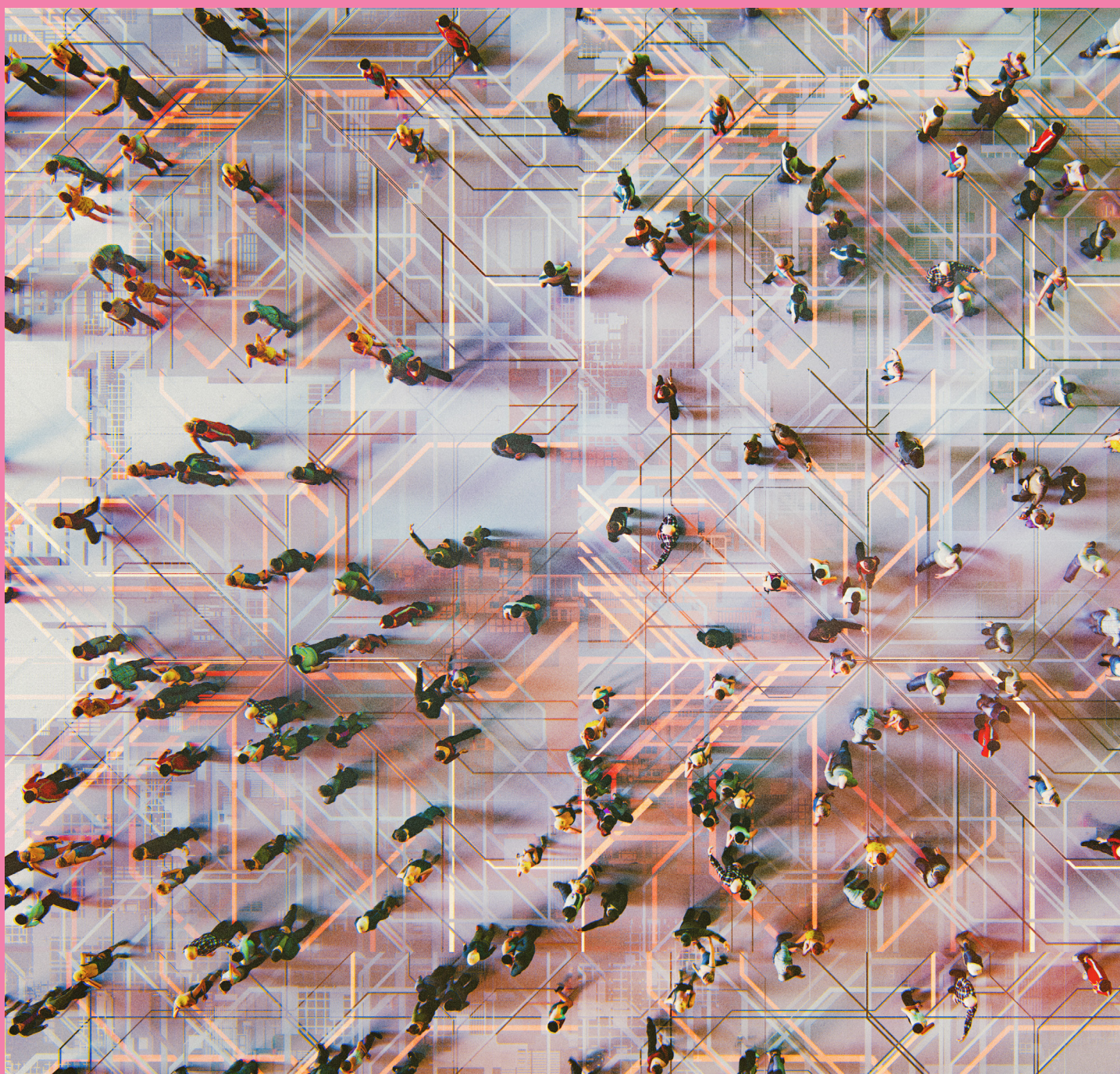
Checkmarx

The world runs on code. We secure it.

Die Facetten von Modern Application Development

Leitfaden für Führungskräfte und Experten aus der Praxis – Teil 1



Inhaltsverzeichnis

Einleitung

In einer Welt, die sich auf den technischen und logischen Grundlagen des Internets entwickelt hat, sind wir völlig abhängig von Softwareanwendungen (Apps), die nahezu jeden Aspekt unseres Lebens steuern. Nur wenige würden bestreiten, dass Software die Grundlage für die Zukunft bildet.

