

Die Folien sind für den persönlichen Gebrauch im Rahmen des Moduls gedacht. Eine Veröffentlichung oder Weiterverteilung an Dritte ist nicht gestattet. (G. Neugebauer)



# Development, Security and Operations (DevSecOps)

## (Bachelor Wahlfach, Modul 55803)

Wintersemester 2022/2023

### **Kapitel 3: Security Testing & SAST**

Prof. Dr. Georg Neugebauer

*Lehrgebiet IT-Sicherheit*

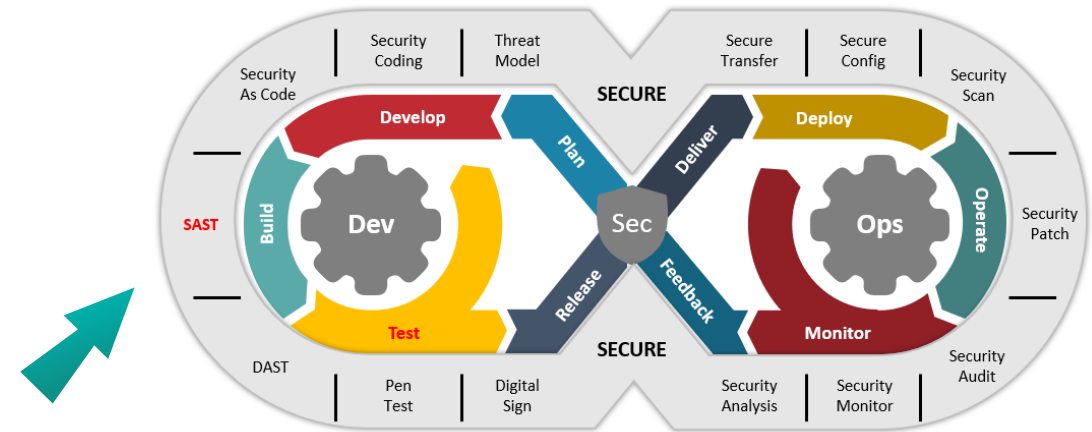
*Fachbereich 5 – Elektrotechnik und Informationstechnik*

*FH Aachen*

*Email: [g.neugebauer@fh-aachen.de](mailto:g.neugebauer@fh-aachen.de)*

# Security Testing & SAST im DevSecOps-Modell

- Hauptziele des Security Testing
  - Testen ist eine wichtige Tätigkeit im SDLC, insbesondere bevor der Kunde die Software erhält
  - Security Testing legt einen Fokus auf Fehler, die als Schwachstellen ausgenutzt werden können
    - Hier Fokus auf automatisierte Tests
- Hauptziele des Static Application Security Testing
  - Fehler im Source Code und potentielle Schwachstellen frühzeitig finden (während der Entwicklung)
  - Automatisierte Tools unterstützen die Entwickler und machen eine sichere Entwicklung erst möglich



# Umfrage zum Testen

---

Besuchen Sie [www.menti.com](https://www.menti.com) und benutzen Sie den Code 8369 7274

<https://www.menti.com/alfdodzngo4h>



# Beispiel Apple Bug - 2014

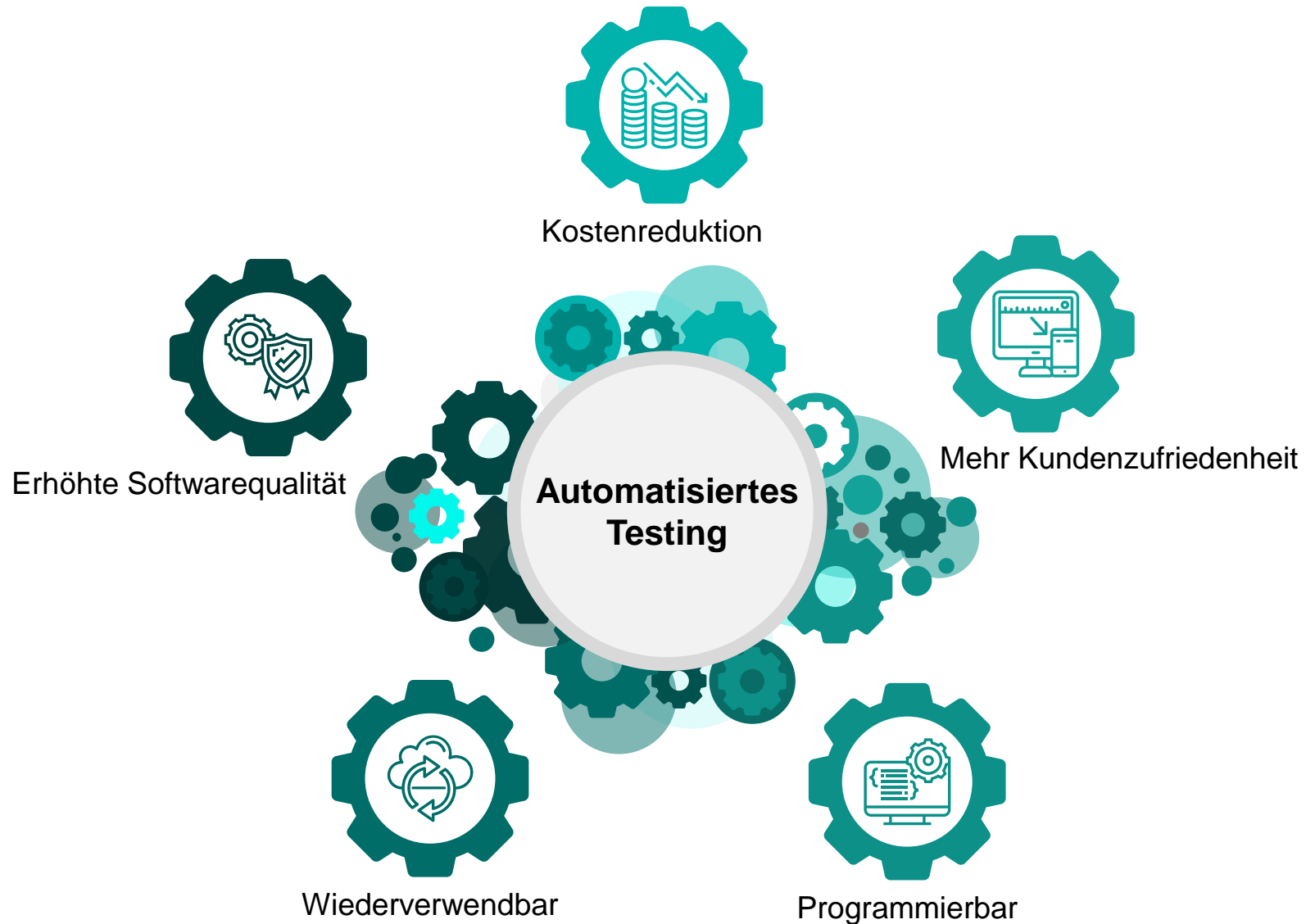
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                uint8_t *signature, UInt16 signatureLen)
{
    OSStatus      err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...

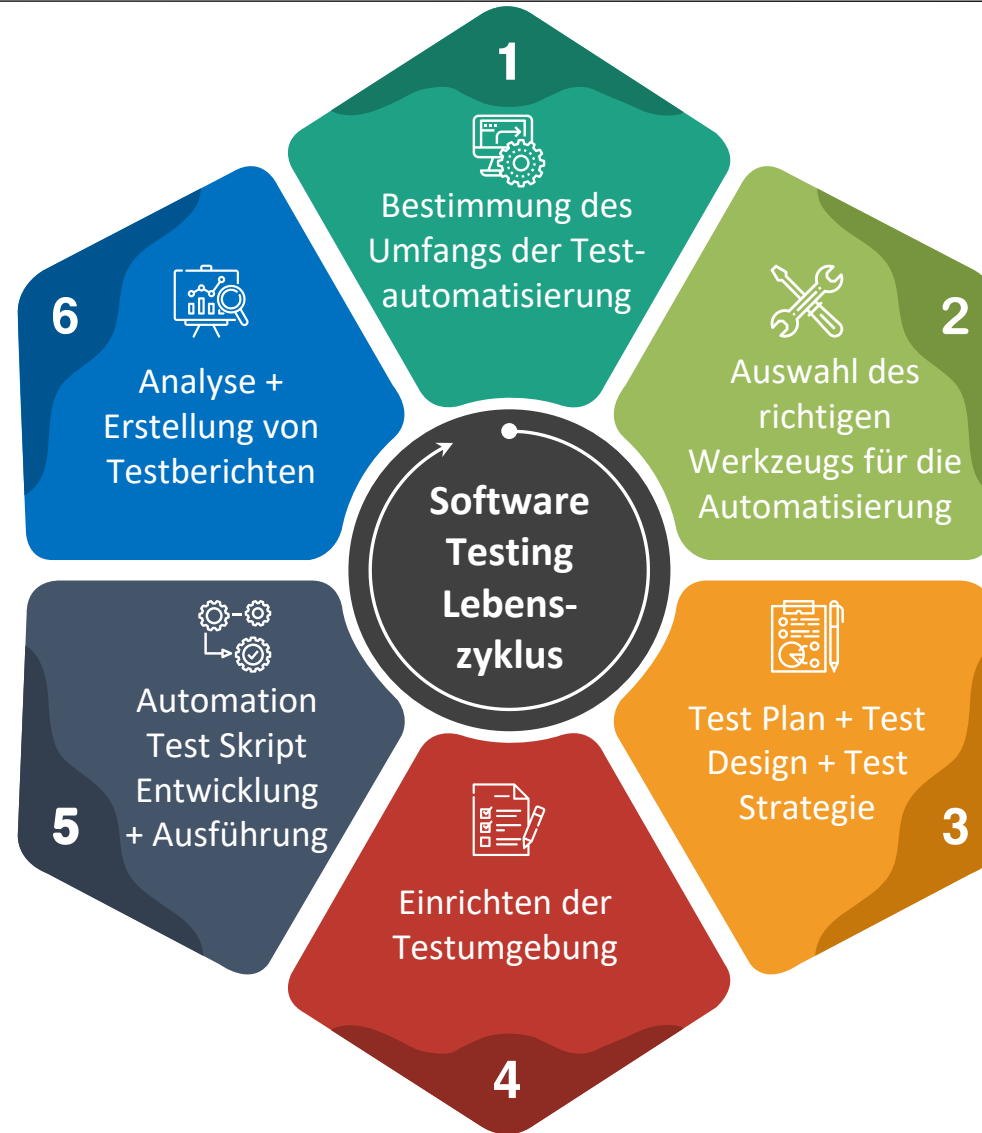
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

**Wer sieht den Fehler?**

# Motivation



# Automatisierter Software Testing Life Cycle



# Use Cases für Automatisiertes Testing



## Regression Testing

Regressionstests eignen sich für automatisierte Tests, da der Code häufig geändert wird und die menschliche Fähigkeit zur rechtzeitigen und qualitativen Durchführung von Tests überfordert ist.



## Load Testing

Es soll überprüft werden, ob das System/die Anwendung die erwartete Anzahl von Transaktionen bewältigen kann, und das Verhalten des Systems/der Anwendung sowohl unter normalen als auch unter Spitzenlastbedingungen überprüft werden.



