from hundreds of millions of sensors worldwide, over networks, endpoints and mobiles. The intelligence is enriched with AI-based engines and exclusive research data from Check Point Research – The intelligence & research arm of Check Point Software Technologies.

Software Composition Analysis (SCA)

- So, Open Source is a bad and dangerous thing? Of course not!
- But you need to use it careful and mostly you need to clearly know what are you using!
- Constantly run a scan on your dependencies is crucial to understand known vulnerabilities on your supply chain
- Knowing the vulnerabilities and their severity you may define you plan to fix them
- Know what you're using means knowing your dependency graph! Your direct dependency have its own dependencies. That dependencies have their own dependencies and so on

Dependency Graph

- You select only one package but look to your attack surface!
- Another risk is about how open source project is maintained
- On image, red means only one maintainer.
- Is a risk you may want to take, but you have to clearly know it!



SCA Tooling

Dependabot

MEND
Formerly WhiteSource

snyk

DEPENDENCY-TRACK

Demo: Dependabot + Snyk

Secure DevOps

Lab 03: Enable SCA

Secure DevOps

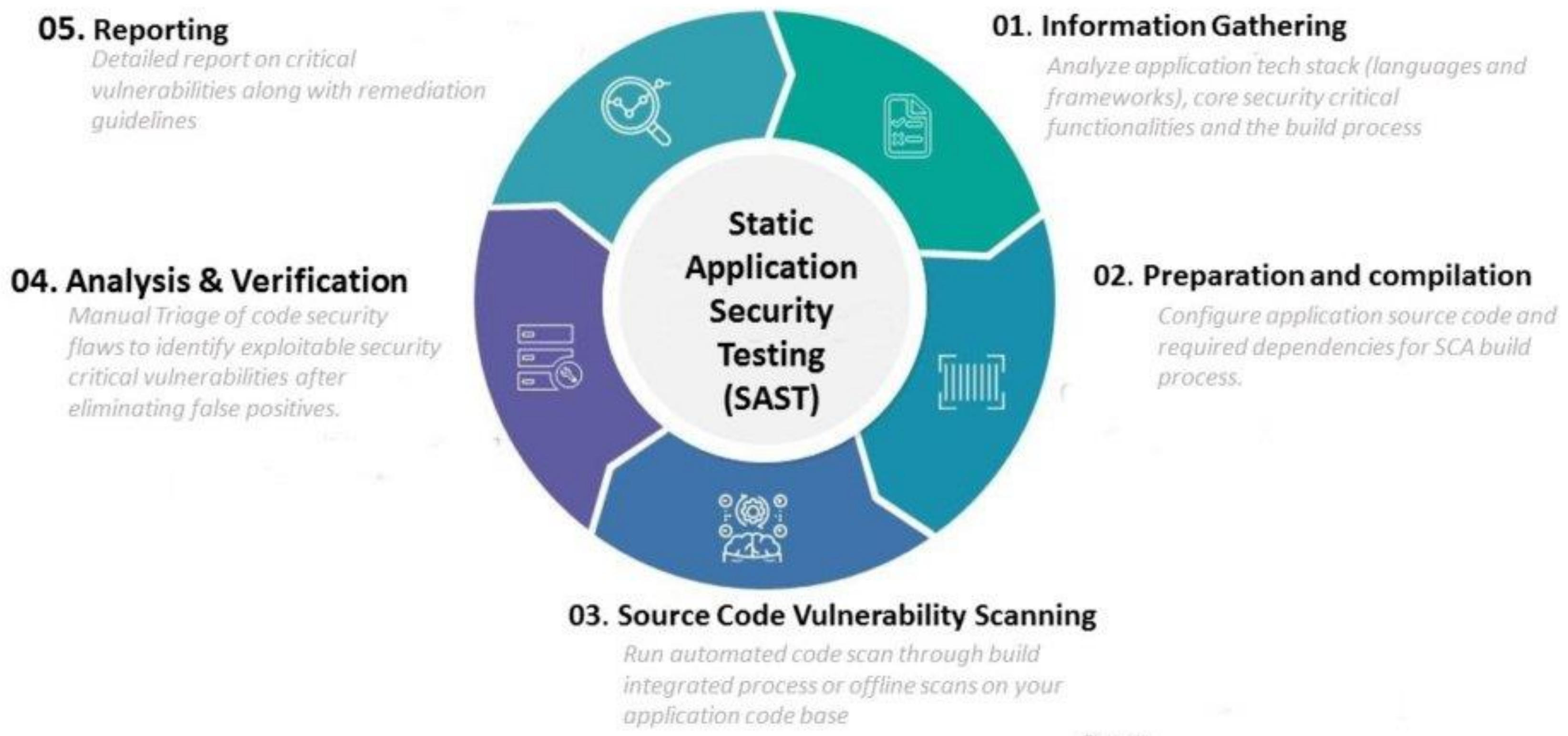
Static Application Security Testing (SAST)

Secure DevOps

Static Application Security Testing (SAST)

- Improve code security and quality on an easy and cost-effective way
- Makes an analysis on your source code and return insights about security, performance, maintainability
- Fully automated and can be shift-left for developers IDE
- Runs to answer by with the question “Is the code secure?”
  - Is it vulnerable to injections (like SQL)?
  - Does it use any weak encryption algorithms?
  - Are cookies used with the right flags?

