Techniken für sichere agile Software

Ulrich Viefhaus - seism0saurus.de

Ulrich Viefhaus

Softwareentwickler

Java, Spring Boot, Angular, CI-/CD

2016 – 2019	BRANDAD Systems
2019 – heute	Apollo-Optik Holding

Certified Ethical Hacker

Seit September 2019

Certified Application Security Engineer (Java)

Seit März 2021

Agilist

SCRUM, DevOps

Inhalt

Klassisch vs. Agil
Wo liegen die Unterschiede in den Modellen und was bedeutet das für die Sicherheit.

Phase I: Vision etablieren
Wie kann bereits bei der Erstellung einer Vision auf Sicherheits geachtet werden.

Phase II: Planung
Einplanen von Sicherheitsfeatures.

Phase III: Ausführen
Best Practises zu sicherer Softwareentwicklung im (agilen) Umfeld.

Phase IV: Anpassen
Neue Ziele finden, um die Vision umzusetzen.

Abschluss
Zusammenfassung und Buchempfehlungen.

Klassisch vs Agil

Warum wir bei Security umdenken müssen

Wasserfallmodell

Klassische Softwareentwicklung

Anforderungs-analyse
Erstellung eines vollständigen Lastenhefts

Systemspezifi-kation
Definition einer Softwarearchitektur

Programmier-ung und Modultest
Erstellung der eigentlichen Software

Integrations- und Systemtest
Durchführen von Integrations- und Systemtests

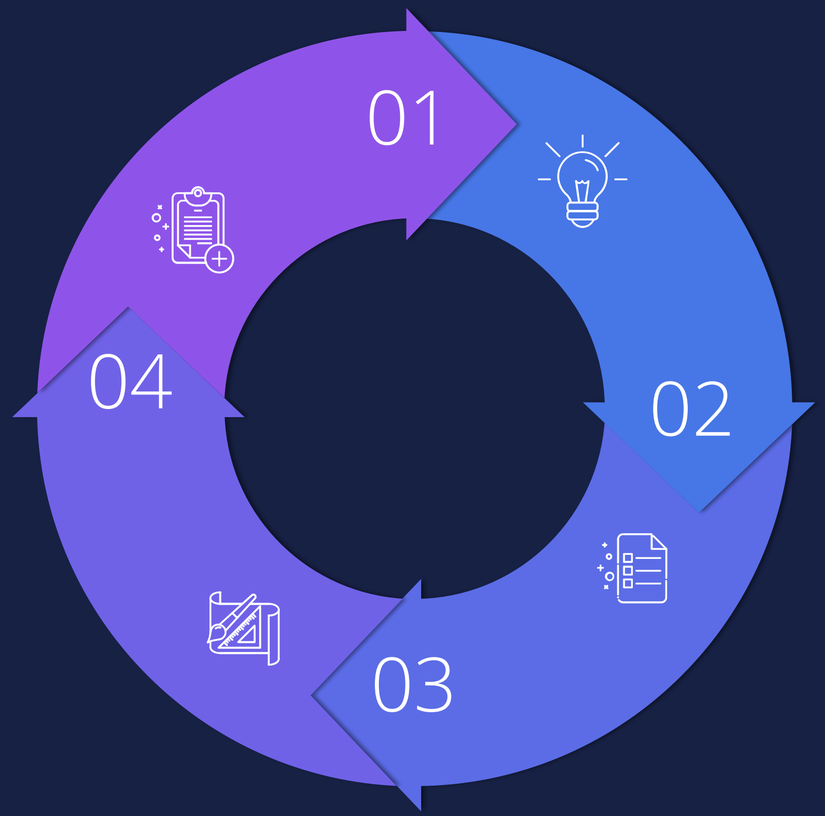
Auslieferung, Einsatz und Wartung
Software wird installiert und gewartet

Das Wasserfallmodell geht davon aus, dass alle Anforderungen an ein System vollständig und korrekt erfasst werden können. Daher können die Phasen von links nach rechts durchlaufen werden. Es stammt ursrpünglich aus dem Bau- und Produktionsgewerbe.

Agiler Entwicklungsprozess

Nicht nur für SCRUM

Agile Prozesse leben von kurzen Entwicklungszyklen. Nach jedem Zyklus sollte ein für sich auslieferbares Produkt stehen. Dabei wird die Vision ständig an sich verändernde Umstände angepasst. Auch das Team past seine Vorgehensweise ständig an, um besser zu werden. Das Produkt kann geändert werden, wenn sich ein Weg als Fehler herausgestellt hat.



Vision etablieren

Eine Vision des Produktes richtet alle Prozesse auf ein Ziel aus.

Planen

Die nächsten Schritte werden danach ausgewählt, ob sie am meisten zur Erreichung des Ziels beitragen.

Ausführen

Die geplanten Schritte werden umgesetzt.