## Performance Testing

Bei dieser Art von Tests werden die Geschwindigkeits-, Skalierbarkeits- und/oder Stabilitätseigenschaften des zu prüfenden Systems oder der zu prüfenden Anwendung ermittelt oder validiert.

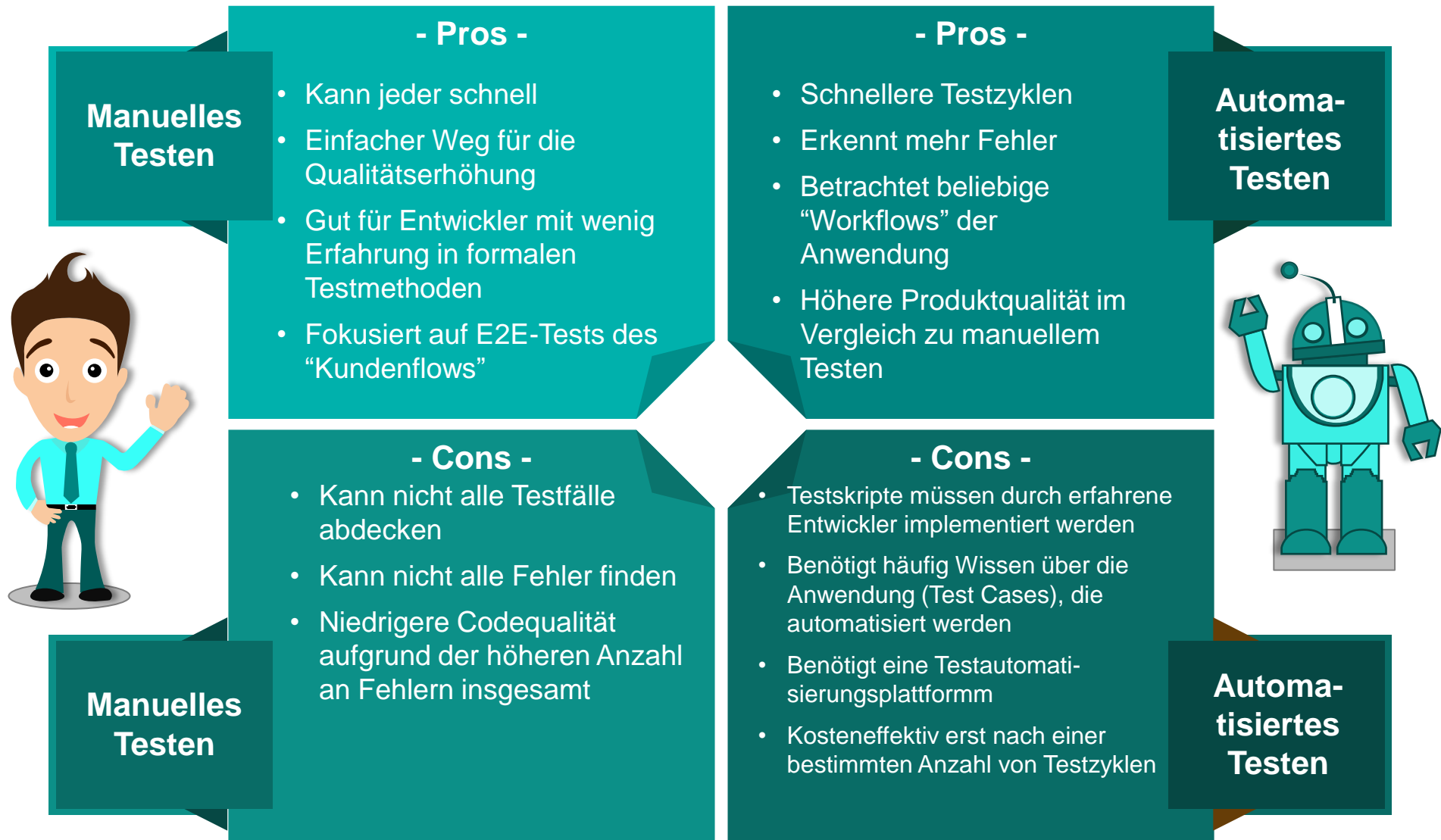
- Weitere geeignete Use Cases
  - End-to-End Testing
    - Wertvoll, aber manchmal schwierig zu automatisieren
  - Unit-Tests
  - Integrationstests

Continuous Delivery

Continuous Integration

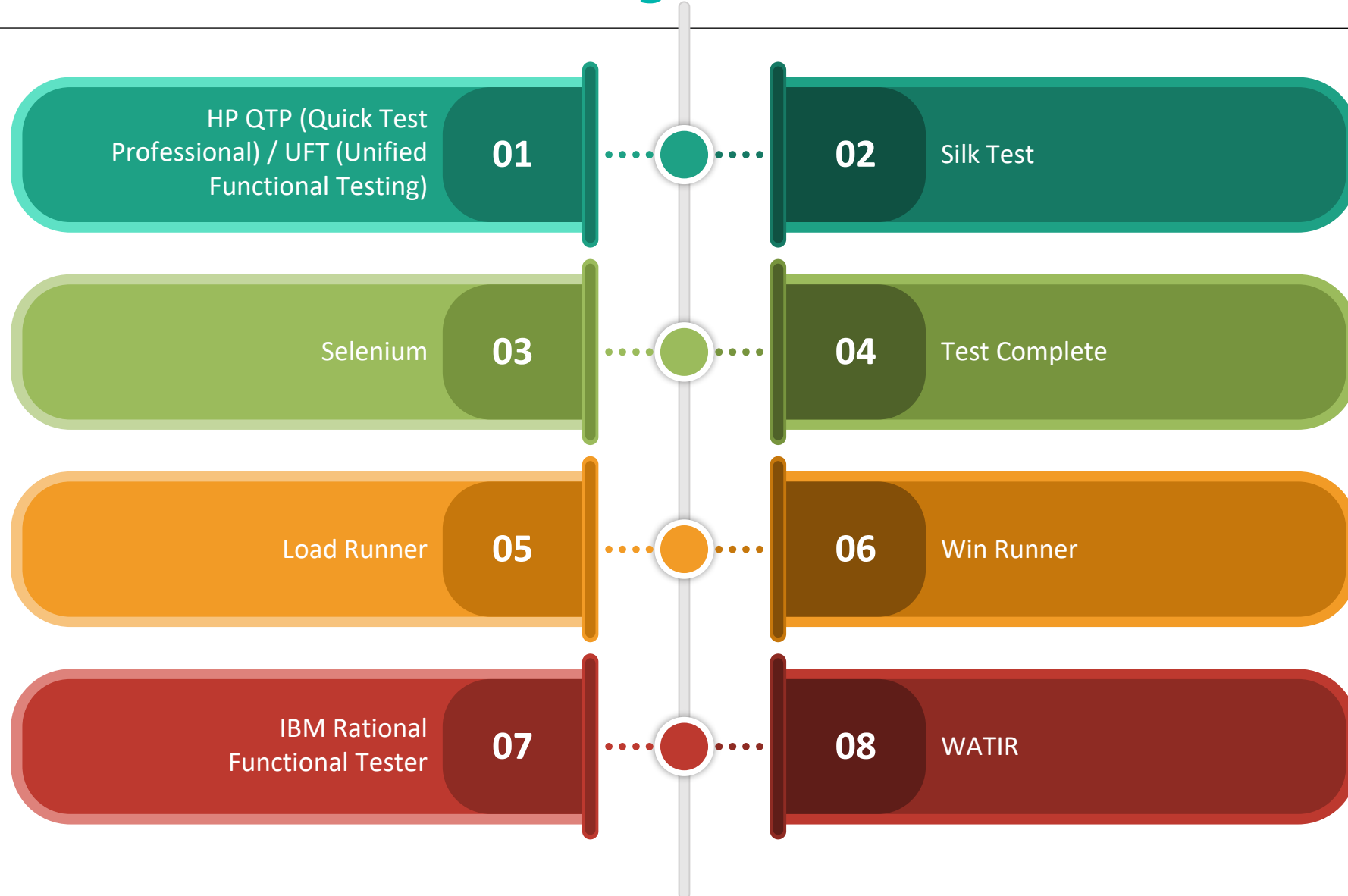
Automated Testing

# Vergleich Automatisiertes Testen vs Manuelles Testen





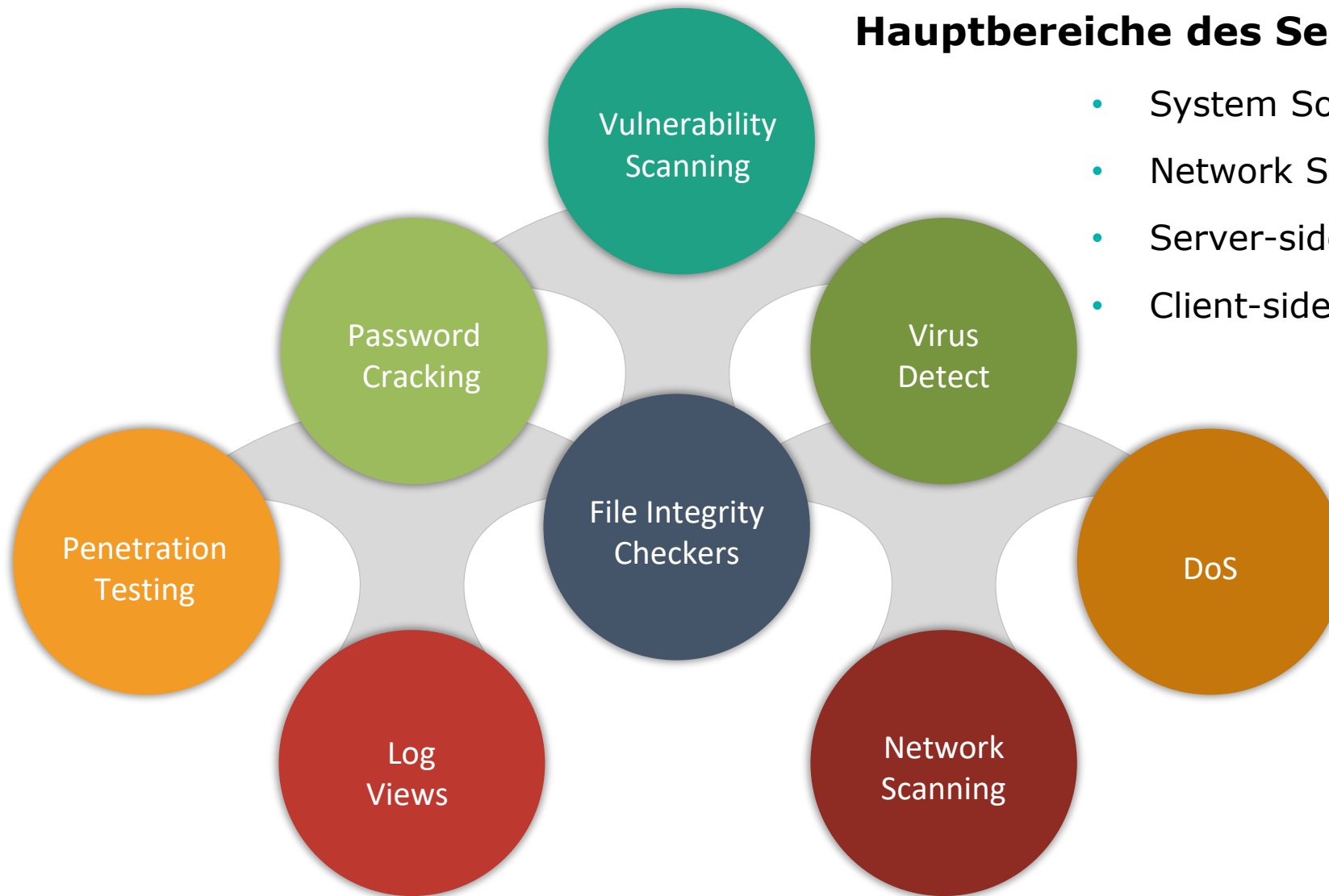
# Bekannte Testautomatisierungsframeworks



