KAPITEL 1

# Prozesse für sichere Entwicklungszyklen

**Am Ende dieses Kapitels**

- verstehen Sie einige der Prozesse, die für die Entwicklung sicherer Software erforderlich sind.
- können Sie innerhalb Ihrer Organisation dazu beitragen, eine Sicherheitskultur zu entwickeln.
- sind Sie in der Lage, den Zweck der verschiedenen Arten von Umgebungen für die Entwicklung bis hin zur Produktion zu erläutern und welche unterschiedlichen Sicherheitsmaßnahmen sie erfordern.

## Entwickler sind die Hauptursache für Kompromittierungen

Die Hauptursache für Gefährdungen sind nicht Hacker, Angreifer oder andere ruchlose Akteure. Vielmehr sind wir, die Softwareentwickler, die Hauptursache für Sicherheitslücken. Laut einer Analyse von Contrast Security aus dem Jahr 2020 sind fast 50 Prozent aller Kompromittierungen auf Schwachstellen in Anwendungen zurückzuführen – Schwachstellen, die letztlich von Softwareentwicklern geschaffen wurden. Eine Zusammenfassung des Berichts können Sie hier lesen: *https://azsec.tech/lvz*.

Als Softwareentwickler können wir nicht viel gegen Angriffe tun. Sie werden auf jeden Fall stattfinden. Was wir aber tun können, ist, die Sicherheit unseres Codes zu verbessern. Wir kommen nicht darum herum, dass das Design unseres Systems und die Qualität unseres Codes den Unterschied zwischen einem fehlgeschlagenen und einem erfolgreichen Angriff ausmachen können. Wir müssen die Art und Weise ändern, wie wir Software entwerfen und entwickeln, um die Sicherheit so nahtlos wie möglich und mit so wenig Reibungsverlusten wie möglich zu verbessern. Das entscheidende Wort hier ist *Reibung*. Sicherheit wird oft als eine Art Steuer angesehen, die Entwickler zahlen müssen und die die Entwicklung behindert. Sie steht einfach im Weg. Wir müssen Prozesse und Aufgaben einbeziehen, die die Sicherheit erhöhen, die so reibungslos wie möglich sind und einfach als ein weiterer wichtiger Aspekt bei der Erledigung der Aufgabe angesehen werden.

Natürlich sind auch Werkzeuge wichtig. Aber sie sollten nicht blindlings eingesetzt oder als einzige Quelle für die Sicherheit Ihrer Lösung betrachtet werden. Ganz gleich, wie viele Tools oder Automatisierungsfunktionen Sie bei Ihren Entwicklungsverfahren einsetzen, letztendlich sind es Menschen, die Software erstellen, und auch Ihre Sicherheitslage hängt von Menschen

ab. Wie das Sprichwort sagt: »Ein Dummkopf mit einem Werkzeug ist immer noch ein Dummkopf.« Wir müssen also nicht nur in die neuesten Sicherheitstools investieren, sondern auch in menschliches Sicherheitskapital und Prozesse.

**Dieses Kapitel beschäftigt sich sowohl mit dem Prozess und den menschlichen Aspekten der Praktiken für die Softwareentwicklung; aber auch die technischen Aspekte kommen nicht zu kurz. Das Ziel ist, wie erwähnt, bei der Bereitstellung von sicheren Softwarelösungen so reibungslos zu sein wie möglich.**

# Einführung in den Microsoft Security Development Lifecycle

Der Microsoft Security Development Lifecycle (SDL) entstand Anfang der 2000er-Jahre und wurde im Laufe der Jahre angewandt und angepasst. Ein Sprichwort sagt: »Es gibt nichts Neues unter der Sonne«, und das trifft auf den SDL zu. Der SDL unterscheidet sich jedoch durch die Menge an unterstützender Dokumentation, Werkzeugen, Forschungsergebnissen und Vordenkern, die Microsoft öffentlich zugänglich gemacht hat.

Was also ist der SDL? Der SDL besteht aus einer Reihe von Praktiken zur Verbesserung der Softwaresicherheit. Er verfolgt zwei übergreifende Ziele:

- Verringerung der Anzahl von Sicherheitslücken in Ihrem Code
- Verringerung der Schwere der Schwachstellen, die Sie übersehen

Diese beiden Ziele führen zu einer gewissen Spannung in Ihrer Sicherheitsstrategie. Sie möchten die sicherste Software entwickeln, müssen aber gleichzeitig damit rechnen, dass Ihnen etwas entgeht und dass sich die Strategien der Angreifer mit der Zeit weiterentwickeln. Was heute sicher und korrekt ist, kann morgen angreifbar sein.

**Um Ihr Wissen zu vervollständigen, empfehlen wir Ihnen, die vollständige Liste der SDL-Anforderungen zu lesen, die Sie hier finden:**
***https://www.microsoft.com/securityengineering/sdl/practices*.**

## Qualität ≠ Sicherheit

Wir hören oft, dass Leute sagen: »Wenn man die Qualität verbessert, dann verbessert sich auch die Sicherheit.« Diese Aussage klingt zwar plausibel, aber es gibt keine Beweise, die diese Aussage stützen. Keine. Softwarequalitätsprogramme finden nur selten Sicherheitsprobleme, denn Sicherheitsprobleme sind etwas anderes als Qualitätsprobleme. Außerdem wird Sicherheit, wie wir später in diesem Buch erörtern, oft als »zusätzliche« Funktionalität definiert.

Angenommen, Sie haben eine Anwendung, die lediglich die folgenden Datenbankoperationen durchführt:

- Hinzufügen eines neuen Benutzers (Erstellen/*Create* in CRUD)
- Lesen der Details eines Benutzers (Lesen/*Read* in CRUD)
- Bearbeitung der Benutzerdaten (Aktualisierung/*Update* in CRUD)
- Löschen eines Benutzers (Löschen/*Delete* in CRUD)
- Drucken der Benutzerdaten

Sie erstellen einige Tests, die erfolgreich oder (absichtlich!) fehlschlagen sollen, und überprüfen dann diese Erfolge und Fehlschläge. Wenn alle Tests, die den Erfolg überprüfen sollen, erfolgreich sind und alle Tests, die das Scheitern überprüfen sollen, pflichtgemäß scheitern, könnte man zu dem Schluss kommen, dass die Anwendung keine Fehler aufweist. Dies ist jedoch nicht der Fall. Wenn die Anwendung beispielsweise eine SQL-Injection-Schwachstelle aufweist, die es einem Tester (oder Angreifer) ermöglicht, alle Benutzer zu lesen oder eine Datenbanktabelle zu löschen, wird die Anwendung dennoch alle Erfolgstests bestehen und alle Fehlertests nicht bestehen. Die Moral von der Geschichte ist, dass Sie Ihren Softwareentwicklungsprozessen Sicherheit als eigenen Faktor hinzufügen *müssen*.

## Sicherungsmerkmale vs. Sicherheitsmerkmale

Das Microsoft SDL konzentriert sich auf die Sicherung Ihrer Software, nicht nur auf das Hinzufügen weiterer Sicherheitsfunktionen. Sicherheitsfunktionen sind wichtig, aber Sie können nicht einfach jedes beliebige Sicherheitsprodukt in Ihre Lösung einbauen und sie als sicher bezeichnen. Die Funktionen, die Sie Ihren Lösungen hinzufügen, müssen ebenfalls sicher sein. Diese philosophische Perspektive stellt einen wichtigen Sinneswandel für viele dar, die glauben, sie könnten ein Produkt kaufen und es als erledigt betrachten – vor allem, wenn so viele Unternehmen ihre Produkte als Allheilmittel verkaufen.