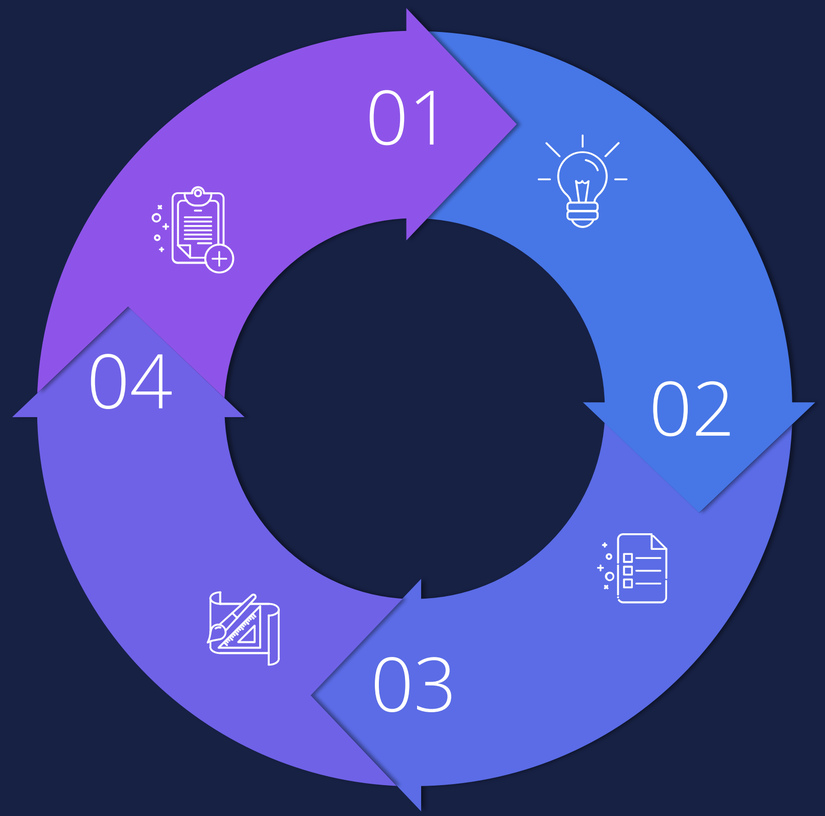
Anpassen

Das Ergebniss wird verifiziert und die Ziele werden an die aktuelle Situation angepasst.

Agiler Entwicklungsprozess

Nicht nur für SCRUM

Agile Prozesse leben von kurzen Entwicklungszyklen. Nach jedem Zyklus sollte ein für sich auslieferbares Produkt stehen. Dabei wird die Vision ständig an sich verändernde Umstände angepasst. Auch das Team past seine Vorgehensweise ständig an, um besser zu werden. Das Produkt kann geändert werden, wenn sich ein Weg als Fehler herausgestellt hat.



Vision etablieren

Eine Vision des Produktes richtet alle Prozesse auf ein Ziel aus.

Planen

Die nächsten Schritte werden danach ausgewählt, ob sie am meisten zur Erreichung des Ziels beitragen.

Ausführen

Die geplanten Schritte werden umgesetzt.

Anpassen

Das Ergebniss wird verifiziert und die Ziele werden an die aktuelle Situation angepasst.

Anti-Personas

Was?

Anti-Personas sind Archetypen von AngreiferInnen, die das Produkt bedrohen. Sie haben individuelle Motive, Hintergründe und Fähigkeiten.

Warum?

Unterstützt bei der Analyse der Bedrohungslage. Man kann sich besser verteidigen, wenn man die Angreifer kennt.

Wie?

Workshops mit SecurityexpertInnen, EntwicklerInnen und Fachleuten des Produkts.

Anti-Personas

Wer bedroht meine Anwendung?

01 Person

Name, Alter, Job...

02 Skills

IT-Ausbildung, Securitytools, Programmiersprachen, Kreativität...

03 Motivation

Ziele, Beweggründe, Weltsicht...

04 Resourcen

Geld, Hardware, Software, Zeit...

Ulf

Skriptkiddy

Geht auf eine weiterführende Schule und hat grundegende IT-Kentnisse. Er weiß, wie er mit Google Securitytools und sie auf Systeme ansetzt. Ulf kann keine eigenen Exploits schreiben. Ihm geht es um Ruhm und Aufmerksamkeit. Dabei geht er wahllos und ungeduldig vor.