# Security Testing – Auswahl an Techniken

## Hauptbereiche des Security Testing

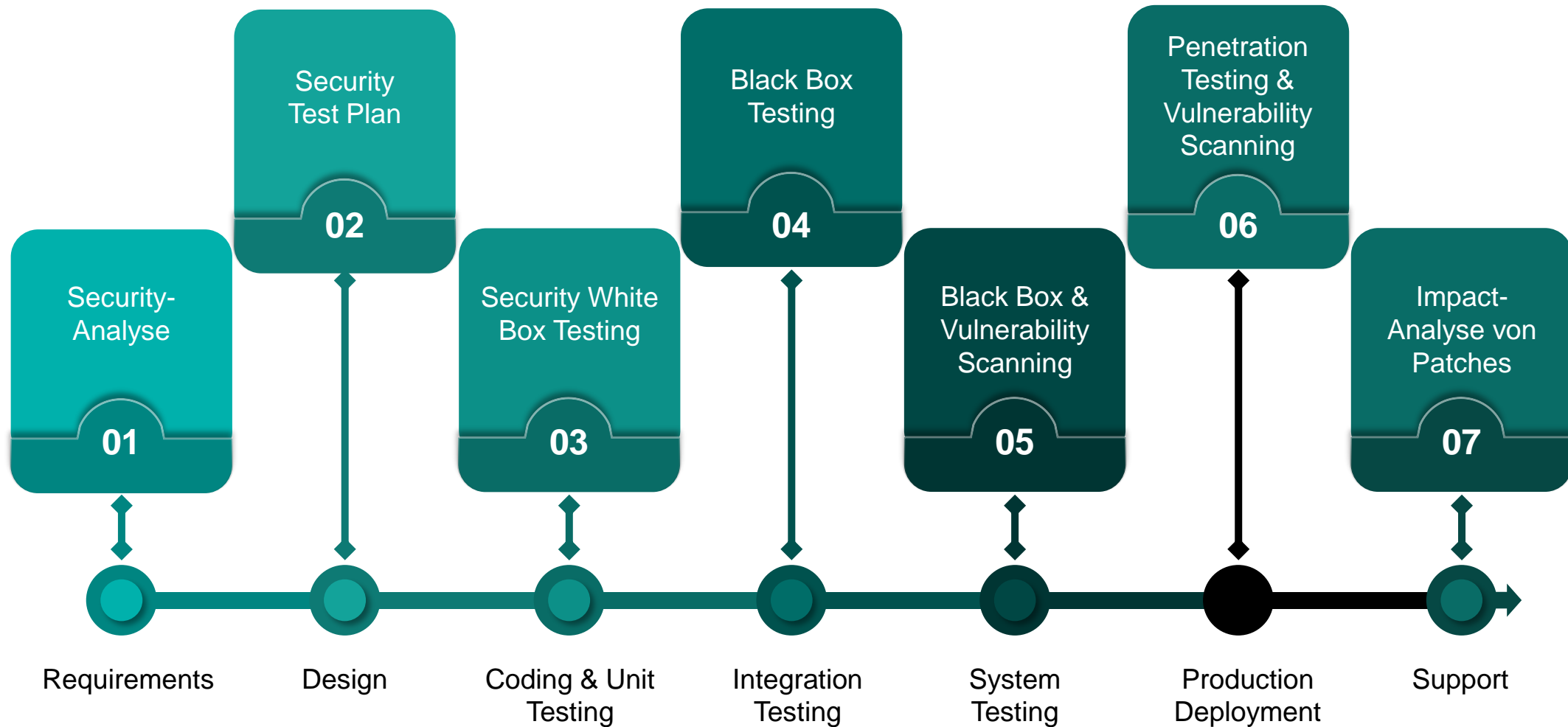
- System Software Security
- Network Security
- Server-side Application Security
- Client-side Application Security



# Hauptschutzziele des Security Testing



# Security Testing in den unterschiedlichen Phasen des SDLC



# Definition SAST

## Definition

- Static Application Security Testing (SAST), oder statische Analyse, ist eine Testmethode, die den Quellcode analysiert, um Sicherheitslücken zu finden.
  - SAST scannt die Anwendung, bevor der Code kompiliert wurde
  - White Box Testing
  - Hilfreich für Entwickler, um Schwachstellen frühzeitig zu erkennen und zu beheben
- Wir schauen uns unterschiedliche Features von SAST-Tools anhand von Snyk<sup>1</sup> an



<sup>1</sup> <https://snyk.io/>

# Umfrage zu SAST

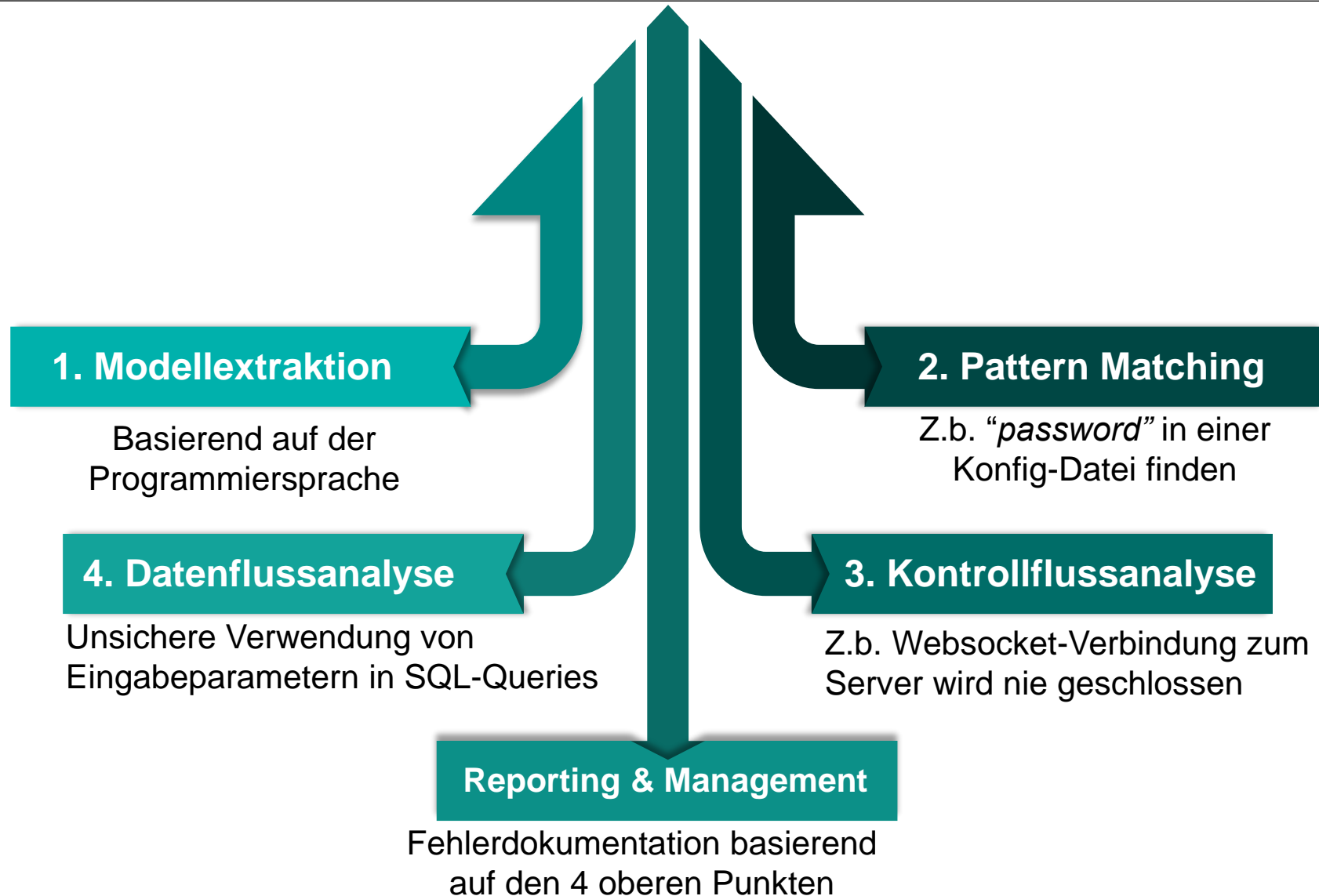
---

Besuchen Sie [www.menti.com](https://www.menti.com) und benutzen Sie den Code 6445 4629

<https://www.menti.com/al1tom1x9gjt>



# SAST-Tool – Ablauf und Prozess



# Vor- und Nachteile von SAST

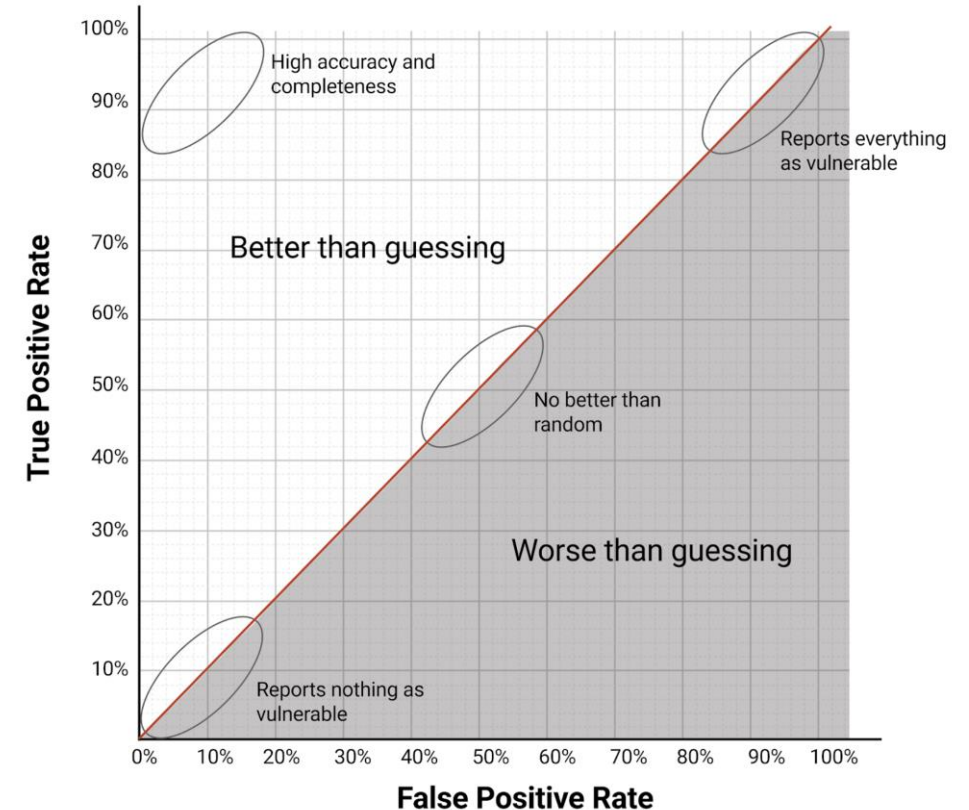
- Je früher eine Schwachstelle im Softwareentwicklungsprozess behoben wird:
  - Weniger Kosten und Ressourcen (Dev/Test/Production – 1/10/100)
- ✓ Großer Vorteil: 100 % Source Code Abdeckung möglich + Line of Code Information
- Nachteil, frühe Integration im SDLC: Sehr viele False-Positives (Bugs, Schwachstellen)
- Wichtig: **Tooling**-Auswahl + Grad der **Automatisierung** + klarer **Prozess** sind entscheidend!