**Kurz, *Sie* müssen sich auf die Disziplin der Softwareentwicklungssicherheit konzentrieren. Sie können diese Verantwortung nicht auf ein Produkt abwälzen.**

# SDL-Komponenten

Die wichtigsten Aufgaben und Anforderungen von Microsoft SDL sind wie folgt:

- Sicherheitsschulung
- Definieren Ihrer Bug Bar
- Analyse der Angriffsfläche
- Modellierung von Bedrohungen
- Definieren Ihrer Toolchain
- Verbotene Funktionalität vermeiden

- Werkzeuge zur statischen Analyse verwenden
- Dynamische Analysetools verwenden
- Review des Sicherheitscodes
- Reaktionsplan bei Zwischenfällen zur Hand haben
- Penetrationstests durchführen

Schauen wir die einzelnen Punkte genauer an.

### Agile SDL

Sie werden vielleicht denken, dass diese Liste mit Anforderungen vor allem für ein Wasserfallmodell gilt. Tatsächlich handelt es sich aber nur um eine Liste von Aufgaben, die erledigt werden müssen; sie gibt nicht an, wann Sie sie erledigen müssen. Es kann sein, dass Sie nur einen Sprint damit verbringen, an einigen Aufgaben zu arbeiten, und diese Arbeit wird dann für alle zukünftigen Sprints gelten. Wir werden erläutern, wie man jede dieser Aufgaben in einer agilen Umgebung am besten anwendet.

## Sicherheitsschulung

Sicherheitsschulungen sind ein Muss. Aber mit Sicherheitsschulung meinen wir nicht die Schulung »Passwörter nicht wiederverwenden!«, sondern die Schulung »Sicherheit bei der Anwendungsentwicklung«.

Lange Zeit verlangte Microsoft von allen *technischen* Mitarbeitern, dass sie an allgemeinen Sicherheitsschulungen teilnehmen, und die Mitarbeiter können dies auch heute noch tun, wenn sie wollen. Heutzutage verwenden wir jedoch ein schlankeres Schulungsmodell, das eher bereichsspezifisch ist. So verlangen wir zum Beispiel von unseren technischen Mitarbeitern, dass sie sicherheitsrelevante Kurse besuchen, die sich auf ihre Rolle beziehen, anstatt eine allgemeine Schulung zu absolvieren.

Wenn ein Entwickler beispielsweise serverseitigen Node.js-JavaScript-Code für eine Webanwendung schreibt, muss er Cross-Site-Scripting-Probleme (XSS), die sichere Verwendung von Cookies und andere Web- und HTTP-bezogene Sicherheitsprobleme und Abwehrmaßnahmen verstehen. Wenn dieselbe Anwendung mit einer SQL-Datenbank kommuniziert, sind solide Kenntnisse über SQL Injection, Datenbankverbindungen mit geringsten Rechten und das sichere Speichern von Verbindungszeichenfolgen ebenfalls wichtig. Es ist jedoch sehr wahrscheinlich, dass dieser Entwickler die potenziellen Probleme der Speicherbeschädigung bei der Verwendung von `strcpy()` in C nicht verstehen muss. Er könnte also einfach lesen, ein Video ansehen oder eine Onlineschulung zu den für ihn wichtigen Themen absolvieren.

**Es gibt mehrere Schulungsmodi, daher sollten Sie den Modus wählen, der für Sie als Entwickler, Architekt oder Tester am besten geeignet ist. Unterschätzen Sie jedoch nie den Wert eines kurzen Videos, das den Kern des Problems auf den Punkt bringt!**

Aus agiler Sicht sollte es niemandem erlaubt werden, an einem Sprint mitzuarbeiten, solange er nicht über ein Grundniveau an Wissen über Sicherheit verfügt. Vergewissern Sie sich, dass alle Mitglieder des Entwicklungsteams über die Anforderungen des Unternehmens an die Erstellung sicherer Software informiert sind. Wenn es in Ihrem Unternehmen keine Liste der empfohlenen Ressourcen für den Entwurf und die Entwicklung sicherer Anwendungen gibt, sollten Sie eine erstellen. Sie könnten sogar in Erwägung ziehen, dieses Buch zur Pflichtlektüre für alle Ihre technischen Mitarbeiter zu machen.

### Praktische Erfahrungen: »Pflichtlektüre«

Als Michael (Howard) zusammen mit David LeBlanc »Writing Secure Code« (die deutsche Ausgabe erschien unter dem Titel »Sichere Software programmieren«) schrieb, las Bill Gates das Buch und erklärte die zweite Auflage zur Pflichtlektüre bei Microsoft. Dadurch wurde der kollektive Sicherheits-IQ im gesamten Unternehmen schnell erhöht.

## Definieren Ihrer Bug Bar, Ihres Klassifizierungsschemas

Nicht alle Sicherheitslücken sind gleich; die Festlegung einer Methode zur Berechnung des Schweregrads Ihrer Sicherheitslücken ist entscheidend. Es gibt viele Möglichkeiten, den Schweregrad zu berechnen, aber die gängige Methode ist das Common Vulnerability Scoring System (CVSS). Laut NIST ist das CVSS »ein offenes Framework für die Kommunikation der Merkmale und des Schweregrads von Softwareschwachstellen«. Das CVSS gibt das mit einer bestimmten Schwachstelle verbundene Risiko anhand einer numerischen Punktzahl an, die von 1 (niedrig) bis 10 (kritisch) reicht.

**First.org stellt eine Liste von CVSS-Beispielen zur Verfügung, mit deren Hilfe Sie lernen können, wie man Schwachstellen bewertet. Sie finden sie hier: *https://azsec.tech/wt7*.**

Kapitel 8, »Compliance und Risikomanagement«, behandelt CVSS ausführlich. Im Moment ist es wichtig, dass Sie definieren, was die CVSS-Risiko-Scores für Sie bedeuten. Nehmen wir zum Beispiel an, Sie sind dabei, einen Code in die Produktion zu überführen, aber Sie stellen fest, dass er eine Sicherheitslücke mit einem CVSS-Score von 7,3 enthält. Was sollten Sie tun? Hätte dieselbe Schwachstelle einen CVSS-Wert von 1,7, wäre es ein Bug mit geringem Risiko – kein Fehler, der Sie davon abhalten sollte, den Code in Produktion zu geben, sondern etwas, das Sie wahrscheinlich irgendwann einmal beheben sollten. Ein Wert von 7,3 stellt eher ein Dilemma dar. Sollten Sie den Code in die Produktion überführen und den Fehler danach schnell beheben? Oder sollten Sie die Bereitstellung um ein oder zwei Tage verschieben, um die Fehlerbehebung sofort vorzunehmen und sie vorher zu testen? Es ist eine schwierige Entscheidung, aber im Allgemeinen ist es am besten, nach dem Motto »Vorsicht ist die Mutter der Porzellankiste« zu handeln.

KAPITEL 2

# Sicherer Entwurf

**Am Ende dieses Kapitels**

- können Sie bei der Entwicklung von Cloudlösungen Sicherheitsaspekte in die DevOps-Prozesse Ihres Unternehmens integrieren.
- sind Sie in der Lage, grundlegende Aspekte des Zero Trust-Konzepts in einem Entwicklungskontext anzuwenden.
- können Sie Verbesserungsmöglichkeiten bei der sicheren Gestaltung von Lösungen in der Cloud ermitteln.
- können Sie bei der Entwicklung von Azure-Lösungen grundlegende Sicherheitsdesignprinzipien anwenden.

## Die Cloud, DevOps und Sicherheit

Die Cloud hat die Art und Weise, wie wir Lösungen entwickeln, revolutioniert. Unternehmen nutzen die Cloud wegen ihrer Flexibilität und ihrer Agilität. Dies hat Entwicklungsteams dazu gezwungen, die Art und Weise, wie sie Lösungen implementieren, weiterzuentwickeln, um die Versprechen der Cloud zu erfüllen. Aus diesem Grund ist es fast eine Notwendigkeit, bei der Entwicklung von Lösungen für die Cloud DevOps und agile Ansätze einzusetzen.