Für Unternehmen, die sich zu softwaregesteuerten Einheiten entwickelt haben, ist die Marschrichtung eindeutig: Es gilt, schnellere, intelligentere und userfreundlichere Anwendungen zu entwickeln, um Umsatz und Marktanteil zu steigern. Dabei verändern sich die Anforderungen an die Entwicklung und Bereitstellung innovativer Apps in bisher unvorstellbarem Tempo. Losgelöst von den Fesseln der zugrundeliegenden Internet-Infrastruktur ist das Potenzial in puncto Userfreundlichkeit, Skalierbarkeit und Reichweite quasi unbegrenzt.

Aus diesem Innovationsdruck heraus entstand Modern Application Development (MAD), das inzwischen aus den Kinderschuhen herausgewachsen ist und in fast allen öffentlichen und privaten Organisationen, deren Geschäftsmodell auf Softwareentwicklung basiert, an vorderster Front steht. Im Hinblick auf die Wertschöpfung durch Software findet ein Umdenken statt, und MAD ist der Schlüssel zur Modernisierung. Diejenigen, die sich auf diese neue Ära einlassen, werden von den Vorteilen profitieren, und diejenigen, die dies nicht tun, bekommen sehr wahrscheinlich die Quittung in Form einer hohen Fehlerquote.

In Teil 1 dieses eBooks werden wir die vielen Facetten von MAD beleuchten, um Ihnen dabei zu helfen, ein grundlegendes Verständnis des Konzepts zu entwickeln, einschließlich der Frage, wie Ihr Unternehmen den Übergang zu diesem neuen Entwicklungs- und Bereitstellungsparadigma bewerkstelligen kann. Die Themen, die wir behandeln, sind vor allem für Führungskräfte von Interesse, aber auch für Experten aus der Praxis gibt es einiges zu entdecken. Unternehmen, die die MAD-Reise antreten, müssen sich im Klaren darüber sein, dass ein langsames und vorsichtiges Vorgehen sie möglicherweise davon abhält, jemals die Ziellinie zu überqueren. In der Welt von MAD ist „langsam“ ein Fremdwort.

Was ist Modern Application Development?

Stellen Sie sich vor, im neuen Zeitalter der Softwareentwicklung Anwendungen zu entwickeln, die überall – und auf jeder Commodity-Infrastruktur – skaliert laufen können. MAD macht dies durch cloudbasierte, native Ressourcen möglich, die die Services, die Software bietet, von den bisherigen Design-, Entwicklungs- und Bereitstellungsprozessen entkoppeln und es Teams ermöglichen, in größerem Umfang als bisher zu arbeiten. MAD basiert auf der Vorstellung, dass alles Code ist, und geht davon aus, dass nichts mit den alten operativen Aspekten früherer monolithischer Software-Builds definiert ist.

In der Vergangenheit wurde erwartet, dass Software auf traditionelle, genau definierte Weise entwickelt und On-Premises in einem Data Center bereitgestellt wird. Dieser Ansatz erforderte eine Vielzahl von physischen Servern, auf denen die verschiedenen Funktionen des Codes ausgeführt wurden: Ein Server konnte für das webbasierte User Interface verwendet werden, ein anderer für einen Data Access Layer, ein weiterer für einen Data Store und so weiter.

MAD entkoppelt Softwareinnovation von operativen Grenzen, sodass Teams mehr Zeit auf die Entwicklung innovativer Funktionen verwenden können, anstatt sie damit zu verschwenden, die Auswirkungen auf die zugrundeliegenden Systeme zu berücksichtigen. Bei MAD verursachen oder beeinflussen kleine Fehlfunktionen keine Systemausfälle, da die Services von den Ressourcen, auf denen sie laufen, entkoppelt sind. Bei MAD dreht sich alles um die Generierung von zusätzlichem Transaktionswert, was Entwickler zu einem grundlegenden Umdenken zwingt.