Skills: Angriff

40%

Skills: Tarnung

20%

Skills: Kreativität

30%

Resourcen: Zeit

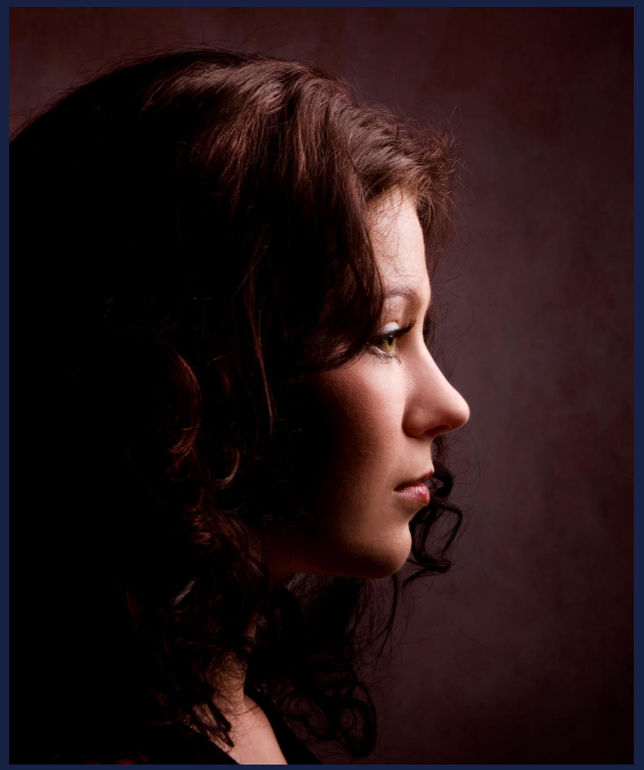
30%

Resourcen: Technik

20%

Resourcen: Finanzen

15%



Mrs. Smith

NSA IT-Expertin

Hat einen sehr guten Abschluss in Informatik und mehrere Jahre Erfahrung im Bereich IT-Security. Über die NSA verfügt sie über Zugriff auf neuste Technologien und Supercomputer. Angriffe werden über Monate vorbereitet und Ziele genau ausgewählt. Über zahlreiche Tarnfirmen können Angriffe verschleiert werden.

Skills: Angriff

95%

Skills: Tarnung

95%

Skills: Kreativität

80%

Resourcen: Zeit

85%

Resourcen: Technik

90%

Resourcen: Finanzen

95%

H3x

Aktivistin

Hat eine IT-Ausbildung und engagiert sich seit mehreren Jahren in der Community und politischen Gruppen. Kennt viele Angriffswerkzeuge und kann sie kombinieren, um ans Ziel zu gelangen. Ihre Gruppe greift Firmen an, die in politische oder ökologische Skandale verwickelt waren, um sie bloßzustellen.

Angriff
60%

Verschleierung
40%

Kreativität
50%

Zeit als Resource
70%

Technische Resourcen
45%

Finanzielle Resourcen
30%

Mark

Mitglied einer Cybercrime Gang

Ist von der C-Programmierung in der IT-Security gelandet und entwickelt für eine Cybercrime Gang Erpressungstrojaner. Ziele sind alle Systeme im Besitz zahlungskräftige Firmen, die bekannte Sicherheitslücken aufweisen. Die Infrastruktur wird im Ausland gehostet. In den letzten zwei Jahren hat die Gang bereits gut verdient.

Angriff
60%

Verschleierung
70%

Kreativität
50%

Zeit als Resource
70%

Technische Resourcen
60%

Finanzielle Resourcen
85%

Johann

Kollege

Hat von Computern wenig Ahnung. Hat BWL studiert und arbeitet seit Jahren ohne besondere Vorkommnisse in der Verwaltung. Da er aus seiner Sicht zu wenig verdient, verkauft er Firmengeheimnisse an die Konkurrenz. Er lässt sich dabei Zeit, um möglichst lange von der Datenquelle zu profitieren.

Angriff
15%

Verschleierung
15%

Kreativität
20%

Zeit als Resource
75%

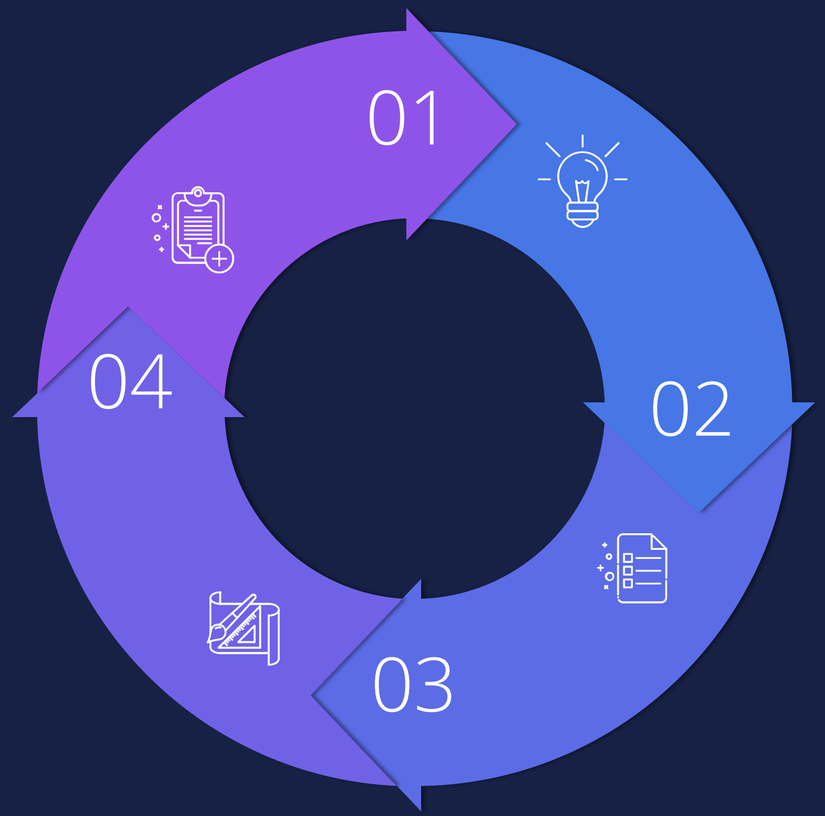
Technische Resourcen
20%

Finanzielle Resourcen
40%

Agiler Entwicklungsprozess

Nicht nur für SCRUM

Agile Prozesse leben von kurzen Entwicklungszyklen. Nach jedem Zyklus sollte ein für sich auslieferbares Produkt stehen. Dabei wird die Vision ständig an sich verändernde Umstände angepasst. Auch das Team past seine Vorgehensweise ständig an, um besser zu werden. Das Produkt kann geändert werden, wenn sich ein Weg als Fehler herausgestellt hat.



