DevSecOps

dynamischer, schneller, besser
und vor allem sicherer

swisscom

Impressum

Herausgeberin	Swisscom AG
Autor	Group Security
Redaktion	René Mosbacher, Faktor Journalisten AG, Zürich
Grafik	Agentur Nordjungs, Zürich
Copyright	© März 2019 by Swisscom AG, Group Security, Bern

Inhaltsverzeichnis

1 Vorwort

Heute leben wir unser Leben online. Das Internet kennt keine Geografie und es kennt keine Grenzen. Indem die Menschen dieses weltumspannende Netz schufen, haben sie aber auch eine Büchse der Pandora geöffnet, in der es weder schützende Grenzen noch erkennbare Feinde gibt. Das hat zur Folge, dass wir die die erste Generation in der Geschichte der Menschheit sind, die eher im Cyberspace Opfer eines Verbrechens wird als in der realen Welt. Das ist ein tiefgreifender Wandel.

In dieser neuen Welt sind auch die Unternehmen mit neuen Arten von Risiken konfrontiert. Dies lässt sich etwa daran ablesen, dass Datenabflüsse aus grossen Unternehmen zum Dauerthema in den Medien geworden sind. Ein Grund dafür ist: Heute ist eigentlich jede grössere Firma ein Softwareunternehmen, mit all den typischen Verwundbarkeiten. Schon nur deshalb wird die Online-Sicherheit zur Pflicht. Eigentlich sollte sie ein fixes Traktandum in jeder Geschäftsleitungssitzung sein.

Am Ende gibt es nur zwei Arten von Sicherheitsproblemen: technische und menschliche. Die technischen lassen sich durch Patches lösen – die menschlichen hingegen nicht. Es gibt keinen Patch gegen Dummheit. Und es spielt keine Rolle, wie oft man davor warnt: In der Praxis werden immer irgendwelche Nutzer jedem noch so dubiosen Link folgen. Sie werden immer jeden Anhang öffnen und sie werden ihr Passwort auf Phishing-Seiten preisgeben. Das ist so klar wie Klossbrühe.

Schwachstellen in unseren Systemen sind letztlich nichts anderes als Fehler im Code. Weshalb gibt es Fehler im Code? Weil Programme von Menschen geschrieben werden und Menschen nun mal Fehler machen. Wie aber können wir die Anzahl der Fehler minimieren? Es braucht bessere Sicherheitsdesigns. Es braucht eine bessere Sicherheitskultur. Sicherheit sollte integriert, statt bloss nachträglich um Applikationen und Daten herumdrappiert werden. Sicherheit sollte ein integraler Teil des gesamten Lebenszyklus einer Software sein. Auf keinen Fall darf sie nach der Software-Entwicklung und den Sicherheitsprüfungen enden.

Egal, welche Art von System aufgebaut wird, man sollte immer davon ausgehen, dass es früher oder später eine Sicherheitsverletzung geben wird. Jemand wird eine Sicherheitslücke finden, an die man nicht gedacht hat. Deshalb sollten sich Unternehmen darauf konzentrieren, wie sie Sicherheitsverletzungen feststellen und wie sie darauf reagieren können. Widerstandsfähigkeit ist ein zentraler Faktor.

Wir alle haben nur begrenzte Ressourcen und Budgets, um unsere Netze zu verteidigen. Wenn wir aber verstehen, wie der Feind denkt, können wir unsere Ressourcen dort einsetzen, wo sie am meisten nützen. Wenn wir also Hacker bekämpfen wollen, dann sollten wir mit Hackern zusammenarbeiten. Es ist ganz einfach: Wir brauchen gute Hacker, um die bösen Hacker zu fassen.

Das funktioniert am besten, wenn wir das Finden von Fehlern belohnen. Ich rede hier von den sogenannten Bug-Bounty-Programmen. Solche Programme für gute Hacker einzuführen, erfordert zwar viel Arbeit, es eröffnet uns aber auch eine neue Sicht auf die Sicherheit der Systeme. Mir gefällt, dass viele Grossunternehmen mittlerweile ganz offen solche Programme durchführen. Das vereinfacht es mir, junge, ehrgeizige Hacker auf den legalen Weg zu verweisen: «Du willst also ein Unternehmen hacken? Ok: Geh und hacke Microsoft oder Google oder Apple. Das ist ganz legal, solange du es im Rahmen ihres Belohnungsprogramms tust. Sie bezahlen dich sogar dafür.»

Ich arbeite nun seit fast 30 Jahren im Sicherheitsbereich und es ist nicht absehbar, dass uns die Arbeit hier irgendwann ausgehen wird. Ihnen wünsche ich für die Zukunft viel Erfolg dabei, Systeme sicherer zu machen. Diese Broschüre über DevSecOps zeigt Ihnen anhand von praktischen Tipps, was Sie tun können, um die Welt um uns herum sicherer zu machen.

Vielen Dank für Ihre Arbeit.

Mikko Hypponen
Chief Research Officer, F-Secure

2 Über diese Publikation

2.1 Warum dieses Booklet?

Swisscom ist mit ungefähr 20'000 Mitarbeitenden ein Grossunternehmen mit Wurzeln in der Telekommunikation. Über die Jahre sind weitere Geschäftsfelder hinzugekommen und heute ist Swisscom auch klassischer IT-Dienstleister und das in der Schweiz bedeutendste Unternehmen der Informations- und Kommunikationstechnik (IKT).

2016 wurde entschieden, das Unternehmen nach agilen Prinzipien auszurichten. Den ersten Schritt machte hier die Innovationsabteilung, dann folgten die Entwicklungsabteilungen und mittlerweile ist die ganze Firma mit an Bord. Wie Swisscom DevOps (Begriffserklärung *siehe Seite 56*) konkret anwendet, wird in den Kapiteln *3.3* und *3.4* erläutert.

Im Laufe der Transformation wurde klar, dass die bisherige Unterstützung durch die zentrale Sicherheitsorganisation im agilen Umfeld nur noch beschränkt anwendbar ist. Auf Basis dieser Erkenntnis wurden die Security-Initiativen, die in diesem Booklet beschrieben sind, gestartet oder weiter ausgebaut.

Die vorliegende Publikation bietet eine Sammlung von Best Practices aus dem Dev(Sec)Ops-Umfeld. Sie wurde aus dem Blickwinkel des Enterprise-Umfelds verfasst und dokumentiert auch die eigenen Erfahrungen, die im Unternehmen bisher gemacht wurden. Davon sollen andere Firmen und Fachleute profitieren können, die vor ähnlichen Aufgaben stehen.

> 2016 wurde im Bereich Entwicklung bei Swisscom eine erste Abteilung geschaffen, die ganz nach den DevOps-Prinzipien aufgebaut war. Damit konnte das Unternehmen lernen, wie sich Agilität auf einen Bereich mit mehr als 100 Personen skalieren lässt.
>
> Zudem stellte man fest, dass mit der «Agilisierung» der Entwicklung die Probleme im Betrieb nicht geringer wurden. Zwar wurden mehr und mehr Betriebsleute in die Entwicklung mit einbezogen, die dann auch dafür sorgten, dass wirklich betreibbare Lösungen gebaut wurden. Aber für den eigentlichen 7×24-Stunden-Betrieb all der bestehenden Applikationen stand immer weniger Personal zur Verfügung.
>
> Deshalb wurde die Organisation – ganz nach dem Motto «you build it, you run it» – auf das Jahr 2017 hin nochmals umgestellt. Anstelle der klassischen Unterteilung zwischen Entwicklung und Betrieb wurde die IT neu in einen Software- und einen Infrastrukturbereich gegliedert. Innerhalb dieser Bereiche

fallen aber alle Aufgaben (Entwicklung, Test, Security, Betrieb) in die Verantwortung des zuständigen Teams. So entstand und entsteht ein viel grösseres Verantwortungsbewusstsein für den entwickelten Code.

Mit diesen Veränderungen wurden immer mehr Bereiche in die Arbeitsweise von DevOps eingeführt. Bald zeigte sich aber, dass es ein Skalierungs-Framework braucht. Swisscom entschied, das Scaled Agile Framework (SAFe) einzuführen und überall dort anzuwenden, wo eine Skalierung notwendig ist.

Durch die Anpassungen hat sich die operative Agilität markant erhöht. Gezeigt hat sich indessen auch, dass sich die Vorteile von DevOps nicht nur auf Entwicklung und Betrieb beschränken – im Gegenteil. Im Kern von DevOps stand schon immer das Bestreben, den ganzen Wertschöpfungsteil mit einzubeziehen.

So stand 2018 im Zentrum, die Agilität mehr und mehr von der operativen auf die strategische Ebene zu tragen. Das Business sollte noch stärker involviert und Unternehmensprozesse sollten angepasst werden, wenn sie den agilen Prinzipien widersprachen.

Eine DevOps-Reise braucht Zeit und Ausdauer. Und so wird Swisscom auch die folgenden Jahre weiter daran arbeiten, sich stetig weiterzuentwickeln. Das betrifft auf der einen Seite die technischen Fähigkeiten wie das Decoupling, das Continuous Delivery oder die Testautomatisierung. Auf der anderen Seite geht es aber vor allem auch um die kulturelle Ebene, also darum, mehr selbstorganisierte Teams und eine stärkere Flussoptimierung zu erreichen.

2.2 Zielpublikum

Die Publikation richtet sich an Fachleute, die selbst ein DevOps-Umfeld gestalten oder mitgestalten können respektive wollen. Insbesondere sollen Personen angesprochen werden, die auch in einem hochagilen Umfeld Sicherheit gewährleisten müssen. Sie bekommen die grundlegenden Konzepte erklärt und auch immer wieder kurze Erfahrungsberichte aus dem Umfeld von Swisscom.

Es wird vorausgesetzt, dass die Leserinnen und Leser Grundkenntnisse zu DevOps mitbringen. Ressourcen wie etwa Websites oder Bücher, die es erlauben, Themen vertieft zu betrachten, werden verlinkt oder abschliessend aufgeführt.

2.3 Aufbau der Publikation

Der erste Teil dieses Booklets geht auf die grundlegenden Ideen und Prinzipien von DevOps ein. Dabei werden besonders die folgenden Aspekte beleuchtet:

- Why – warum entscheidet sich eine Organisation für DevOps?
- What – was definiert und qualifiziert DevOps?
- How – wie wird DevOps eingeführt?

Auf dieser Basis baut der zweite Teil auf, der speziell die Security-Aspekte in DevOps erläutert. Die Unterkapitel führen jeweils als Erstes auf, welche Punkte bezüglich People, Process und Technology speziell berücksichtigt werden müssen.

Der zweite Teil ist nach einem vereinfachten Software Development Lifecycle (SDLC, siehe Bild 1) gegliedert. Zuerst behandelt er die übergreifenden Themen wie Training & Awareness (T&A) sowie die Art und Weise, wie Security aufgestellt werden muss, damit sie sich in die Dynamik einer DevOps-Organisation einfügen kann. Diese Themen sind entscheidend, wenn die Sicherheitsorganisation DevOps erfolgreich mitgestalten soll. Danach wird Schritt für Schritt entlang des SDLC untersucht, wie Security behandelt werden sollte, damit sie ihre Wirkung im DevOps-Umfeld entfalten kann.

Es folgt ein Kapitel zur sogenannten Deployment Pipeline, der eigentlichen Fabrik also. Dabei spielt die Automation – wenig überraschend – eine wichtige Rolle. Zum Schluss wird schliesslich erläutert, wie die neuen Ansätze gewinnbringend für die schnelle Detektion und Abwehr von Angriffen genutzt werden können.

An ausgewählten Stellen sind Erfahrungsberichte und Lösungsansätze aus dem Umfeld von Swisscom eingefügt. Sie sollen helfen, beschriebene Prinzipien besser zu verstehen und aus bereits gemachten Fehlern zu lernen. Diese Abschnitte sind jeweils blau hinterlegt.

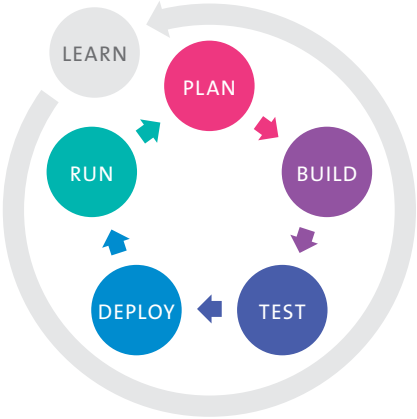

Bild 1 | Ein vereinfachter Software Development Lifecycle.

3 DevOps – Einführung

3.1 Why – warum DevOps?

Faster, better, cheaper, happier – bei DevOps geht es darum, schneller Business Value zu kreieren und robustere Systeme zu erstellen, die zu stärker kundenorientierten Produkten führen. Dabei wird mit geeigneten Formen von Zusammenarbeit und Teamorganisation sichergestellt, dass diese Bestrebungen nicht auf Kosten der Mitarbeiterzufriedenheit gehen. Wer das beherrscht, ist gerüstet für die immer schneller drehende IT-Welt, die sich ferner durch steigende Volatilität, Unsicherheit, Komplexität und Mehrdeutigkeit auszeichnet (Stichwort VUCA: volatility, uncertainty, complexity and ambiguity).

3.2 What – was ist DevOps?

DevOps hat sich aus verschiedenen, bereits bestehenden Bewegungen entwickelt. Dazu gehören etwa das Agile Manifesto, das Lean Movement, das Continuous Delivery Movement und das Toyota Kata [1]. DevOps ist also kein Framework, kein Tool und auch keine Organisationsform. Vielmehr kann DevOps als konsequente Integration und Weiterentwicklung der erwähnten Konzepte betrachtet werden. Der IT-Experte John Willis ist einer der DevOps-Pioniere und prägt die Szene stark. Er hat zur Beschreibung von DevOps das Akronym CALMS eingeführt (erst hiess es nur CAMS – dann kam von Jezz Humble das L dazu). [2] CALMS setzt sich zusammen aus:

- **Culture** – eine funktionsübergreifende Zusammenarbeit mit einer Kultur der geteilten Verantwortung
- **Automation** – so viele Tasks wie nur möglich werden automatisiert
- **Lean** – visuelle Darstellung des Wertflusses und der zu jedem Zeitpunkt fliessenden Arbeitspakete
- **Measurement** – Hypothesen bilden, validieren und lernen [3]
- **Sharing** – Verantwortung und Erfolge teilen, sowohl zwischen Betrieb und Entwicklung als auch über die Teamgrenzen hinweg

[1] The DevOps Handbook von Patrick Debois, Jez Humble, Gene Kim, John Willis

[2] DevOps Culture (Part 1) von John Willis, URL: *itrevolution.com*

[3] Hypothesis-Driven Development von Jeffrey L. Taylor, URL: *drdobbs.com*

3.3 How – was braucht DevOps?

3.3.1 Fertigkeiten

Eine gute Übersicht über die erforderlichen Fertigkeiten gibt das IBM-Referenzmodell «DevOps: The IBM approach – Continuous delivery of software-driven innovation».[4]

Swisscom setzt auf eine adaptierte Version des IBM-Referenzmodells. Das Unternehmen nutzt es als ganzheitliches System, das nicht nur die Entwicklung abdeckt, sondern alle relevanten Fertigkeiten entlang des gesamten Wertstroms. Das Swisscom-Modell beschreibt Fertigkeiten, die für einen DevOps-Betrieb relevant sind, es ist aber kein Ablaufdiagramm.

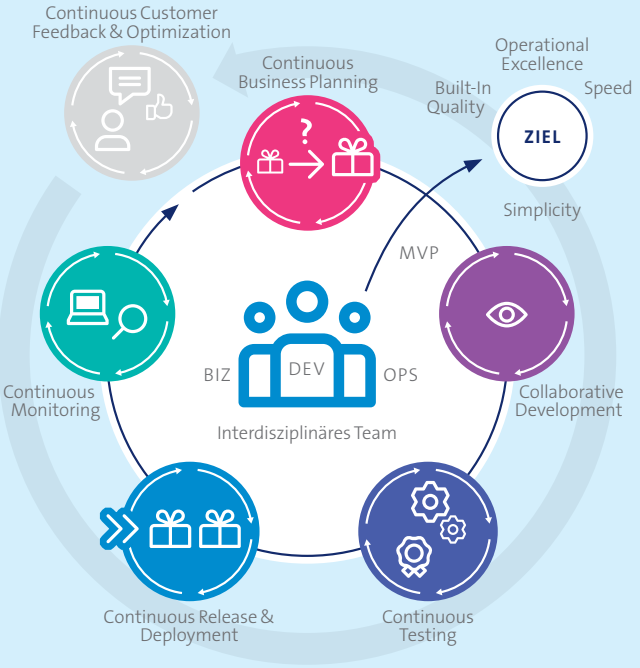

Bild 2 | Swisscom-Modell der Fertigkeiten, die es für DevOps braucht.