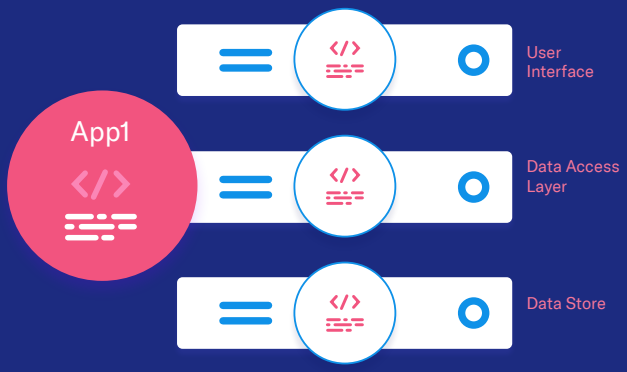
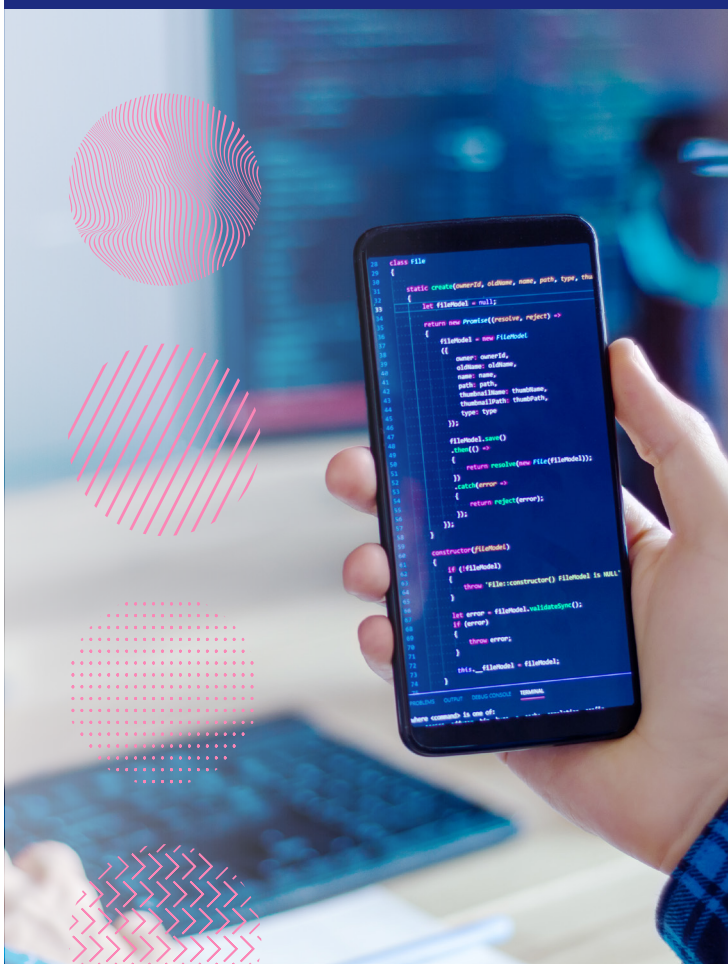


Abbildung 1 zeigt das Konzept einer traditionellen Architektur. Dank MAD sieht das heute anders aus.



MAD – Geschwindigkeit

MAD läuft in Hochgeschwindigkeit ab. Zu Anfang kann dies für Entwickler aufgrund der linearen Denkweise, die sie oft verinnerlicht haben, Stress bedeuten. Im MAD ist alles integriert und automatisiert, wodurch alle von Menschen verursachten Verzögerungen beseitigt oder überwunden werden.
MAD beeinflusst die Entwicklungs- und Bereitstellungsprozesse so radikal, dass viele die Vorteile anfangs gar nicht erkennen – aus Unverständnis oder aus Angst vor dem Unbekannten. Fünf Jahre Softwareentwicklung in einem Jahr liefern: das ist MAD.

Nachdem wir nun einen kurzen Blick auf MAD aus einer theoretischen Perspektive geworfen haben, lassen Sie uns etwas tiefer einsteigen und MAD mit dem vergleichen, was gemeinhin als Cloud Native bezeichnet wird.

MAD vs. Cloud Native

In der Vergangenheit stellten IT- und Operations-Teams Server mit Betriebssystemen und Anwendungen bereit. Kabel, Switches, Router, Load Balancer und vieles mehr wurden physisch bereitgestellt und implementiert. Heute kann all dies in Codezeilen abstrahiert werden, die alles bereitstellen, was für die Ausführung einer App und die Bereitstellung eines Service nötig ist.

Im Vergleich zu MAD ist Cloud Native eine Methode zur Entwicklung und Ausführung von Anwendungen, die die Vorteile des Cloud-Computing-Deployments nutzt. Die Softwareentwicklung in einer Cloud-Native-Umgebung setzt auf bewährte DevOps-Ansätze und macht sich die Grundlagen von Continuous Integration, Continuous Delivery und Continuous Deployment (CI/CD) zunutze. Dabei kann Cloud Native oft als Teilmenge von MAD betrachtet werden, da nicht alle Projekte vollständig für Cloud Native konzipiert sind. Software, die On-Premises, in der Cloud oder hybrid bereitgestellt wird, kann dennoch nach den Prinzipien von MAD entwickelt werden.