Vision etablieren

Eine Vision des Produktes richtet alle Prozesse auf ein Ziel aus.

Planen

Die nächsten Schritte werden danach ausgewählt, ob sie am meisten zur Erreichung des Ziels beitragen.

Ausführen

Die geplanten Schritte werden umgesetzt.

Anpassen

Das Ergebniss wird verifiziert und die Ziele werden an die aktuelle Situation angepasst.

Attacker Stories

Was?

Attacker Stories sind das Gegenstück zu User Stories. Sie beschreiben aus sicht einer Anti-Persona, was diese nicht mit dem System tun darf.

Warum?

Agile Teams sind die Arbeit mit User Stories bereits gewohnt. Sicherheit wird explizit addressiert.

Wie?

Das Team versetzt sich gemeinsam in die Rolle einer Anti-Persona und notiert Ablehnungskriterien.

Als [Anti-Persona] möchte ich [etwas böses tun], um [Schaden anzurichten].

Ablehnungskriterium 1

Die Anti-Persona erhält keinen Zugrif auf…

Ablehnungskriterium 2

Die Anti-Persona kann nicht…

Ablehnungskriterium 3

Das Feature ist durch XY geschützt…

Als Skriptkiddie Ulf möchte ich das Passwort des Admins per Bruteforce knacken, um meinen Namen auf der Startseite zu hinterlassen.

Ablehnungskriterium 1

Der Admin kann nur ein Passwort wählen, dass mindestens 8 Zeichen lang ist und neben Groß- und Kleinbuchstaben auch Zahlen und Sonderzeichen enthält.

Ablehnungskriterium 2

Nach jedem fehlgeschlagenem Anmeldeversuch wird der nächste Versuch um weitere 10 Sekunden verzögert.

Ablehnungskriterium 3

Der Admin wird über Anmeldeversuche per Mail informiert.

Attack Tree

Was?

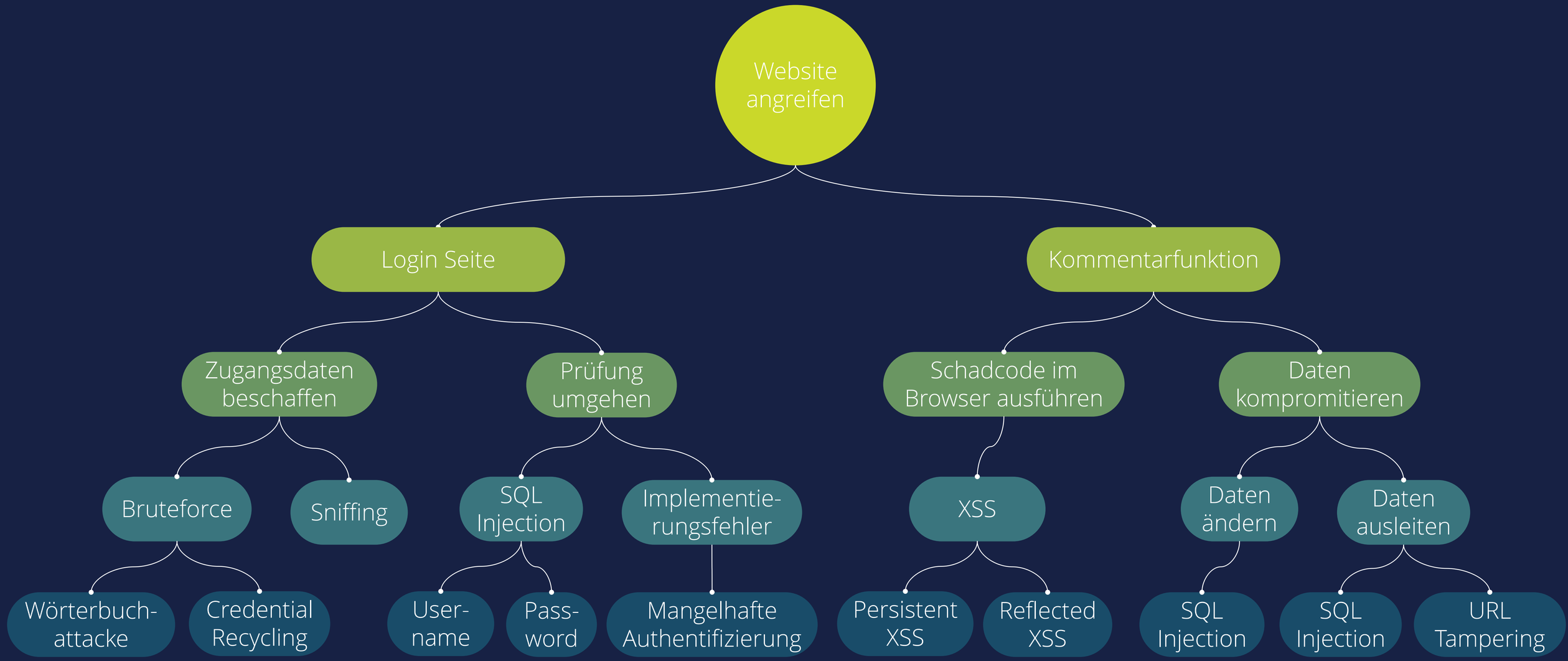
Ein Attack Tree ist eine Möglichkeit, Angriffswege zu sammeln und zu visualisieren. Dabei wird sichtbar an welchen Stellen des Produkts welche Angriffe möglich sind.