Static Application Security Testing (SAST)


05. Reporting
Detailed report on critical vulnerabilities along with remediation guidelines
01. Information Gathering
Analyze application tech stack (languages and frameworks), core security critical functionalities and the build process
Static Application Security Testing (SAST)
04. Analysis & Verification
Manual Triage of code security flaws to identify exploitable security critical vulnerabilities after eliminating false positives.
02. Preparation and compilation
Configure application source code and required dependencies for SCA build process.
03. Source Code Vulnerability Scanning
Run automated code scan through build integrated process or offline scans on your application code base

SAST: Tooling

Semmle/CodeQL

sonarcloud

snyk

klocwork

Spectral

Checkmarx

https://owasp.org/www-community/Source_Code_Analysis_Tools

Demo: CodeQL + SonarCloud

Secure DevOps

Lab 04: Enable SAST

Secure DevOps

Dynamic Application Security Testing (DAST)

Secure DevOps

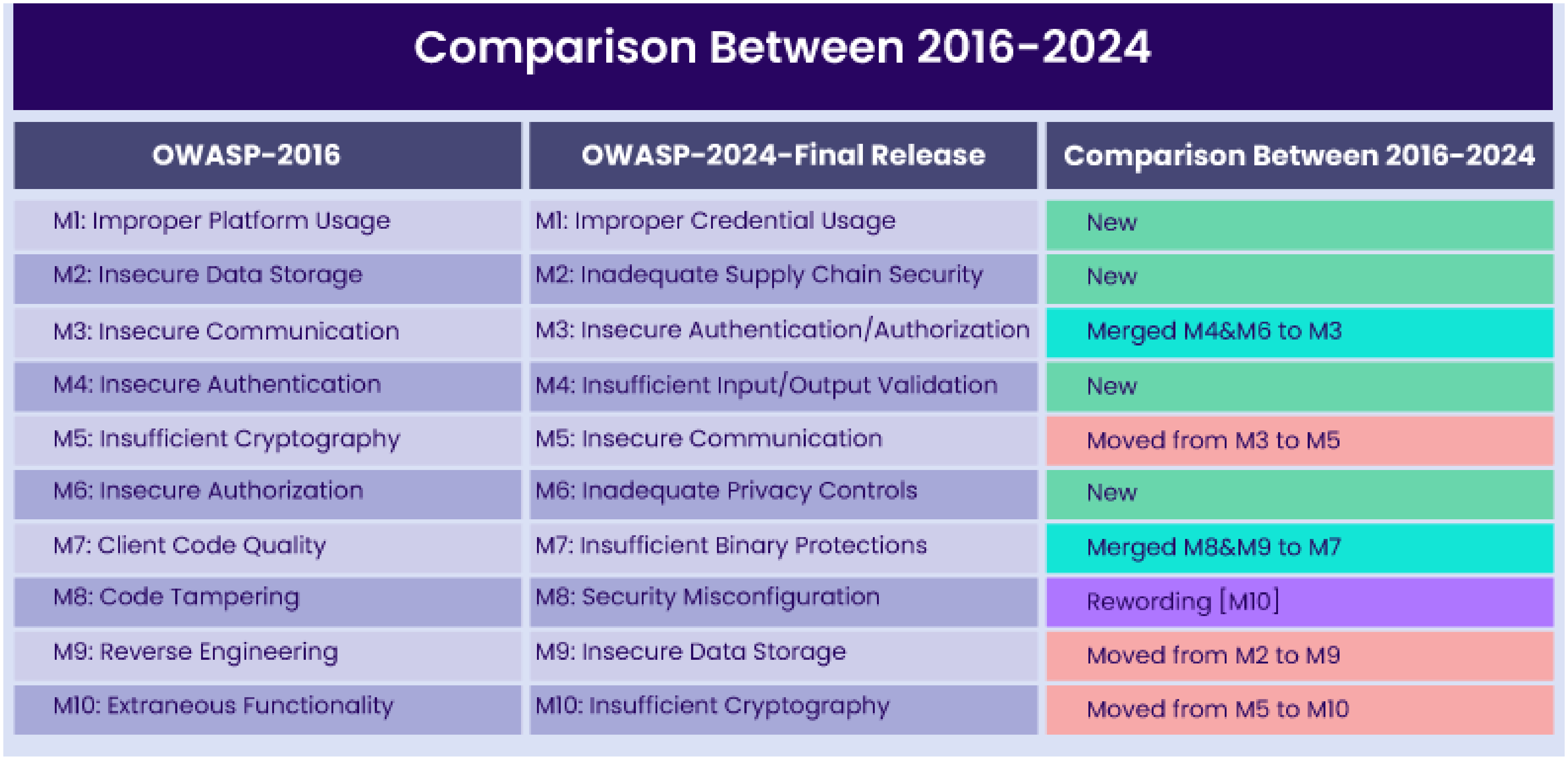
OWASP (Open Web Application Security Project)