Die Cloud Native Computing Foundation sagt: „Cloud Native-Technologien ermöglichen es, skalierbare Anwendungen in modernen und dynamischen Umgebungen wie Public Clouds, Private Clouds und Hybrid Clouds zu entwickeln und auszuführen. Container, Service Meshes, Microservices, unveränderliche Infrastruktur und deklarative APIs sind Beispiele für diesen Ansatz. Sie ermöglichen lose gekoppelte Systeme, die widerstandsfähig, einfach zu managen und beobachtbar sind. In Kombination mit einer robusten Automatisierung ermöglichen sie es, tiefgreifende Änderungen schnell, planbar und mit minimalem Aufwand vorzunehmen.“[1]

Obwohl Cloud Native eine Komponente von MAD ist, sollten wir die Faktoren beleuchten, die MAD in den Fokus innovativer Unternehmen rücken.

[1] „Who we are,” Cloud Native Computing Foundation, abgerufen am 30. August 2021, https://www.cncf.io/about/who-we-are.

MAD – Warum jetzt?

Die traditionelle Softwareentwicklung wird oft mit dem Zusammensetzen eines Zuges in einem Rangierbahnhof verglichen: Waggon für Waggon wird an eine Lokomotive gekoppelt, bis ein langer Zug entsteht. Die Verbindung der Waggons, die Sicherstellung ihrer Betriebsbereitschaft bis hin zur Freigabe der Fahrt ist ein langwieriger und kostspieliger Prozess – einfach ausgedrückt kann der Zug aber erst dann in Richtung seines Ziels aufbrechen, wenn dieser fertiggestellt ist.

Traditionelle Softwareentwicklungsmethoden sahen ähnlich aus: Unternehmen, die schnell neue Software bereitstellen wollten, mussten Verzögerungen in Kauf nehmen: Teams steckten Aufwand in Low-Value Deliverables und warteten auf den Abschluss anderer interner Tasks – Verzögerungen, die wenig mit der Softwareentwicklung zu tun hatten. MAD ersetzt lineare Abläufe zugunsten eines parallelen Bereitstellungsansatzes. Denn die heutige wettbewerbsorientierte Marktlandschaft erfordert in puncto Time-to-Market vor allem eines: Schnelligkeit.

MAD ermöglicht es Entwicklern, die Vorteile nativer Cloud-Technologien zu nutzen, um den Designprozess zu beschleunigen, und wendet dann dieselbe Logik auf die Phasen Modellierung, Entwicklung und Bereitstellung an. Durch den komponentenbasierten Ansatz zur Optimierung der Softwareentwicklung, Beschleunigung der Bereitstellung und Verbesserung der Qualität ihrer Deliverables macht es MAD Entwicklern leicht, innovativ zu sein. Es reißt die Mauern zwischen Entwicklung, Betrieb und Infrastruktur ein, und mit der voranschreitenden Digitalisierung beginnen Unternehmen, MAD – und die Veränderungen der traditionellen Kultur, die damit einhergehen – immer besser zu verstehen. Lassen Sie uns einen Blick auf diese Veränderungen werfen.

MAD – Kultur

Früher gingen Unternehmen davon aus, dass ihre Kunden alles, was sie entwickeln, auch kaufen. In der heutigen Welt der Software und Services ist das nicht mehr der Fall. MAD kehrt das Konzept um und stellt den Kunden in den Mittelpunkt. Unternehmen treffen ihre Geschäftsentscheidungen also auf der Grundlage eines konstanten Flusses von Kundenfeedback. So können sie ihre Produkte und Services kontinuierlich verbessern, um die Erwartungen der Kunden zu erfüllen. Welche Art von Kultur braucht MAD vor diesem Hintergrund?

Der wichtigste kulturelle Wandel, der für eine erfolgreiche MAD-Umgebung erforderlich ist, liegt darin, Innovation durch Eigenverantwortung zu ermöglichen. In der Vergangenheit arbeiteten Entwickler an „Projekten“: Sie erhielten eine Aufgabe und erledigten sie. Im MAD übernehmen die Entwickler die Verantwortung für die Produkte, die sie liefern sollen, und für die Entwicklung des Produkts und seinen Erfolg. Sie bestimmen selbst, wie und wo ihr Produkt läuft, halten es up-to-date und verbessern es, um mit den Kundenanforderungen Schritt zu halten. So können Entwickler in einer Atmosphäre arbeiten, in der sie experimentieren können, anstatt sich um Aspekte zu sorgen, die keinen Business Value generieren.