[4] DevOps: The IBM approach – Continuous delivery of software-driven innovation (vor allem Seite 6), URL: *developer.ibm.com/community*

Damit DevOps gelingt, müssen im Wesentlichen folgende Voraussetzungen erfüllt sein:

1. **Das Team steht im Zentrum:** Ein interdisziplinäres, voll dediziertes, autonomes Team steht im Zentrum, das auf Augenhöhe kommuniziert *(siehe auch 3.2, Teil «Culture» in CALMS)*.

2. **Continuous Business Planning:** Gemeint ist die Fähigkeit, schnell auf Kundenbedürfnisse zu reagieren und entsprechende Rückmeldungen laufend einzubauen. Dabei steht immer das Kundenbedürfnis im Zentrum und definiert das Produkt.

3. **Collaborative Development:** Dieser Begriff bezeichnet die Fähigkeit, in kurzen Iterationen und interdisziplinären Teams, bestehend aus Vertretern von Business, Entwicklung, Security, Test und Betrieb, kontinuierlich Wert zu generieren. Dies setzt voraus, dass Code kontinuierlich ins Code Repository eingepflegt wird – im besten Fall mehrmals täglich. Dabei wird alles als Code angesehen – das betrifft neben dem eigentlichen Source Code auch Infrastruktur, Deployment Scripts, Monitoring, Testdaten und Weiteres.

4. **Continuous Testing:** Dies umfasst die Fähigkeit, den Code auf reproduzierbare Art und wiederkehrend zu testen. Automatische Tests erlauben es, ihn gleich nach dem Einchecken ins Repository in vielerlei Hinsicht zu untersuchen. So kann früh und kontinuierlich getestet werden, statt erst am Schluss einer Prozesskette, kurz vor Einführung in die Produktion. Damit wird das Testen Teil des Gesamtprozesses. Dies wiederum verbessert die Kosteneffizienz, da Fehler früh im Zyklus und somit günstiger[5] behoben werden können.

5. **Continuous Release & Deployment:** Darunter wird die Fähigkeit verstanden, automatisch einen Build durchzuführen und auf einer beliebigen Umgebung auszuliefern. Das ist notwendig, um das volle Potenzial von automatisierten Tests zu nutzen. Das Code Check-in im Repository erlaubt es, eine Reihe von automatischen Prozessen durchzuführen, vom Build über Tests bis zum Deployment. Dies reduziert die manuellen Tätigkeiten signifikant.

6. **Continuous Monitoring:** Dabei geht es darum, das gesamte System auf allen Plattformen jederzeit zu überwachen. Hierdurch wird gleich nach dem Einbau eines Features ersichtlich, ob es die Leistungsfähigkeit des Gesamtsystems beeinflusst. Dieses frühe Feedback hilft, die Qualität weiter zu verbessern.

7. **Continuous Customer Feedback and Optimization:** So bezeichnet man die Fähigkeit, eine Software, basierend auf Messungen und Feedback sowohl vom Kunden als auch aus dem Betrieb, laufend zu verbessern. Dabei wird das Kundenverhalten sehr bewusst analysiert und es werden verschiedene Kanäle genutzt, um früh und umfassend Feedback zu bekommen.

[5] Figure: Relative Cost of Fixing Defects von Maurice Dawson, URL: *researchgate.net*

3.3.2 Skalierung – DevOps in grossen Organisationen

Alle agilen Ansätze stellen das Team ins Zentrum. Das ist gut so, weil der Wert eines kleinen (idealerweise 5-7 Personen umfassenden), dedizierten Teams gar nicht hoch genug eingeschätzt werden kann. Solche Teams können im besten Fall komplett autonom ihre Services entwickeln und Code ausliefern.

Gerade in grossen Organisationen mit stark gekoppelter IT-Architektur sind aber unabhängige Teams kaum möglich. Deshalb stellt sich die Frage, wie die Skalierung gelöst werden kann. Hier hilft beispielsweise das Scaled Agile Framework (SAFe) weiter. Es ist eine Sammlung von Best Practices, die eine mögliche Skalierung beschreibt.[6]

> Swisscom hat sich entschieden, auf SAFe zurückzugreifen, um «Agile at Scale» praktizieren zu können. In der Realität heisst das: Wenn ein Produkt zu komplex und zu gross ist, um von einem autonomen Team allein getragen zu werden, setzt Swisscom SAFe-Strukturen ein. Das Framework hilft dabei, ein gemeinsames Verständnis zu schaffen, wie «Agile at Scale» funktioniert. Angewendet werden aber nur die Elemente und Rollen von SAFe, die effektiv einen Mehrwert bieten. Einige Elemente übernimmt Swisscom, so wie sie sind; andere werden an die Bedürfnisse angepasst, wie etwa die Rolle der Security.

3.4 Impact – was bewirkt DevOps?

Wer DevOps in seiner Organisation einführen will, sollte sich im Klaren sein, dass er damit die gesamte Wertschöpfungskette beeinflusst. Im Folgenden wird auf die relevanten Aspekte eingegangen.

Beim Kunden

Eigentlich ist es den Kunden egal, wie die Services produziert werden. Sie haben den berechtigten Anspruch, dass ihre Bedürfnisse schnell und zuverlässig erfüllt werden. Ihre Systeme sollen verfügbar sein, fehlerfrei und sicher funktionieren. In unserer sich schnell verändernden Welt ist dies mit klassischen Modellen immer schwieriger zu leisten. Mit DevOps hingegen erhält der Kunde besser auf seine Bedürfnisse abgestimmte Lösungen, die schneller implementiert werden können.

Im eigenen Unternehmen

Im State of DevOps Report[7] wird seit einigen Jahren erhoben, wie DevOps-Praktiken auf Unternehmen einwirken. Dabei konnte klar nachgewiesen werden, dass Unternehmen mit einer hohen DevOps-Reife wirtschaftlich erfolgreicher sind. Wichtig ist in diesem

[6] SAFe. Scaled Agile Inc., URL: *scaledagileframework.com*

Zusammenhang aber, darauf hinzuweisen, dass DevOps kein Kostensparprogramm sein kann. Eine höhere Effizienz ist das Resultat von gut eingeführten DevOps-Praktiken – sie kann aber nicht das Ziel sein.

Im organisatorischen Aufbau
Agilität und DevOps setzen stark auf autonome Teams. Entscheide werden möglichst dezentral gefällt. Ein solches Umfeld erfordert natürlich spezielle Führungsqualitäten. Wichtig ist, dass die Teams divers und «cross-funktional» sind. Dies führt erfahrungsgemäss zu besseren Lösungen.

Die Zusammensetzung der Teams soll möglichst konstant sein. Das begünstigt gemeinsames und gegenseitiges Lernen. Daraus ergibt sich auch ein wichtiger Paradigmenwechsel für die Prozesse: Die Arbeit soll zum Team gebracht werden und nicht das Team zur Arbeit. Das wiederum hat zur Folge, dass viel stärker an der kontinuierlichen Entwicklung eines Produkts gearbeitet wird als an in sich geschlossenen Projekten. Wichtig ist, dass Entscheide möglichst dort getroffen werden, wo auch die Konsequenzen entstehen. Auch deshalb sind interdisziplinäre Teams so wichtig.

DevOps soll zudem die organisatorischen Silos reduzieren. So erhält der Wertefluss ein viel grösseres Gewicht als die organisatorische Zugehörigkeit. Deshalb wird auch die Ablauforganisation viel wichtiger als die Aufbauorganisation.

> Die Umsetzung von DevOps hat bei Swisscom besonders in der IT zu organisatorischen Anpassungen geführt. Zuvor waren Entwicklung und Applikationsbetrieb in separaten Abteilungen mit unterschiedlichen Zielen angesiedelt. «Dev» wurde gemessen an der Anzahl der gelieferten Funktionen. «Ops» hingegen wurde an der Stabilität der Systeme gemessen und hat deshalb versucht, die Anzahl der Änderungen klein zu halten. Um diesen Zielkonflikt lösen und die gemeinsame Verantwortung stärken zu können, wurden diese Abteilungen zusammengelegt. Nun ist jedes Team nicht nur für die Entwicklung, sondern auch für den Betrieb seiner Applikationen verantwortlich.

Im Team und beim Individuum
DevOps setzt den Schwerpunkt stark beim Team und bei der Teamverantwortlichkeit. Ziele werden im Team beschlossen, und die Teammitglieder unterstützen sich gegenseitig dabei, sie zu erreichen. Im Vordergrund stehen die Ziele des Teams und nicht die des einzelnen Mitglieds.

[7] State Of DevOps Report von Puppet + Splunk, URL: *puppet.com*

DevOps verkürzt die Zyklen und intensiviert die Zusammenarbeit im Team massiv. Deshalb muss die direkte Kommunikation zwischen allen Beteiligten bewusst gefördert werden. Konzepte dazu liefert etwa das Agile Manifesto[8].

In seinem Buch «Drive» schreibt Daniel Pink[9], dass Motivation vor allem von den Faktoren Autonomy, Mastery und Purpose abhängt. DevOps wird diesen sehr gut gerecht:

- Autonomy entsteht durch dezentrales Entscheiden und die Eigenständigkeit des Teams.
- Mastery wird durch das notwendige kontinuierliche Lernen gefördert.
- Purpose, also der Sinn und Zweck, wird klarer, weil die Verantwortlichkeiten und die Auswirkungen der Arbeit sehr direkt erlebt werden können. Man ist nicht nur ein Rädchen in einer grossen Maschine, sondern verantwortet ein konkretes Produkt bis in die Produktion. Das schätzen wohl die meisten Mitarbeitenden.

Der Entwicklungsprozess wird mit DevOps generell kürzer und schneller. Dadurch verkürzen sich die Feedback-Loops. Dies wiederum führt dazu, dass Information und Wissen und somit auch die Verantwortung schneller an den Prozessstart – also zur Entwicklung zurückfliessen. Dieser Effekt wird auch als «Shift Left» bezeichnet.[10]
Mit DevOps arbeiten die Mitglieder immer möglichst dediziert für ihr Team. Das hilft, Prioritätskonflikte zu vermeiden, eine gesunde Teamkultur zu entwickeln und das gemeinsame Lernen zu fördern. Zudem gilt, was schon zum Einfluss von DevOps auf den organisatorischen Aufbau beschrieben wurde: Die Arbeit des Teams richtet sich eher auf die kontinuierliche Entwicklung von Produkten aus als auf die Abwicklung von einzelnen Projekten.

[8] Manifesto for Agile Software Development by Kent Beck et al., URL: *agilemanifesto.org*

[9] Drive von Daniel Pink, URL: *danpink.com*

[10] Shifting Left – Approach and Practices von Paul Bahrs, URL: *slideshare.net*

4 DevSecOps – Security im DevOps-Kontext

Security ist per Definition [11] schon stark in DevOps abgebildet. Deshalb erscheint es auf den ersten Blick unnötig, ihr mit einem weiteren Kunstterm wie DevSecOps zusätzliches Gewicht zu verleihen. Die Erfahrung zeigt aber, dass im Zuge von «Shift Left» immer wieder Themen vernachlässigt werden – gerade die Sicherheit ist hier besonders gefährdet. Im Wesentlichen ergeben sich durch DevOps drei wichtige Herausforderungen:

- **Wegfall von fixen Prozess-Tollgates respektive Meilensteinen**
  Was in iterativen Projektführungsprozessen bereits Einzug gehalten hat, gilt bei DevOps noch verstärkt: Einzelne Meilensteine, die eine manuelle Freigabe bedingen, sind bei DevOps kontraproduktiv, da sie die Durchlaufzeit von der Idee bis zur Produktion verlängern. Deshalb können bei DevOps auch keine Security-Reviews vor der «Inproduktionssetzung» verankert werden, wie dies in klassisch geführten Projekten häufig der Fall ist.

- **Wegfall der Gewaltentrennung zwischen System und Applikation (Segregation of Duties)**
  Durch die Diversifizierung der Skills und die damit einhergehende End-to-End-Verantwortung innerhalb der DevOps-Teams erhalten alle Teammitglieder die gleichen Berechtigungen. Die Konsequenz daraus ist, dass viel mehr einzelne Personen Zugriff auf wertvolle Daten erhalten.

- **Unsicherheit bezüglich der Verantwortlichkeiten für Risiken**
  Was in streng hierarchisch aufgestellten Organisationen über die Linienfunktion verankert wurde, ist in einer DevOps-Organisation an die Rolle gebunden. Risiken sollten einen Eigner haben, der sie trägt oder mitigiert.

Um dem Rechnung zu tragen, haben die Security-Fachleute den Kunstbegriff DevSecOps geschaffen. Er soll dabei helfen, Sicherheit im Umfeld von DevOps näher zu beleuchten. Er illustriert aber auch, dass alle Aktivitäten rund um den Produktlebenszyklus stark miteinander verknüpft sein müssen. Für die Security bedeutet dies zunächst: Nur wenn die mit ihr zusammenhängenden Aktivitäten iterativ anwendbar sind, eignen sie sich auch für DevOps.

[11] The DevOps Handbook von Patrick Debois, Jez Humble, Gene Kim, John Willis

Die Rahmenbedingungen für die Security lassen sich anhand der drei Säulen People, Process und Technology darstellen:

People	Process	Technology
Personen, die DevOps leben, werden dazu gebracht, intrinsisch auch die Sicherheit eines Produkts zu berücksichtigen	Prozesse unterstützen die sichere Entwicklung eines Produkts In mutierenden Organisationen müssen die relevanten Ansprechpartner hinsichtlich Sicherheit einfach zu finden sein	«Security as Code» wird konsequent umgesetzt – damit wird sie nachvollziehbar, reproduzierbar und mutierbar

Diese drei Säulen werden am Anfang jedes Unterkapitels wieder aufgegriffen. Damit sollen jeweils die wichtigsten Punkte im jeweiligen Aktivitätsbereich umschrieben werden. Die drei Säulen lassen sich dabei wie folgt definieren:

- **People:** Aktivitäten und Massnahmen, die auf die im Unternehmen gelebte DevOps-Kultur einzahlen
- **Process:** Alles, was es rund um Prozesse und die organisatorische Aufstellung braucht, damit die Sicherheit in DevOps ihren gebührenden Platz erhält
- **Technology:** Der Kern von DevOps ist die Automatisierung von sich wiederholenden Abläufen – hier geht es um den Umgang damit und die Chancen, die sich daraus ergeben

In einer Produktbereitstellung nach DevOps ist es schwierig, einzelne Aktivitäten scharf einer der drei Säulen zuzuordnen. Deshalb werden die Säulen im Weiteren auch immer gemeinsam dargestellt.

Die folgenden Unterkapitel beschreiben die essenziellen Sicherheitsfähigkeiten (Capabilities), die in einer DevOps-Organisation mit Konzerndimension angestrebt werden sollten. Ebenso zeigen sie, wo zwischen den verschiedenen Security-Aktivitäten Synergiepotenziale bestehen. Das hilft, den Zusatzaufwand und die zusätzliche Reibung auf das Notwendige zu minimieren. Die Abfolge der Unterkapitel orientiert sich dabei an den verschiedenen Aktivitäten des vereinfachten Software Development Lifecycle *(siehe Kapitel 2.3)*, also an:

- Training und Organisation
- Planung und Implementierung eines Produkts/Features
- Deployment und Betrieb

4.1 Training & Awareness

In diesem Kapitel zentral		
People	**Process**	**Technology**
Wissen, wo Security berücksichtigt werden muss	Training & Awareness ist für die relevanten Stakeholder Pflicht	Skalierbarkeit und Nachhaltigkeit der Awareness- und Trainingsmethode(n)

Wenn es darum geht, Entscheide, Priorisierungen oder andere steuernde Aktivitäten fachlich zu unterfüttern, ist es wichtig, dass alle Beteiligten Security Training & Awareness (T&A) erhalten. In einem DevOps-Umfeld muss jedem Teammitglied klar sein, wo ein Fehlverhalten oder ein systemischer Fehler grosse Konsequenzen haben kann. Ebenso muss allen bewusst sein, welches die grundlegenden zu schützenden Assets (zum Beispiel Daten) überhaupt sind.

Eine gute Awareness hilft, ein Gefühl für unsichere Dinge zu entwickeln. Das ist sehr nützlich, weil Sicherheit im Vergleich zu anderen Anforderungen oft weniger sichtbar und spürbar ist. Ein gesundes Bauchgefühl erleichtert es den Beteiligten, an die Konsequenzen ihres Handelns zu denken.

4.1.1 Für wen?

Grundsätzlich sind in einer Organisation alle Mitglieder für die Sicherheit mitverantwortlich. Deshalb braucht es ein Training-&-Awareness-Programm, das breites Grundwissen vermittelt und somit für alle Mitarbeitenden relevant ist (beispielsweise im sicheren Umgang mit Daten). Zusätzlich muss ein solches Programm auch rollenspezifisches Wissen vermitteln. Das erlaubt es Mitarbeitenden, sich in ihrem Kompetenzbereich weiterzubilden, wenn nötig bis auf Expertenniveau.