Warum?

Unterstützt bei der Analyse der Bedrohungslage. Wenn mögliche Angriffe bekannt sind, können sie nach Risiko priorisiert und verhindert werden.

Wie?

Initialer Workshop mit SecurityexpertInnen, EntwicklerInnen und Fachleuten des Produkts. Später regelmäßiges Aktualisieren in kleinen Workshops.

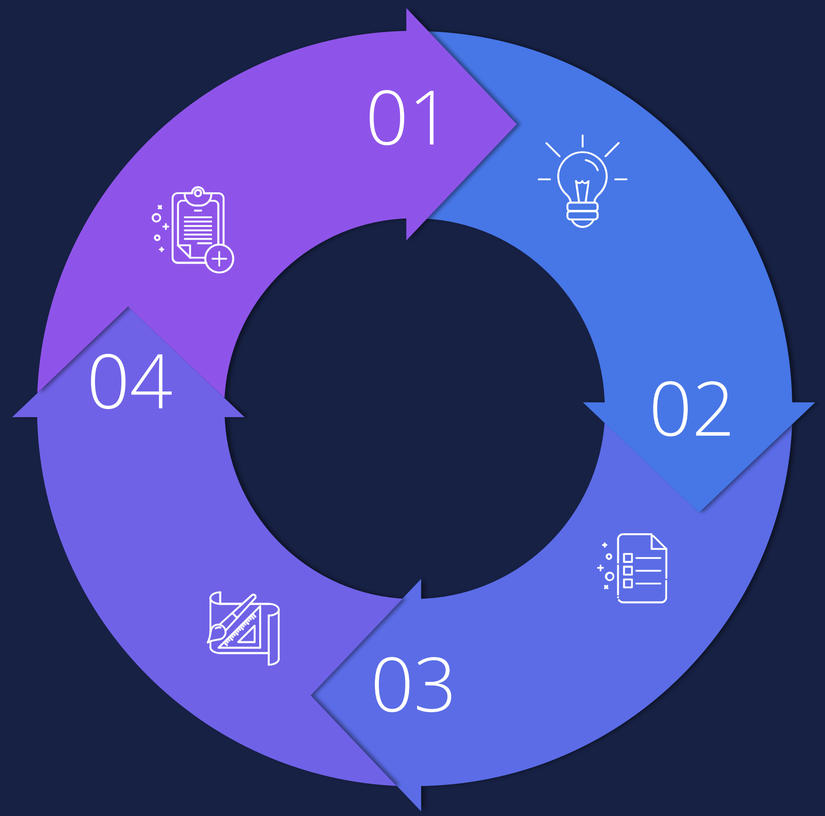


Ausschnitt eines Attack Trees für eine Webseite

Agiler Entwicklungsprozess

Nicht nur für SCRUM

Agile Prozesse leben von kurzen Entwicklungszyklen. Nach jedem Zyklus sollte ein für sich auslieferbares Produkt stehen. Dabei wird die Vision ständig an sich verändernde Umstände angepasst. Auch das Team past seine Vorgehensweise ständig an, um besser zu werden. Das Produkt kann geändert werden, wenn sich ein Weg als Fehler herausgestellt hat.



Vision etablieren

Eine Vision des Produktes richtet alle Prozesse auf ein Ziel aus.

Planen

Die nächsten Schritte werden danach ausgewählt, ob sie am meisten zur Erreichung des Ziels beitragen.

Ausführen

Die geplanten Schritte werden umgesetzt.

Anpassen

Das Ergebniss wird verifiziert und die Ziele werden an die aktuelle Situation angepasst.

Lieferketten absichern

Was?

Nur Libraries verwenden, die keine bekannten Lücken haben. Sichere Docker Images verwenden.

Warum?

Durch Dependencies und Images können große Sicherheitsprobleme in die Anwendung geholt werden.

Wie?

Libraries mit Dependency Scannern testen (OWASP Dependency Check). Aktuelle und minimale Docker Images verwenden. Scanner für Docker Images verwenden (Clair).