Agile ist mehr eine Philosophie als eine einzelne Methode zur Softwareentwicklung. Es handelt sich um eine Reihe von Grundsätzen und Techniken zur Verbesserung des Softwareentwicklungsprozesses, indem die Fähigkeit zur Anpassung an Veränderungen und schnelleren Bereitstellung von Werten verbessert wird. Beispiele für agile Frameworks sind Scrum und Extreme Programming. Die wichtigsten Merkmale der agilen Entwicklung werden hier erörtert: *https://agilemanifesto.org*.

Bei DevOps handelt es sich um eine Reihe von Praktiken, die darauf abzielen, den Lebenszyklus der Software- und Systementwicklung zu beschleunigen, indem die Entwicklung mit den IT-Aktivitäten verbunden wird. Es beinhaltet und unterstützt Prinzipien wie kontinuierliche Integration (continuous integration, CI) und kontinuierliche Bereitstellung (continuous delivery, CD), um die Zeit zu verkürzen, die für die Bereitstellung von Updates für Produktionssysteme erforderlich ist. In seiner reinsten Form ermöglicht DevOps, dass täglich Code in die Produktion übergeht, möglicherweise sogar mehrmals täglich.

Ein Hauptprinzip von DevOps besteht darin, alltägliche Aufgaben wie Erstellung, Test und Bereitstellung durch Automatisierung zu beschleunigen, um sie vorhersehbarer und wiederholbar zu machen. Ein Grundsatz von iterativen und agilen Prozessen wie Scrum ist es, mit jeder

Iteration einen Mehrwert zu schaffen. Wenn diese Methoden richtig angewendet werden, führen sie zur frühen Erstellung begrenzter, aber funktionierender Produkte mit schnellen inkrementellen Verbesserungen.

Üblicherweise werden spezialisierte Sicherheitstools eingesetzt, um den Code automatisch zu analysieren. Eine Beschreibung einiger gängiger Kategorien finden Sie in Kapitel 1, »Prozesse für sichere Entwicklungszyklen«. Diese Tools bewerten den für eine Lösung entwickelten Code und suchen nach allgemeinen Sicherheitsproblemen oder bekannten Schwachstellen. Manchmal können Sie diese automatisierten Tests durch manuelle Tests ergänzen, zum Beispiel durch Sicherheits-Code-Reviews oder Penetrationstests. Diese Ansätze sind komplementär, da sie sich auf unterschiedliche Kategorien von Problemen konzentrieren. Es ist nicht ungewöhnlich, dass einer dieser Ansätze Probleme aufdeckt, die die anderen übersehen haben.

**Diese Werkzeuge decken im Allgemeinen keine Sicherheitsprobleme ab, die mit dem Entwurf der Lösung zusammenhängen. Penetrationstests können manche davon aufdecken, aber sie werden nur gelegentlich durchgeführt, oft erst später im Projekt, und die Abdeckung ist nie vollständig. Daher müssen Sie sich um die Sicherheit des Designs kümmern. Dieses Kapitel trägt einige der wichtigsten Konzepte zu diesem Thema zusammen.**

Bei all diesen Werkzeugen zur Gewährleistung der Sicherheit Ihrer Lösung ist es nur normal, sich zu fragen, ob man sie alle braucht. Wie Sie sich vielleicht denken können, gilt die übliche Antwort: Es kommt darauf an. Die allgemeine Empfehlung lautet, sie alle einzubeziehen. Wenn Ihre Lösung nur eingeschränkt datenschutzrelevant und nicht angreifbar ist, können Sie sie als geringes Risiko für die Organisation und ihre Benutzer einstufen. Dies könnte Ihnen erlauben, auf einige dieser Praktiken zu verzichten. Unserer Erfahrung nach ist sicheres Design eine der wichtigsten und effektivsten Praktiken, und Sie sollten sie daher nicht außer Acht lassen.

## IaaS vs. PaaS vs. SaaS und die gemeinsame Verantwortung

Einer der ersten Schritte, die Sie bei der Entwicklung einer Lösung unternehmen müssen, ist die Entscheidung für ein Cloud-Sicherheitsmodell. Diese Entscheidung ist von entscheidender Bedeutung, denn sie bestimmt, was Sie tun können, wofür Sie verantwortlich sind und welche Aufgaben der Cloudanbieter hat.

Das Modell der gemeinsamen Verantwortung gilt für alle Anbieter von Clouddiensten (Cloud Service Providers, CSPs), einschließlich Microsoft. Es lassen sich zwar einige gemeinsame Merkmale feststellen, die von der Cloud Security Alliance (CSA) unter *https://azsec.tech/cu0* zusammengefasst wurden, aber es gibt auch einige Unterschiede, die zu beachten sind. In diesem Buch konzentrieren wir uns auf Microsoft Azure.

Anhand des in Abbildung 2-1 dargestellten Modells der gemeinsamen Zuständigkeit bzw. Verantwortung in Azure lässt sich sagen, dass die Verantwortung je nach Art der verwendeten Dienste variiert. Microsoft kategorisiert die Azure-Dienste in drei Hauptkategorien:

- Infrastructure as a Service (IaaS) umfasst alle Dienste, die grundlegende Ressourcen bereitstellen. Zu dieser Kategorie gehören zum Beispiel virtuelle Computer (virtuelle Maschinen, VMs), Netzwerke und einfacher Speicher wie virtuelle Festplatten.
- Platform as a Service (PaaS) umfasst alle Dienste, die auf den grundlegenden Ressourcen aufbauen und Funktionen bieten, die Sie zur Entwicklung Ihrer Lösungen nutzen können. Zum Beispiel gehören App Services (die die Entwicklung von Webanwendungen und APIs ermöglichen), Azure SQL und Azure Data Lake Storage zu dieser Kategorie.
- Software as a Service (SaaS) umfasst Komplettlösungen, die Sie unverändert nutzen oder um Ihre spezifischen Funktionen erweitern können. Microsoft 365 und Power BI fallen unter diese Kategorie.

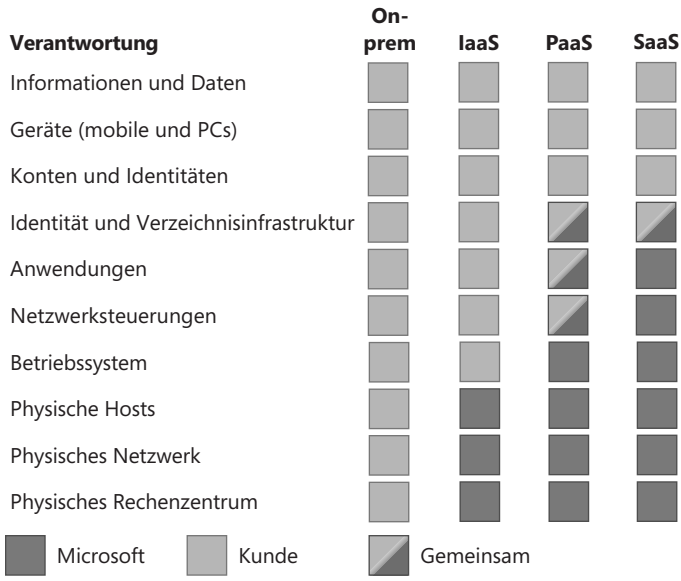


**Abbildung 2-1** Das Modell der gemeinsamen Verantwortung. Quelle: *https://learn.microsoft.com/azure/security/fundamentals/shared-responsibility*.

Nichts hindert Sie daran, Lösungen zu erstellen, die mehrere Arten von Ressourcen umfassen. Es ist durchaus üblich, IaaS- und PaaS-Dienste gleichzeitig einzusetzen. Es ist auch üblich, SaaS-Lösungen mit IaaS oder PaaS zu erweitern.