- OWASP is a worldwide nonprofit organization focused on improving the security of software (mainly web applications)
- Provides resources and guidance to developers, security professionals, and organizations.
- Organizes conferences, training sessions, and other events around the world, as well as maintains a community forum for developers and security professionals to share knowledge and best practices related to web application security
- Maintains various open-source projects and tools that help identify and mitigate security vulnerabilities in web applications, including the OWASP Zed Attack Proxy (ZAP) and the OWASP Web Security Testing Guide.

OWASP TOP 10

- One of the most well-known contributions of OWASP is the OWASP Top Ten Project
- Provides a regularly-updated list of the top 10 web application security risks based on community feedback and research.
- The latest version is the OWASP Top 10 2021, which includes risks such as injection flaws, broken authentication and session management, and security misconfigurations.
- This Top 10 is updated regularly but is interesting to observe that the main vulnerabilities don’t change a lot since 2013

OWASP TOP 10 2016 vs 2024



Comparison Between 2016-2024

OWASP-2016	OWASP-2024-Final Release	Comparison Between 2016-2024
M1: Improper Platform Usage	M1: Improper Credential Usage	New
M2: Insecure Data Storage	M2: Inadequate Supply Chain Security	New
M3: Insecure Communication	M3: Insecure Authentication/Authorization	Merged M4&M6 to M3
M4: Insecure Authentication	M4: Insufficient Input/Output Validation	New
M5: Insufficient Cryptography	M5: Insecure Communication	Moved from M3 to M5
M6: Insecure Authorization	M6: Inadequate Privacy Controls	New
M7: Client Code Quality	M7: Insufficient Binary Protections	Merged M8&M9 to M7
M8: Code Tampering	M8: Security Misconfiguration	Rewording [M10]
M9: Reverse Engineering	M9: Insecure Data Storage	Moved from M2 to M9
M10: Extraneous Functionality	M10: Insufficient Cryptography	Moved from M5 to M10

OWASP Top Ten Web Application Security Risks | OWASP

OWASP TOP 10

- If your application is not affected by OWASP Top 10 vulnerabilities, can you say your application is secure and solid?

OWASP TOP 10

- If your application is affected by any OWASP Top 10 vulnerabilities, can you say your application is insecure?

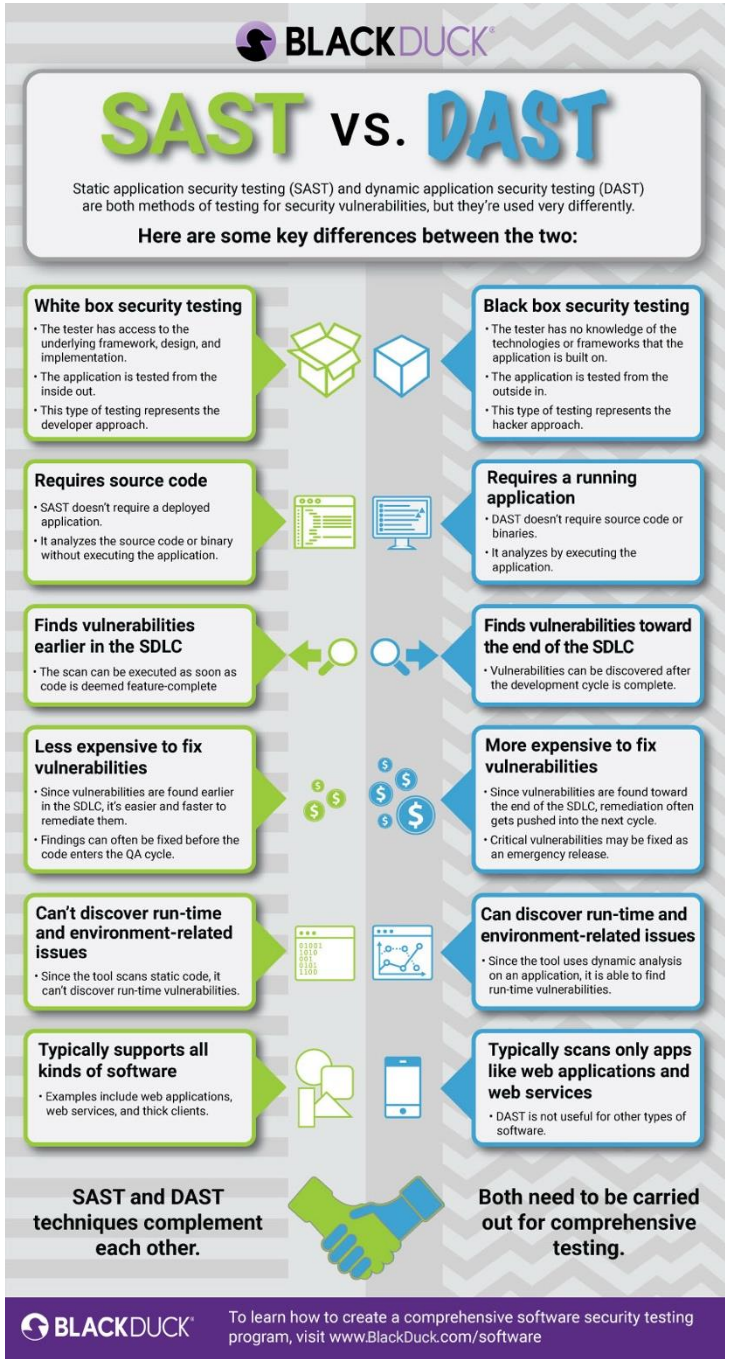
Dynamic Application Security Testing (DAST)