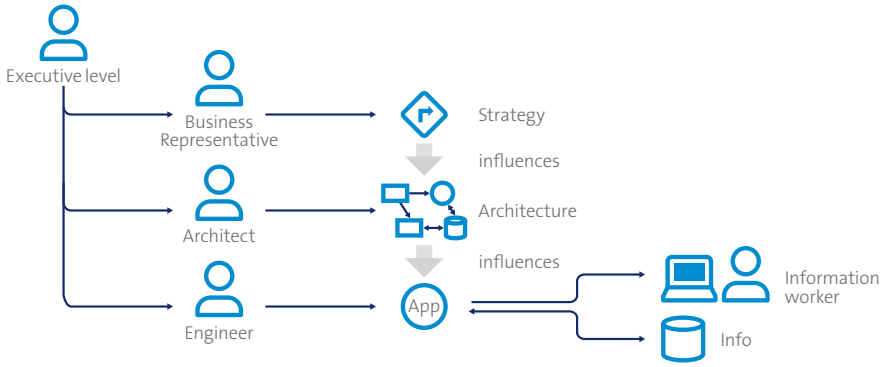

Bild 3 | Involvierte Rollen im DevOps-Grundwissen und allgemeine Awareness

Jedes Mitglied der Organisation muss ein minimales Grundwissen in Security und Datenschutz haben. Deshalb sollen Basisschulungen hierzu angeboten werden, etwa in Form von E-Learning. Zur Zielgruppe solcher Schulungen gehören alle in DevOps involvierten Rollen, also:

- **Engineers** – kümmern sich um die technische Bereitstellung von Produkten
- **Architects** – sind verantwortlich für korrekte Interaktionen und Prozesse innerhalb von Produkten und an deren Schnittstellen nach aussen
- **Business Representatives** – priorisieren inhaltliche Aspekte des Produkts und beeinflussen somit die Entwicklungsrichtung
- **Executives** – verantworten und steuern übergreifende Aspekte im Unternehmen
- **Information Workers** – die (internen) Nutzniesser von bereitgestellten Produkten

Rollenspezifische Training-&-Awareness-Einheiten
Rollenspezifische Schulungseinheiten für Training & Awareness helfen den Teammitgliedern dabei, die relevanten Themen zu Sicherheit und Datenschutz zu erkennen und ihr Wissen in der Praxis grob anzuwenden. Hier soll also ein Grundverständnis vermittelt werden, das es zum Beispiel einem Architekten erlaubt, die nötigen Sicherheitskomponenten einzuplanen. Oder es soll etwa dem Business Representative helfen, die nötigen Freiräume und Ressourcen für Sicherheit und Korrekturen vorzusehen. Damit das gelingt, müssen die Trainingsinhalte auf die jeweilige Rolle zugeschnitten sein.

Training & Awareness für spezialisierte Bereiche
Sehr tiefgreifende, spezifische Schulungen braucht es für Engineering und Development. Das Ziel muss sein, in den DevOps-Teams mindestens eine spezialisierte Person *(siehe 4.2.2 → Security Champion)* zu haben, die über erweitertes Wissen in Sicherheit und Datenschutz verfügt. Sie soll als Bindeglied zur zentralen Sicherheitsorganisation wirken. Spezifische Trainings sind daher vor allem für die Engineers sinnvoll.

4.1.2 Training-&-Awareness-Konzept

Das Konzept für Training & Awareness ruht auf folgenden drei Hauptpfeilern:

1. **Kontinuierliche Schulung** mit kurzen Lerneinheiten (Micro Learnings), die teils auch sich wiederholende Inhalte bieten. Lange Lerneinheiten werden, wo immer möglich, in mehrere kürzere aufgeteilt. Das erleichtert das Aufnehmen der Inhalte und steigert die Akzeptanz unter den Mitarbeitenden.
2. **Awareness-Aktionen** machen die Mitarbeitenden immer wieder auf mögliche Gefahren aufmerksam (etwa Fake-Phishing-Kampagnen). Damit soll ein «Grundrauschen» erzeugt werden, das die Security Awareness dauerhaft in den Köpfen verankert.
3. **Spezielle Kommunikationskanäle** helfen, Sicherheitsthemen und vor allem die Training-&-Awareness-Angebote rollenspezifisch zu verbreiten.

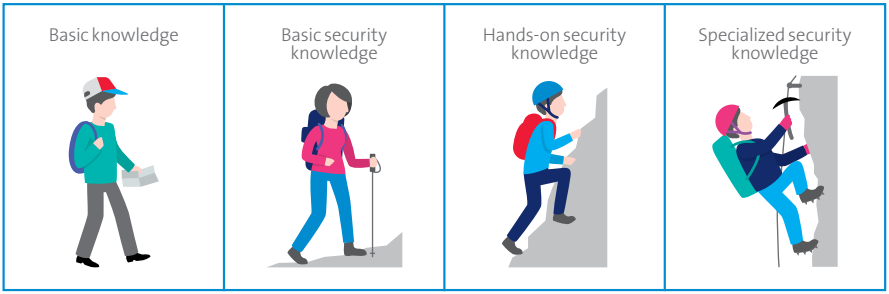

Bild 4 | Die vier Maskottchen der Swisscom-Zertifizierung im Bereich DevSecOps: Trekker, Hiker, Mountaineer, Alpinist

Inhaltlich sieht das Training-&-Awareness-Konzept bei Swisscom vier Stufen vor. Mit jeder Stufe werden die Inhalte spezialisierter, womit sich auch das Zielpublikum verkleinert. Die Teilnehmenden können auf jeder Stufe ein Swisscom-internes Zertifikat erlangen. Die erste Stufe bilden die sogenannten **Trekker-Module**. Sie dienen primär dazu, intern verfügbare und zentral verwaltete Services rund um DevOps bekannt zu machen. Vermittelt werden auch grundlegende DevOps-Ansätze wie Source Code Management, Continuous Integration/Continuous Deployment. Die Teilnehmenden lernen zudem das gemeinsame Vokabular, das sich dann durch die folgenden Stufen zieht. Am Ende sollen selbst Einsteiger die nötigen Informationen zur Hand haben und sichere DevOps-Services kennen, mit denen sie rasch produktiv arbeiten können. Auf das Vermitteln von spezifischen Security-Inhalten wird bewusst verzichtet.

In den **Hiker-Modulen** werden zum einen Security-Services vorgestellt. Zum anderen erhalten die Teilnehmenden eine Einführung in die wichtigsten Verwundbarkeiten wie Cross-Site-Scripting oder SQL Injection. Zur Priorisierung der Verwundbarkeitsklassen wurden Daten aus dem Bug-Bounty-Programm von Swisscom *(siehe Kapitel 4.6)* herangezogen, alternativ bietet das Open Web Application Security Project (OWASP) das Top-10-Projekt an. Diese beiden ersten Stufen sind einfach und in automatisierter Form mittels Multiple-Choice-Quiz kontrollierbar.

Die **Mountaineer-Module** werden von Personen besucht, die wissen wollen, wie man mit den im Hiker-Modul kennengelernten Verwundbarkeiten in der Praxis umgeht. Hierfür sollen Teilnehmende, die sonst Systeme oder Produkte bauen, für ein bis zwei Tage die Seite wechseln können, um die Sicht des Angreifers kennenzulernen. Von den Teilnehmenden, die dieses Zertifikat erlangen wollen, wird erwartet, dass die Erkenntnisse im «Alltagsjob»

angewendet werden können. Die gemachten Erfahrungen sollen auch mit der DevOps-Community geteilt werden.

Auf der Stufe **«Alpinist»** wird ein hoher Spezialisierungsgrad vorausgesetzt. Hier wird ein externes, am Markt anerkanntes Zertifikat angestrebt, beispielsweise der Certified Secure Software Lifecycle Professional (CSSLP). Die Absolventen sollen am Ende die organisationsinterne Sicherheitskultur mit beeinflussen können.

4.2 Organisatorische Skalierung

In diesem Kapitel zentral		
People	**Process**	**Technology**
Kommunikationswege kurz halten Matrixkommunikation fördern	Verantwortlichkeiten klar dokumentieren und kommunizieren	Übersicht über die Organisation und die verschiedenen Rollen zu jedem Zeitpunkt ermöglichen

Security als Haltung in der Organisation zu verankern, bleibt auch im Zeitalter von DevOps eine Herausforderung. Im Gegensatz zu anderen, nichtfunktionalen Attributen wie Performance oder Betriebsstabilität ist ein Mangel an Security bei einem Produkt nicht unmittelbar spürbar. Wir kennen das aus dem Alltag: Wenn eine Applikation träge reagiert, werden die Kunden recht schnell aktiv. Fehlt es aber an Sicherheit, bemerken sie das – wenn überhaupt – erst viel später, wenn bereits etwas schiefgelaufen ist. Es gilt also, der Sicherheit die Visibilität zu verleihen, die sie verdient. Gehen Daten verloren oder werden sie manipuliert, sind die Auswirkungen nicht selten fatal.

Klassischerweise stehen in der IT pro 100 Entwickler rund 10 Betreiber, aber nur 1 Sicherheitsspezialist zur Verfügung. Eine Eins-zu-eins-Betreuung der Mitarbeitenden in Sicherheitsfragen ist also von Anfang an unrealistisch. Das wird sich auch in einer DevOps-Organisation nicht ändern. Deshalb ist es wichtig, dass alle in DevOps involvierten Parteien (Business, Entwicklung, Test, Betrieb; *siehe auch ‹Collaborative Development› in Kapitel 3.3*) vermehrt Sicherheitskompetenz aufbauen und dabei durch die zentrale Sicherheitsorganisation unterstützt werden. Hier hilft auch die sogenannte Matrixkommunikation. Darunter versteht man die Vernetzung von Personen, die innerhalb der verschiedenen Teams die gleiche Rolle einnehmen. So können sie sich ständig quer über das Unternehmen mit ihren Fachkollegen austauschen und ihr Wissen ins eigene Team einbringen.

Wie bereits in *Kapitel 2.1* beschrieben, nutzt Swisscom das Scaled Agile Framework (SAFe) in angepasster Form. Einer der angepassten Bereiche ist die Integration von Security in die Organisation – sie ist für Swisscom-Zwecke im Basis-Framework zu wenig ausgestaltet. In diesem Kapitel wird näher betrachtet, wo und warum die Umsetzung der Security bei Swisscom vom SAFe abweicht. Gezeigt wird auch, wie Security organisatorisch eingebettet ist.

4.2.1 Skill-Diversität in der DevOps-Organisation

Durch den «Shift Left»-Ansatz werden viele Verantwortlichkeiten zum DevOps-Team hin verschoben. Das bedeutet, dass das Team neben dem funktionalen Verhalten des Produkts auch viele andere qualitative Aspekte im Auge behalten muss. Die Bandbreite der zu bearbeitenden Themen verbreitert sich also wesentlich. Es braucht daher eine ganzheitliche Sicht auf das Produkt und die ist nur möglich, wenn das Team heterogen und diversifiziert zusammengestellt ist.

Gibt es im Team Spezialisten aus vielen unterschiedlichen Gebieten, hat das einzelne Mitglied weniger Probleme, relevante Bedrohungsszenarien zu formulieren. Es wird auch besser verstehen, warum bestimmte, nicht unmittelbar gewinnbringende Aktivitäten ausgesprochen wichtig sein können. Insgesamt steigt also das Sicherheitsbewusstsein im Team durch die Diversität an Skills.

4.2.2 Security-Rollen in der Organisation

Damit das gewonnene Sicherheitsbewusstsein seine volle Wirkung entfalten kann, muss es durch ein skalierbares System von sicherheitsrelevanten Rollen gestützt werden. In der Security Community ist der Nutzen dieser Skalierbarkeit unbestritten. Umsetzen lässt sie sich beispielsweise mithilfe eines Security-Champion-Modells [12].

Typischerweise sind Security Coaches und Security Officer der zentralen Sicherheitsorganisation angegliedert. Der Security Officer hat eine übergeordnete Sicht und überblickt technische Risiken wie auch daraus resultierende Geschäftsrisiken. Seine Gegenüber sind die Business-Entscheidungsträger. Mit ihnen muss er also die Risiken diskutieren. Dabei treten die Vertreter des Business als Risiko-Eigner auf. Und der Security Officer hilft ihnen, die Risiken richtig einzuordnen, um sie bei der Weiterentwicklung eines Produkts oder Services richtig priorisieren zu können.

Der Security Coach setzt sich je nach Gesprächspartner primär mit Bedrohungen [13] und den daraus entstehenden Risiken oder Verwundbarkeiten auseinander. Er unterstützt die

[12] Security Champions. Open Web Application Security Project, URL: *owasp.org,* Security Champions Playbook. Open Web Application Security Project, URL: *owasp.org*

[13] Wir verwenden in dieser Publikation das deutsche Wort «Bedrohung» synonym mit dem englischen Begriff «Threat».

Security Champions in den Teams. Die wiederum sind feste Mitglieder der DevOps-Teams und tragen dort den Security-Hut. Zusammen mit den Engineers im Team geht der Security Champion Verwundbarkeiten nach und bearbeitet auch technische Bedrohungen.

Getreu dem «Shift Left»-Ansatz konzentriert sich so die Verantwortung für Verwundbarkeiten und Bedrohungen direkt im DevOps-Team. Die zentrale Sicherheitsorganisation wirkt dabei unterstützend und befähigend und steuert die Behandlung von Sicherheitsrisiken.

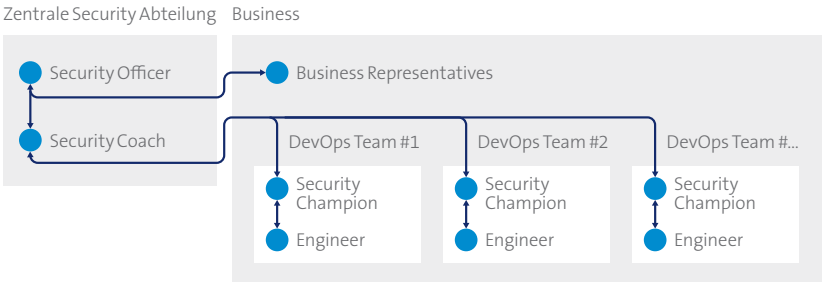

Bild 5 | Organisatorische Aufstellung der Security-Rollen

Weil DevOps-Organisationen sehr dynamisch sind, kann es helfen, wenn die Rollen in einem zentralen Register gepflegt werden. Damit ist stets klar, wer jeweils welche Rolle innehat und in welcher Form er informiert werden muss. In jedem Fall sollte mit der Definition von neuen Rollen sparsam umgegangen werden, da jede zusätzliche Rolle neue Schnittstellen und damit zusätzlichen Koordinationsaufwand generiert. Damit die Organisation möglichst schlank bleibt und schnell auf wechselnde Anforderungen reagieren kann, ist die schrittweise Einführung der Security-Rollen sinnvoll. Beispielsweise können in einem ersten Schritt die oben beschriebenen Rollen des Security Officer und des Security Coachs kombiniert werden.

> Wenn in einem Grossunternehmen eine DevOps-Transformation durchgeführt wird, generiert dies üblicherweise viel Unsicherheit (im unternehmerischen Sinn). In solch einem Umfeld bietet das Security-Champions-Modell der zentralen Sicherheitsorganisation die Möglichkeit, sich (als Teilorganisation) früh zu involvieren und die nötigen Strukturen zu schaffen.

Security Champions

Für die Rolle des Security Champions kursieren in der Industrie verschiedene, leicht unterschiedliche Definitionen. Einig ist man sich aber darüber, dass er vornehmlich die

Ansprechperson im Team für alles rund um die Sicherheit sein soll. In dieser Funktion fördert er die Awareness. Er hilft, Bedrohungen und Verwundbarkeiten zu erkennen, macht sie transparent und stellt sicher, dass sie angegangen werden.
Die Verantwortung für die Sicherheit eines Produkts soll aber nicht allein beim Security Champion liegen, sondern insgesamt beim Team und bei der Organisation. Es ist auch nicht vorgesehen, dass der Champion alle Security-Themen allein angeht. Vielmehr tut er dies in ständigem Austausch mit seinem Team, seinem Security Coach, aber auch mit anderen Security Champions *(siehe weiter unten: Security Community)*.

Beim Austausch mit seinem Coach profitiert der Champion vor allem von dessen Know-how. Im Gegenzug kann sich der Coach auch ein Bild der (Security-)Lage im Team respektive im Projekt machen. So ermöglicht der Security Champion eine konsolidierte, teamorientierte Sicht, die die konkreten Bedürfnisse abbildet.

In der Praxis hat sich bei Swisscom Folgendes gezeigt: Für die Akzeptanz und damit den Erfolg eines Security-Champion-Modells in einem Enterprise-Kontext ist es wichtig, dass der Security Champion nicht nur ins DevOps-Team integriert ist, sondern auch seine technische Sprache spricht. Wenn verschiedene Teams an einem gemeinsamen Produkt arbeiten, darf zudem die übergeordnete Security-Sicht nicht vernachlässigt werden. Weil üblicherweise jedes Team einen Teil des Endprodukts entwickelt, kann sonst der Gesamtüberblick verloren gehen. Deshalb ist es wichtig, dass man auch auf Business-Ebene einen Champion nominiert. Dieser «Champion of Champions» behält vor allem die Architektur und das Zusammenspiel der verschiedenen Teams bezüglich Sicherheit im Auge.