Sie fragen sich vielleicht, ob es nicht besser wäre, eine reine IaaS-Lösung anstelle einer PaaS- oder SaaS-Lösung zu wählen. Dies ist ein Dilemma, das Sie nur lösen können, wenn Sie verstehen, wie sehr das Unternehmen die Kontrolle über die Qualität schätzt und wie sehr es die Bindung an einen Anbieter fürchtet. Der Grund dafür ist, dass etwas, das Sie für sich selbst entwickeln, ausgehend von Ihren spezifischen Anforderungen Ihnen die größte Kontrolle über das Endergebnis gibt, aber nicht so getestet und optimiert werden kann wie etwas, das von Hunderten oder Tausenden anderer Benutzer auf der ganzen Welt genutzt wird. Außerdem ist eine über IaaS erstellte Lösung leicht auf andere Systeme übertragbar, was bei Lösungen, die auf PaaS oder SaaS basieren, nicht unbedingt der Fall ist. Aber betrachten wir nur die Sicherheit

als entscheidenden Faktor. In der Regel ist es besser, SaaS anstelle von PaaS und PaaS anstelle von IaaS zu verwenden, einfach weil es einfacher ist, ein System zu sichern, bei dem unsere Verantwortung begrenzt ist.

An diesem Punkt ist die Frage berechtigt, ob es einen Unterschied macht, eine Lösung auf IaaS zu betreiben, anstatt sie on-prem, also lokal, zu nutzen. Es gibt verschiedene Vor- und Nachteile für jede Option, und zwar selbst dann, wenn wir uns nur auf den Sicherheitsaspekt konzentrieren. Sie wissen vielleicht, dass Microsoft umfassende Sicherheitsrichtlinien für seine Rechenzentren implementiert. Diese Richtlinien umfassen die physische Sicherheit und die Art und Weise, wie die Microsoft-Administratoren auf Ihre Systeme und Daten zugreifen. Das allgemeine Prinzip ist, dass kein Microsoft-Administrator Zugang zu Ihren Daten hat, abgesehen von einigen wenigen Ausnahmen, da alles normalerweise automatisch gehandhabt wird. Microsoft hat mehrere Sicherheitskontrollen eingerichtet, um die Notwendigkeit des Zugriffs eines Microsoft-Mitarbeiters auf Ihre Daten zu minimieren. In Fällen, in denen Sie den Zugriff zulassen möchten, können Sie bei Microsoft angeben, ob Sie dies tun möchten. Wenn Sie beispielsweise einen Supportfall eröffnen, können Sie den Zugriff auf erweiterte Diagnoseinformationen, die vertrauliche Daten enthalten, zulassen oder verweigern.

Wenn Sie Kunden-Lockbox für Azure einsetzen, definieren Sie einen Eskalationspfad, der es den Microsoft-Supporttechnikern ermöglicht, Sie bei Bedarf um zusätzlichen Zugriff zu bitten. Wenn Sie sich entscheiden, Kunden-Lockbox für Azure nicht zu übernehmen, können die Microsoft-Supporttechniker immer noch den Zugriff erhalten, den sie benötigen, um die von Ihnen initiierten Supporttickets zu bearbeiten, aber der Eskalationspfad, um den Zugriff zu erhalten, ist vollständig intern bei Microsoft. Dies kann den Support beeinträchtigen, da die benötigten Informationen nicht verfügbar sind und es keinen Weg gibt, auf sie zuzugreifen. Dieser Ansatz schützt Ihre Daten, weil er deren Schutz als primäre Voraussetzung betrachtet.

**Weitere Informationen zu Kunden-Lockbox für Microsoft Azure finden Sie hier: *http://azsec.tech/o4z*.**

Ein Vorteil von Azure IaaS gegenüber On-Premise-Lösungen ist, dass es die Einhaltung von Vorschriften erleichtert. Dieser Vorteil ist darauf zurückzuführen, dass Microsoft sich darauf konzentriert, für alle geltenden Gesetze und Standards zertifiziert zu werden, einschließlich DSGVO (Datenschutz-Grundverordnung der EU) und ISO/IEC 27018 für den Datenschutz und ISO/IEC 27001 und CIS Benchmark für die Sicherheit. Die wichtigste Quelle für Informationen über die Erfüllung von Compliancestandards in Azure ist das Azure Trust Center (*https://azsec.tech/hen*). Microsoft veröffentlicht die vollständige Liste der unterstützten Vorschriften und Standards unter *https://learn.microsoft.com/compliance/regulatory/offering-home*. Dies ist eine hilfreiche Ressource, wenn Sie Azure vor der Einführung auf Compliance prüfen oder die Compliance Ihrer über Azure erstellten Lösungen nachweisen müssen. Das Microsoft Azure Trust Center ermöglicht es Ihnen auch, Complianceberichte herunterzuladen und einzusehen, die von externen Zertifizierungsstellen für Microsoft erstellt wurden.

**Kapitel 8, »Compliance und Risikomanagement«, erläutert die Bedeutung der Compliance und wie Azure Ihnen dabei helfen kann, die Erfüllung von Compliancestandards zu erreichen.**

In Anbetracht der zahlreichen Microsoft-Maßnahmen sind Sie vielleicht schon davon überzeugt, dass die von den Azure-Rechenzentren gebotene Sicherheit mindestens gleichwertig, wenn nicht sogar besser ist als die vor Ort. Es gibt aber auch noch andere Gründe für einen Wechsel in die Cloud. Wie Sie in Abbildung 2-1 gesehen haben, liegt beispielsweise bei IaaS die Verantwortung für die Wartung der Hosts für virtuelle Computer bei Microsoft. Dies ist ein wesentlicher Vorteil gegenüber dem On-Prem-Szenario. Microsoft wendet auch Sicherheitsfixes für die Hosting-Umgebung an. Schwachstellen, wie die berüchtigte Log4Shell-Lücke, können für Unternehmen massive Probleme verursachen, von denen die meisten nicht darauf vorbereitet sind, Fixes schnell anzuwenden. Mehr über diese Schwachstelle können Sie hier lesen: *https://azsec.tech/9tz*. All dies bedeutet, dass es sinnvoll sein kann, wichtige Komponenten Ihrer Infrastruktur, wie die Hosting-Umgebung, von Microsoft verwalten und patchen zu lassen.

**In der Anfangsphase von Azure stellte Microsoft fest, dass Angreifer noch Zeit hatten, die entsprechenden Schwachstellen auszunutzen, wenn sie die Sicherheitskorrekturen nach der Veröffentlichung der Bulletins anwendeten. Aus diesem Grund hat das Unternehmen damit begonnen, die Bulletins erst dann zu veröffentlichen, nachdem die Infrastruktur gepatcht wurde.**

Natürlich haben Sie vielleicht immer noch Bedenken oder sogar Angst davor, Ihre Umgebung mit vielen anderen zu teilen, von denen einige möglicherweise bösartige Absichten haben. Um diese Bedenken zu zerstreuen, können Sie Features wie Azure Dedicated Hosts und Azure Confidential Computing nutzen.

**Weitere Informationen zu Azure Dedicated Hosts finden Sie unter *https://azure.microsoft.com/products/virtual-machines/dedicated-host*. Informationen über Azure Confidential Computing finden Sie unter *https://azure.microsoft.com/solutions/confidential-compute*. Dieses Thema wird auch in Kapitel 11, »Confidential Computing«, behandelt.**

Gemeinsame Verantwortung bedeutet genau das: Sie sind für die Auswahl der richtigen Dienste und deren korrekte Nutzung verantwortlich, während Microsoft dafür sorgt, dass sie sicher sind, indem sie nach den besten verfügbaren Verfahren entwickelt und betrieben werden, einschließlich des Microsoft SDL.