Lücken in und durch Libraries

Veracode – State of Software Security Open Source Edition

60%

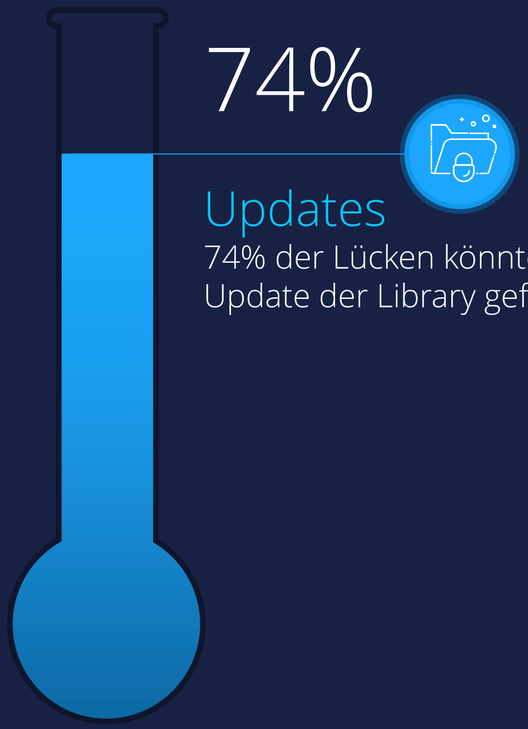
Schwachstellen

Je nach Sprache haben bis zu 60% der untersuchten Bibliotheken Schwachtstellen

70%

Anwendungen

70% der untersuchten Anwendunen enthielten Libraries mit Schwachstellen



74%

Updates

74% der Lücken könnten mit einem Update der Library gefixt werden

Lücken durch Docker Images

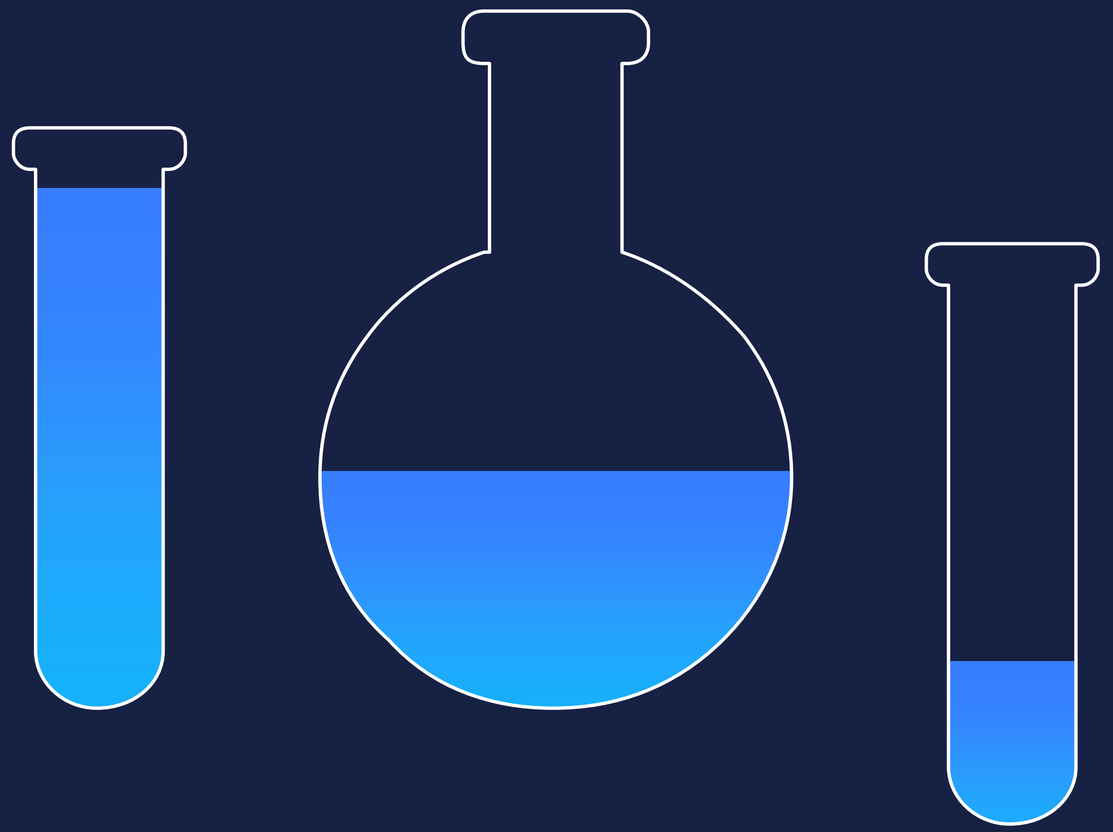
Snyk – The State of Open Source Security Report 2020

44%

Der gescannten Docker Images enthielten Lücken und es gab bereits eine neuere Version

95%

Wenn es von einem Image eine “slim” Variante gibt, enthält diese bis zu 95% weniger Lücken



22%

Der Lücken hätten durch einen einfachen Rebuild gefixt werden können

Secure Coding

Was?

Best Practices für sichere Softwareentwicklung anwenden.

Warum?

Damit keine Programmierfehler oder fehlerhafte Designs Sicherheitslücken in der Anwendung aufreißen.

Wie?

Geeignete Best Practices auswählen.
Fortbilden.
Üben.
Reviewen.