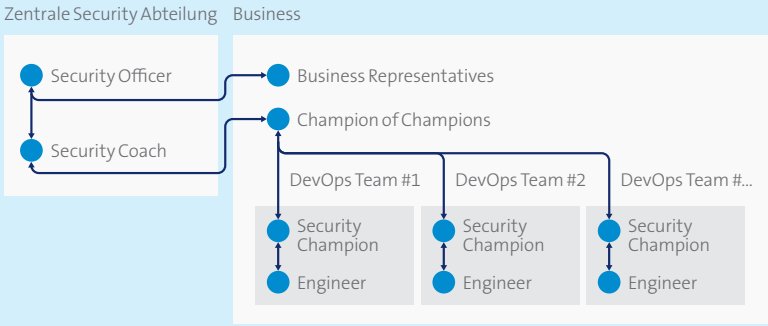

Bild 6 | Der Security Champion of Champions behält den Überblick bei grösseren Produkten.

Je grösser die Organisation, desto weniger sinnvoll ist es, dass der Security Champion ein eigentlicher Sicherheitsspezialist wird. Damit in einem

Grossunternehmen trotzdem das nötige Know-how in das Endprodukt fliesst, kann die Rolle des Champions auf mehrere Personen innerhalb eines Produktteams verteilt werden.
Die Ernennung von Security Champions kann zu Spannungen führen. Das passiert besonders dann, wenn im DevOps-Team niemand freiwillig in diese Rolle schlüpfen will. Häufig springt dann ein Mitglied über seinen Schatten – oder es wird in die Rolle gedrängt. In solchen Fällen braucht es allseits so viel Umsicht und Pragmatismus, dass sich der «Champion wider Willen» trotzdem wohl in seiner Rolle fühlt.

Ganz wesentlich: Mit der Nomination der Security Champions ist die Arbeit weder für das Business noch für die zentrale Sicherheitsorganisation getan. Damit die Champions ihre Rolle ausfüllen können, muss nämlich dafür gesorgt werden, dass sie auch die nötigen Ressourcen (Zeit, Geld, Entscheidungskompetenz) erhalten. Zudem gilt es sicherzustellen, dass sie das erforderliche Wissen vermittelt erhalten, das es ihnen erlaubt, im Team die Verantwortung für die Sicherheit zu tragen.

Man sollte sich aber im Klaren sein: Auf Unternehmensebene liegt die Verantwortung für Sicherheitsrisiken und die damit einhergehenden Massnahmen immer bei der Stelle, die abschliessend über ein Produkt oder einen Service entscheidet. Zwar wird mit DevOps ein grosser Teil der Verantwortung und damit auch der Entscheidungsbefugnis an die Teams delegiert. Die Sicherheitsorganisation muss die Teams aber über das Coaching-Modell begleiten. Und: Sie muss die Kompetenz haben, einzuschreiten und die Reissleine zu ziehen, wenn die Sicherheit grob vernachlässigt wird.

Security Coaches
Die Aufgabe der Security Coaches ist es in erster Linie, die Security Champions zu befähigen, ihren Job zu machen.

Dieser Punkt darf nicht unterschätzt werden. Wenn es einem Security Champion nämlich an Know-how mangelt, führt dies zu Mehraufwand bei den Security Coaches. Wird die Lücke nicht entdeckt, entstehen blinde Flecken bei der Sicherheit, die schlimmstenfalls zu unentdeckten (und daher unberechenbaren) Sicherheitslücken führen.

Der Security Coach soll den Security Champions in der täglichen Arbeit beratend zur Verfügung stehen. Er soll Synergien zwischen den Bedürfnissen mehrerer Teams erkennen, bündeln und bedienen. Bei kritischen Sicherheitsthemen kann der Coach den Champion

und das Projekt durchaus vertieft, nötigenfalls auch inhaltlich unterstützen.
Zum Beispiel kann er dem Champion bei Security Audits helfen. Der Coach ist typischerweise Ansprechperson für mehrere Teams. Dabei konzentriert er sich nicht auf implementierungstechnische Spezifika. Vielmehr hilft er auch beim Zusammentragen der relevanten Rahmenbedingungen, bei Reviews zu Architekturanpassungen und bei der Umsetzung eines sinnvollen Threat-Modeling-Prozesses.

Besonders wirksam ist ein Security Coach dann, wenn er direkt auf passgenaue Lösungsmodule, Prozeduren oder Musterimplementierungen zur Umsetzung der Vorgaben verweisen kann. Idealerweise kann er sich dabei eines durchsuchbaren Katalogs bedienen, der Vorgaben mit möglichen Lösungsvarianten verknüpft. Von solch einem Katalog kann übrigens auch der Security Champion profitieren. Steht er den DevOps-Teams frei zur Verfügung, können sie die beste Lösung selbst finden, die gleichzeitig auch alle Sicherheitsanforderungen erfüllt.
Aufgabe des Security Coaches ist es zudem, technische Sicherheitsrisiken transparent zu machen, die im Dialog mit den Champions erkannt werden. Nötigenfalls muss er dafür sorgen, dass alle Stakeholder diese Risiken auch verstehen.

Security Officer
Je nach Grösse der Organisation können weitere Security-Rollen eingeführt werden. Besonders bei komplexen und verschachtelten Organisationen ist es sinnvoll, weitere Abstraktionsebenen einzuführen.
Der Security Officer ist mit den Business-Stakeholdern verknüpft und hat den Gesamtüberblick über die verschiedenen Produkte und deren Zusammenspiel. Diese erweiterte Sicht erlaubt es ihm, besser einzuschätzen, wo grosse Risiken lauern und wie die verschiedenen Elemente, Teams und Abteilungen zusammenspielen. Der Security Officer setzt sich also hauptsächlich mit den Risiken auseinander. Dank seiner Sicht auf das Business kann er sowohl technische als auch geschäftliche Risiken identifizieren und gegenüber den zuständigen Stellen kommunizieren. Bei der Priorisierung von Mitigationsmassnahmen steht der Security Officer unterstützend zur Seite. Der Security Officer tauscht sich regelmässig mit den Coaches in seinem Handlungsfeld aus. Dabei werden wichtige Informationen geteilt und die strategische Ausrichtung definiert.

Security Community
Wenn neue dezentrale Security-Rollen geschaffen werden, empfiehlt es sich, das Knowhow der Rolleninhaber zu verknüpfen und so eine sogenannte Matrixkommunikation zu ermöglichen. Das heisst, der Security Champion ist mit seiner Rolle in seinem Team zwar einzigartig, hat aber die Möglichkeit, sich mit fachlichen Peers im Rahmen der Security Community auszutauschen.

Dieser Erfahrungsaustausch ist gerade in der Security wichtig. Um ihn zu fördern und zu pflegen, kann es sinnvoll sein, eine sogenannte Security-Gilde zu gründen. Sie sorgt dafür,

dass sich die Community-Mitglieder gegenseitig unterstützen und Lösungen zu Sicherheitsherausforderungen auch gemeinsam diskutiert und erarbeitet werden können. Erfahrungsgemäss sind viele Aspekte, die bei einem Security Champion auftauchen, auch für andere Teams relevant. Deshalb lassen sich die erarbeiteten Lösungen auch von anderen Teams wiederverwenden. Werden Erfahrungen und Lösungen innerhalb einer Gilde geteilt und weiterentwickelt, erhöht sich die Sicherheitsmaturität der ganzen Firma. Dies wiederum entlastet die zentrale Sicherheitsorganisation.

> Bei Swisscom wurde die Security-Gilde folgendermassen umgesetzt: Alle Security Champions und Coaches treffen sich mehrmals jährlich zum gemeinsamen Arbeiten an Sicherheitsproblemen. Um das Know-how zu vergrössern, den Austausch und die Motivation zu fördern, werden regelmässig Workshops, Schulungen und Anlässe organisiert. Dazu gehört beispielsweise der jährliche «Capture The Flag»[14]-Hackathon. Dort erhalten die Teilnehmenden die Gelegenheit, Security-Aufgaben in spielerischer Form und anhand von Praxisbeispielen zu lösen. Die Gilde unterhält zusätzlich einen Chat, der den Austausch ermöglicht und beschleunigt.

4.3 Sicherheit in Planungsaktivitäten

In diesem Kapitel zentral		
People	**Process**	**Technology**
Berücksichtigen von Sicherheitsanforderungen Verständnis schaffen, wie sich die im agilen Umfeld typische iterative Planung auf die Sicherheit auswirkt	Effiziente und iterativ anwendbare Prozesse zur Identifizierung von Bedrohungen und Risiken verfügbar machen	Unterstützung durch Tools, wo möglich, um Informationen zu versionieren, zu teilen und zu annotieren

«Everything as Code» ist eines der zentralen Prinzipien von DevOps. Dabei darf aber nicht vergessen werden, dass Code nicht ohne Vorbereitung und reifliche Überlegung niedergeschrieben werden kann. Grundsätzlich sollen Sicherheitsaktivitäten bereits in den abstrakten Phasen «Requirements gathering» und «Architecture & Design» des Software Development Lifecycle berücksichtigt werden.

[14] CTF? WTF?, CTFtime team, URL: *ctftime.org*

4.3.1 Sicherheitsanforderungen definieren

Einfluss auf den Anforderungskatalog einer Produktentwicklung haben in jedem Fall die folgenden massgebenden Quellen:

- Gesetzliche Vorschriften hinsichtlich der verarbeiteten Informationen (z. B. das Schweizer Fernmelde- oder Datenschutzgesetz)
- Die zu erfüllende Compliance (etwa zum Rundschreiben der Schweizer Finanzmarktaufsicht Finma oder zu ISO/IEC 27001)
- Die Handhabung von Daten und Informationen gemäss firmeninterner Datenklassifizierung und entsprechenden Vorgaben sowie, je nach Fall, Kundenanforderungen

In einem Enterprise-Kontext sind diese Anforderungsquellen geschäftskritisch. Ein Verstoss kann unter Umständen schwere Folgen nach sich ziehen – von Geldstrafen und einem Reputationsschaden bis hin zum Verlust von Geschäftsgeheimnissen und damit der Geschäftsgrundlage.
Ist der Anforderungskatalog einmal durchdacht, muss er als Grundlage für das komplette Produkt und dessen Architektur, Implementierung und Bereitstellung allen beteiligten Personen kommuniziert werden. Selbstverständlich muss auch dafür gesorgt werden, dass er angepasst wird, wenn der Einsatz des Produkts auf neue Bereiche erweitert wird.

4.3.2 Threat Modeling

Um die verarbeiteten Informationen adäquat schützen zu können, muss die technische Lösung gegen Angriffe gewappnet sein. Eine Bedrohungsmodellierung (Threat Modeling) erlaubt es, mögliche Angriffsformen zu identifizieren, denen das Produkt ausgesetzt wird. Basiert die Bedrohungsmodellierung auf einem systematischen Ansatz (z. B. STRIDE[15]), können zusätzliche Aspekte wie Nachvollziehbarkeit und Reproduzierbarkeit berücksichtigt werden. Oft ist es sinnvoll, den grundlegenden Ansatz mit Erfahrungswerten zu ergänzen und zu verfeinern.

> Die Autoren dieses Booklets glauben, dass Threat Modeling den grössten Nutzen für die Sicherheit eines Produkts bietet. Seine konsequente Umsetzung birgt aber viele Herausforderungen und setzt ein hohes Mass an Security Awareness bei allen Beteiligten voraus. Gerade angesichts der Diversität der Aktivitäten in einem Grossunternehmen wird das Sicherstellen der durchgängigen Qualität von Bedrohungsmodellen zur Herkulesaufgabe. Konsequentes Threat Modeling kann deshalb durchaus als Qualitäts- und Maturitätsmerkmal einer Organisation verstanden werden.

[15] STRIDE ist ein Akronym und steht für die 6 Bedrohungsklassen Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service und Elevation of Privilege; STRIDE wurde ursprünglich von Microsoft-Mitarbeitenden begründet.

Resultate eines Threat Models

Das gewinnbringende Resultat eines Threat Models sind nicht in erster Linie die erarbeiteten Bedrohungen. Obwohl sie für die Nachvollziehbarkeit wichtig sind, sind die daraus hergeleiteten Massnahmen sowie die Test- und Detektionsszenarien wertvoller.

Bei der Wahl der anzuwendenden Mitigation gilt mit absteigender Wichtigkeit:

- Bedrohungen sollten möglichst durch eine Architekturanpassung vermieden werden.
- Wenn die Architektur nicht angepasst werden kann, soll die technische Mitigation möglichst mit einer Standardlösung umgesetzt werden.
- Wenn keine Standardlösung verfügbar ist, soll die eigene Lösung mindestens durch einen Spezialisten geprüft werden.
- Ist auch eine technische Lösung unmöglich, muss die (verbleibende) Verwundbarkeit als Risiko dokumentiert und kommuniziert werden.

Entsprechend der gewählten Mitigation können daraus abgeleitet Test- und Detektionsszenarien definiert werden.
Alle im Threat Model erarbeiteten Resultate beeinflussen die Security-Aktivitäten in den folgenden Phasen des SDLC:

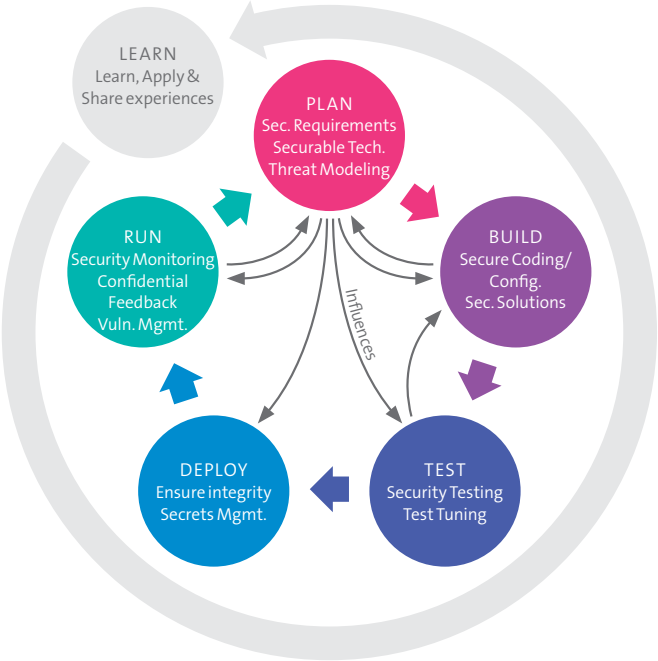

Bild 7 | Ein Threat Model beeinflusst alle anderen Aktivitäten in einem SDLC

- **Implementierung:** Die technischen Mitigationen werden direkt oder via Einbinden einer Lösung umgesetzt.
- **Test/Integration:** Die umgesetzten Mitigationen werden automatisch getestet. Hierbei können dedizierte Security-Scanning-Tools und -Services herangezogen werden. Wenn nötig werden sie mit den Erkenntnissen aus Threat Models spezifischer auf das Produkt ausgerichtet.
- **Deployment:** Bedrohungen, die bei der Verteilung von Software, bei der Konfiguration oder bei vertraulichen Informationen wie Passwörtern, Private Keys oder Tokens auftreten, können frühzeitig mit passender technischer Mitigation angegangen und überwacht werden.
- **Betrieb:** Damit Angriffe während der Laufzeit erkannt werden können, müssen zwei wichtige Voraussetzungen erfüllt sein. Zum einen müssen die identifizierten Detektionsszenarien geplant und umgesetzt werden. Zum anderen muss dafür gesorgt werden, dass das Produkt die nötigen Informationen aus den Logs auf einer zentralen Plattform aggregiert.

Verwaltung der Threat-Modeling-Dokumentation
In der Praxis gibt es verschiedene Ansätze zum Verwalten einer Threat-Modeling-Dokumentation. Sie unterscheiden sich im Wesentlichen durch die Art der Handhabung:

- Beim zentralen Threat Model arbeitet das ganze Team an der gleichen, zentral abgelegten Dokumentation. Hier muss bei der Umsetzung einzelner Features jeweils auf das zentrale Threat Model referenziert werden.
- Beim verteilten Threat Model werden neu entstehende Bedrohungen und die zugehörigen Mitigationen direkt Teil des entsprechenden Arbeitspakets (Ticket, User Story etc.). Deshalb werden sie direkt in diesem Kontext spezifiziert und umgesetzt.