Der kulturelle Wandel fördert Eigenverantwortung für innovative Ergebnisse. Wenn Entwickler die Möglichkeit haben, mit mehr Autonomie neue und aufregende Ergebnisse zu liefern, öffnet das die Tür für mehr Kreativität, die sich in einer zunehmend risikobereiten Kultur entfalten kann. Vertrauen ist der Kern von MAD: Entwickler haben nicht mehr das Gefühl, nur ein Teil des Puzzles zu sein – sie sind die Designer des Puzzles, und es liegt letztlich an ihnen, wie alle Teile zusammenpassen. MAD kann eine Kultur schaffen, in der Entwickler gerne arbeiten und sich wohlfühlen – und das erfordert ein neues Architekturmodell. Schauen wir uns dieses Konzept als Nächstes an.

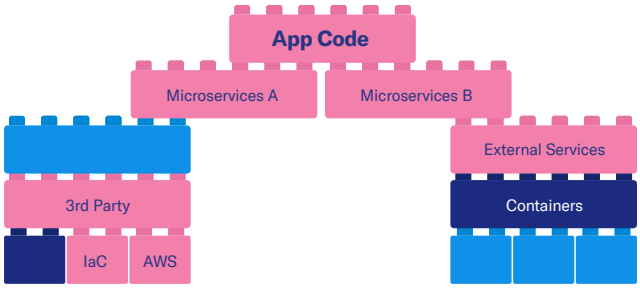
MAD – Architektur

Im Zentrum von MAD steht ein iteratives Mindset: MAD etabliert interne und externe Feedbackschleifen und bindet diese – dank zunehmender Automatisierung – nativ in den Entwicklungsprozess ein. Bei MAD dreht sich alles um die Qualität des Endprodukts, kontinuierliche Funktionstests und Application Security Testing sind unverzichtbar.

MAD stützt sich nicht auf einen monolithischen, linearen Ansatz für die Softwareentwicklung und -bereitstellung. Stattdessen könnte man es mit einem Gebilde aus Legosteinen vergleichen: Jeder Baustein hat einen etwas anderen Zweck, aber sie passen alle perfekt zusammen, weil sie mit dieser Absicht konzipiert wurden. Die Komponenten von MAD, die diese neue Architektur unterstützen, sind:

- **Anwendungscode**
  - Open-Source-(Third-Party)-Code
  - Microservices/Serverless Code
  - API-Code
- **Container-Code**
- **Infrastructure-as-Code**

Abbildung 2 zeigt, wie in der „legolisierten“ MAD-Architektur jedes Teil einen Zweck erfüllt und alle Teile ineinandergreifen. Die Entwickler können die Codeteile bei Bedarf jederzeit aktualisieren, verbessern und bereitstellen – dabei ist die Geschwindigkeit und Flexibilität, die MAD bietet, ein echter Game Changer.



Nachdem wir nun viele der positiven Eigenschaften von MAD betrachtet haben, sollten wir uns auch mit den Herausforderungen befassen, vor denen Unternehmen bei der Umsetzung ihrer MAD-Projekte stehen.

MAD – Herausforderungen

Ja, MAD gilt als revolutionär – das Ergebnis ist aber immer noch Software. Input und Output, Networking, Storage, Databases und so weiter existieren immer noch. Entwickler genießen die Freiheit, die MAD bietet, aber während sie sich mehr und mehr an die Verwendung von Development Frameworks gewöhnen, nimmt auch der Abstraktionsgrad zu. Mit anderen Worten: Funktioniert eine Anwendung nicht wie gewünscht, wissen Entwickler oft nicht sofort, wo sie ansetzen sollen – denn sie können nicht nachvollziehen, wie diese Anwendung in der Vergangenheit funktioniert hat.

Als Code noch in COBOL geschrieben und ein COBOL Assembler verwendet wurde, konnten Entwickler, die mit dem Code vertraut waren, Probleme schnell beheben. Mit MAD nimmt der Abstraktionsgrad weiter zu. Ein Beispiel sind JavaScript Frameworks: Es gibt so viele Abstraktionen, dass das Troubleshooting – wenn etwas nicht funktioniert – mitunter viel Zeit in Anspruch nimmt. Entwickler müssen Abhängigkeiten, gegenseitige Abhängigkeiten und Abhängigkeiten von Abhängigkeiten vollständig nachvollziehen, was eine große Herausforderung darstellen kann.