Secure Coding

Templates

Stellt einfach nutzbare und sichere Templates für Verschlüsselung, Validierung und andere sicherheitsrelevante Funktionen bereit.

Input validieren

Jeder Input kann bösartig sein und muss validiert werden.

Default Deny

Standardmäßig sollte das Programm alles ablehnen und Aktionen nur nach expliziter Erlaubniss genehmigen.

Secure Coding

Einfachheit

Komplexe Lösungen erhöhen die Chance auf Fehler und Sicherheitsprobleme.

Sanitize Data

Übergebene Daten müssen für den jeweiligen Einsatzzweck unschädlich gemacht werden.

Defense in Depth

Verwendet mehre Abwehrtechniken

Statische Codeanalyse

Was?

Static Application Security Testing

Warum?

Hilft früh potentielle Bugs und Sicherheitslücken zu finden.
Menschliche TesterInnen können sich auf kritische Stellen konzentrieren.

Wie?

IDE Integration von Scannern.
Dedizierte Server mit CI/CD Integration.

Statische Code-analyse

Einheitliche Codingstandards

Vereinfacht das Verständnis fremden Codes

Erkennung bekannter Schwachstellen

Tools finden automatisch bestimmte Schwachstellen wie z.B. SQL Injection

Identifizieren relevante Bereiche

Anhand der Berichte kann eine Vorauswahl von Bereichen getroffen werden, die auf jeden Fall manuell geprüft werden müssen

Testautomatisierung

Was?

So viele Tests automatisieren, wie es sinnvoll möglich ist.

Warum?

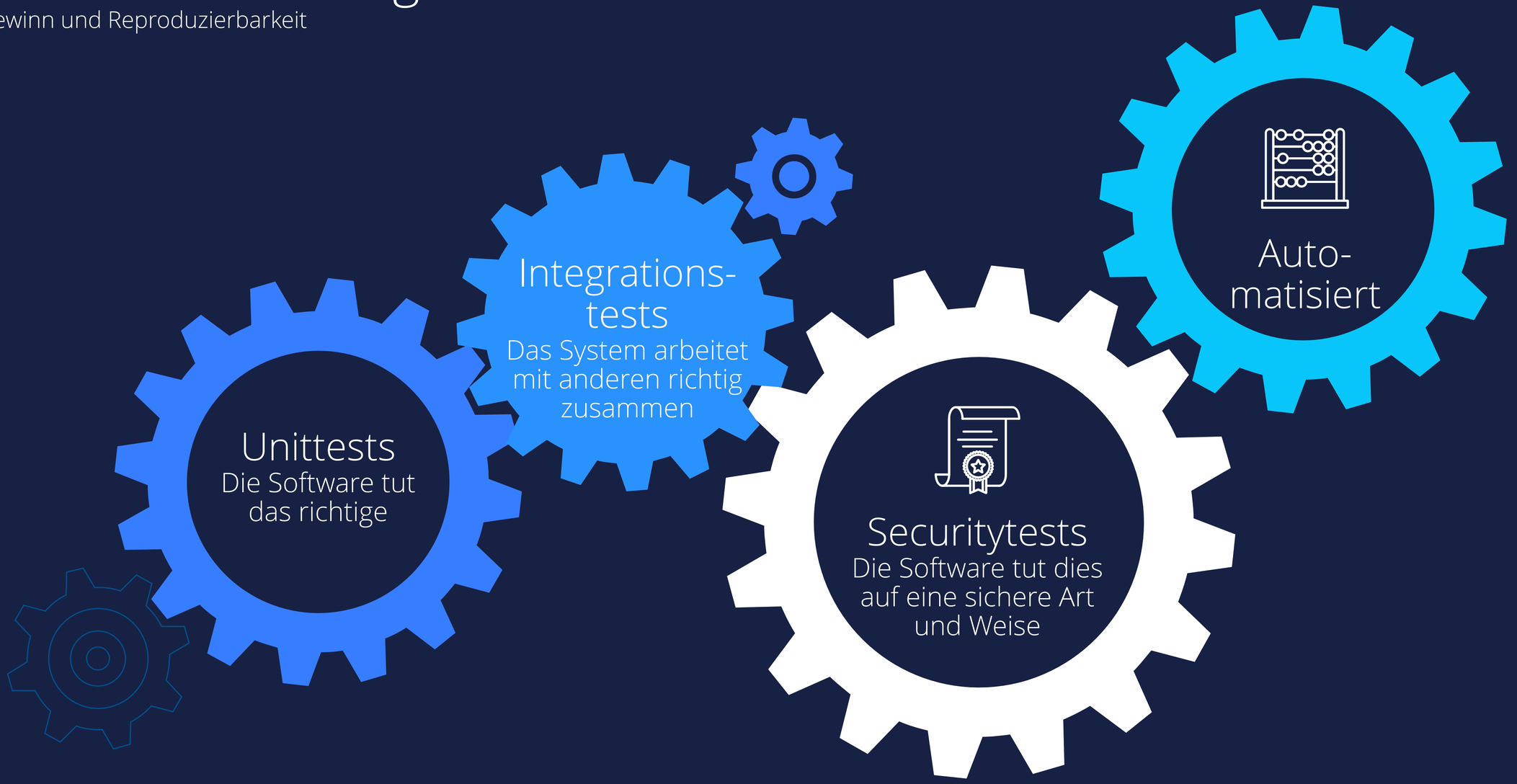
Um schnell Änderungen umsetzten zu können, muss ein Grundgerüst aus Tests sicherstellen, dass alles wie erwartet funktioniert und nicht ausgenutzt werden kann.