Beide Ansätze haben spezifische Vor- und Nachteile, je nachdem, wer auf welcher Stufe damit arbeitet. Zudem liefern sie eine etwas unterschiedliche Sicht.
Ein zentrales Threat Model ermöglicht die schnelle Orientierung in einem Produktkontext. Damit lassen sich Synergiepotenziale einfach erkennen. Es verursacht aber bei der Umsetzung von einzelnen Features, User Stories oder Task kognitiven Aufwand seitens des DevOps Engineers. Er muss dann die für ihn relevanten Punkte zuerst aus dem Ganzen heraussuchen. Ein isoliertes und vom eigentlichen Produkt losgelöstes Threat Model kann zudem rasch asynchron zum Produkt werden. Deshalb ist es oft aufwändig, damit zu arbeiten und es aktuell zu halten.

Das verteilte Threat Model ist näher am DevOps Engineer und isoliert den für das jeweilige Feature relevanten Teil. Weil bereits ein Ticket als Arbeitsgrundlage vorhanden ist, lässt sich das verteilte Modell einfacher pflegen. Hier entsteht umgekehrt zusätzlicher Aufwand, wenn es darum geht, eine komplette Übersicht über die Bedrohungen und die umgesetzten Mitigationen zu erhalten. Dann müssen zuerst alle Einzelstücke zusammengesetzt werden.

Je nach Sicht, die man gewinnen möchte, verursachen also beide Ansätze Mehraufwand. Damit der Umgang mit dem Modell, seine Pflege und die Folgeaktivitäten geordnet ablaufen, sollen sie in einem einfachen und zweckdienlichen Threat-Modeling-Prozess abgebildet werden.

DevOps und Threat Modeling

Threat Modeling ist im Lebenszyklus eines Produkts massgebend für dessen Sicherheit. Die saubere Integration in den Ablauf der Entwicklung ist aber anspruchsvoll. Auch hierfür gibt es verschiedene Herangehensweisen.

So oder so müssen Bedrohungen immer identifiziert werden, wenn eine Architekturerweiterung geplant wird. Um dies tun zu können, ist es notwendig, dass sich eine Person in einen Angreifer hineinversetzen kann. Das ist anspruchsvoll, weil man dabei von seiner systematisch-konstruktiven Haltung (als Entwickler oder Betreiber) in die kreativ-destruktive Rolle eines Angreifers schlüpfen muss. Häufig ist genau das die grösste Herausforderung, wenn das Entwicklerteam selbst ein Threat Model erarbeiten muss.

Alternativ kann man Threats auch automatisch durch ein Tool generieren lassen. Häufig fehlt den Ergebnissen dabei aber die nötige Spezifität zum Produktkontext. Deshalb werden viele der generisch gehaltenen Threats von den Nutzern der Tools als nicht anwendbar eingestuft. Das hat zur Folge, dass die Ergebnisse des Tools allgemein weniger genau analysiert werden und so ihr Effekt wieder verpufft.

Letztlich profitiert die Qualität eines Bedrohungsmodells wohl am meisten, wenn ein erfahrener Bedrohungsmodellierer das Entwicklerteam unterstützt. Bei diesem Ansatz entstehen aber oft Skalierungsprobleme. Speziell im Enterprise-Kontext fehlt es den Sicherheitsexperten oft an technischem oder am thematischen Verständnis vom Produkt. Deshalb entstehen hier Engpässe, mit denen umgegangen werden muss.

Um diese Lücke zu schliessen, ist ein durchgängiger und einfacher Threat-Modeling-Prozess nötig. Er soll die nötigen Schnittstellen schaffen, etwa zu den folgenden Schritten im Product Lifecycle, und dafür sorgen, dass die Beteiligten ohne Reibungsverluste einander zuarbeiten können. Beispielsweise soll ein erfahrener Threat Modeler mit wenig Ahnung vom Produkt ein Team unterstützen können, das wiederum wenig Ahnung von Security Threats hat.

> Weil DevOps ein iteratives Konzept ist, kann die Architektur eines Produkts mit jeder neuen User Story verändert werden. Deshalb ist es nicht praktikabel, ein initiales Bedrohungsmodell zu erstellen und über längere Zeit zu referenzieren, ohne dass es gepflegt wird.

Stattdessen sollte darauf geachtet werden, dass das Modell sowohl als Ganzes als auch in Form von einzelnen, spezifischen Teilstücken verarbeitet werden kann. Ersteres ist wichtig, um eine gesamtheitliche Security-Sicht zu erhalten, und Letzteres, um Bedrohungen passgenau angehen zu können. Die stetige Veränderung des Bedrohungsmodells im Entwicklungsprozess macht seine Pflege jedenfalls nicht einfacher.

4.4 Technische Security-Aktivitäten

In diesem Kapitel zentral		
People	**Process**	**Technology**
Das Team muss die Einsatzgebiete von technischen Lösungen kennen und verstehen	Entdeckte Verwundbarkeiten müssen getrackt werden und es muss klar sein, wer dafür verantwortlich ist Es braucht fachliche Unterstützung bei der Integration, beim Aufbau und beim Bereitstellen der Security Solutions	Security Solutions sollen einfach zu integrieren sein

Unter technischen Security-Aktivitäten sind jene Schritte zusammengefasst, die entweder unmittelbar technisch umsetzbar sind oder die Implementierung eines Produkts direkt beeinflussen.

4.4.1 Security Solutions

Im Rahmen von «Shift Left» werden dem DevOps-Team nicht nur Sicherheitsthemen übertragen – die für sich ja schon eine Mammutaufgabe sind. Vielmehr kommt noch Weiteres dazu, wie Stabilität, Change-Management, betriebliches Monitoring und so weiter. Dabei muss stets berücksichtigt werden, dass sich all dies durch den gesamten Produktlebenszyklus zieht.

Die Verantwortung, die dem Team übertragen wird, ist also gross. Wenn hier eine adäquate Unterstützung fehlt, können leicht Fehler mit schwerwiegenden Konsequenzen entstehen. An diesem Punkt kommt die zentrale Sicherheitsorganisation ins Spiel. Sie kann durch Bereitstellen von integrierbaren, skalierbaren und konfigurierbaren Security Solutions das nötige Expertenwissen einbringen.

Pfannenfertige Lösungen
Damit das Sicherheitsniveau im Unternehmen gehalten oder erhöht werden kann, soll den DevOps-Teams ein Katalog mit pfannenfertigen Lösungen, Patterns & Referenzimplementierungen bereitgestellt werden. Die passenden technischen Integrationen können via Inner Source (einem innerorganisatorischen Open-Source-Ansatz) angeboten und, auch durch die DevOps-Teams, erweitert werden.

Damit der Katalog standardisiert und übersichtlich bleibt, sollte die inhaltliche Steuerung in den Händen der Sicherheitsspezialisten bleiben. Sie steuern das Portfolio aufgrund von Sicherheitsrisiken und Nutzer-Feedback. Ziel sollte es sein, die Anwendung von Solutions so einfach und unkompliziert wie möglich zu gestalten.

> Es hat sich als sinnvoll erwiesen, generisch einsetzbare Security Solutions von einem separaten DevOps-Team aus einer Organisationsperspektive zentral weiterzuentwickeln und zu betreiben. Bei Swisscom übernimmt das, als Teil der zentralen Sicherheitsorganisation, das DevSecOps Tooling Competence Center. Ziel eines solchen Set-ups muss es sein, das spezifische Expertenwissen aus der Sicherheitsorganisation möglichst automatisiert in die Produkt-Deployment-Pipelines zu bringen.
>
> Dabei gilt es zu beachten, dass dieses Team auch die üblichen DevOps-Aufgaben wie Support und Dokumentation wahrnehmen muss. Speziell beim Angebot von Security Solutions ist die korrekte Einbindung in die Zielsysteme entscheidend.

Um sich im Solutions-Katalog zu orientieren und die relevanten Solutions zu identifizieren, kann einmal mehr ein Bedrohungsmodell herangezogen werden. So können statt Vorgaben zur Mitigation bestimmter Bedrohungen direkt Security Solutions referenziert werden. Das hilft nicht nur dem DevOps-Team, weil es keine eigenen Lösungen entwickeln muss, sondern auch den Security Coaches und Security Officers. Sie können dann sicher sein, dass der zentral angebotene und von einem spezialisierten Team betriebene Security-Service wasserdicht ist.

Einige Beispiele für Security Solutions
Die Liste von Security Solutions, die über einen Katalog angeboten werden können, ist lang. Einige Beispiele, die sich an der bekannten STRIDE-Methode zur Bedrohungsidentifikation orientieren:

- **Authentisierung:** (Multifaktor-)Authentisierung als Service, der in eine Applikation eingebunden werden kann
- **Integrität:** Public Key Infrastructure (PKI) als automatischer Service, um Signierung und Verschlüsselung von Artefakten und Nutzdaten zu ermöglichen
- **Nachvollziehbarkeit:** Logging-Lösungen, die ein Security Monitoring erlauben
- **Vertraulichkeit:** Bibliotheken und Services zur Verschlüsselung von und für den Zugriff auf sensitive Informationen, wie betrieblich benötigte Geheimnisse (Passwörter, SSH Keys, Zertifikate etc.)
- **Verfügbarkeit:** Bibliotheken und Services, mit denen Angriffe auf Applikations- und Netzwerkebene erkannt und abgewehrt werden können
- **Autorisierung:** Identity-Management-Lösungen, deren Einbindung eine einfache Autorisierung auf Applikationsebene erlaubt

4.4.2 Automatisiertes Security Testing

Das automatisierte Verifizieren von Produkten bildet den Kern von DevOps. Automatisierte Tests ermöglichen es, Fehler schnell zu erkennen und zu beheben. Dabei gilt: Je früher ein Mangel aufgedeckt wird, desto einfacher kann er behoben werden und desto weniger Aufwand entsteht dabei.

Die Security bildet hier keine Ausnahme. Im Gegenteil – komplett automatisierte Deployment Pipelines (inklusive impliziter Security-Freigabe) erlauben es erst, Sicherheitsaspekte bereits während der Implementierung sichtbar zu machen. Die Automatisierung des Security Testing ist also nicht nur eine Qualitätsprüfung, sondern auch ein Instrument zur Steigerung der Security Awareness.

Damit all die Vorteile zum Tragen kommen, braucht es einige Vorarbeiten. So müssen beispielsweise Security-Testing-Services bereitgestellt werden, die den Bedürfnissen der DevOps-Teams im Zusammenhang mit der Automatisierung entsprechen. Dazu gehören unter anderem:

- Die problemlose technische Integration der Security-Testing-Lösung in die Deployment Pipeline
- Keine Verlangsamung des Builds (Test Performance)
- Minimaler Onboarding-Aufwand

Wer diese Prinzipien befolgt, kann die Akzeptanz der Security-Testing-Services verbessern. Das wiederum erleichtert die Durchsetzung der erforderlichen Services.

Es kommt immer wieder vor, dass einzelne DevOps-Teammitglieder weniger Wert auf Security legen. Häufig stehen sie dann auch Security-Tools kritisch gegenüber und kommunizieren dies auch so.

Um solche, auf die Security gerichtete Frustration zu vermeiden, braucht es eine geeignete Supportstelle. Sie hilft bei Problemen rund um die Integration respektive Anwendung von Security Tooling und beantwortet auch inhaltliche Fragen. Bei Swisscom übernimmt diese Aufgabe wieder das DevSecOps Tooling Competence Center. Dieses multidisziplinäre Team baut die Security-Tools aus und betreibt sie. Gleichzeitig ist es Anlaufstelle für Anwender und nimmt auch Feedback entgegen, das wiederum in bestehende oder neue Solutions einfliesst.

Anforderungen an Security-Testing-Tools
Kernstück der Automatisierung bilden die Tools, die die Tests am Ende durchführen. Bei deren Auswahl gilt es, einige Grundaspekte zu beachten:

- **Zukunftstauglichkeit:** Bedürfnisse ändern sich besonders in einer DevOps-Organisation schnell. Deshalb muss ein Tool flexibel integrierbar und auf verschiedene Arten ansprechbar sein. Dies lässt sich erreichen, wenn alle Arbeitsschritte via API automatisiert werden können. Idealerweise wird das Security-Tool nach einem API-first-Ansatz entwickelt.
- **Anwenderorientierung:** Das Tool soll die Hauptanwendergruppen zufriedenstellen, also:
  - Die DevOps-Teams: Sie sind technisch unterwegs und müssen deshalb ihre Kerntechnologien unterstützt wissen. Das Tool soll auch die technische Weiterentwicklung unterstützen.
  - Die Security Champions und Coaches: Sie helfen den Teams dabei, Sicherheit ins Produkt mit einzubauen. Darum soll das Tool die Möglichkeit bieten, Defects (also gefundene Schwachstellen) so zu aggregieren, dass sich Bereiche identifizieren lassen, in denen das Team noch Unterstützung braucht.
- **Mandantenfähigkeit:** Speziell in grossen Firmen ist es interessant, Testresultate nach Geschäftsbereichen auszuwerten. Das kann beispielsweise dem Umstand geschuldet sein, dass Bereiche unterschiedlich kritisch für die Organisation sind. Oder es ermöglicht eine Übersicht, in welchen Bereichen noch in Security investiert werden sollte. Ein Tool soll solche spezifischen Betrachtungen unterstützen.

Gerade in serviceorientierten Umgebungen werden im Alltag oft viele verschiedene Technologien eingesetzt. Deren Unterstützung am Markt variiert häufig stark. So werden häufig Services in veralteten Programmiersprachen weitergewartet, während gleichzeitig Services mit Technologien entwickelt werden, die so neu sind, dass sie noch kaum jemand kennt.

Dieser Vielfältigkeit lässt sich auf verschiedene Weise begegnen. Entweder wird ein Tool respektive ein Service pro Technologie oder Programmiersprache bereitgestellt, oder aber man findet ein Produkt auf dem Markt, das gleich das ganze Spektrum abdeckt. So oder so: Im Enterprise-Umfeld soll hier nach der 80-20-Regel vorgegangen werden, gesteuert durch die Kritikalität der zu scannenden Produkte.

Test Scope
Mit automatisierten Security-Tests können verschiedene Aspekte untersucht werden. In der Praxis lassen sich mindestens die folgenden Fälle unterscheiden:

- **Selbst entwickelter Source Code:** Wird ein Produkt selbst entwickelt, kann der Source Code geprüft werden. Verwundbarkeiten lassen sich so direkt angehen und mitigieren oder entfernen.
- **Third Party Libraries:** Fast jede Software basiert an irgendeiner Stelle auf Software-Artefakten, die von Dritten verantwortet werden. Bei Open Source Libraries beispielsweise wird häufig jegliche Verantwortung dem Anwender übertragen. Umso mehr ist der darauf angewiesen, über alle bekannten Verwundbarkeiten informiert zu sein.
- **Infrastruktur:** Jede Software setzt auf irgendeiner Art von Infrastruktur auf. Diese Basis muss sicher konfiguriert und aktuell gehalten werden. Schwachstellen wie unnötig exponierte Kommunikations-Ports, unsichere Kryptographie-Algorithmen oder fehlende Security Headers müssen vermieden werden.

Testtypen
Es gibt Typen von Security-Tests, die komplementär eingesetzt werden können und häufig auch sollten. Auch wenn sich die Test-Scopes einzelner Typen überlappen, können damit trotzdem andere Verwundbarkeiten gefunden werden. Im Folgenden sind die am weitesten verbreiteten Security-Testing-Typen beschrieben. Sie können für sich und untereinander spezifischer auf das zu testende Produkt ausgerichtet werden, wenn Informationen aus dem Threat Model herangezogen werden.

- **Static Application Security Testing (SAST):** Bei der statischen Analyse wird der Source Code an sich untersucht. Je nach Technik wird nach Stichwörtern gesucht, die bekanntermassen zu unsicherem Verhalten führen. Fortgeschrittene Analyse-Tools bauen, vom Source Code ausgehend, Graphmodelle, die anschliessend auf bestimmte Muster untersucht werden.
  Allgemein gesehen sind SAST-Lösungen effizient, da sie sehr tiefen Einblick in den Source Code gewähren. Ihr grosser Nachteil ist allerdings die Anfälligkeit auf False Positives, die oft manuell aufgearbeitet werden müssen. Weiter hinken SAST-Tools häufig der Weiterentwicklung von Programmiersprachen und Frameworks nach.
  Da SAST den Applikationskontext nicht berücksichtigt, werden zudem logische Fehler oft nicht entdeckt.