# Integration von SAST in CI/CD

- So früh wie möglich -> Alle Repos, Alle Branches, Jeden Commit scannen - z.B. Jenkins + SonarQube – oder sogar noch vorher in der IDE
- Prozess definieren – Wer ist wofür zuständig – Stakeholder einbinden - Fehlerkultur etablieren
- False positives abarbeiten
  - Security Know-how
  - Programmiersprachenerfahrung
  - Anwendungsexpertise
  - Domänenwissen
  - Deploymentkenntnisse
- Tooling-Wahl wichtig und entscheidend



Praetorian, 03/2020

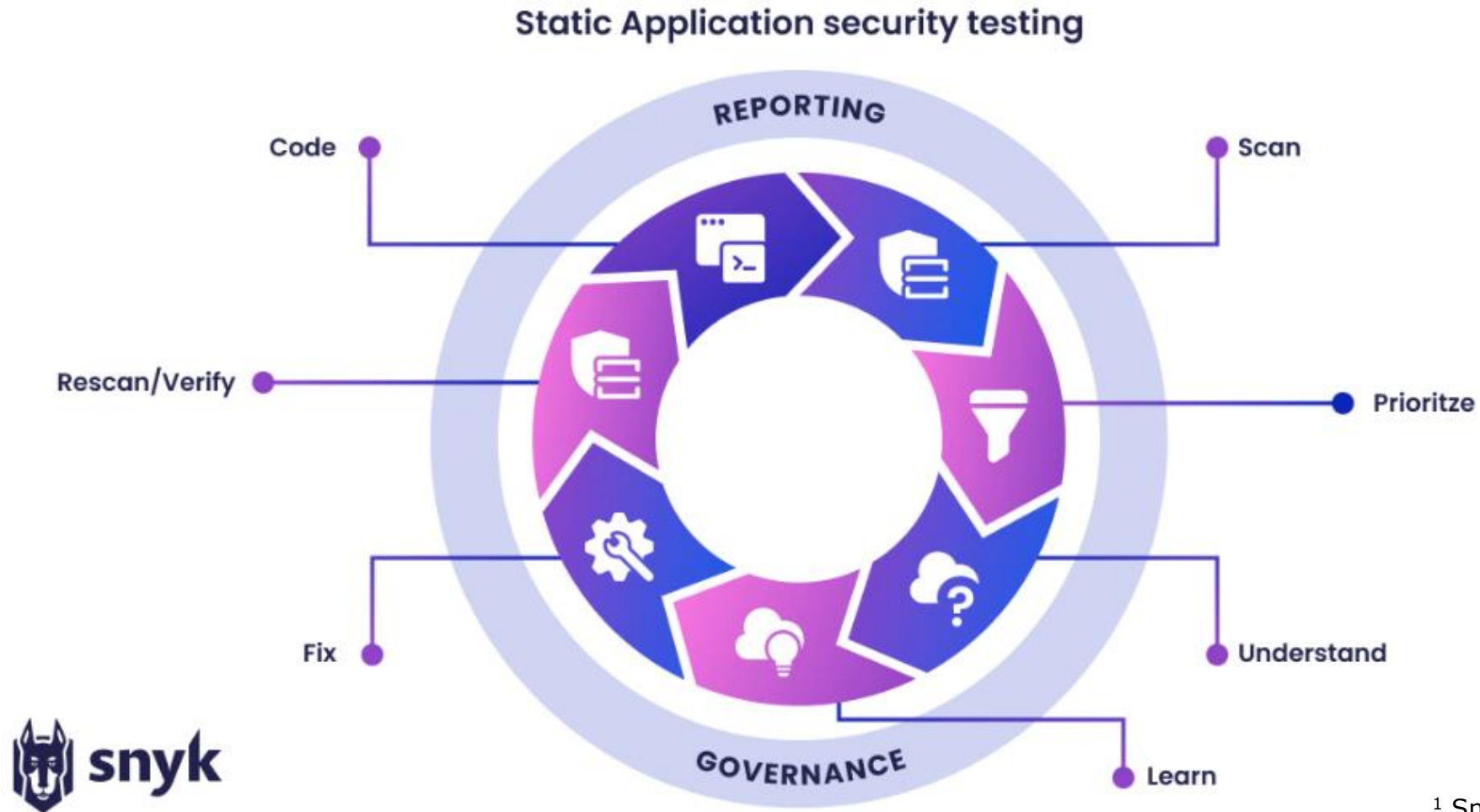
# Toolvorstellung Snyk



- Snyk ist eine branchenführende Security Intelligence Lösung für das automatisierte Finden und Beheben von Schwachstellen
  - Im Source Code – Snyk Code (SAST)
  - In Open-Source-Abhängigkeiten und Bibliotheken – Snyk Open Source (SCA)
  - in Containern – Snyk Container
  - in Infrastruktur als Code – Snyk Infrastructure as Code
  - In Cloud-Umgebungen – Snyk Cloud
- Wir schauen uns im folgenden Snyk Code und Snyk Container genauer an...<sup>1</sup>

<sup>1</sup> Alle Screenshots stammen von einer snyk.io und einem Beispielprojekt über einen registrierten Account.

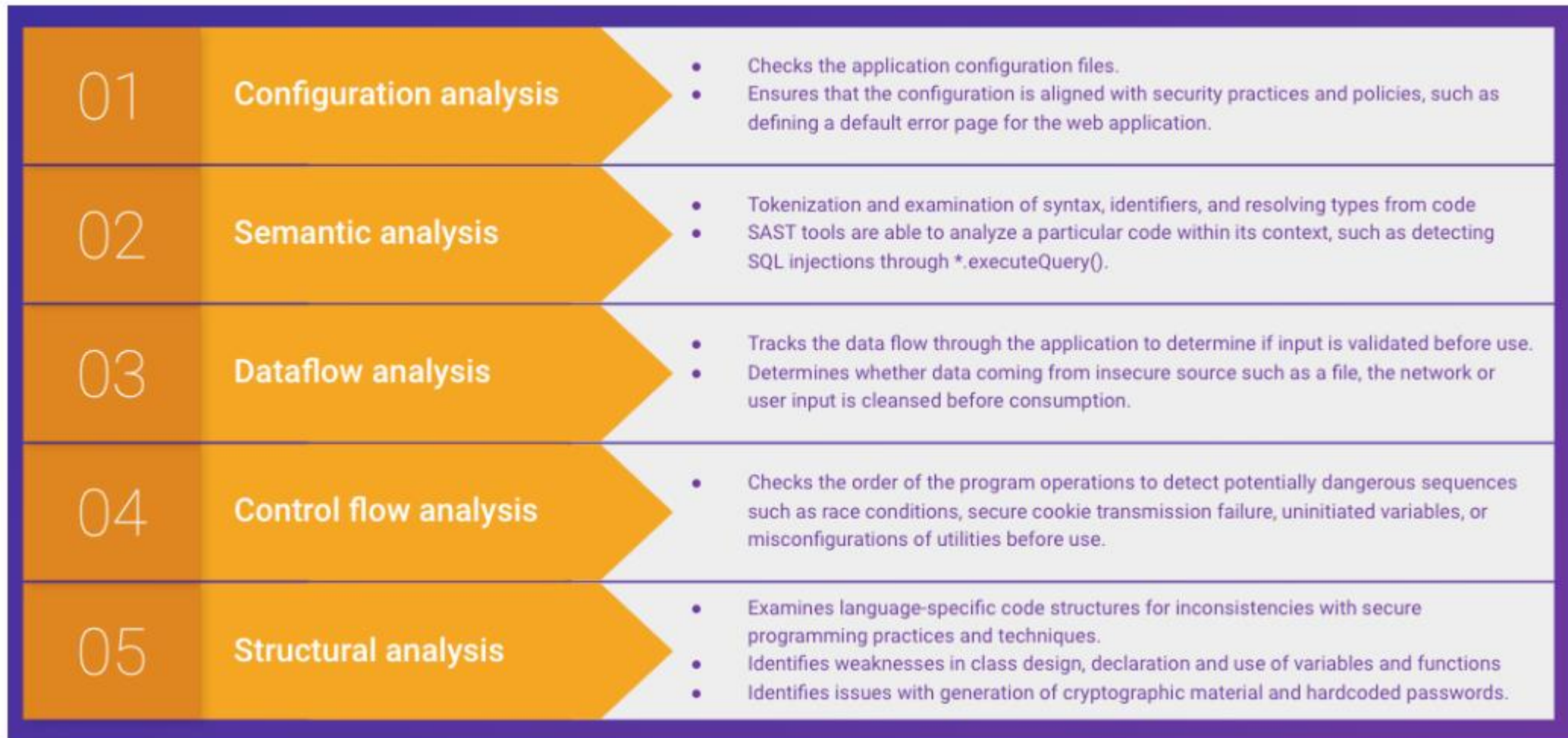
# Snyk SAST



<sup>1</sup> Snyk Learn

<sup>1</sup> <https://snyk.io/learn/application-security/static-application-security-testing/>

# Snyk SAST - Analysearten



<sup>1</sup> Snyk Learn

<sup>1</sup> <https://snyk.io/learn/application-security/static-application-security-testing/>














# Snyk Dashboard

- Dashboard enthält Informationen zum Unternehmen und Projekten
- Support für mehrere Organisationen, Teammanagement, diverse Integrationen und detailliertes Reporting

The screenshot displays the Snyk dashboard for the organization 'gneugeb-fhaachen'. The left sidebar contains navigation links: Organization, Dashboard, Projects (selected), Reports, Integrations, Members, and Settings. The main content area shows a search bar for targets, a filter for 'SHOW' (With issues, Without issues), and a section for 'INTEGRATIONS' (CI/CLI, GitHub). A notification banner states: 'The last import successfully processed 7 projects. View the last import log for more details.' Below this, a project entry for 'gneugeb-fhaachen/vulnerablerepo' is shown with a GitHub icon, 28 stars, and a lock icon. To the right of the project name are vulnerability counts: 111 C, 471 H, 548 M, 1.2k L. At the bottom, a prompt asks 'Ready to import another project?' with the subtext 'Secure your entire stack with Snyk' and an 'Add project' button.