Die Frage ist also: Wie können Sie sichere Lösungen auf Azure entwerfen und entwickeln? Um die Antwort auf diese Frage zu erhalten, müssen Sie den Rest des Buches lesen!

KAPITEL 3

# Sicherheitsmuster

**Am Ende dieses Kapitels**

- können Sie die vorgeschlagenen Muster übernehmen, um die sichere Gestaltung Ihrer Lösungen zu verbessern.
- erkennen Sie zahlreiche weitere Azure-Sicherheitsmuster, um Ihr Verständnis von Azure weiter zu verbessern.

## Was ist ein Muster?

Entwurfsmuster sind in der Informationstechnologie nicht neu, aber sie spielen nach wie vor eine grundlegende Rolle. Entwurfsmuster wurden von einem britisch-amerikanischen Architekten österreichischer Herkunft namens Christopher Alexander entwickelt. Im Jahr 1977 schrieb Alexander ein Buch über wiederkehrende Lösungen für häufige Probleme im Zusammenhang mit dem Bau physischer Strukturen. Dieses Buch wurde jedoch über seinen ursprünglichen Bereich hinaus einflussreich. Alexanders Arbeit inspirierte nämlich vier Informatiker und Forscher – Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides – dazu, die gleichen Konzepte auf die Softwareentwicklung anzuwenden. Das Ergebnis war ein Buch mit dem Titel »Design Patterns: Elements of Reusable Object-Oriented Software«, das auch heute noch weit verbreitet ist. (Die deutsche Ausgabe trägt den Titel »Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software«.)

In seinem Buch definiert Christopher Alexander Muster wie folgt:

> *»Jedes Muster beschreibt ein Problem, das in unserem Umfeld immer wieder auftritt, und beschreibt dann den Kern der Lösung für dieses Problem, und zwar so, dass man diese Lösung millionenfach anwenden kann, ohne es jemals zweimal auf die gleiche Weise zu tun.«*

Es geht hier darum, dass Muster einen strukturierten Ansatz zur Lösung gemeinsamer Probleme darstellen. Sie sind eine Möglichkeit, Know-how zu sammeln und weiterzugeben, das sich in vielen Disziplinen, einschließlich des Softwaredesigns, bewährt hat. Da sich dieses Buch auf die Entwicklung von sicheren Lösungen auf Azure bezieht, konzentrieren wir uns hier auf Entwurfsmuster in diesem Kontext.

## Unsere Meinung zu Azure-Sicherheitsmustern

Azure verwendet in großem Umfang Entwurfsmuster. Sie sind fast überall zu finden, zum Beispiel bei dem Schutz von ruhenden Daten, bei der Implementierung der Benutzerauthentifizierung und bei zu vielen anderen Szenarien, um sie alle hier aufzuführen. Anstatt das Rad neu zu erfinden, verwendet Azure dieselben Muster und manchmal sogar dieselben Dienste, um wichtige Funktionen bereitzustellen und allgemeine Probleme zu lösen.

Es ist wichtig, die Sicherheitsmuster von Azure zu kennen, da sie den besten Weg darstellen, um allgemeine Sicherheitsprobleme zu lösen. Sie können Sie sogar in die Lage versetzen, sichere Lösungen zu entwerfen, ohne anspruchsvollere Ansätze wie die Bedrohungsmodellierung zu verwenden, die in Kapitel 4, »Bedrohungsmodellierung«, eingeführt wird. Dieses Kapitel stellt einige dieser Muster vor und enthält die folgenden Informationen zu jedem Muster:

- Der Name des Musters
- Die Absicht und Motivation hinter dem Muster
- Eine Beschreibung des Musters
- Beispiele für die Implementierung des Musters in Azure
- Verwandte Sicherheitsgrundsätze (erörtert in Kapitel 2, »Sicherer Entwurf«)
- Verwandte Muster

Außerdem sind die Muster in Kategorien unterteilt, um ihre Identifizierung zu erleichtern:

- **Authentifizierung** Diese Muster befassen sich mit der Authentifizierung der Kontrahenten einer Interaktion.
- **Autorisierung** Bei diesen Mustern geht es um die Kontrolle des Zugriffs auf Ressourcen.
- **Verwaltung von Geheimnissen** Bei diesen Mustern geht es darum, wie die Lösung Schlüssel und andere Geheimnisse verwaltet.
- **Verwaltung vertraulicher Informationen** Diese Muster befassen sich mit dem Umgang mit vertraulichen Informationen.
- **Verfügbarkeit** Bei diesen Mustern geht es darum, sicherzustellen, dass die Ressourcen für rechtmäßige Benutzer zugänglich sind.

Auf den folgenden Seiten werden einige Muster beschrieben. Diese Liste ist keineswegs erschöpfend. Sie enthält lediglich einige der gängigsten Muster, die wir in unserer Praxis gesehen haben und die sich ausschließlich auf den Lösungsentwurf konzentrieren. Dieses Kapitel befasst sich nicht mit Implementierungs- und Bereitstellungsmustern, wie zum Beispiel solchen, die sich auf die Lieferkette beziehen. Überlegungen, die sich auf diese Muster beziehen, werden an anderer Stelle im Buch, in Kapitel 9, »Secure Coding«, behandelt. Indem wir hier einige der wichtigsten Muster erörtern und klar darlegen, warum Sie sie übernehmen müssen, möchten wir Ihnen eine einheitliche Sichtweise vermitteln.

**Eine vollständige Liste der Muster finden Sie unter *https://learn.microsoft.com/azure/security/fundamentals/best-practices-and-patterns*.**

Sobald Sie die Muster und ihre Bedeutung in Azure kennen, können Sie vielleicht auch andere Muster in den von Ihnen verwendeten Diensten erkennen. Dieses Verständnis von Entwurfsentscheidungen ermöglicht es Ihnen, bessere Lösungen zu entwerfen, indem Sie dieselben Konzepte übernehmen und Azure-Dienste richtig nutzen.

## Das Azure Well-Architected Framework

Microsoft hat eine Reihe von Richtlinien zur Implementierung solider Architekturen auf Azure veröffentlicht. Diese Richtlinien sind als Teil des Azure Well-Architected Framework (WAF) verfügbar, das hier zu finden ist: *https://learn.microsoft.com/azure/well-architected*. Diese Richtlinien decken mehrere Aspekte der Implementierung von Architekturen ab, darunter Zuverlässigkeit, Sicherheit, Kostenoptimierung, optimaler Betrieb, Leistungseffizienz, Arbeitslasten und Dienste. Für unsere Zwecke ist der Abschnitt über Sicherheit am interessantesten. Er behandelt Themen wie Governance, Zielzonen, Identitäts- und Zugriffsmanagement und vieles mehr. Viele Konzepte innerhalb des Azure Well-Architected Framework lassen sich auf die in diesem Kapitel beschriebenen Entwurfsmuster abbilden, aber das würde den Rahmen unseres Buches sprengen. Wir empfehlen daher, einen Blick darauf zu werfen.

# Authentifizierungsmuster

Authentifizierung ist eine wesentliche Eigenschaft jeder Lösung. Es geht darum, Ihr Gegenüber in einem Gespräch mit einiger Sicherheit zu identifizieren.

Wir bezeichnen den Grad der Gewissheit üblicherweise als *Authentifizierungsstärke*. Nehmen wir zum Beispiel an, jemand erklärt, wer er ist, ohne einen Beweis zu liefern. Dies wäre keine Authentifizierung, sondern eher eine Identifizierung. Wenn sich das Gegenüber etwas mehr Mühe gibt und ein Kennwort angibt, wäre dies eine schwache Authentifizierung. Natürlich können Sie Einschränkungen einführen, damit ein Angreifer die Kennwörter der Benutzer nicht einfach erraten kann, aber das hat nur eine begrenzte Auswirkung auf die Sicherheit.