- **Dynamic Application Security Testing (DAST):** Bei der dynamischen Analyse wird die Software in einem geschützten Umfeld ausgeführt und mit sicherheitsrelevanten Eingaben aufgerufen. Eingaben können entweder sinnvoll formatiert sein oder zufällig generiert werden (sogenanntes Fuzzing). Je nachdem wie sich die Applikation verhält oder was sie zurückgibt, kann auf Verwundbarkeiten zurückgeschlossen werden. DAST-Tools generieren weniger False Positives. Weil der Applikationskontext mitläuft, eignen sie sich auch besser zum Entdecken von logischen Fehlern. Wegen des limitierten Einblicks in die Software werden aber nur unmittelbar exponierte Verwundbarkeiten entdeckt. Selbstverständlich können die Tools in gewissem Rahmen den Zielapplikationen angepasst werden. Es bleibt aber schwierig, einen kreativ vorgehenden Angreifer realistisch zu simulieren.
- **Interactive Application Security Testing (IAST):** Das «Interactive» in IAST bezieht sich auf die Interaktion zwischen den beiden bereits vorgestellten Ansätzen. Damit wird versucht, die positiven Aspekte aus SAST und DAST zusammenzuführen. So erlaubt IAST beispielsweise abzuschätzen, wie ein bestimmter Input in einem Datenfluss verarbeitet wird. Dies senkt den Anteil an False Positives. Allerdings sind IAST-Lösungen häufig sehr technologiespezifisch.

Alle drei Konzepte zählen zu den dedizierten Application-Security-Lösungen.

Während eine SAST-Lösung als Selfservice für die Teams eingeführt wurde, konnte Swisscom folgende wertvolle Erfahrung gewinnen: Die DevOps Engineers, oft mit wenig Know-how zu SAST und Security, haben teilweise ihren kompletten Source Code, inklusive Test-Verzeichnissen, Fixture-Daten etc., scannen lassen. Dies führte dazu, dass der einzelne Scan sehr viel Zeit in Anspruch nahm (mehrere Stunden) und deshalb nicht in automatische Build-Prozesse eingebunden werden konnte. Folglich wurden diese Scans nur manuell und sporadisch durchgeführt.

Durch das unlimitierte «Hineinkippen» von Code wurde den Anwendern auch eine riesige Menge an Defects präsentiert – in den meisten Fällen False Positives. Dadurch verkam die Interpretation der Scan-Resultate zur sprichwörtlichen Suche nach der Nadel im Heuhaufen.

Entsprechend sank die Akzeptanz des neuen SAST-Services massiv. Wegen der negativen Erfahrungen entwickelten viele Nutzer eine regelrechte Aversion gegen den Service und weigerten sich, ihn weiter zu nutzen.
Die wichtigste Erkenntnis aus dieser Situation ist, dass ein SAST-Tool nicht ohne spezifische Informationen über das Zielobjekt (Kontext) auskommt. Um dies in den Griff zu bekommen, wurde primär an zwei Stellen angesetzt:

- Der komplette Onboarding-Prozess wurde neu gestaltet.
- Das Zielpublikum wurde neu justiert, primär, um ohne Vorbehalte neu starten zu können.

Zudem wurde der Service unter anderem Namen neu lanciert. Damit konnte das negative Image abgestreift und mit dem an sich guten Tool noch einmal von vorne begonnen werden.

Heute wird der Service von einem spezialisierten SAST-Team mit entsprechendem Know-how betrieben. DevOps-Teams, die ihr Produkt neu für den SAST-Service registrieren, durchlaufen beim Onboarding drei Schritte:

- In einem initialen Threat Model werden Datenflüsse analysiert und irrelevante Code-Teile identifiziert.
- Anschliessend wird der Scan vom SAST-Team spezifisch auf das Produkt abgestimmt, um False Positives zu eliminieren, False Negatives aufzudecken und die Durchlaufzeit zu optimieren.
- Nach dem ersten Scan werden die gefundenen Defects in einem Fact Finding Meeting besprochen. Dort erhalten die Teams auch Unterstützung, die es ihnen erleichtert, die Defects zu beheben und in Zukunft zu vermeiden.

Mit diesen Änderungen am Prozess konnten fast 95 % der False Positives verhindert werden. Die Scan-Durchlaufzeit hat sich von mehreren Stunden auf wenige Minuten reduziert.

Mit der kurzen Laufzeit und der hohen Relevanz der Befunde können die Scans nun automatisiert werden. Insgesamt hat sich dadurch die Akzeptanz bei den Anwendern auch merklich verbessert.

Grundsätzlich können Sicherheitsaspekte auch ohne dedizierte Security-Testumgebung getestet werden. So lässt sich etwa die Input-Validierung direkt durch automatisierte **Unit-Tests** abdecken.

Es liegt auf der Hand, dass die Anzahl der eingeschleppten Verwundbarkeiten durch Third Party Libraries so klein wie möglich gehalten werden sollte. Um dies zu erreichen, kann der Code zuerst mit SAST hinsichtlich eingebundener externer Abhängigkeiten durchsucht werden. Mit den erhaltenen Resultaten können anschliessend relevante Verwundbarkeiten über die Korrelation von CPE (Common Platform Enumeration) und CVE (Common Vulnerability Enumeration) identifiziert werden.

Ein anderer Ansatz, der ergänzend eingesetzt werden kann, führt den Scan erst am Ende der Deployment Pipeline auf dem fertig assemblierten Artefakt aus. Um Informationen über inkludierte Third Party Libraries zu gewinnen, kann das Artefakt entpackt werden. Das liefert ein Inventar der darin enthaltenen Softwarekomponenten und -versionen, anhand dessen sich gefundene Verwundbarkeiten korrelieren lassen.
Damit auch zu einem späteren Zeitpunkt Verwundbarkeitskorrelationen möglich sind, kann das Inventar persistiert werden. Dieser Vorgang wird unter *4.6.1 Security Monitoring* behandelt. Er wird als Artefact Vulnerability Management (AVM) oder auch Software Component Analysis (SCA) bezeichnet.

Um die **Infrastruktur** auf die sichere Konfiguration hin (manchmal auch «Härtung» genannt) zu testen, werden verschiedene Tools eingesetzt. Dazu gehören nmap[16], nessus[17], sslscan[18] und Ähnliches. Die Tools können entweder einzeln, via Kommandozeilenintegration oder als Bestandteil von etablierten Testframeworks automatisiert werden. Solche Frameworks können beispielsweise mit der Abstraktionssprache «Gherkin»[19] orchestriert werden, die sich im End-to-End-Testing etabliert hat. Gherkin ist eine Syntax, die im Behaviour Driven Development (BDD) verwendet wird. Sie soll es auch Nichttechnikern ermöglichen, Testfälle zu schreiben, indem Szenarien durch Prosa-Text formuliert werden *(siehe Bild 8: Beispiel eines Testfalles in Gherkin-Syntax)*.

Grundsätzlich sollten sich auch bei diesem Ansatz die Infrastrukturtestfälle direkt vom Bedrohungsmodell ableiten oder sogar in automatisierter Weise generieren lassen. In der Community spricht man dann üblicherweise von Security as Code. Das fügt sich natürlich schön in die DevOps-Forderung «Everything as Code» ein.

Es gibt verschiedene automatisierte Security-Testing-Methoden. Wichtig zu verstehen ist, dass jede von ihnen die Applikation aus einem anderen Blickwinkel betrachtet. Eine einzelne Methode für sich liefert also nur eine ungenügende Sicht auf die Applikation. Optimale Sicherheit wird erreicht, wenn man verschiedene automatische Testing-Methoden kombiniert und mit einem Penetration Testing abschliesst. Dann kann man sicher sein, dass auch die Kreativität eines Angreifers ins Resultat mit einfliesst.

[16] Nmap von Gordon Fyodor Lyon, URL: *nmap.org*

[17] Nessus (Software), Wikipedia: *de.wikipedia.org/wiki/Nessus_(Software)*

[18] sslscan, rbsec, URL: *github.com/rbsec/sslscan*

[19] Gherkin Syntax, Cucumber Community, URL: *docs.cucumber.io/gherkin*

```
Feature: Google Searching
      As a web surfer, I want to search Google, so that I can
      learn new things.

Scenario: Simple Google search
      Given a web browser is on the Google page
      When the search phrase "panda" is entered
      Then results for "panda" are shown
```

Bild 8 | Beispiel eines Testfalles in Gherkin-Syntax *(Quelle: automationpanda.com)*

4.4.3 Manuelles Security Testing

Viele technische Security-Tests können automatisiert werden. Ihre Qualität hängt allerdings davon ab, wie gut sie gewartet werden. Somit zeichnen die Resultate von automatisierten Tests zwar ein gutes Bild von der Basissicherheit. Sie können aber nicht die Kreativität eines Angreifers abbilden, der Informationen in einen Kontext zu setzen vermag.

Penetration Testing

Deshalb können automatisierte Tests die periodischen technischen Penetrationstests nicht ersetzen. Dabei spielt es keine Rolle, ob es sich um White-, Gray- oder Black-Box-Tests [20] durch geschultes Fachpersonal handelt. Optimal ist aber, wenn die Software vor dem Penetrationstest bereits automatisiert getestet und die dabei gefundenen Verwundbarkeiten behoben wurden. Dann können sich die Tester besser auf die komplexen Verwundbarkeiten konzentrieren und werden weniger durch auftauchende «einfache» Verwundbarkeiten abgelenkt.

Koordinierte Audits

Periodische Audits sind primär für kritische und exponierte Produkte sinnvoll. Getreu dem DevOps-Ansatz kann für den Penetrationstest eine dedizierte Produktumgebung instanziiert werden.

Die Tester sollten mit möglichst vielen Informationen zum Produkt versorgt werden. Das ermöglicht es ihnen, auch fortgeschrittene Angriffsszenarien von Anfang an auszuprobieren oder auszuschliessen. Ziel der Aktion sollte sein, möglichst effizient zu effektiven Resultaten zu gelangen (entspricht einem White-/Grey-Box-Test). Defensive Haltungen helfen in diesem Set-up nicht, die Produktqualität realitätsnah zu prüfen und einzuschätzen.

[20] Bei White-Box-Tests erhält der Tester Quellcode und die Konfiguration eines zu testenden Produkts. Bei Grey-Box-Tests wird nur ein Teil der Informationen preisgegeben. Bei Black-Box-Tests hat der Tester keine vorgängigen Informationen zur Verfügung.

Red Teaming

Während es beim Penetrationstest darum geht, die Schwachstellen einer bestimmten Anwendung zu finden, liegt der Fokus beim Red Teaming [21] darin, eine Schwachstelle auszunutzen, um möglichst weit und unbemerkt in ein Unternehmensnetzwerk einzudringen. Red Teams gehen dabei unter realistischen Bedingungen vor. Das heisst, sie verfügen nur über wenige interne Informationen und erarbeiten sich diese im Zuge ihrer Aktion. Das Red Team in der Angreiferrolle ist der Gegenspieler des Blue Teams in der Verteidigerrolle. Das Blue Team bezeichnet alle defensiven und reaktiven Massnahmen, die ein Unternehmen im Sicherheitsbereich einsetzt. Im weitesten Sinn sind neben den Security-Teams, die an der Attack Response beteiligt sind *(siehe Kapitel 4.6)* also auch die DevOps-Teams Teil des Blue Teams.

An sich bleibt das Konzept des Red Teaming auch in einer DevOps-Organisation unverändert. Die Unterschiede liegen primär in der potenziellen Reaktionszeit und in den administrativen Auswirkungen eines Angriffs. Da ein Produkt bei DevOps gleichzeitig vom selben Team entwickelt und betrieben wird, können beispielsweise Anomalien beim Datenverkehr viel schneller und unkomplizierter angegangen werden *(siehe Kapitel 4.6)*.

Abschliessende Red-Team-Publikationen sind oft unterhaltsame und einfach zu lesende «War Stories», die doch mit viel technischem Inhalt aufwarten. Sie erlauben den DevOps-Teams, direkt einen Bezug aufzubauen und Vergleiche zum eigenen Produkt herzustellen. In Bezug auf die Security Awareness ist eine solche Aufarbeitung also sehr effizient und wertvoll. Natürlich lassen sich aus den Red-Team-Aktionen auch systematische Massnahmen ableiten.

> Red Teaming wird bei Swisscom seit 2015 betrieben. Pro Jahr werden ein bis zwei Angriffssimulationen mit einem sich jeweils ändernden Team durchgeführt. Das Spektrum der Einsätze reicht von Social Engineering über Phishing bis hin zu aktiven Übernahmen von Swisscom-Systemen mittels Malware. Alles dies geschieht natürlich, ohne tatsächlichen Schaden anzurichten. Ziel dieser Angriffe ist es einerseits, Schwächen und Risiken zu identifizieren, bevor echte Hacker sie ausnutzen. Andererseits soll dabei das Blue Team (Swisscom SOC, CSIRT und der Betrieb) mit realitätsnahen Szenarien konfrontiert werden, um Schwächen in Fähigkeiten und Abläufen zu identifizieren.
>
> Wie sich gezeigt hat, ist die grösste Herausforderung, die im Red Team aktiven Mitarbeitenden aus dem Alltagsgeschäft herauszulösen, damit sie sich die nötige Zeit fürs Hacken nehmen können. Da auch das Blue Team seine

[21] Red team, Wikipedia: URL: *en.wikipedia.org/wiki/Red_team*

Detektionskompetenzen ausbaut und stets dazulernt, wird die Aufgabe für das Red Team mit der Zeit immer schwieriger.

Wie beschrieben, hat sich die interne Aufarbeitung und Kommunikation der sogenannten Red Team Cases als sehr wertvoll erwiesen. Von den aufbereiteten Storys und Berichten fühlen sich Mitarbeitende in allen Geschäftsbereichen und auf allen Stufen konkret angesprochen. Dies führt wiederum zu Anpassungen in ihrem Umfeld, die der Security zugutekommen.

4.5 Deployment Pipeline Security

In diesem Kapitel zentral		
People	**Process**	**Technology**
Das Team kennt die Bedrohungslage hinsichtlich der zentralen Deployment-Pipeline-Systeme (also der «Fabrik») Es herrscht ein Bewusstsein für Risiken bzgl. eigener Assets und solchen von Partnerteams, die die gleiche Plattform verwenden	Es gibt eine Gewaltentrennung nach dem Need-to-know-Prinzip	Die Deployment Pipeline unterstützt die modulare Erweiterung mit Komponenten zur Qualitätsüberprüfung Die zentralen Systeme sind adäquat abgesichert

Weil eine der Kernfähigkeiten von DevOps-Organisationen in der Automatisierung liegt, müssen Integrität und Vertraulichkeit der Automatisierungsservices stets gewahrt sein. Zu den Automatisierungsservices können die folgenden Systeme gezählt werden:

- **Workplace** – das System, an dem Code geschrieben wird
- **Version Control System (VCS)** – dort wird der Code aufbewahrt und versioniert
- **Continuous Integration/Continuous Deployment (CI/CD) System** – hier wird der Code verarbeitet
- **Scanning Services** – dienen zum Testen von Code und Artefakten
- **Artefakt-Ablagen** – hier werden Dateien für Softwarepakete und weiteres gelagert
- **Laufzeitumgebungen** – sind erforderlich für die Werterbringung mittels des Softwareprodukts

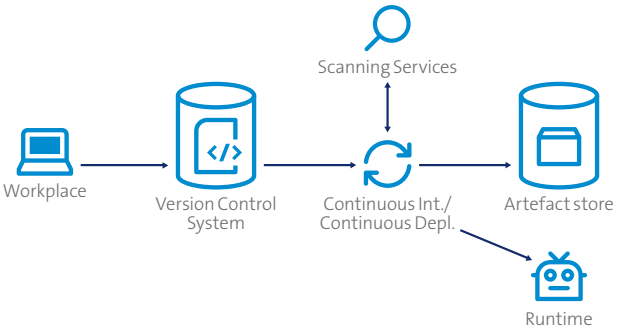

Bild 9 | Abstrakte Darstellung einer Deployment Pipeline mit den verschiedenen Komponenten, die es zu sichern gilt.

In diesem Kapitel geht es vor allem um die Interaktion mit externen Partnern. Diese ist wichtig, weil die zunehmend zentrale Rolle der Deployment Pipeline Schnittstellen zu Systemen von Partnern und damit neue Angriffsvektoren ermöglicht.

Zum DevOps-Prinzip gehört, dass die Software und deren Konfiguration getrennt aufbewahrt werden. Das heisst: Neben der Software muss auch ihre Konfiguration in die Produktionsumgebung eingebracht werden. Weil Konfigurationen häufig auch Parameter enthalten, die für den Datenaustausch mit Drittsystemen verwendet werden, muss auch diese Kommunikation gesichert werden. In den meisten Fällen geschieht dies anhand von Zertifikaten und Geheimnissen zur Authentisierung. Mit DevOps entstehen hier neue Anforderungen.

Der Einfluss von «Everything as Code»
Die Maxime «Everything as Code» bedeutet bei DevOps-Organisationen, dass alle Tests, Builds, Ablagen und Deployments durch Code beschrieben werden. Dies hat enorme Auswirkungen: So kann eine versehentliche Anpassung an der falschen Stelle Tests scheitern lassen. Im schlimmsten Fall kann eine mutwillig bösartige Änderung am Code Geheimnisse exfiltrieren. Allein schon der lesende Zugriff auf eine Codebase kann Angreifern eine ganzheitliche Sicht auf das betriebene Produkt liefern. Das erleichtert ihnen die Suche nach Schwachstellen massiv.