# Snyk Projektinfos

- Auflistung aller gefundenen Fehler in den unterschiedlichen Source Code Dateien
  - Klassifikation: Critical, High, Medium, Low
  - Anzahl der gefundenen Fehler und Testdatum

▼  28 gneugeb-fhaachen/vulnerablerepo 	111 <b>C</b> 471 <b>H</b> 548 <b>M</b> 1.2k <b>L</b>	
 Code analysis	0 <b>C</b> 11 <b>H</b> 37 <b>M</b> 3 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 dvna/Dockerfile	54 <b>C</b> 209 <b>H</b> 235 <b>M</b> 557 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 dvna/Dockerfile-dev	54 <b>C</b> 209 <b>H</b> 235 <b>M</b> 557 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 dvna/ package.json	3 <b>C</b> 16 <b>H</b> 12 <b>M</b> 1 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/Dockerfile	0 <b>C</b> 1 <b>H</b> 15 <b>M</b> 54 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/node_modules/accepts/ package.json	0 <b>C</b> 1 <b>H</b> 0 <b>M</b> 0 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/node_modules/body-parser/ package.json	0 <b>C</b> 1 <b>H</b> 0 <b>M</b> 2 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/node_modules/debug/ package.json	0 <b>C</b> 0 <b>H</b> 0 <b>M</b> 2 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/node_modules/ejs-locals/node_modules/ejs/ package.json	0 <b>C</b> 2 <b>H</b> 3 <b>M</b> 0 <b>L</b>	Tested a month ago <a href="#">Activate</a>
 VulnNodeApp/node_modules/ejs-locals/ package.json	0 <b>C</b> 2 <b>H</b> 3 <b>M</b> 0 <b>L</b>	Tested a month ago <a href="#">Activate</a>



# Snyk – Issue Übersicht

SEVERITY

☐ Critical 54
 ☐ High 209
 ☐ Medium 235
 ☐ Low 557

PRIORITY SCORE

Scored between 0 - 1000

FIXABILITY

☐ Fixable 0
 ☐ Partially fixable 213
 ☐ No fix available 842

EXPLOIT MATURITY

☐ Mature 11
 ☐ Proof of concept 2
 ☐ No known exploit 1042
 ☐ No data 0

STATUS

☒ Open 1055
 ☐ Patched 0
 ☐ Ignored 0

OS BINARIES

☐ OS packages 22
 ☐ Node 22

1055 of 1055 issues

Sort by highest priority score ▾

NEW

Did you know...

You can reduce the backlog of existing vulnerabilities at a manageable pace with prioritized fix pull requests - enable for your GitHub integration.

C

curl/libcurl4-openssl-dev - Double Free

SCORE 714

VULNERABILITY

CWE-415 [↗](#) | CVE-2019-5481 [↗](#) | CVSS 9.8 [↗](#) **CRITICAL** | SNYK-DEBIAN9-CURL-466508 [↗](#)

Introduced through

curl/libcurl4-openssl-dev@7.52.1-5+deb9u9, curl/libcurl3-gnutls@7.52.1-5+deb9u9 and others

Fixed In

curl/libcurl4-openssl-dev@7.52.1-5+deb9u10

Exploit maturity

NO KNOWN EXPLOIT

Show more detail ▾

Ignore

C

curl/libcurl4-openssl-dev - Out-of-bounds Write

SCORE 714

VULNERABILITY

CWE-120 [↗](#) | CVE-2019-5482 [↗](#) | CVSS 9.8 [↗](#) **CRITICAL** | SNYK-DEBIAN9-CURL-466505 [↗](#)

Introduced through

curl/libcurl4-openssl-dev@7.52.1-5+deb9u9, curl/libcurl3-gnutls@7.52.1-5+deb9u9 and others

Fixed In

curl/libcurl4-openssl-dev@7.52.1-5+deb9u10

Exploit maturity

NO KNOWN EXPLOIT

Show more detail ▾

Ignore

C

dpkg/libdpkg-perl - Directory Traversal

SCORE 714

VULNERABILITY

CVE-2022-1664 [↗](#) | CVSS 9.8 [↗](#) **CRITICAL** | SNYK-DEBIAN9-DPKG-2847943 [↗](#)

© FH AACHEN UNIVERSITY OF APPLIED SCIENCES | Development, Security and Operations (DevSecOps) – Kapitel 3 – Prof. Dr. Georg Neugebauer

23

FH AACHEN  
UNIVERSITY OF APPLIED SCIENCES

# SAST – Benötigte Informationen über Issues

- Titel und Beschreibung
- Details und Erläuterungen zum Fehler und CVE, CWE, CVSS (falls vorhanden)
- Scoring des Fehlers und Klassifikation gemäß SAST-Tool und Konfiguration
- Bei Bibliotheken: Wann aufgetreten und mit welcher Version behoben?
- Exploit maturity – Wie schnell muss ich den Fehler beheben...
- Art des Fehlers (SQL Injection, Cross-site Scripting (XSS), ...)
- Beschreibung für die Fehlerbehebung und ggf. Automatisierung





# Snyk – Details eines Fehlers

**H** mariadb-10.1/libmariadbclient18 - Arbitrary Code Injection [↗](#)  
VULNERABILITY | [CWE-78](#) | [CVE-2021-27928](#) | [CVSS 7.2](#) **HIGH** | [SNYK-DEBIAN9-MARIADB101-1087460](#)

SCORE  
**571**

Introduced through mariadb-10.1/libmariadbclient18@10.1.41-0+deb9u1, mariadb-10.1/libmariadbclient-dev-compat@10.1.41-0+deb9u1 and others  
Fixed in mariadb-10.1/libmariadbclient18@10.1.48-0+deb9u2

Exploit maturity **MATURE**

Show less detail ^

## Detailed paths


- Introduced through: node@carbon › mariadb-10.1/libmariadbclient18@10.1.41-0+deb9u1  
Fix: No remediation path available.
- Introduced through: node@carbon › mariadb-10.1/libmariadbclient-dev-compat@10.1.41-0+deb9u1  
Fix: No remediation path available.
- Introduced through: node@carbon › mariadb-10.1/libmariadbclient-dev@10.1.41-0+deb9u1  
Fix: No remediation path available.

## Security information


Factors contributing to the scoring:

- Snyk: [CVSS 7.2](#) - High Severity
- NVD: [CVSS 7.2](#) - High Severity
- Debian Security Rating: Not yet assigned

Why are the scores different? [Learn how Snyk evaluates vulnerability scores](#)

 Ignore

# Snyk – Weiteres Beispiel


**pg** - Arbitrary Code Execution

SCORE  
**801**

VULNERABILITY | [CWE-94](#) | [CVE-2017-16082](#) | [CVSS 8.3](#) **HIGH** | [NPM:PG:20170813](#)

Introduced through
 

pg-promise@4.8.1

Exploit maturity
 

MATURE

Fixed in
 

pg@2.11.2, @3.6.4, @4.5.7, @5.2.1, @6.0.5, @6.1.6, @6.2.5, @6.3.3, @6.4.2, @7.0.2, @7.1.2

[Show less detail](#)

**Detailed paths and remediation**

- Introduced through: vulnerable-node-source@0.0.0 › pg-promise@4.8.1 › pg@5.1.0

Fix: [Upgrade to pg-promise@5.9.2](#)

**Security information**

Factors contributing to the scoring:

- Snyk: [CVSS 8.3](#) - High Severity
- NVD: [CVSS 9.8](#) - Critical Severity

[Why are the scores different? Learn how Snyk evaluates vulnerability scores](#)

**Overview**

pg is a non-blocking PostgreSQL client for node.js.

Affected versions of this package are vulnerable to Arbitrary Code Execution. When parsing results of a query, it goes through a form of `eval`, and with a specially crafted column name, an attacker can cause code to run remotely on the server.

PoC:

```
const { Client } = require('pg')
const client = new Client()
client.connect()

const sql = `SELECT 1 AS "\'/*", 2 AS "\'*/\n + console.log(process.env)] = null;\n/`

client.query(sql, (err, res) => {
  client.end()
});
```

NEW
[Learn about this type of vulnerability](#)

# Snyk – Weiteres Beispiel

## SQL Injection

SNYK CODE | [CWE-89](#)

SCORE  
**808**

```
30
31 |         $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
32 |         #print $query;
33 |         try {
34 |             $results = $sqlite_db_connection->query($query);
35
```


Unsanitized input from *an HTTP parameter flows* into *query*, where it is used in an SQL query. This may result in an SQL Injection vulnerability.


 Webapp/vulnerabilities/sqli/source/low.php

8 steps in 1 file

NEW

 [Learn about this type of vulnerability and how to fix it](#)

 Ignore

 Full details

# SAST – Automatisierung für die Fehlerbehebung

API-Call, um direkt über CI den PR zu erstellen...

Recommendations for upgrading the base image

	BASE IMAGE	VULNERABILITIES	SEVERITY
Current image	node:carbon	1055	<div>54 C</div> <div>209 H</div> <div>235 M</div> <div>557 L</div>
Alternative upgrades	node:16.17-bullseye-slim	40	<div>0 C</div> <div>0 H</div> <div>0 M</div> <div>40 L</div>

[Show more upgrade types](#)

[View docs](#)

[Open a fix PR](#)

Debian 9 is no longer supported by the Debian maintainers. Vulnerability detection may be affected by a lack of security updates.



# Snyk – SCA

- Snyk hat auch eine Integration für die Software Composition Analysis (SCA)
- Ergebnisse werden in separatem Tab angezeigt
- In Zukunft: Support für SBOM-Generierung und Analyse

Issues 14 Fixes Dependencies 93						
<div> <input type="text" value="Search..."/> </div>						
DEPENDENCY ^	LATEST	LAST PUBLISHED	ISSUES			
destroy@1.0.4			0	C	0	2
ee-first@1.1.1			0	C	0	6
ejs@2.7.4			0	C	1	1
ejs@0.8.8			0	C	2	1
ejs-locals@1.0.2			0	C	0	1

# Snyk – Docker Image Analyse

- Imageanalyse und Behebungsvorschläge
- Bei Codeanalyse kann man sich die Issues des Containers anschauen

<b>IMPORTED BY</b>  g.neugebauer@fh-aachen.de	<b>PROJECT OWNER</b> <a href="#">+ Add a project owner</a>	<b>SOURCE</b>  GitHub	<b>TARGET OS</b> debian:9
<b>IMAGE TAG</b> carbon	<b>BASE IMAGE</b> node:carbon	<b>REPOSITORY</b> <a href="#">vulnerablerepo</a>	<b>MANIFEST</b> <a href="#">dvna/Dockerfile</a>
<b>ENVIRONMENT</b> <a href="#">+ Add a value</a>	<b>BUSINESS CRITICALITY</b> <a href="#">+ Add a value</a>	<b>LIFECYCLE</b> <a href="#">+ Add a value</a>	<b>LINKED IMAGES</b> <a href="#">+ Add a value</a>

Recommendations for upgrading the base image

	BASE IMAGE	VULNERABILITIES	SEVERITY
Current image	node:carbon	1055	<div> <div>54 C</div> <div>209 H</div> <div>235 M</div> <div>557 L</div> </div>
Alternative upgrades	node:16.17-bullseye-slim	40	<div> <div>0 C</div> <div>0 H</div> <div>0 M</div> <div>40 L</div> </div>