Bei der Authentifizierung werden in der Regel ein, zwei oder alle der folgenden Parameter verwendet:

- **Etwas, das Sie wissen** Dies könnte ein Kennwort sein.
- **Etwas, das Sie haben** Dies kann Ihr Smartphone oder ein physischer Token sein.
- **Etwas, das Sie sind** Dies können biometrische Informationen sein.

Wenn Sie die Authentifizierung nur auf das Kennwort stützen, das heißt auf etwas, das Sie kennen, haben Sie eine Ein-Faktor-Autorisierung, die leicht missbraucht werden kann. Um die Sicherheit zu erhöhen, sollten Sie sie mit einem zweiten Faktor koppeln, zum Beispiel mit Ihrem Benutzernamen und Ihrem Kennwort (das Sie kennen) und Ihrem Handy (das Sie besitzen). Die

Idee dahinter ist, dass ein Angreifer zwar leicht einen der beiden Faktoren kompromittieren könnte, es aber viel schwieriger wäre, beide gleichzeitig zu kompromittieren. Schließlich können Sie den dritten Faktor – etwas, das Sie sind – hinzufügen, um die Sicherheit noch weiter zu erhöhen. Dieser Ansatz wird üblicherweise als Multi-Faktor-Authentifizierung (MFA) bezeichnet, da er sich auf mehr als einen Faktor stützt.

**Kapitel 5, »Identität, Authentifizierung und Autorisierung«, enthält weitere Informationen zu diesem Thema.**

Die Authentifizierung stellt keinen Wert an sich dar, ist aber für die Sicherheit Ihrer Anwendung und der Daten von entscheidender Bedeutung.

Dieser Abschnitt befasst sich mit einem gängigen Muster für die Sicherung Ihrer Lösung durch Authentifizierung: die Verwendung eines zentralisierten Identitätsanbieters für die Authentifizierung.

## Zentralisierten Identitätsanbieter für die Authentifizierung verwenden

### Intention und Motivation

Die Cloud ist eine bequeme Plattform für das Hosting von Anwendungen – so bequem sogar, dass Sie möglicherweise viele Anwendungen darauf hosten werden, die alle eine Authentifizierung erfordern. Es ist nur natürlich, dass Unternehmen einen zentralen Ansatz für die Identitätsverwaltung suchen, um mit einem einzigen Satz von Anmeldedaten auf alle verschiedenen Dienste zugreifen zu können. Dies ist nicht nur aus Gründen der Einfachheit erforderlich, sondern auch, um die Kontrolle und den Überblick zu behalten.

Die Zentralisierung der Identitätsverwaltung ermöglicht den Einsatz von Tools zur Analyse des Benutzer- und Entitätsverhaltens (User and Entity Behavior Analysis, UEBA). Mit diesen Tools können Sie feststellen, ob ein Konto oder ein System für die Organisation ein potenzielles Risiko darstellt; hierzu analysieren Sie das Verhalten, wobei häufig Algorithmen der künstlichen Intelligenz (KI) eingesetzt werden, die in der Lage sind, Veränderungen in den Nutzungsmustern zu erkennen.

Ein weiterer Vorteil zentraler Identitätssysteme besteht darin, dass sie einen einzigen Ort für die Verwaltung von Identitäten und Berechtigungen bieten. Sie ermöglichen auch die Integration des Identitätsmanagements in die HR-Prozesse, zum Beispiel das Entfernen oder Deaktivieren eines Benutzerkontos, sobald der Benutzer seine Beziehung zum Unternehmen beendet. Schließlich können Sie mit diesen Identitätssystemen zugewiesene Rechte überprüfen und bei Bedarf entfernen.

### Beschreibung

Microsoft Entra ID stellt einen vollständigen und einheitlichen Ansatz für das Identitätsmanagement auf Azure dar. Es bietet grundlegende Funktionen, wie verwaltete Identitäten und

Zugriffsüberprüfungen, und kann mit zusätzlichen Diensten erweitert werden, um die Sicherheit zu erhöhen. Diese Dienste umfassen:

- **Microsoft Entra ID Identity Protection** Dieser Dienst ermittelt, welche Identitäten gefährdet sind, indem er Signale aus vielen Quellen analysiert, zum Beispiel Bedrohungsdaten, durchgesickerte Anmeldeinformationen und Microsoft Defender for Cloud Apps (Microsofts Cloud Access Service Broker, CASB).
- **Microsoft Defender for Cloud Apps** Sie können diesen Dienst nutzen, um die Übernahme von Anwendungen zu kontrollieren und einzuschränken und das Vorhandensein von Schatten-IT in Ihrer Organisation zu erkennen.
- **Microsoft Defender for Endpoints** Dies ist eine Lösung zur Analyse des Benutzer- und Entitätsverhaltens (UEBA), die in Windows 10, Windows 11 und verschiedene Geräte integriert werden kann.
- **Microsoft Entra ID Privileged Identity Management (PIM)** Mit diesem Dienst können Sie bei Bedarf Zugriffsrechte zuweisen, wobei vor der Ausführung der Zuweisung eventuell die Genehmigung einer dritten Partei erforderlich ist.

Microsoft Entra ID bietet außerdem Zero-Trust-Sicherheit für die Implementierung von Identitäten durch die Verwendung von bedingtem Zugriff. Hierbei werden Richtlinien definiert, um den Zugriff auf Dienste durch Benutzer zu verhindern, die nicht vertrauenswürdig genug sind. Nehmen wir zum Beispiel an, dass Microsoft Entra ID Identity Protection ein potenzielles Sicherheitsrisiko in Bezug auf einen bestimmten Benutzer identifiziert hat. Dann kann dieser Benutzer gezwungen werden, sich mit MFA zu authentifizieren, wenn er auf vertrauliche Ressourcen zugreift.

**Microsoft Entra ID ist nicht Ihre einzige Option für die Identitätsverwaltung. Sie können stattdessen auch ein Tool eines Drittanbieters einsetzen. Es gibt in der Tat verschiedene Gründe, einen Identitätsanbieter eines Drittanbieters zu verwenden. Möglicherweise hat Ihr Unternehmen bereits für Ihre lokale Umgebung oder zur Unterstützung verschiedener Clouds einen anderen Anbieter eingesetzt, wie AWS oder GCP.**

## Beispiele

- Führen Sie bedingten Zugriff ein, um MFA für privilegierte Benutzer wie die Lösungsadministratoren zu verlangen.
- Verwenden Sie Microsoft Entra ID, um benutzerdefinierte Anwendungsrollen zu definieren, um zu steuern, wie die Lösung genutzt wird. Mehr darüber erfahren Sie hier: *https://learn.microsoft.com/azure/active-directory/develop/howto-add-app-roles-in-apps*.
- Obwohl einige Dienste wie Azure SQL-Datenbank, SQL Managed Instances, Cosmos DB und Azure Storage verschiedene Möglichkeiten zur Authentifizierung bieten, einschließlich Microsoft Entra ID-Anmeldeinformationen, sollten Sie wann immer möglich Microsoft Entra

KAPITEL 4

# Bedrohungsmodellierung

**Am Ende dieses Kapitels**

- können Sie erläutern, warum die Bedrohungsmodellierung für die Verbesserung der Sicherheit jeder Lösung einen besonderen Beitrag darstellt.
- können Sie den Microsoft STRIDE-Bedrohungsmodellierungsansatz beschreiben.
- sind Sie in der Lage, ein Tool für die Bedrohungsmodellierung auszuwählen, das Ihren spezifischen Anforderungen gerecht wird.
- haben Sie mit Threats Manager Studio eine einfache Übung zur Modellierung von Bedrohungen durchgeführt.