Früher wurden Anwendungen, die für den Betrieb in einem Three-Tier-Datacenter gedacht waren, oft von einem einzigen Team entwickelt. Und analog dazu war auch ein AppSec-Team für die Sicherheit und ein Operations-Team für den Support der Anwendung zuständig. Heute ist alles dezentralisiert, und jedes Framework stellt eigene Anforderungen an Entwicklung, Sicherheit und Bereitstellung. Im MAD können Entwickler die Frameworks und Sprachen wählen, die sie verwenden möchten, was die Gesamtkomplexität des Endprodukts weiter erhöht. Bevor wir fortfahren, sollten wir daher einen genauen Blick auf das Thema Abstraktion im Zusammenhang mit MAD werfen.

MAD – Abstraktion

Erinnern Sie sich noch an die Zeiten, in denen man ein Datacenter-Netzwerk mit Hilfe eines Visio-Diagramms aufbaute und – streng nach Vorlage – Hunderte von Cat-5-Kabeln an die Switches, Router, Load Balancer und Server im 19-Zoll-Rack anschloss? Heute lässt sich dieser Prozess mit dem simplen Aufruf einer Infrastructure-as-Code-Funktion abbilden. Das ist die Art von Abstraktion, von der wir hier sprechen.

Angenommen, ein Unternehmen stellt fest, dass sein Amazon Elastic Block Store (EBS) aufgrund einer kürzlich vorgenommenen Änderung nicht mehr mit der Amazon Elastic Compute Cloud (EC2) funktioniert. Da Entwickler vollkommen losgelöst vom Endresultat agieren und die Detailkenntnisse fehlen, weiß niemand, wie man das Problem in den Griff bekommt. Verlagert sich die Entwicklung dann auch noch von der Verwendung eines Assemblers auf Golang, haben wir es mit Abstraktionen von Abstraktionen zu tun. Wenn dann ein Problem in der Produktion auftritt, hört man häufig: „Ich kenne mich nur mit diesem Layer aus, mit dem Rest nicht."

Aufgrund des zunehmenden Abstraktionsgrads haben Entwickler häufig keinen echten Überblick über die Security mehr – angefangen bei den Basics auf dem Network Layer. Statt auf der Basis eines umfassenden Verständnisses von Ports, Protokollen und Network Layer robuste Ansätze zu entwickeln, definieren sie weiche Policies – sprich: Sie lassen die Tür zum Netzwerk sperrangelweit offen stehen. Dies sind nur einige der Herausforderungen, die es bei MAD zu berücksichtigen gilt. Sehen wir uns noch ein paar weitere an.

MAD Skalierbarkeit und Transparenz

Auch wenn Geschwindigkeit bei MAD das A und O ist, dürfen Skalierbarkeit und Transparenz nicht vernachlässigt werden. Unternehmen stehen vor der Frage, wie sie bei MAD skalieren können – denn ältere Verfahren sind oft denkbar ungeeignet. In MAD-Umgebungen können Provisionierung, Support, Feintuning, Troubleshooting und so weiter mehr Zeit in Anspruch nehmen. Stellt ein Unternehmen zum Beispiel auf Cloud Native um, können der Support-Bedarf und die Komplexität extrem hoch sein.

Neben der Skalierbarkeit gilt es auch die Transparenz im Blick zu haben. Gibt es Stabilitätsprobleme – zum Beispiel Prozesse, die online und schnell wieder offline gehen – und ist kein Observability Layer vorhanden, werden sich viele Teams fragen: „Was ist da gerade passiert?“ Ihnen fehlen schlichtweg die Details, um einen reibungslosen Ablauf zu gewährleisten. Kurzum: Die kompliziertesten Bestandteile einiger neuerer Architekturen sind die Logging- und Observability-Funktionen.

Bei der Überwachung einer monolithischen Anwendung, die in einem Data Center läuft, lässt sich der gesamte Stack einfach überblicken. Mithilfe von Monitoring-Tools können Entwickler, Security- und Operations-Teams einfach, schnell und zuverlässig einen Überblick aus der Vogelperspektive erhalten. Bei MAD sieht es mit der Transparenz völlig anders aus. Nehmen Sie zum Beispiel Microservices. Wenn potenziell Hunderte von Microservices (wenn nicht mehr!) für eine einzige Anwendung ausgeführt werden und viele von ihnen von vor- oder nachgelagerten Services abhängen, ist lückenlose Transparenz das A und O – unabhängig davon, wo oder wie sie bereitgestellt werden.