Mit der Konzentration der Logik im Code entfallen Segmentierungen und damit auch manuelle Kontrollmechanismen. In einer Transformationsphase hin zu DevOps gilt es, speziell die Qualitätskontrollen durch automatisierte Massnahmen zu ersetzen. Sie werden, wiederum als Code, spezifisch auf das Produkt abgestimmt abgelegt. Dadurch entsteht eine Art logische Redundanz, die integritätsbewahrend wirkt.

Infolge der Übersetzung (fast) aller Aspekte in Code vervielfacht sich der Wert respektive die Kritikalität der Code-Ablage, also des Version Control Systems (VCS). Ist dieses System kompromittiert, hat das weitreichende Konsequenzen.

Eine weitere kritische Komponente ist die Artefakt-Ablage. Hier sind die Herausforderungen vergleichbar mit denen beim VCS. Können Artefakte verändert oder gar kompromittierte Artefakte eingeschleust werden, sind womöglich ganze Produktionsplattformen in Gefahr.

Bei beiden Systemen ist die Wahrung der Integrität von verwalteten Assets am wichtigsten. Ungewollte Veränderungen können zu fatalen Problemen in Produktionsumgebungen führen. Am zweitwichtigsten ist die Vertraulichkeit. Der Abfluss von Daten aus einem VCS oder einer Artefakt-Ablage kann dazu führen, dass Geheimnisse und Interna verloren gehen.

Selbstverständlich gibt es im Zusammenhang mit «Everything as Code» noch weitere Bedrohungen. Allerdings sind die eher spezifisch für die jeweils eingesetzten Technologien oder Prozesse.

Integrität und Vertraulichkeit von Code und Artefakten
Um die Integrität von Code und Artefakten zu schützen, sollte beim Git-basierten [22] VCS die Signierung von Commits durchgesetzt werden. Technisch gesehen müssen Commits mit unbekannten Signaturen zurückgewiesen werden. Nur so lässt sich sicherstellen, dass ausschliesslich autorisierte Nutzer Code-Veränderungen vornehmen können.

Bei Artefakten sollte die Integrität mindestens mit einer Prüfsumme feststellbar sein, also einer Quersumme über das komplette Artefakt. Im optimalen Fall wird aber auch für Artefakte eine Signatur durchgesetzt.

Bei der Wahrung der Vertraulichkeit von Interna und Geheimnissen sind die Nutzer ebenso gefordert wie die Technik. So gilt es etwa, Zugriffe mit einem geeigneten Identity & Access Management (IAM) zu steuern. Damit lässt sich etwa sicherstellen, dass nur auf die Assets zugegriffen werden kann, die auch benötigt werden. Exzessive Zugriffe sollten via Monitoring aufgedeckt und via Throttling/Rate Limiting gebremst werden können. Gegenmassnahmen können IP- oder Account-Blocking sein, um mögliche Angriffe zu unterbinden *(mehr dazu in Kapitel 4.6)*.

Aber auch die Verfasser von Code oder Systemspezifikationen sind gefordert. Eine konsequent durchgesetzte Clean-Coding-Vorgabe kann das Risiko eines Abflusses von operativen Geheimnissen (Zertifikate, Schlüssel, Passwörter etc.) verringern. Geheimnisse haben in einem VCS nichts verloren, da sie selbst nach dem Entfernen aus dem fertigen Code noch in alten Versionen gefunden werden können. Ähnliches gilt für Geheimnisse in Artefakten.

[22] Git hat sich als Versionierungstechnologie mehrheitlich durchgesetzt, Git, Git Community, URL: *git-scm.com*

Geschützt werden sollen auch Algorithmen, sofern es sich dabei um kommerziell verwertbare Innovationen handelt. Sie sind als Assets zu betrachten. Oft sind hier komplett segregierte VCS gerechtfertigt.

Zusammenarbeit mit Partnern und Lieferanten
Wertbringende Daten oder Systeme sind primär für den Eigner wertvoll. Weil Partner und Lieferanten weniger Bezug zu diesen Assets haben, sind sie oft nicht besonders willig, zusätzliche Sicherheitsmassnahmen umzusetzen. Umso mehr braucht es auch gegenüber den Partnern vertraglich geregelte Sicherheitsvorkehrungen. Darauf aufbauend lassen sich beispielsweise Prozesse auditieren oder kontrollieren, ob Abmachungen eingehalten werden.

Weil Personalrochaden recht häufig und kurzfristig auftreten, sollte der eingesetzte IAM-Prozess ein schnelles Onboarding ermöglichen. Auch das «Offboarding» sollte nicht vernachlässigt werden, etwa indem dafür gesorgt wird, dass einzelnen Mitarbeitern wenn nötig rasch Zugriffe entzogen werden können.

Sehr vorteilhaft ist es in einem DevOps-Kontext, wenn die eingesetzten Good Practices technisch und vollautomatisiert überprüft werden. So können beispielsweise unsignierte Commits von Zulieferern zurückgewiesen werden. Dasselbe gilt, wenn Zulieferer Commits von solchen Mitarbeitern pushen, die beim Kunden nicht angemeldet wurden. Fasst man das Konzept etwas weiter, lassen sich auch automatisch erkannte Security Defects direkt an die Partner zurückspielen.

Insgesamt zeigen die Beispiele, dass die Zusammenarbeit mit Partnern und Lieferanten gut durchdacht sein muss. Die Prozesse sollten es erlauben, Fehler rasch zu korrigieren. Die Ansprechpartner beidseits sollten sich dazu auf einem technischen Level unterhalten können.

Zugriff auf produktive Daten und Systeme
Das ultimative Ziel einer DevOps-Organisation sollte sein, komplette Infrastrukturen als Code abbilden zu können. So werden Anpassungen während der Laufzeit ausgeschlossen. Der Zugriff auf produktive Systeme ist dann nur noch in Ausnahmefällen möglich, etwa um spezielle Vorkommnisse und Umstände verstehen zu können. Eigentlich wären auch solche Zugriffe nicht nötig, weil Problemfälle via sauberes Logging nachvollzogen und ausgewertet oder zumindest in einer nicht-produktiven Umgebung reproduziert werden sollten.

Die Aktionen im Verlauf des Zugriffes können zwecks Nachvollziehbarkeit aufgezeichnet werden. Im optimalen Fall würde ein einzelner Systemknoten nach manuellen Zugriffen direkt durch eine neue Instanz ersetzt.

Aus Sicht der Security ist der Nutzen eines Infrastructure-as-Code-Ansatzes gross – um nur einige Aspekte zu nennen:

- Die Systemhärtung (sichere Konfiguration) kann anhand von Code-Analysen überprüft werden
- Anpassungen an der Systemhärtung können eindeutig nachvollzogen werden
- Instanziierungen von dedizierten, zur Produktionsumgebung identischen Testumgebungen sind ohne Zusatzaufwand möglich, etwa fürs Penetration Testing
- Systeme, die potenziell durch manuelle Aktionen geschwächt wurden, können unmittelbar ersetzt werden (auch: Immutable Infrastructure)
- Produktionsdaten sind besser geschützt, da in der Regel kein manueller Zugriff auf produktive Systeme stattfindet

Auch für den Schutz von Daten und Systemen ist ein funktionierendes IAM Voraussetzung. Können Identitäten nicht zweifellos zugeordnet werden, lassen sich auch die Zugriffe nicht mehr vertrauenswürdig Personen zuordnen. Damit kann auch die Nachvollziehbarkeit von Aktionen nicht mehr gewährleistet werden. Eine weitere Herausforderung entsteht, wenn nicht menschliche Nutzer auf Services zugreifen, sondern technische Accounts.

Verwaltung von Geheimnissen und Zertifikaten
Unweigerlich mit DevOps verbunden ist die Frage der automatisierten Verwaltung von Geheimnissen, die für den Zugriff von Systemen auf andere Systeme (z. B. Datenbanken oder weitere Services) nötig sind. Seit jeher entspricht es einer Best Practice, Geheimnisse wie Passwörter von Zeit zu Zeit zu ersetzen. Das reduziert das Risiko, dass Angreifer Zugriff auf ein altes Passwort erlangen und damit unerlaubterweise und anonym Daten einsehen oder gar verändern können.

In einer Organisation mit Enterprise-Dimensionen laufen sehr viele Systeme, auf die von automatisierten Prozessen aus zugegriffen wird. Deshalb entsteht sehr viel Aufwand, wenn Systemadministratoren periodisch alle Passwörter von Hand auswechseln müssen. Der manuelle Passwortwechsel kann auch betriebliche Probleme auslösen, wenn ein zugreifendes System noch ein altes Passwort verwendet und plötzlich den nötigen Zugriff nicht mehr erhält. Das kann etwa dann passieren, wenn ein Passwort im Source Code verankert ist. Davon abgeleitet entstehen weitere Probleme. Dazu gehört etwa, wie vorgegangen werden soll, wenn ein Systemadministrator die Firma verlässt, der wahrscheinlich die verwendeten Passwörter kennt.

Ein weiterer Punkt, der für die automatische Geheimnisverwaltung spricht, ist auch der verkürzte technische Lebenszyklus von Produkten im DevOps-Umfeld. Spätestens dadurch werden manuelle Lösungen definitiv unpraktikabel. Geheimnisse müssen dann unab-

hängig von der Produktversion rotiert und in das Produkt eingefügt werden. Genau dasselbe gilt für (Web-)Zertifikate und ihren jeweiligen Gegenpart, den privaten Schlüssel.

Wer diese Themen mit einem DevOps-würdigen Automatisierungsgrad angeht und trotzdem ein angemessenes Sicherheitsniveau garantieren will, findet auf dem Markt verschiedene Secrets-Management-Lösungen. Die gibt es sowohl als kommerzielle Produkte wie auch in Form von Open Source.

Bei den Zertifikaten gilt es, eine Public Key Infrastructure (PKI) einzubinden, die zum Automatisierungsbedürfnis passt. Hierbei muss entschieden werden, ob ein Gratis-Service verwendet wird oder ob sich die Organisation eine eigene PKI aufbauen soll. In letzterem Fall kann die Automatisierung mit dem De-facto-Standard Automated Certificate Management Environment (ACME) adressiert werden.

4.6 Produktiver Betrieb und Attack Response

In diesem Kapitel zentral		
People	**Process**	**Technology**
Kommunikation und Vorbereitung der Bedrohungsszenarien	Monitoring und Response-Prozesse bewirtschaften Für eine Organisation sorgen, die die Reaktionsfähigkeit sicherstellt	Einsatz von Lösungen zur Erkennung von Angriffen

Security Incidents passieren – mit oder ohne DevOps. Diese unangenehme Wahrheit lässt alle präventiven Sicherheitsmassnahmen in einem anderen Licht erscheinen. Sie zeigt auch, dass es nicht nur die eine (präventive) oder die andere (reaktive) Seite von Security gibt. Vielmehr erstrecken sich die Herausforderungen über den kompletten Produktlebenszyklus.

In grösseren Organisationen gibt es immer Bereiche, die bezüglich der Implementierung von Security-Massnahmen etwas weniger im Fokus stehen. Sie müssen jeweils mit weniger Unterstützung auskommen als andere Bereiche. Aus diesem Blickwinkel gesehen ist es wichtig, über einfache Schnittstellen und Prozeduren zum Sammeln von sicherheitsrelevanten Logs und Feedbacks zu verfügen. Auch das Abarbeiten respektive Auswerten der Daten und Feedbacks soll aus Sicht des DevOps-Teams möglichst einfach und nachvollziehbar ablaufen.

In einem effizienten Software-Entwicklungszyklus sollten alle Arten von Feedback auf eine Kernaussage reduziert werden, damit sie in den präventiven Teil des SDLC zurückfliessen können. Beispielsweise lassen sich Hinweise auf nicht berücksichtigte Bedrohungen in einen Bedrohungskatalog einpflegen, der während der Planungsaktivitäten beim Threat Modeling wieder herangezogen werden kann. So werden Bedrohungskatalog und -modell iterativ erweitert mit dem Ziel, eine möglichst komplette Sicht zu erhalten.

Anwender-Feedback
Zu den Grundsätzen von DevOps gehört es, implizites (messbares) wie auch explizites Feedback von den Anwendern einzuholen. Dabei muss sicherheitsrelevantes Feedback zu einem Produkt in der Regel vertraulich behandelt werden. Es darf auch nur einem limitierten Kreis an Personen zugänglich gemacht werden. Andernfalls riskiert man, dass potenzielle Angreifer Kenntnisse über bestehende Schwachstellen erhalten und dem Unternehmen schaden können.

Um zu verhindern, dass sicherheitsrelevante Beurteilungen in Foren oder sozialen Medien zu lesen sind, gilt es, zunächst legale und vertrauliche Kanäle zu schaffen, über die Schwachstellen gemeldet werden können. Zusätzlich müssen Anreize geschaffen werden, die externe Sicherheitsforscher überhaupt dazu veranlassen, Feedback zu geben. Anreize braucht es gewöhnlich auch, um die Qualität des Feedbacks hochzuhalten. Mehr dazu später unter «*Bug Bounty*».

Vulnerability Management
In der Praxis kann es vorkommen, dass während des produktiven Einsatzes einer Lösung neue Verwundbarkeiten für einzelne ihrer Komponenten publiziert werden. Die Informationen dazu können aus Quellen innerhalb oder ausserhalb der eigenen Organisation stammen *(siehe 4.4.2)*, etwa aus der Common Vulnerability Enumeration (CVE). Damit man rasch reagieren kann, gilt es, solche Quellen ständig zu überwachen. Der generelle Ablauf beim Behandeln von Verwundbarkeiten wird formal vom ISO-Standard 30111 zu «Vulnerability Handling Processes» beschrieben.

DevOps ermöglicht hier neue Ansätze. Zum einen vereinfacht es, betroffene Komponenten auszutauschen, da der komplette Applikationsaufbau durch Quellcode beschrieben ist. Zum anderen helfen vollautomatisierte Tests, rasch zu entscheiden, ob die Funktionalität eines Produkts immer noch gewährleistet ist. Damit der gesamte Prozess reibungslos vonstattengeht, muss ein präzises Inventar vorhanden sein, aus dem abgeleitet werden kann, welche Produktinstanzen von einer Verwundbarkeit betroffen sind.

Eine Möglichkeit, mit Schwachstellen umzugehen, ergibt sich aus der Kombination von zentralen und dezentralen Ansätzen: Die DevOps-Teams sind grundsätzlich für die Schwachstellen und ihre Behebung verantwortlich (dezentral). Dabei erhalten sie Unterstützung aus der zentralen Sicherheitsorganisation. Sie bietet im Rahmen einer Art Taskforce-Struktur Unterstützung beim Beheben von kritischen Schwachstellen und sorgt damit für kurze Reaktionszeiten.

Bug Bounty

Eine zunehmend beliebte Massnahme, um Feedback bezüglich Sicherheit einzuholen, sind Bug-Bounty-Programme. Sie vereinen den vertraulichen Feedback-Kanal mit dem Vulnerability Management, indem eine weitere Informationsquelle für Verwundbarkeiten geschaffen wird. Dabei setzen sie den ISO-Standard 29174 zu «Vulnerability Disclosure» um.

Bug-Bounty-Programme funktionieren grob gesagt so[23]: Ein Anwender meldet eine gefundene Schwachstelle im Produkt inklusive nachvollziehbarer Dokumentation an das Unternehmen. Wird die Schwachstelle bestätigt, erhält er eine Belohnung, sofern er sich zur Geheimhaltung während einer bestimmten Zeit verpflichtet. Diese Zeit nutzt das Unternehmen, um die Schwachstelle zu beheben. Dieses Vorgehen ist in Fachkreisen auch als «Responsible Disclosure Model» bekannt.

In DevOps-Organisationen grosser Firmen entstehen infolge der zugrundeliegenden Agilität laufend relevante Herausforderungen bei der Sicherheit. Deshalb wird das Bug-Bounty-Programm am besten von zentraler Stelle aus betrieben. Voraussetzung ist aber, dass die Betreiber die gesamte Angriffsfläche der Organisation kennen. Sie ergibt sich aus der Summe der exponierten Produkte.

Damit Bug-Bounty-Meldungen effizient abgearbeitet werden können, braucht es ein möglichst komplettes und aktuelles Verzeichnis mit den nötigen Angaben zu Produkten und Personen *(siehe auch: «Alarm – und dann?» unter 4.6.1)*. Ein solches Verzeichnis à jour zu halten, ist aber aufwändig, weil sich die Produkte, Verantwortlichkeiten und Ansprechpersonen dauernd ändern. Zusätzlich ist es empfehlenswert, den technischen Geltungsbereich des Bug-Bounty-Programms genau abzustecken und für die Allgemeinheit zu dokumentieren. In einer solchen Dokumentation steht dann genau, welche Applikationen und Services ein Unternehmen im Internet exponiert. Nur auf diesen dürfen die Sicherheitsforscher Verwundbarkeiten suchen.

[23] Die Bedingungen von Bug-Bounty-Programmen können je nach Unternehmen variieren. Die hier angeführte Definition entspricht dem generellen Verständnis.