## TL;DR

Die Modellierung von Bedrohungen ist eine wichtige Fähigkeit, die Sie mit der Zeit immer besser beherrschen werden. Aber es ist ein umfangreiches Thema – daher ist dies ein umfangreiches Kapitel. Aber glauben Sie uns: Selbst, wenn wir versuchen würden, in diesem Kapitel lediglich all das zusammenzufassen, was wir im Hinblick auf die Bedrohungsmodellierung für wichtig halten, wäre es immer noch ziemlich lang. Aber lassen Sie sich von dem Umfang nicht abschrecken! In diesem Kapitel finden Sie alle Informationen, die Sie brauchen, egal ob Sie bereits ein Experte für Bedrohungsmodellierung oder ein völliger Neuling sind.

Das Kapitel besteht aus acht Abschnitten:

- Was ist Bedrohungsmodellierung?
- Die vier Hauptphasen der Bedrohungsmodellierung
- Der STRIDE-Ansatz zur Klassifizierung von Bedrohungen
- Das Problem mit der Bedrohungsmodellierung
- Auf der Suche nach einem besseren Prozess zur Modellierung von Bedrohungen
- Ein besserer Weg, Bedrohungsmodelle zu erstellen: Die fünf Faktoren
- Tools zur Bedrohungsmodellierung
- Wie man ein Bedrohungsmodell erstellt: Ein Praxisbeispiel

**Wenn Sie gerade erst in das Thema Bedrohungsmodellierung einsteigen, empfehlen wir Ihnen, dieses Kapitel von vorne nach hinten durchzuarbeiten. Wenn Sie bereits über fundierte Kenntnisse verfügen, können Sie mit dem Abschnitt »Das Problem mit der Bedrohungsmodellierung« beginnen und danach dann direkt beim Abschnitt »Tools zur Bedrohungsmodellierung« weitermachen.**

## Was ist Bedrohungsmodellierung?

Das »Threat Modeling Manifesto« (*https://www.threatmodelingmanifesto.org*) definiert Bedrohungsmodellierung als eine Sicherheitspraxis für die Analyse von »Darstellungen eines Systems, um Bedenken hinsichtlich der Sicherheits- und Datenschutzmerkmale aufzuzeigen«. Mit anderen Worten: Es ist eine Möglichkeit, die Sicherheit einer bestimmten Lösung zu bewerten und ihren sicheren Entwurf zu unterstützen.

Die Bedrohungsmodellierung zielt darauf ab, Sicherheitsprobleme zu ermitteln und zu bestimmen, wie sie zu lösen sind. Als Sicherheitsentwurfsprozess ermöglicht Ihnen die Bedrohungsmodellierung, Lösungen von Grund auf sicher zu gestalten, da Sie alle möglichen Angriffe in Betracht ziehen und die notwendigen Schutzmaßnahmen einbauen müssen, um sie zu verhindern.

Diese beiden Verwendungszwecke für die Bedrohungsmodellierung heben zwei einzigartig kombinierte Merkmale hervor: den Fokus auf den Entwurf der Lösung und auf die Identifizierung der am besten geeigneten Schutzmaßnahmen (was zu tun ist) und deren Verknüpfung mit den Bedrohungen selbst (warum die Schutzmaßnahmen erforderlich sind). Diese Kombination führt zu zwei wünschenswerten Ergebnissen:

- Sie sind bereit, in einem sehr frühen Stadium des Entwicklungsprozesses eine Bedrohungsmodellierung durchzuführen. Wie wichtig dies ist, um die Effektivität zu maximieren und gleichzeitig die Kosten zu senken, werden Sie auf den folgenden Seiten sehen.
- Sie beseitigt die meisten Unklarheiten darüber, warum Sie die vorgeschlagenen Gegenmaßnahmen umsetzen müssen.

Andere Sicherheitsansätze, die auf bewährten Verfahren beruhen, können nicht dieselben Vorteile bieten. Nehmen wir zum Beispiel eine Maßnahmenbibliothek wie Microsoft Cloud Security Benchmark (die bis Oktober 2022 als Azure Security Benchmark vermarktet wurde) . Dabei handelt es sich um einen Satz allgemeiner Maßnahmen für Azure als Ganzes und ein Referenztool für die Erstellung dienstspezifischer Anleitungen, die sogenannten Baselines. Microsoft hat Microsoft Cloud Security Benchmark und Azure Security Baselines entwickelt, um häufige Sicherheitsangriffe auf relevante Systeme abzuwehren. Jede enthaltene Maßnahme dient einem bestimmten Zweck. Das Problem ist nur, dass Sie möglicherweise nicht wissen, was es ist. Sie können nicht zwischen »Must-haves« und »Nice-to-haves« unterscheiden, weil es keine klare Verbindung zwischen den vorgeschlagenen Maßnahmen und ihrem Zweck gibt. Im Zweifelsfall könnten Sie also den Wunsch verspüren, alle Maßnahmen zu implementieren.

**Weitere Informationen über Microsoft Cloud Security Benchmark und den Vorgänger Azure Security Benchmark finden Sie hier: *https://learn.microsoft.com/security/benchmark/azure/introduction*. Sie können ihn zusammen mit den zugehörigen Baselines unter *https://azsec.tech/fyc* herunterladen.**

Andere Bibliotheken wie die Benchmarks des Center of Internet Security (CIS) liefern eine Begründung für ihre Empfehlungen. Bei den CIS-Benchmarks handelt es sich um Sammlungen von Methoden, die zur Absicherung bestimmter Produkte wie Windows, Linux, Oracle oder sogar Azure eingesetzt werden können. Dennoch kann es schwierig sein, die genaue Mischung von Gegenmaßnahmen zu bestimmen, die Sie benötigen, weil Sie über keinen Plan verfügen, der es Ihnen ermöglicht, die Folgen der verschiedenen Gegenmaßnahmen für Ihr spezifisches Szenario zu verstehen, was zu Einschränkungen führen könnte. Sie wollen also das, was unbedingt erforderlich ist, und nicht mehr. In jedem Fall sind diese Bibliotheken von Natur aus unvollständig, denn sie entsprechen den Szenarien anderer, nicht unbedingt Ihren eigenen Szenarien.

**CIS-Benchmarks ist hier verfügbar: *https://azsec.tech/uki*.**

Eine Lösung, die Strichcodes verwendet, ist ein hervorragendes Beispiel. Nehmen wir an, dass Sie bei der Entwicklung der Lösung nur bewährte Verfahren und Richtlinien wie CIS Benchmarks und Microsoft Cloud Security Baselines berücksichtigt haben. Dies stellt ein Problem dar, denn Sie könnten Strichcodes als Angriffsvektor übersehen. Böswillige Akteure könnten Ihre Benutzer dazu zwingen, Barcodes zu scannen, die URLs enthalten, die auf schadhafte Nutzdaten wie Skripte verweisen, die von Ihrer Anwendung ausgeführt werden und alle möglichen unangenehmen Dinge verursachen würden.