🔗 Open a fix PR

# Snyk – Lizenzen und Copyright (Lizenz Compliance Tool)

- Analyse des Projekts und Infos darüber, welche Open-source Lizenzen verwendet werden (auch in Abhängigkeiten). Hilfreich für Compliance-Analysen.
- Warnung, falls problematische Lizenz entdeckt wird
- Individuelle Konfiguration des „Severity Levels“ einer Lizenz

LICENSE	DEPENDENCIES	PROJECTS
MIT	<a href="#">867 dependencies</a>	<a href="#">84 projects</a>
ISC	<a href="#">76 dependencies</a>	<a href="#">70 projects</a>
Apache-2.0	<a href="#">139 dependencies</a>	<a href="#">70 projects</a>
BSD-3-Clause	<a href="#">63 dependencies</a>	<a href="#">65 projects</a>
BSD-2-Clause	<a href="#">50 dependencies</a>	<a href="#">64 projects</a>
Unknown	<a href="#">117 dependencies</a>	<a href="#">49 projects</a>
Multiple licenses: <a href="#">BSD-2-Clause</a> , <a href="#">MIT</a> , <a href="#">Apache-2.0</a>	<a href="#">1 dependency</a>	<a href="#">47 projects</a>
Dual license: <a href="#">MIT</a> , <a href="#">X11</a>	<a href="#">4 dependencies</a>	<a href="#">11 projects</a>
CC0-1.0	<a href="#">14 dependencies</a>	<a href="#">9 projects</a>
BSD-3-Clause OR MIT	<a href="#">1 dependency</a>	<a href="#">8 projects</a>
Dual license: <a href="#">BSD-3-Clause</a> , <a href="#">GPL-2.0</a>	<a href="#">1 dependency</a>	<a href="#">6 projects</a>
Dual license: <a href="#">MIT</a> , <a href="#">Apache-2.0</a>	<a href="#">3 dependencies</a>	<a href="#">5 projects</a>
All licenses	<a href="#">1 dependencies</a>	<a href="#">2 projects</a>

MEDIUM SEVERITY

NEW

EPL-1.0 license

Module: junit:junit

Introduced through: org.apache.maven:maven-core@3.1.0, org.apache.maven:maven-project@2.2.0 and others

Detailed paths

- Introduced through: snyk/snyk-maven-plugin@snyk/snyk-maven-plugin#f21227c61e6e61dbe35947a597bdd49e41df670a › org.apache.maven:maven-core@3.1.0 › org.apache.maven:maven-settings-builder@3.1.0 › org.sonatype.plexus:plexus-sec-dispatcher@1.3 › org.codehaus.plexus:plexus-container-default@1.0-alpha-9-stable-1 › junit:junit@3.8.1
- Introduced through: snyk/snyk-maven-plugin@snyk/snyk-maven-plugin#f21227c61e6e61dbe35947a597bdd49e41df670a › org.apache.maven:maven-project@2.2.0 › org.apache.maven:maven-artifact-manager@2.2.0 › org.codehaus.plexus:plexus-container-default@1.0-alpha-9-stable-1 › junit:junit@3.8.1
- Introduced through: snyk/snyk-maven-plugin@snyk/snyk-maven-plugin#f21227c61e6e61dbe35947a597bdd49e41df670a › org.apache.maven:maven-project@2.2.0 › org.apache.maven:maven-profile@2.2.0 › org.codehaus.plexus:plexus-container-default@1.0-alpha-9-stable-1 › junit:junit@3.8.1

...and 4 more


[More about this issue](#)

# Open-Source Lizenzen

- Viele Entwickler missverstehen den Begriff "Open Source,,!
  - Software kann **nicht** nach Belieben verwendet, kopiert, verändert und weitergegeben werden kann.
  - Hier gibt es Open-Source Lizenzen, die klare Vorgaben zur Nutzung machen, u.a.
    - Korrekte Verwendung, Einbindung und Wiederverwendung des Codes
    - Teilen und Modifizieren des Codes
    - Verteilung des Codes in „neuen“ eigenen Programmen oder Anwendungen
  - Public Domain oder Shareware ermöglicht die freie Verwendung ohne spezielle Genehmigungen oder Lizenzen.
- Closed-Source Software (Proprietary) verbietet im Allgemeinen den Zugriff zum Code und auch jegliche Modifikation oder Weiterverbreitung
- Auch wenn Sie nur Open-Source Software einsetzen, müssen Sie die gesetzlichen Anforderungen an die Softwarenutzung (Compliance) einhalten!



# Open-Source Lizenzen – Overview Snyc

snyc	Copyleft					Permissive			
									
Permissions in addition to commercial use, distribution, modification:									
Patent use	●	●	●	●	●	●	●	●	●
Patent use	●	●	●	●	●	●	●	●	●
Conditions									
Disclose source	●	●	●	●	●	●	●	●	●
License & copyright notice	●	●	●	●	●	●	●	Source	●
Network use is distribution	●	●	●	●	●	●	●	●	●
Same license	●	●	Library	●	File	●	●	●	●
State changes	●	●	●	Some	●	●	●	●	●
Limitations/Disclaimers									
Liability	●	●	●	●	●	●	●	●	●
Warranty	●	●	●	●	●	●	●	●	●
Trademark use	No explicit limitation				●	●	●	●	●

<sup>1</sup> Snyc Learn

<sup>1</sup> <https://snyc.io/learn/open-source-licenses/>

## Microsoft, GitHub und OpenAI verklagt: KI-Programmierhilfe Copilot kopiert Code<sup>1</sup>

- Sammelklage verlangt von Microsoft, GitHub und OpenAI 9 Milliarden Dollar Schadenersatz.
  - Hauptgrund: KI-Tool Copilot sammle Open-Source-Code ohne Lizenzangaben.
  - Kläger: Programmierer und Rechtsanwalt Matthew Butterick
  - Hochrechnung: 3.6 Millionen DMCA-Verstöße (pro Verstoß 2500 Dollar)
- Was ist Copilot?
  - KI-gestützte Programmierhilfe bei GitHub seit Juni 2022 im Einsatz
  - KI-System Codex von OpenAI
  - Funktionsweise: Basierend auf natürlicher Sprache werden Vorschläge für Code-Abschnitte oder passende Funktionen dem Entwickler präsentiert.
  - Lernphase der KI: Basierend auf zahlreichen öffentlichen GitHub-Repositories
- Wir schauen uns mal kurz GitHub Copilot an<sup>2</sup>...

<sup>1</sup> [https://www.heise.de/news/Microsoft-GitHub-und-OpenAI-verklagt-KI-Programmierhilfe-Copilot-kopiert-Code-7331566.html?wt\\_mc=nl.red.ho.ho-nl-newsticker.2022-11-07.link.link](https://www.heise.de/news/Microsoft-GitHub-und-OpenAI-verklagt-KI-Programmierhilfe-Copilot-kopiert-Code-7331566.html?wt_mc=nl.red.ho.ho-nl-newsticker.2022-11-07.link.link)

<sup>2</sup> <https://github.com/features/copilot>

# Diskussion zu GitHub Copilot

---

❖ **Diskussion:** Wie schätzen Sie Copilot aus IT-Sicherheitsperspektive ein?

---

Besuchen Sie [www.menti.com](https://www.menti.com) und benutzen Sie den Code 5877 8755

<https://www.menti.com/alwtobuc1g1p>



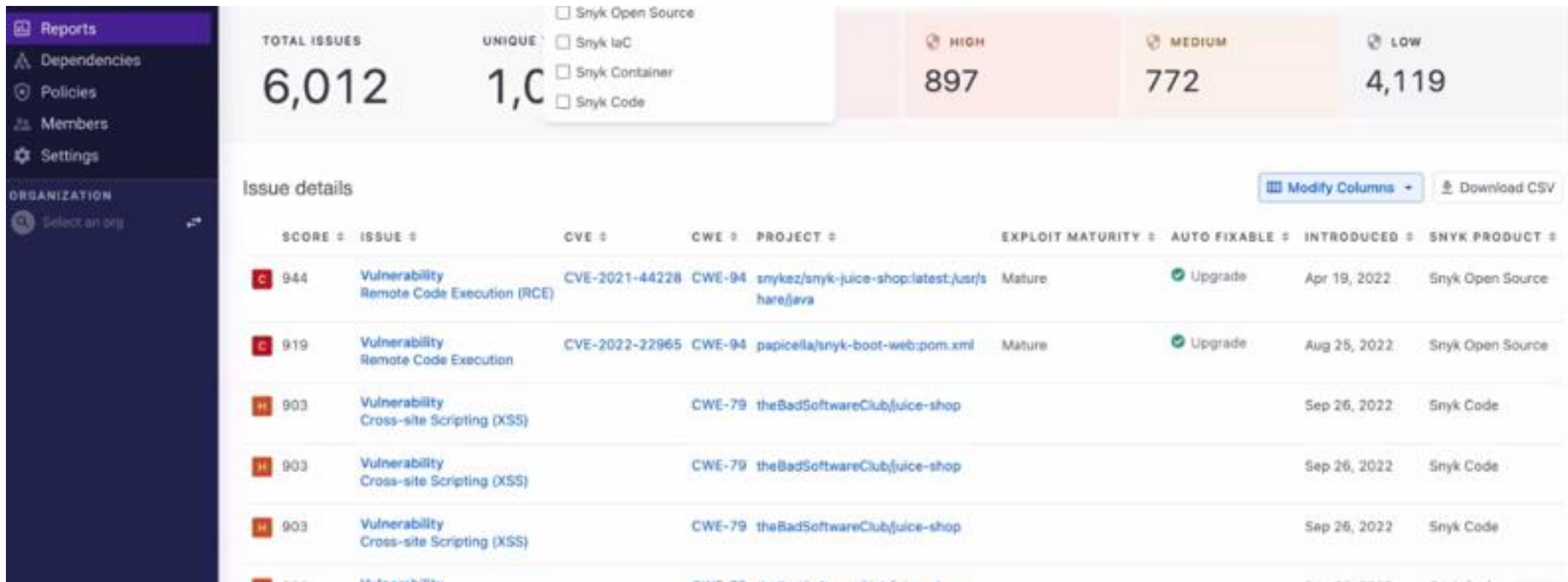
# Snyk – Integrations

- SCM
- Container-Registries
- Container orchestrators
- CI/CD
- IDE Plugins
- Notifications
- Vulnerability Management