Wie?

Neben Unit- und Integrationtests auch Securitytests in die Pipeline einbauen.

Testautomatisierung

Zeitgewinn und Reproduzierbarkeit



CI-/CD-Pipeline und Betriebsumgebung absichern

Was?

Buildserver und Pipeline absichern.
Betriebsumgebungen absichern.

Warum?

Angriffe auf Pipeline und die Server, auf denen die Anwendung läuft verhindern.

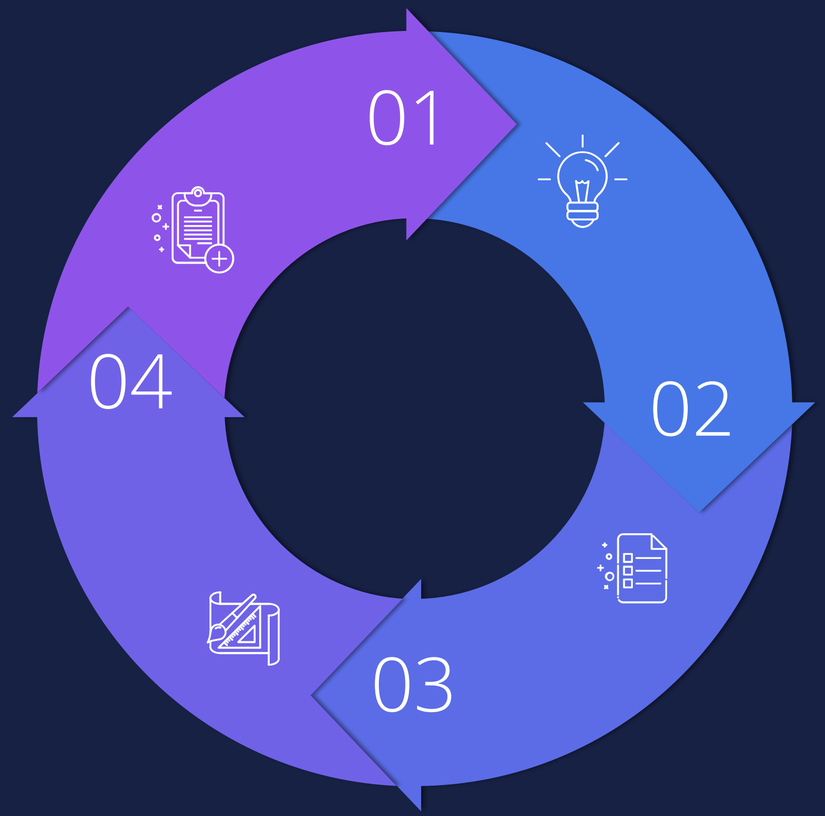
Wie?

Klassisches Server Hardening.
Minimale Zugriffsrechte.
Docker Images signieren.
Zugangsdaten mit geeigneten Tools verwalten.
Umsetzung regelmäßig und automatisch testen.

Agiler Entwicklungsprozess

Nicht nur für SCRUM

Agile Prozesse leben von kurzen Entwicklungszyklen. Nach jedem Zyklus sollte ein für sich auslieferbares Produkt stehen. Dabei wird die Vision ständig an sich verändernde Umstände angepasst. Auch das Team past seine Vorgehensweise ständig an, um besser zu werden. Das Produkt kann geändert werden, wenn sich ein Weg als Fehler herausgestellt hat.



Vision etablieren

Eine Vision des Produktes richtet alle Prozesse auf ein Ziel aus.

Planen

Die nächsten Schritte werden danach ausgewählt, ob sie am meisten zur Erreichung des Ziels beitragen.

Ausführen

Die geplanten Schritte werden umgesetzt.

Anpassen

Das Ergebniss wird verifiziert und die Ziele werden an die aktuelle Situation angepasst.

Externe Penetration Tests

Was?

Penetrationtest durch einen externen Dienstleister.

Warum?

Externe ExpertInnen sind weder befangen noch betriebsblind und bringen neue Perspektiven mit.

Wie?

Beauftragung über entsprechende Unternehmen.

Hackdays

Was?

Ein Team wird zu einer Art Bedrohung geschult und versucht einen Tag lang das System entsprechend anzugreifen.

Warum?

Wissensverteilung. Team lernt, wie AngreiferInnen zu denken.

Wie?

Schulung über interne oder externe SicherheitsexpertInnen. Zeit nehmen.

Abschluss

Wie kann es weitergehen?

Start
Small

Never
Stop

Buch-empfehlungen

Agile Application Security

enabling security in a countinuous delivery pipeline

Site Reliability Engeneering

how google google runs production systems

Securing DevOps

Security in the cloud

Danke

für eure Aufmerksamkeit