**Die Tatsache, dass Sie diese Probleme mit der Bedrohungsmodellierung lösen können, bedeutet nicht, dass Sie Bibliotheken wie CIS Benchmarks und Microsoft Cloud Security Benchmark ignorieren sollten. Im Gegenteil, diese Bibliotheken sind für die schnelle Behebung der häufigsten Probleme unerlässlich. Sie haben sich bei der Verbesserung der Sicherheit einer Lösung bewährt, wenn Sie Ansätze wie die Bedrohungsmodellierung nicht anwenden können oder wollen. Dennoch haben sie ihre Grenzen – vor allem die Tatsache, dass sie nicht speziell auf Ihre Lösung zugeschnitten sind. Daher sind sie möglicherweise nicht ausreichend, wenn das System so kritisch ist, dass mehr erforderlich ist.**

Es gibt verschiedene Methoden zur Bedrohungsmodellierung, die alle so unterschiedlich sind, dass es schwierig sein kann, den Prozess der Bedrohungsmodellierung genau zu definieren. Dieses Buch bezieht sich auf eine Variante des STRIDE-Ansatzes von Microsoft, der speziell zur

Verbesserung der DevOps-Integration erweitert wurde. Dieses Kapitel ist nur als einführende Anleitung gedacht; für weitere Informationen empfehlen wir die folgenden Bücher, die hier in der Reihenfolge ihrer Veröffentlichung aufgeführt sind:

- Adam Shostack. Threat Modeling: Designing for Security, 1. Auflage, Wiley, Februar 2014
- Brook S. E. Schoenfield. Securing Systems, 1. Auflage, CRC Press, Mai 2015
- Izar Tarandach und Matthew J. Coles. Threat Modeling: A Practical Guide for Development Teams, 1. Auflage, O'Reilly, Dezember 2020

## Die vier Hauptphasen der Bedrohungsmodellierung

Die Bedrohungsmodellierung besteht in der Regel aus vier Hauptphasen. Diese Phasen entsprechen den vier Schlüsselfragen im Vier-Fragen-Framework. Dieses Framework wurde von Adam Shostack entwickelt und wird in seinem Buch beschrieben. Das Framework wurde auch in das »Threat Modeling Manifesto« aufgenommen.

**Das »Threat Modeling Manifesto« ist ein Dokument, das von einer Gruppe führender Experten für Bedrohungsmodellierung entwickelt wurde, um zu beschreiben, was Bedrohungsmodellierung ist. Es abstrahiert die Merkmale aktueller Methoden, indem es gemeinsame Grundsätze und Werte aufzeigt. Sie können es hier lesen: *https://www.threatmodelingmanifesto.org*.**

Die Phasen sind wie folgt:

1. **Analyse** Diese Phase entspricht der Frage »Woran arbeiten wir?« im Vier-Fragen-Framework. Sie beinhaltet die Analyse des Systems, was typischerweise (aber nicht unbedingt) zu einem Diagramm führt, das das System selbst darstellt. Der Ansatz von Microsoft bevorzugt die Erstellung von Datenflussdiagrammen (Data Flow Diagrams, DFDs), die sich darauf konzentrieren, wie Daten innerhalb des Systems fließen.
2. **Identifizierung von Bedrohungen** In dieser Phase – die mit der Frage »Was kann schiefgehen?« in Shostacks Framework korrespondiert – werden Bedrohungen identifiziert, die mögliche Angriffe auf das zu untersuchende System darstellen. Jede Bedrohung ist in der Regel ein kontextualisiertes Angriffsszenario.
3. **Identifizierung von Gegenmaßnahmen** In dieser Phase, die mit der Frage »Was werden wir dagegen tun?« korrespondiert, werden Gegenmaßnahmen identifiziert, die sich mit den im vorherigen Schritt aufgedeckten Bedrohungen befassen. Dieser Schritt stellt sicher, dass alle Gegenmaßnahmen eine Daseinsberechtigung haben.
4. **Validierung** In diesem Schritt validieren Sie die in den vorangegangenen Schritten geleistete Arbeit und schließlich ihre Integration oder Korrektur. Er entspricht der Frage: »Haben wir unsere Arbeit gut genug gemacht?«

Wie bereits erwähnt, spielen DFDs (siehe Abbildung 4-1) in vielen Prozessen zur Bedrohungsmodellierung eine wichtige Rolle, so auch bei Microsoft. Daher kann es hilfreich sein, ein paar

grundlegende Dinge über sie zu wissen. DFDs gliedern das System in Teile, die als Prozesse (die Daten verarbeiten) und Datenspeicher (die Daten speichern) bezeichnet werden.

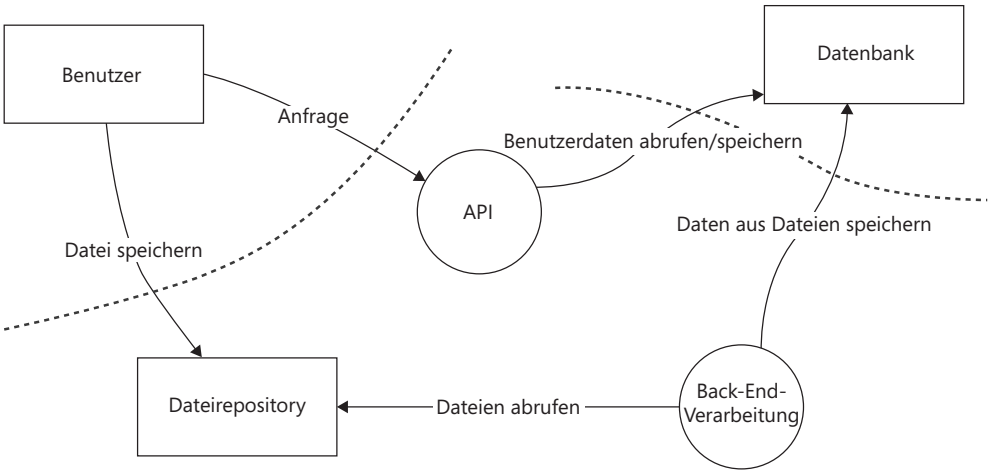


**Abbildung 4-1** Ein Beispiel für ein Datenflussdiagramm

**DFDs sind nicht die einzige Möglichkeit, ein System und seine Angriffspunkte darzustellen. Sie können auch UML, Angriffsbäume, Prozessflussdiagramme oder sogar einfache Tabellen in Excel-Dateien verwenden.**

Der Unterschied zwischen Prozessen und Datenspeichern ist nicht immer klar zu erkennen. Gelegentlich gibt es Prozesse, die einen Teil des Speichers enthalten (wie Azure Databricks, das hauptsächlich Daten verarbeitet, aber über einen dedizierten Azure Storage-Speicher verfügt), oder Datenspeicher, die Geschäftslogik enthalten (wie Azure SQL, dessen Hauptzweck darin besteht, Daten zu speichern und darauf zuzugreifen, der aber auch Logik enthalten kann, beispielsweise in Form von gespeicherten Prozeduren). Ein gängiger Trick, um Unklarheiten zu beseitigen, besteht darin, die Komponente in zwei Teile aufzuteilen. Eine Azure SQL-Datenbank ist beispielsweise ein Datenspeicher und ein Prozess, der gespeicherte Prozeduren und die andere Logik enthält.

DFDs zeigen auch externe Entitäten an, die als Rechtecke dargestellt werden. Diese stellen alles dar, was das System nutzt oder vom System genutzt wird. Externe Entitäten sind zum Beispiel Benutzer und Administratoren, externe Dienste und manchmal auch lokale Systeme. Da sie außerhalb des Systems liegen, werden externe Entitäten als Blackboxes betrachtet. Ihr Innenleben ist daher nicht von Belang. Es wäre zum Beispiel ein Fehler, in einem DFD eine externe Entität darzustellen, die einen Dienst außerhalb der Lösung selbst aufruft.

All diese Elemente interagieren über den Austausch von Daten, der als *Datenfluss* oder, einfacher ausgedrückt, als *Fluss* bezeichnet wird. DFDs stellen Flüsse als gerichtete Pfeile dar, die die verschiedenen Einheiten miteinander verbinden. Es gibt verschiedene Ansätze, um diese Pfeile zu zeichnen. Wenn man die Richtung betrachtet, ist die Darstellung, die den Ursprüngen der DFDs am ehesten gerecht wird, normalerweise die Darstellung der Datenbewegung. Wenn

# Inhaltsverzeichnis

## Kapitel 2



## Kapitel 3

## Kapitel 4

## Kapitel 5

## Kapitel 6



## Kapitel 7

## Kapitel 8

# Teil 2: Sichere Implementierung

## Kapitel 9



## Kapitel 10

## Kapitel 12



## Kapitel 13