[24] Bug Bounty, Swisscom AG, URL: *swisscom.ch*

Damit die gesamte Angriffsfläche abgedeckt werden kann, sollte zudem der Fokus auf die organisationsinternen Akteure erweitert werden. Reife Organisationen stellen hierzu auch den internen Anwendern einen vertraulichen Kommunikationskanal für sicherheitsrelevantes Feedback zur Verfügung.

> Swisscom war in der Schweiz eines der ersten Unternehmen, das ein Bug-Bounty-Programm [24] angeboten hat. Seit der Einführung im Jahr 2015 hat sich der Kanal in zahlreichen Fällen bestens bewährt. Für das breite Publikum relevante Schwachstellen sind auf dem Portal als «Security Advisories» veröffentlicht und erhalten eine CVE-Nummer (Common Vulnerability Enumeration [25]) für die eindeutige Identifizierbarkeit.
>
> Beim Aufbau des Bug-Bounty-Programms hat Swisscom eine Lernstrecke zurückgelegt. Beispielsweise wurde festgestellt, dass die Sicherheitsforscher von ebenso spezialisierten Vertretern der internen Sicherheitsorganisation betreut werden müssen. Das fördert eine gute Kommunikation und sorgt für adäquate Reaktionen auf die eingegangenen Meldungen. Seither hat sich die Beziehung zwischen den Sicherheitsforschern und Swisscom dahingehend entwickelt, dass es sogar gemeinsame öffentliche Auftritte gibt.

4.6.1 Security Monitoring

Unter Security Monitoring wird gemeinhin die Überwachung hinsichtlich sicherheitsrelevanter Vorkommnisse verstanden. Oft geht dabei vergessen, dass es für eine sinnvolle und produktspezifische Auswertung entsprechende Use- oder Abuse-Case-Spezifikationen braucht. Sie beschreiben, was im Ernstfall falsch laufen kann und welche Informationen zum Aufdecken eines Angriffs benötigt werden. Zwar können Log-Daten generisch ausgewertet werden – oft liefern sie dann aber deutlich zu wenige spezifische Resultate oder gar False Positives.

Eine weitere Komponente, die es zu überwachen gilt, ist das Ökosystem, in dem sich ein Unternehmen oder ein Produkt bewegt. Dabei müssen aber verschiedene Abstraktionsebenen berücksichtigt werden.

Zum wirksamen Monitoring in einer DevOps-Organisation gehört auch, dass bei einem Sicherheitsvorfall alle Ansprechpersonen von vornherein definiert sind. Ist das der Fall, kann das DevOps-Set-up seine Vorteile ausspielen: Weil die Kommunikationswege innerhalb der Teams kurz sind, können reaktive Mitigationen effizient umgesetzt werden.

[25] Common Vulnerabilities and Exposures, The MITRE Corporation, URL: *cve.mitre.org*

Weil die Orientierung gerade in Stresssituationen, also in Notfällen oder Krisen, schwierig ist, müssen alle Szenarien und Notfallpläne vorbereitend geübt werden. Dabei treten Schwächen oder gar Fehler in der Choreographie der beteiligten Parteien (DevOps-Team, beteiligte Security Response Teams, etc.) zutage.

Es liegt in der Natur des Security Monitoring, dass Incidents erst während oder nach deren Auftreten sichtbar werden. Deshalb legt eine effiziente DevSecOps-Organisation Wert auf eine gut funktionierende Kommunikation zwischen reaktiven und präventiven Akteuren. Das ermöglicht es erst, die Erkenntnisse aus den Vorfällen rasch in präventive Massnahmen umzumünzen.

Detektion von bekannten Risiken
Eine der Quellen für Security Monitoring Use Cases ist das Threat Model zum Produkt. Es können sowohl mitigierte als auch unmitigierte Bedrohungen als Monitoring Use Cases formuliert werden. Dies erlaubt zum einen die Validierung des Threat Models und zum anderen die Erkennung von sicherheitsrelevanten Events. Wobei letzteres um einiges höher gewichtet werden sollte, da ein Event unter den gegebenen Voraussetzungen zum Alert und dann zum Incident wird.

Um effektive Auswertungen anstellen zu können, sollte vom gewünschten Resultat aus, also den Monitoring Use Cases, «rückwärtsgerechnet» werden. Dabei stellen sich folgende Fragen:

- Welche Komponenten in der Produktinfrastruktur müssen Daten abliefern? Dazu gehören: das Intrusion Detection System (IDS), Reverse Proxy inkl. Load Balancer, Web Application Firewall (WAF), aber auch die unter der Applikation liegende Infrastruktur, die Datenablage etc.
- Welche Daten werden benötigt? Kann überhaupt auf sie zugegriffen werden oder sind zentral verwaltete Komponenten im Einsatz?
- Welche Daten bringen keinen Mehrwert? Ein sinnvolles «Signal-Rausch-Verhältnis» kann die Qualität des Endresultats begünstigen.

Aus Sicht des Security Monitorings ist es meist nicht sinnvoll, möglichst detaillierte Logs von allen Systemen zu sammeln. Stattdessen sollten etwas mehr Log-Daten gespeichert werden, als gerade benötigt werden. Ein guter Anfang ist es, Logs zu schreiben, die Transaktionen über alle Komponenten eines Systems hinweg anhand eines gemeinsamen Identifikators nachvollziehbar machen. Der Identifikator kann zum Beispiel eine Zufallszahl sein, die von der ersten Komponente generiert wird, die den Zugriff «sieht».

Detektion von Anomalien
Durch den Shift-Left-Ansatz entstehen speziell bei der Sicherheit spannende Synergiepotenziale. So können mit einem Mal intrusive Sicherheitslösungen wie Runtime Application

Self Protection (RASP) Frameworks interessant werden, die zuvor wegen personeller oder organisatorischer Hemmnisse nicht effizient genutzt werden konnten.

Solche Selbstverteidigungs-Frameworks sind häufig im Web-Umfeld anzutreffen. Dort sollen sie Angriffe aufgrund von untypisch verändernden Attributen selbstständig erkennen und abwehren. So kann beispielsweise eine Applikation Alarm schlagen, wenn sie Zugriffe auf einen sogenannten Honey-Endpoint [26] feststellt. In einem anderen Fall kann es verdächtig sein, wenn sich die Agent Identifikation während einer laufenden Session plötzlich ändert.
Natürlich hängt das Feintuning solcher RASP-Lösungen stark vom Applikationscode ab. Deshalb lohnt es sich, grossen Wert auf ihre korrekte technische Integration und die entsprechenden automatisierten Tests zu legen. RASP-Lösungen können auch als weitere Quelle für ein Security Monitoring herangezogen werden.

Alarm – und dann?
Taucht ein (überwachter und bestätigter) Security Incident auf, werden im Idealfall sofort und automatisch die zuvor implementierten Gegenmassnahmen ausgelöst. In geschäftskritischen Produkten stösst eine solche Automatisierung aber wegen nicht ausschliessbarer False Positives rasch an Grenzen. Dann hilft es, wenn dank der Nähe zum DevOps-Team rasch manuelle und/oder technische Gegenmassnahmen ergriffen werden können.

Eine mögliche manuelle Gegenmassnahme ist, die Web Application Firewall (WAF) so anzupassen, dass sie ein sogenanntes Hot Patching erlaubt. So lässt sich allenfalls eine Verwundbarkeit via WAF abschirmen, und das DevOps-Team gewinnt die nötige Zeit, um das Produkt selbst sauber zu korrigieren.

Eine technische Gegenmassnahme wäre beispielsweise die Aufnahme eines Angriffs in das Test-Set eines Produkts. So lässt sich kontinuierlich prüfen, ob die Verwundbarkeit (wieder) auftritt. Ohne solche Regressionstests verpufft die Awareness, die im Zusammenhang mit einem Incident erlangt wurde, in der Regel schnell.

Damit im Ernstfall alle nötigen Informationen für solche Gegenmassnahmen bekannt sind, bedarf es eines systematischen Inventars innerhalb der DevOps-Organisation. Das Inventar sollte folgende Informationen enthalten, die insbesondere dem Security Operations Center (SOC) permanent zur Verfügung stehen:

- Mitgliederverzeichnis der DevOps-Teams (technische Ansprechpersonen)
- Produktverantwortungen (Ansprechpersonen beim Business)
- Technologien (kann zur schnelleren Mitigierung beitragen)

[26] Honey-Endpoint: ein Endpoint, auf den im Normalbetrieb nicht zugegriffen wird und dessen Zweck es ist, Angreifer zu identifizieren, die die komplette Angriffsfläche einer Web-Applikation abscannen.

- Produkt-Inventar (Endpoints, Testresultate etc.)
- Weitere Informationsquellen (VCS, Threat Model etc.)

Eskaliert eine Routineanalyse zu einem Vorfall, tritt das Computer Security Incident Response Team (CSIRT) in Aktion. Es profitiert davon, dass DevOps grundsätzlich die forensischen Nachforschungen zu den Ursachen des Vorfalls und zum angerichteten Schaden unterstützt. Dank der hohen Automatisierung ist stets nachvollziehbar, wo, was, wann am Produkt verändert, den Angreifern exponiert wurde und somit den Angriff überhaupt ermöglicht hat.

Damit die Beteiligten in der Hektik keine Daten vernichten, die Informationen zum Angriff hätten liefern können, müssen sie von Anfang an informiert und trainiert werden. Grundsätzlich werden SOC und CSIRT als organisationsinterne, zentrale Services zur Verfügung gestellt. Sie ergänzen und unterstützen die DevOps-Teams primär durch professionelle Response-Massnahmen.

> Speziell das Security-Champions-Programm von Swisscom hat sich bei der Abarbeitung von Security Incidents bewährt. Es führt einerseits dazu, dass bei einem Vorkommnis der Ansprechpartner auf Seite des DevOps-Teams bereits von vornherein bekannt ist. Andererseits hat der Security Champion durch seine absolvierten Training-&-Awareness-Einheiten bereits ein Grundverständnis zur Sicherheitsterminologie aufgebaut. Das verringert die Reibung in der Interaktion zwischen CSIRT und DevOps-Team und hilft auch in der allgemeinen Kommunikation.

5 Zusammenfassung

Effiziente Security in einer DevOps-Organisation zeichnet sich primär durch die starke Integration der verschiedenen Sicherheitsaktivitäten aus. Je besser verknüpft die einzelnen Schritte sind, desto effektiver kann Security umgesetzt werden. DevOps als Ansatz hilft hier, die nötigen Voraussetzungen zu schaffen. Es reisst störende organisatorische Barrieren (etwa zwischen Entwicklung und Betrieb) ein und führt zusammen, was eigentlich zusammengehört. Das begünstigt eine gesamtheitliche Sicht über den Produktlebenszyklus.

Die Herausforderung liegt allerdings darin, Security als Themengebiet ausreichend zu priorisieren. Weil die DevOps-Teams nie dieselbe Expertise wie Sicherheitsprofis erreichen werden, bleibt die zentrale, unterstützende Sicherheitsorganisation wichtig für den Erfolg. Ihr Motto muss lauten: «Make it easy to do the right thing». In der Praxis heisst das: Sie soll Inhalte, Abläufe und Services so anbieten, dass die einfachste Variante auch die sicherste ist.

Wer die Prioritäten im Rahmen einer DevOps-Transformation nicht ganz klar setzt, riskiert, dass die Sicherheit an Aufmerksamkeit verliert. Um dies zu vermeiden, gilt es folgende Punkte aktiv anzugehen:

- **Allgemeine Awareness:** mit einem Trainingsprogramm, das der Security die nötige Visibilität verschafft.
- **Intrinsische Sicherheit:** indem Security so simpel und zugänglich wie möglich gemacht wird, beispielsweise mit einem Baukasten an Solutions, die «telquel» übernommen werden können → Der Wegfall von Prozess-Tollgates kann durch kontinuierliches Schaffen von Transparenz kompensiert werden.
- **Identity & Access Management:** Abbilden aller DevOps-Rollen, um die Segregierung von Berechtigungen und Nachvollziehbarkeit von Zugriffen gewährleisten zu können → Eine klassische Gewaltentrennung entfällt, jedoch nimmt der Wert von IAM als Hoheit über feingranulare Zugriffsberechtigungen zu.
- **Organisation:** Security soll mit all ihren Aspekten und Rollen in der DevOps-Organisation verankert sein. Damit sie skalierbar wird, müssen Verantwortlichkeiten in die Breite, also an die Teams abgegeben werden. → Dabei muss aber stets klar sein, wer für Risiken und Verwundbarkeiten zuständig ist, und es müssen auf allen Stufen die richtigen Leute gefunden und angesprochen werden können.

Werden die richtigen Massnahmen getroffen, stehen sich DevOps und Security also keineswegs im Weg, sondern eröffnen sich gegenseitig neue Möglichkeiten. So kann es gelingen, die Welt zumindest in technischer Hinsicht zu einem etwas sichereren und berechenbareren Ort zu machen.

Swisscom ist nun seit Mitte 2016 dabei, diese Best Practices zu implementieren und anzuwenden. Wie in agilen Umgebungen üblich, ist dies ein sich stetig verändernder Prozess. Ganz nach dem Motto «Do, Learn and Adapt».

Herausforderungen tauchen bisher speziell in den Bereichen Automatisierung und Kultur auf. In der Zukunft ist geplant, die Maturität auf allen drei Säulen People, Processes & Technology weiter zu verbessern und so den Niveauunterschied zwischen Vorreiter- und Nachzüglerteams zu verringern.

Auch gilt es, die Messbarkeit in allen Bereichen zu verbessern, um datengesteuert Entscheidungen treffen zu können. Ein Beispiel ist die Verknüpfung von Zahlen zu Sicherheitsverwundbarkeiten und dem Security-Training-Zertifizierungsgrad eines Teams. Begleitend dazu sollen Daten einfach dargestellt und bei Bedarf schnell verfügbar sein. Um die erwünschte Transparenz zu erreichen, drängt es sich zudem auf, relevante Kennzahlen auf Live-Dashboards zusammenzufassen.

6 Abkürzungsverzeichnis

ACME	Automated Certificate Management Environment
AVM	Artefact Vulnerability Management
BDD	Behaviour Driven Development
CALMS	Culture, Automation, Lean, Measurement, Sharing
CI/CD	Continuous Integration and Continuous Delivery oder Deployment
CPE	Common Platform Enumeration
CSIRT	Computer Security Incident Response Team
CSSLP	Certified Secure Software Lifecycle Professional
CVE	Common Vulnerability Enumeration
DAST	Dynamic Application Security Testing
DevOps	Kunstterm, zusammengesetzt aus Development (Dev) und Operations (Ops); steht für den kompletten Produktlebenszyklus
DevSecOps	Siehe DevOps; Security (Sec) wurde zwecks stärkerer Gewichtung des Themas eingefügt
IAM	Identity & Access Management
IAST	Interactive Application Security Testing
IDS	Intrusion Detection System
OWASP	Open Web Application Security Project
PKI	Public Key Infrastructure
RASP	Runtime Application Self-Protection

SAFe	Scaled Agile Framework
SAST	Static Application Security Testing
SDLC	Software Development Lifecycle
SOC	Security Operations Center
STRIDE	Steht für die sechs Bedrohungsklassen Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service und Elevation of Privilege
VCS	Version Control System
VUCA	Volatility, Uncertainty, Complexity, Ambiguity
WAF	Web Application Firewall

7 Stichwortverzeichnis

8 Weiterführende Quellen

Dem interessierten Leser empfehlen wir zur Vertiefung in einzelne DevSecOps Unterthemen die folgenden Quellen.

- A Leader's Framework for Decision Making von David J. Snowden und Mary E. Boone, URL: *hbr.org*
- Accelerate: Building and Scaling High Performing Technology Organizations von Nicole Forsgren Phd, Jez Humble, Gene Kim
- Agile Application Security von Laura Bell, Michael Brunton-Spall, Rich Smith, Jim Bird
- Building Microservices von Sam Newman, URL: *samnewman.io/books/building_microservices*
- DevOps Culture (Part 1) von John Willis, URL: *itrevolution.com/devops-culture-part-1*
- The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations von Gene Kim, John Willis, Patrick Debois, Jez Humble, URL: *itrevolution.com/book/the-devops-handbook*
- Drive von Daniel Pink. URL: *danpink.com/drive*
- Manifesto for Agile Software Development von Kent Beck und andere, URL: *agilemanifesto.org*
- The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win von Gene Kim, Kevin Behr, Georg Spafford, URL: *itrevolution.com/book/the-phoenix-project*
- Project to Product: How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework von Mik Kersten
- Securing DevOps – Security in the cloud von Julien Vehent
- Security Champions Playbook/OWASP Wiki, URL: *owasp.org*
- State Of DevOps Report von Puppet + Splunk, URL: *puppet.com*
- Threat Modeling – Designing for Security von Adam Shostack

Notizen

Swisscom AG
Alte Tiefenaustrasse 6
3048 Worblaufen