Im MAD benötigen Unternehmen eine umfassende Single-Pane-of-Glass-Ansicht, die alle laufenden Statistiken und Metriken vollständig korrelieren kann, um im Laufe der Zeit bei einer stabilen Anwendung anzugelangen. Transparenz auf App-Ebene, die Rückschlüsse auf die Customer Experience zulässt, ist optimal. Response Rates, Error Rates, Logs, Traces, Metriken und vieles mehr sind wichtige Bestandteile einer vollständigen Transparenz über ein System. Neben Skalierbarkeit und Beobachtbarkeit gibt es noch eine weitere MAD-typische Herausforderung: die Tool-Vielfalt.

MAD – Wildwuchs von Werkzeugen

Wenn Sie eine MAD-Initiative in Angriff nehmen, hegen Sie wahrscheinlich die Hoffnung, dass MAD die Anzahl der für die Entwicklung und Absicherung ihres Codes eingesetzten Tools reduzieren wird – genau das Gegenteil ist der Fall. In der Vergangenheit benötigten Sie nur eine Handvoll IDE-, CI/CD-, SCM- und AppSec-Tools. Nach dem Wechsel auf MAD heißt es: „Wir brauchen 15 neue Tools, die wir gestern noch nicht gebraucht haben." Und das ist nur die Spitze des Eisbergs.

Nehmen Sie zum Beispiel Kubernetes. Viele Entwickler wissen gerade genug über Container, um sie verstärkt zu nutzen. Das nächste große Problem entsteht, wenn Entwickler Kubernetes erweitern und Open Source einbeziehen. Plötzlich hört das Management: „Wir brauchen ein weiteres Tool (oder zwei) in unserem Tool-Kit, damit das funktioniert."

Jeder ist dabei überzeugt, dass seine Tools die besten sind. Aber Unternehmen müssen trotzdem standardisieren – um Kosten zu kontrollieren und zu senken – und Standardisierung ist nicht so einfach, wie sie klingt. Schließlich will man den Teams die Flexibilität bieten, die sie sich wünschen, um so agil wie möglich arbeiten zu können. Jeder will seine eigenen Tools und bald sind so viele im Einsatz, dass sie kaum zu managen sind. Nachdem wir uns nun einige organisatorische Herausforderungen vergegenwärtigt haben, die mit MAD einhergehen, sollten wir über die neuen Sicherheitsrisiken sprechen, die mit MAD ins Spiel kommen.

MAD – Sicherheitsrisiken

Im Kontext Ihres Unternehmens stehen Sicherheitsrisiken in Form von Datenverlust, Breaches und Ransomware ganz oben auf der Agenda. Bei Software stellen hauptsächlich ausnutzbare Schwachstellen in der Produktivumgebung ein Sicherheitsrisiko dar. In der Regel haben Unternehmen daher Application-Security-Testing-Verfahren etabliert, um Software-Risiken, wie sie in den OWASP Top 10, CWE/SANS Top 25 und anderen vergleichbaren Listen aufgeführt sind, zu erkennen und zu minimieren.

Diese Listen bilden eine hervorragende Ausgangsbasis, um Entwickler und AppSec-Experten auf Secure Coding Practices aufmerksam zu machen und Software-Risiken zu minimieren. Speziell im Hinblick auf MAD kommen jedoch eine Reihe neuer Risiken hinzu, die die Risikolandschaft über die traditionellen, webbasierten Software-Risiken der OWASP Top 10 hinaus ausweiten.

Unternehmen müssen also nicht nur die von OWASP (und anderen) gelisteten Risiken berücksichtigen, sondern auch eine Reihe völlig neuer Risiken, die im Zuge von MAD-Initiativen auftreten:

> Open Source Code

- Inkonstistente Security-Standards
- Unbekannte Quellcode-Herkunft
- Lizenzierungsverstöße

> Microservices

- Zunehmende Komplexität
- Begrenzte Kontrolle der Umgebung
- Unzureichender Schutz der Daten
- Unzureichender Schutz des Netzwerks

> Container

- Container aus unsicheren Quellen
- Gefährdung sensibler Daten durch Container Images
- Zu viel Vertrauen in Image Scanning
- Größere Angriffsfläche
- Aufgeblähte Base Images
- Fehlende strikte Isolierung
- Weniger Transparenz