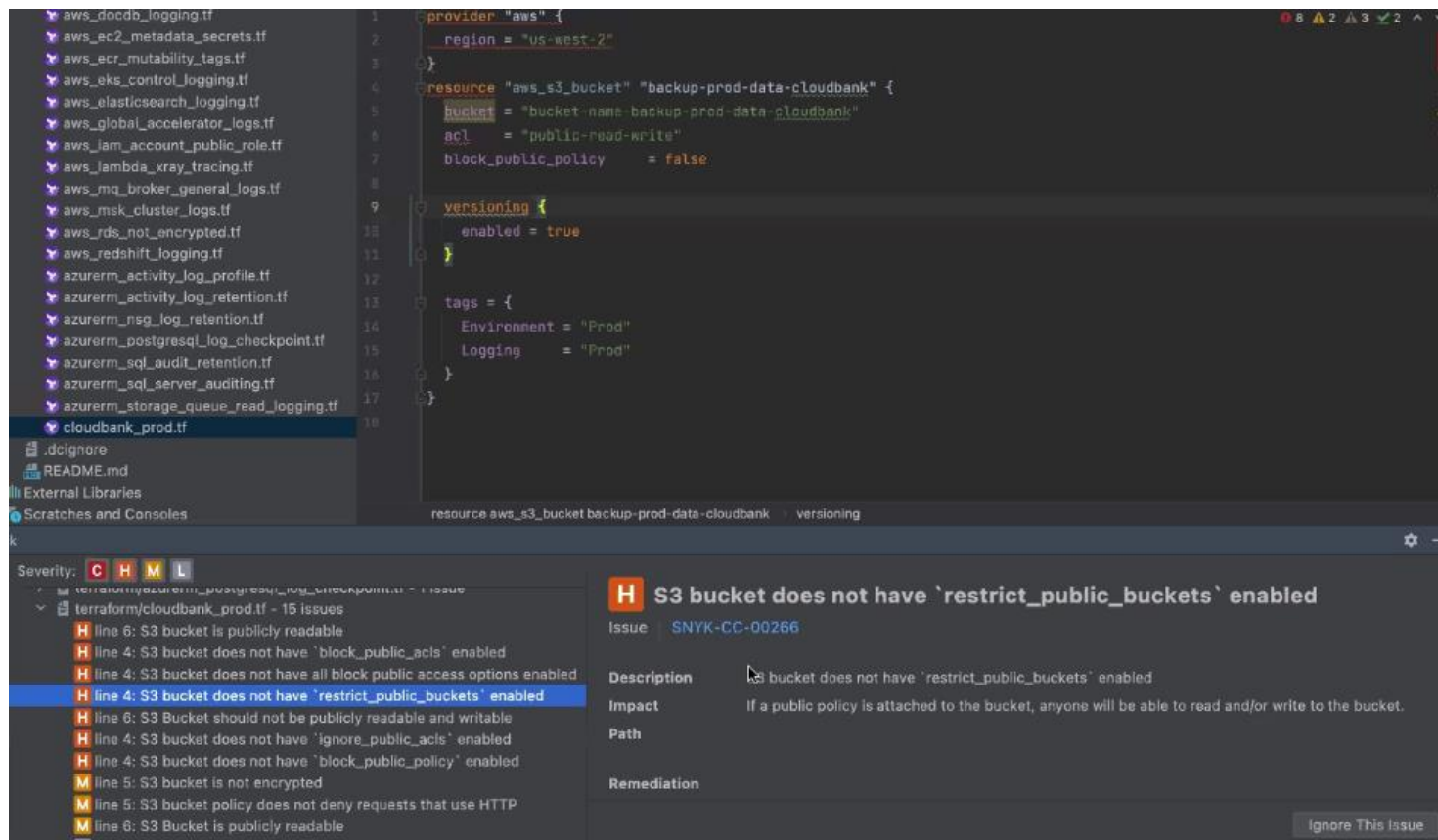
# Snyk – Reports

- Sehr gute Reporting-Funktion in Business-Plänen mit Filteroptionen und Exportfunktionen (csv, PDF, ...)
- Geeignet für Management-Report, Compliance-Nachweise, Externe Audits



# Snyk – Cloud

- Berücksichtigt bei Konfigurationsanalysen oder „Infrastructure as Code“ die Cloud-Umgebung, welche für das Deployment verwendet wird.
- Ein prominentes Tool im Bereich „Infrastructure As Code“ ist Terraform<sup>1</sup>



<sup>1</sup> <https://www.terraform.io/>

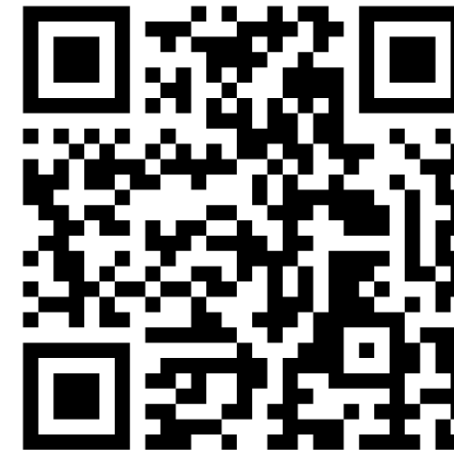


# Umfrage zu Snyk

---

Besuchen Sie [www.menti.com](https://www.menti.com) und benutzen Sie den Code 2596 1048

<https://www.menti.com/alp7yiw9n1x>



# Demo: Snyk

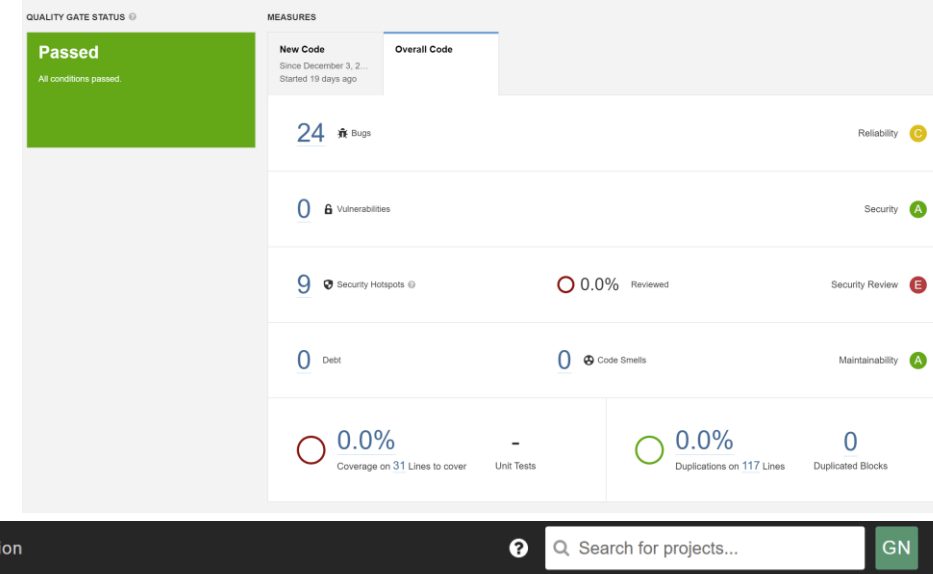
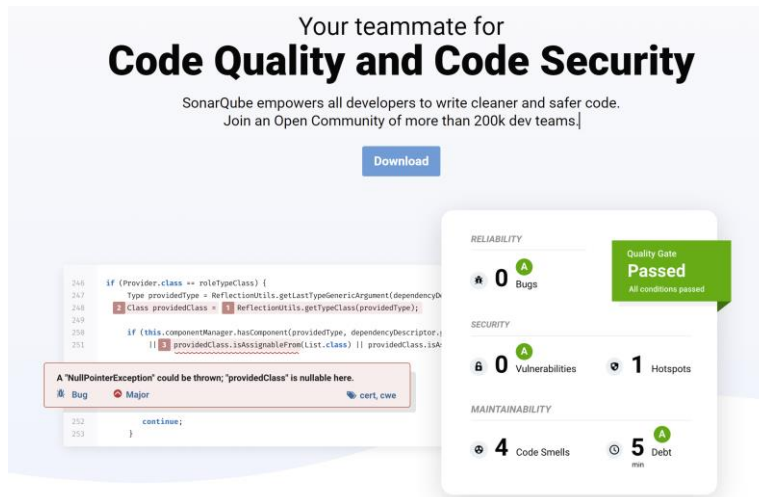
---

Wir schauen mal gemeinsam in ein Snyk-Projekt unter <https://snyk.io/> rein...



# SAST – Beispielanwendung SonarQube

- SonarQube<sup>1</sup> ist eine Open-Source-Plattform, die für die kontinuierliche Überprüfung der Codequalität entwickelt wurde
  - Automatische Überprüfung via statischer Codeanalyse des Codes
  - Erkennung von Code Smells, Bugs, Security Hotspots und Vulnerabilities
  - Mehr als 25 Programmiersprachen unterstützt
  - Reports, Programmierstandards und Empfehlungen, Quality Profile
  - Unterschiedliche Deployment-Varianten existieren (On-Prem, Docker, Cloud)



<sup>1</sup> <https://www.sonarqube.org/> / <https://www.sonarsource.com/products/sonarcloud/>

# SAST – Einfache Beispielanalysen

## Fehlkonfiguration – Weak TLS protocol

```
try {
    trustManager = trustManagerForCertificates(certbuffer.inputStream())
    val sslContext =
        SSLContext.getInstance("TLS")
```

Change this code to use a stronger protocol. Why is this an issue?

## Unsicherer Algorithmus – Weak hash algorithm

```
$this->uniqueid = md5(uniqid(time()));
```



## Debug-Modus aktiviert

```
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
```



## Empfehlung – Min TLS version enforced

[okhttp](#) library:

```
val spec: ConnectionSpec = ConnectionSpec.Builder(ConnectionSpec.MODERN_TLS)
    .tlsVersions(TlsVersion.TLS_1_2) // Compliant
    .build()
```

## Empfehlung – Strong crypt. hash function

```
// for a password
$hash = password_hash($password, PASSWORD_BCRYPT); // Compliant

// other context
$hash = hash("sha512", $data);
```

## Empfehlung – Deactivate before deployment

Recommended Secure Coding Practices

Do not enable debug features on production servers or applications distributed to end users.

See

- [OWASP Top 10 2021 Category A5](#) - Security Misconfiguration
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure
- [MITRE, CWE-489](#) - Active Debug Code
- [MITRE, CWE-215](#) - Information Exposure Through Debug Information