- Dynamic application security testing (DAST) is a method of AppSec testing in which testers examine an application while it’s running
- No knowledge of the application’s internal interactions or designs at the system level, and no access or visibility into the source program
- This “black box” testing looks at an application from the outside in, examines its running state, and observes its responses to simulated attacks made by a testing tool
- An application’s responses to these simulations help determine whether the application is vulnerable and could be susceptible to a real malicious attack.

DAST: What can be solved by?

- Detection of Runtime Vulnerabilities: Identifies security flaws in a running application, such as SQL injection, XSS, and authentication weaknesses, which might not be visible in static code reviews.
- Blind Spots in Source Code Analysis: Detects vulnerabilities in third-party components or dynamically generated content that static analysis might miss.
- Environment-Specific Issues: Captures vulnerabilities caused by misconfigurations, integration errors, or differences between development and production environments.
- Real-World Attack Simulation: Tests applications as an attacker would, identifying exploitable weaknesses in live environments.

SAST vs DAST



BLACKDUCK

SAST vs. DAST

Static application security testing (SAST) and dynamic application security testing (DAST) are both methods of testing for security vulnerabilities, but they're used very differently.

Here are some key differences between the two:

SAST	DAST
White box security testing • The tester has access to the underlying framework, design, and implementation. • The application is tested from the inside out. • This type of testing represents the developer approach.	**Black box security testing** • The tester has no knowledge of the technologies or frameworks that the application is built on. • The application is tested from the outside in. • This type of testing represents the hacker approach.
Requires source code • SAST doesn't require a deployed application. • It analyzes the source code or binary without executing the application.	**Requires a running application** • DAST doesn't require source code or binaries. • It analyzes by executing the application.
Finds vulnerabilities earlier in the SDLC • The scan can be executed as soon as code is deemed feature-complete	**Finds vulnerabilities toward the end of the SDLC** • Vulnerabilities can be discovered after the development cycle is complete.
Less expensive to fix vulnerabilities • Since vulnerabilities are found earlier in the SDLC, it's easier and faster to remediate them. • Findings can often be fixed before the code enters the QA cycle.	**More expensive to fix vulnerabilities** • Since vulnerabilities are found toward the end of the SDLC, remediation often gets pushed into the next cycle. • Critical vulnerabilities may be fixed as an emergency release.
Can't discover run-time and environment-related issues • Since the tool scans static code, it can't discover run-time vulnerabilities.	**Can discover run-time and environment-related issues** • Since the tool uses dynamic analysis on an application, it is able to find run-time vulnerabilities.
Typically supports all kinds of software • Examples include web applications, web services, and thick clients.	**Typically scans only apps like web applications and web services** • DAST is not useful for other types of software.

SAST and DAST techniques complement each other.

Both need to be carried out for comprehensive testing.

BLACKDUCK

To learn how to create a comprehensive software security testing program, visit www.BlackDuck.com/software

DAST Tooling

OWASP
Zed Attack Proxy

BURPSUITE

Probely

Acunetix

Demo: OWASP Zap

Secure DevOps

Lab 05: Enable DAST

Secure DevOps