> Infrastructure-as-Code

- Steile Lernkurve
- Menschliches Versagen
- Configuration Drifts
- Gefährdung sensibler Daten und Ports

> APIs[2]

- Broken Object Level Authorization
- Broken User Authentication
- Übermässige Exponierung von Daten
- Keine Raten- & Ressourcenbegrenzung
- Broken Function Level Authorization
- Massenhafte Attributzuweisung
- Fehlerhafte Security-Konfiguration
- Anfälligkeit für Injections
- Improper Assets Management
- Unzureichendes Logging & Monitoring

[2] "OWASP API Security Top 10 2019," OWASP, abgerufen am 30. August 2021, https://owasp.org/www-project-api-security.

OWASP Top 10 Web Application
Security Risks

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Einige dieser neuen Risiken lassen sich durch die Implementierung besserer Security-Policies, -Prozesse und -Verfahren adressieren. Andere jedoch erfordern neue AST-Lösungen, die einen plattformbasierten Ansatz mit Correlation Layer verwenden, der speziell auf neu aufkommende MAD-Risiken zugeschnitten ist. Unternehmen, die den Übergang zu MAD vollziehen, werden schnell erkennen, dass ihre derzeitigen Security-Testing-Ansätze sich in diesem Modell nicht immer bewähren.

MAD – Nächste Schritte

Wir haben in diesem eBook zunächst die Grundlagen des Modern Application Development betrachtet und erklärt, warum viele Unternehmen weltweit diese neue Methodik anwenden. Wir haben uns die Kultur angesehen, die es für MAD braucht, ebenso wie einige architektonische Komponenten – und hoffentlich gezeigt, dass MAD zwar viele Vorteile bietet, aber wie alles, was Veränderungen mit sich bringt, auch Herausforderungen parat hält.

Anschließend haben wir uns mit diesen Herausforderungen befasst und einige der Probleme aufgezeigt, die wahrscheinlich auftreten werden. Wir haben auch die zunehmenden Sicherheitsrisiken aufgeführt, mit denen sich Unternehmen aufgrund der Funktionsweise moderner Anwendungen konfrontiert sehen. Wir versuchen keineswegs, Unternehmen von der Umstellung auf MAD abzubringen. Ganz im Gegenteil, wir unterstützen MAD-Initiativen voll und ganz – wir bei Checkmarx verwenden dieselben Ansätze. Unsere AST-Lösungen sind von Entwicklern für Entwickler gemacht und kommen sowohl in traditionellen als auch in modernen Entwicklungsumgebungen zum Einsatz.

Da das Konzept ziemlich komplex ist, kann MAD für viel Diskussionsstoff sorgen – aber jetzt sollten Sie eine bessere Vorstellung davon haben, worum es bei MAD geht. In Teil 2 dieses eBooks werden wir alle in Teil 1 erwähnten sicherheitsrelevanten Risiken näher beleuchten und Empfehlungen geben, wie man sie managen und minimieren kann. Für den Moment reicht es aus, anzuerkennen, dass es eine ganze Reihe neuer Risiken gibt.

Über Checkmarx

Checkmarx setzt im Application Security Testing immer neue Maßstäbe, um Security für Entwickler auf der ganzen Welt einfach und intuitiv zu halten und CISOs das notwendige Vertrauen und die richtigen Werkzeuge an die Hand zu geben. Als Marktführer im Bereich AppSec-Testing entwickeln wir bedienfreundliche Lösungen, die Developern und Security-Teams höchste Zuverlässigkeit, einen breiten Leistungsumfang, lückenlose Transparenz und handlungsrelevante Hinweise für die Behebung gefährlicher Schwachstellen in allen Komponenten moderner Software bieten – sowohl im eigenen Code als auch in Open Source, APIs und Infrastructure-as-Code. Mehr als 1.600 Kunden, darunter nahezu die Hälfte der Fortune 50, verlassen sich auf unsere Security-Technologie, unsere Security Research und unsere globalen Services, um sicher, schnell und skaliert zu entwickeln. Für mehr Informationen besuchen Sie unsere Website, lesen unseren Blog oder folgen uns auf LinkedIn..

Checkmarx

Checkmarx auf einen Blick

1,675+
Kunden in 70 Ländern

750
Mitarbeiter in 25 Ländern

45%
der Fortune 50 verlassen sich auf uns

30+
Sprachen & Frameworks

500k+
KICS-Downloads in 2021

The world runs on code. We secure it.