# Security by Design – Black Friday



# Vergleich DAST – Beispiel eines Scans via OWASP ZAP

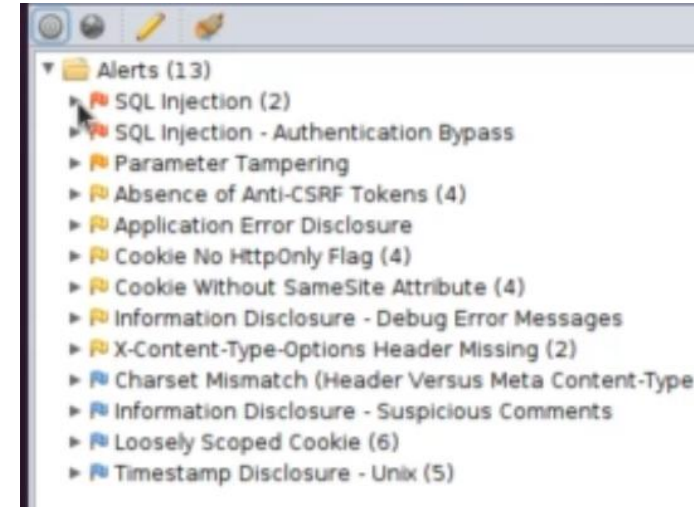
Context: WebGoat Scan Progress

Progress Response Chart

Host: http://localhost:8080

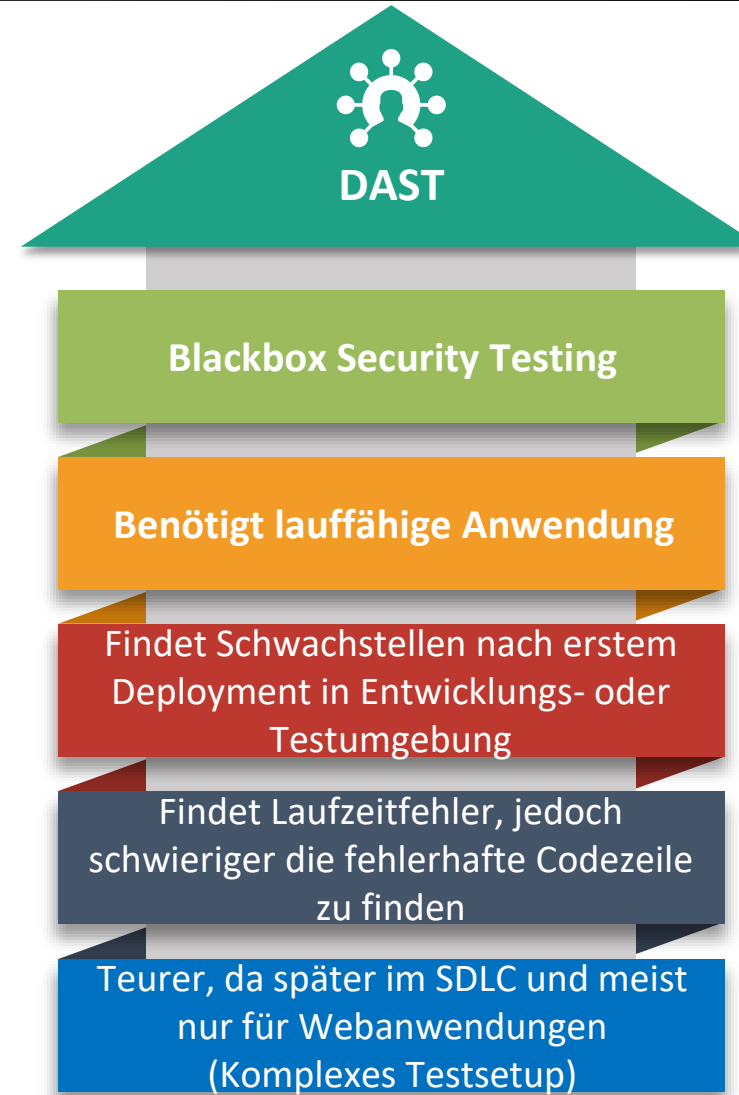
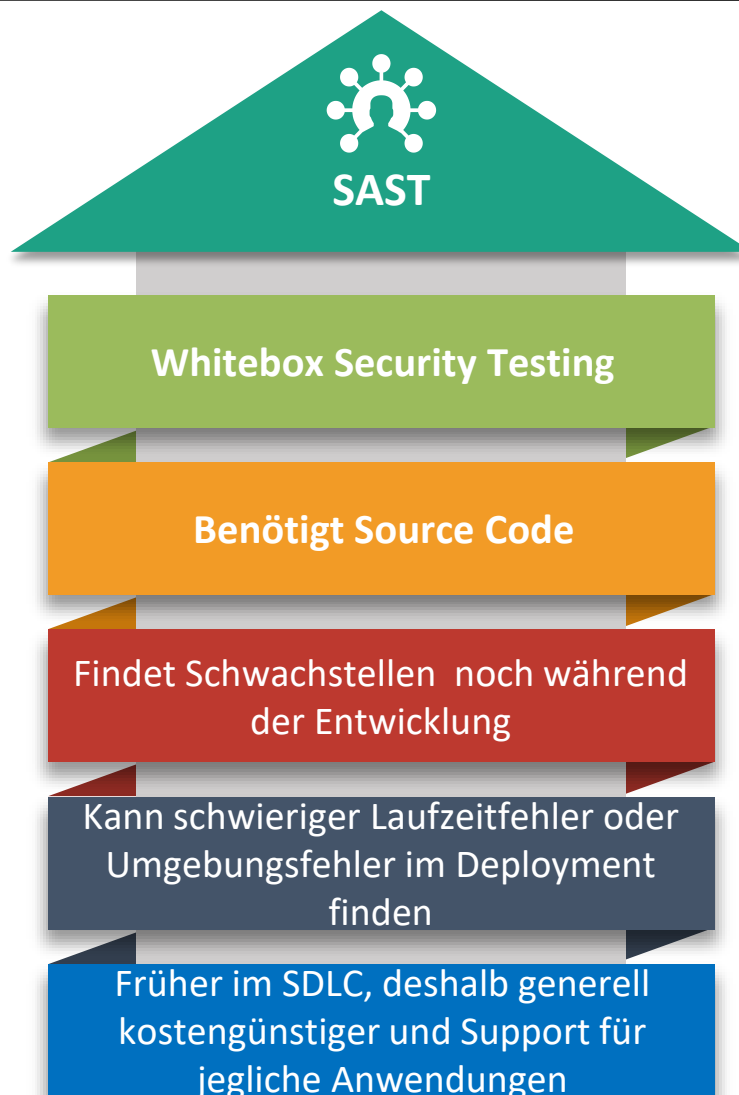
	Strength	Progress	Elapsed	Reqs	Alerts	St...
Analysier			00:00.286	11		
Plugin						
Path Traversal	Medium	<div></div>	00:33.596	732	0	⏏
Remote File Inclusion	Medium			0	0	⏏
Source Code Disclosure - /WEB-INF fol...	Medium			0	0	⏏
External Redirect	Medium			0	0	⏏
Server Side Include	Medium			0	0	⏏
Cross Site Scripting (Reflected)	Medium			0	0	⏏
Cross Site Scripting (Persistent) - Prime	Medium			0	0	⏏
Cross Site Scripting (Persistent) - Spi...	Medium			0	0	⏏
Cross Site Scripting (Persistent)	Medium			0	0	⏏
SQL Injection	Medium			0	0	⏏
Server Side Code Injection	Medium			0	0	⏏
Remote OS Command Injection	Medium			0	0	⏏
Directory Browsing	Medium			0	0	⏏
Buffer Overflow	Medium			0	0	⏏
Format String Error	Medium			0	0	⏏
CRLF Injection	Medium			0	0	⏏
Parameter Tampering	Medium			0	0	⏏
Script Active Scan Rules	Medium			0	0	⏏
Totals			00:33.920	754	0	

Copy to Clipboard Close



- Angriffsziel: WebGoat lokal - unsichere Anwendung von OWASP, um Schwachstellen zu testen, die gängige und beliebte Open-Source-Komponenten verwenden.

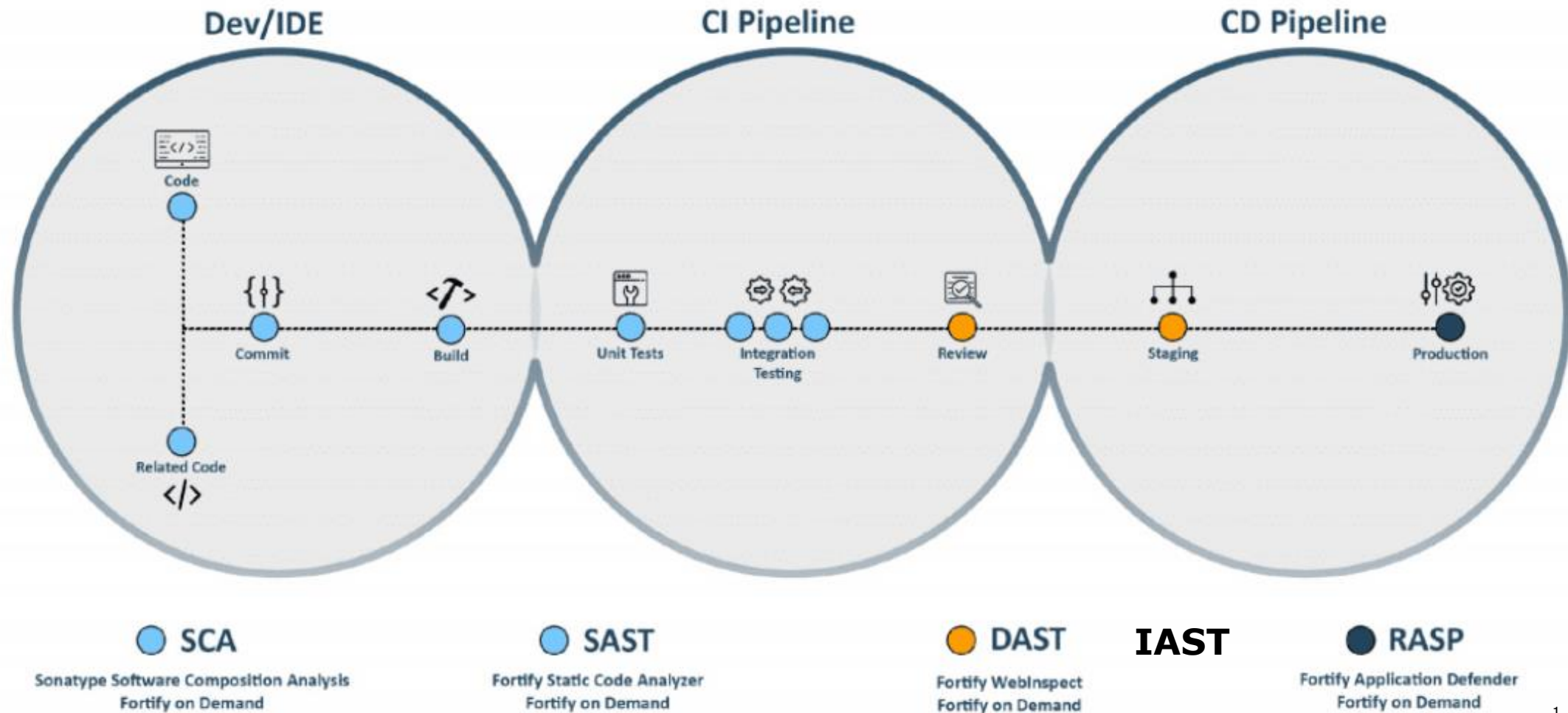
# Vergleich SAST vs DAST



Siehe auch Synopsys<sup>1</sup>

<sup>1</sup> <https://www.synopsys.com/blogs/software-security/sast-vs-dast-difference/>

# Automated Testing in CI/CD



<sup>1</sup> Microfocus

<sup>1</sup> <https://www.microfocus.com/en-us/what-is/sast>

**MAST** für mobile Anwendungen...



# Zusammenfassung

---

- Security Testing ist essentiell, um die Großzahl an Fehlern zu finden, bevor die Anwendung an den Kunden geht und Schwachstellen ausgenutzt werden
- SAST ist ein wichtiges Security-Werkzeug für die statische Codeanalyse im frühen SDLC-Stadium
- Hauptziel ist das Auffinden von Schwachstellen und Bugs in Softwareanwendungen
- Je höher die Softwarequalität, desto sicherer ist die Softwareanwendung
- Klarer Prozess und Automatisierung (CI/CD) sind wichtig
- SAST sollte durch weitere automatisierte Testmethoden wie DAST ergänzt werden
- SAST ist aber nur ein Baustein im DevSecOps-Modell

**Vielen Dank für ihre Aufmerksamkeit!  
Fragen?**