Maßnahmen zur Risikominimierung

Die Gefahr klein halten

Eine Reihe von Maßnahmen hilft, das Datenschutzrisiko zu senken. Doch an welchen Punkten müssen wir dabei ansetzen? Welche Hebel können wir in Bewegung setzen, um die Gefahr von Datenverlust maximal einzudämmen? Ein Weg kann beispielsweise die bewusste Entscheidung sein, bestimmte Informationen erst gar nicht zu erheben und zu speichern, doch auch technische Maßnahmen helfen weiter.

von Stefan Kühnlein und Siegfried Höck

Die wohl effektivste Methode, personenbezogene Daten zu schützen, ist, sie nicht zu erheben. Informationen, die ein Unternehmen nicht besitzt, muss es auch nicht absichern. Deshalb sollte am Anfang immer die Frage stehen: Welche Daten brauche ich überhaupt? Neben diesem ganz allgemeinen Ansatz gibt es auch spezifische technische Maßnahmen, die das Datenschutzrisiko senken. Schon beim Design einer Anwendung lassen sich die Aufbewahrungsfristen festlegen. Damit sollte dann auch die Planung bzw. Etablierung eines entsprechenden Prozesses einhergehen, durch den die Daten zuverlässig gelöscht werden.

Eine wesentliches Thema in der DSGVO ist die Weitergabe von personenbezogenen Daten an Dritte, die sogenannten Datenverarbeiter. Dementsprechend ist es wichtig, bei der Auswahl von Datenverarbeitern darauf zu achten, dass diese ein sehr hohes Maß an Sicherheit bieten und kontinuierlich gewährleisten. In diesem Zusammenhang müssen entsprechende Vereinbarungen unterzeichnet werden, die klarstellen, welche Informationen übertragen und geteilt werden. Hierbei sind auch die Übertragungswege sowie deren Absicherung zu dokumentieren.

Technische Maßnahmen: PETS

Wenden wir uns den konkreten technischen Maßnahmen zu, die Entwicklern zur Verfügung stehen, um Daten abzusichern. Sie lassen sich in vier Kategorien aufteilen: Authentifizierung, sichere Kommunikation bzw. Verschlüsselung, Anonymisierung und Pseudonymisierung sowie der Schutz von personenbezogenen Daten in der Datenbank [1], [2].

Authentifizierung und Autorisierung

Um personenbezogene Daten entsprechend zu schützen, ist die Authentifizierung der Benutzer ein zentrales Prinzip. Authentifizierung und Autorisierung sind die Schlüssel zur Sicherung von IT-Systemen und Anwendungen. Eine starke Authentifizierung stellt einen wichtigen Datenschutzmechanismus dar, wenn sichergestellt wird, dass nur bestimmte berechtigte Personen auf private Informationen von Betroffenen zugreifen können. In heterogenen Enterprise-Architekturen ist eine starke Authentifizierung jedoch nicht so einfach umzusetzen, da eine Vielzahl von Anforderungen gestellt werden:

- Ermittlung der Berechtigungen, abhängig von den Attributen einer Person oder einer Ressource
- Ermittlung der Berechtigungen auf Basis von logischen oder mathematischen Funktionen, die wiederum vor dem Hintergrund einzelner Attribute einer Person zu definieren sind
- Zusammenführung von unterschiedlichen Regeln und Richtlinien (Policies) aus verschiedenen Systemen zu einem einheitlichen Regelwerk
- Ausführung von zusätzlichen Aktionen bei der Ermittlung der Berechtigung
- Flexible und parametrisierbare Regeln

Damit die Richtlinien nicht in jedem relevanten System zu implementieren sind, und um dies zu vereinfachen, entstand die Idee einer einheitlichen Sprache. Mit der eXtensible Access Control Markup Language (XACML) [3] wurde ein XML-Dialekt definiert, der genau diese Richtlinien erfüllt. XACML wurde von OASIS 2003 standardisiert und liegt inzwischen in der Version 3.0 vor. Im Oktober 2017 wurde das JSON-Profil für XACML ausgerollt.

Von PAP bis PIP

Die Architektur eines XACML-Systems besteht aus den folgenden fünf zentralen Komponenten: Im PAP (Policy Administration Point) werden die Regeln hinterlegt, ob

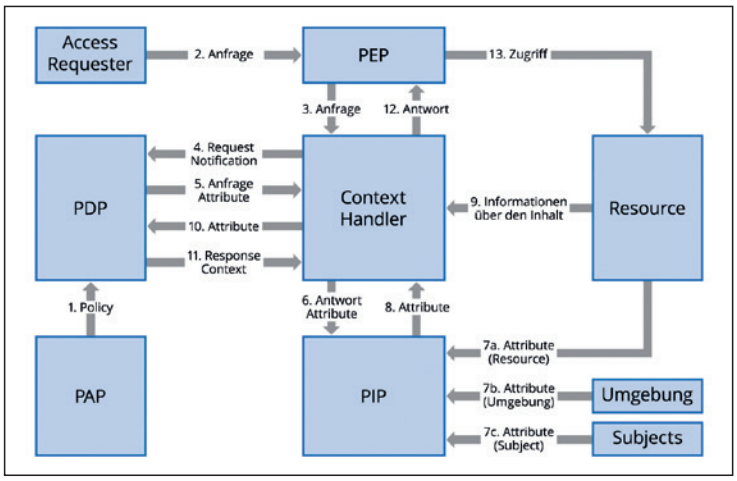

Abb. 1: Rollen und Datenfluss nach dem XACML-Standard

dem Anfragenden die Berechtigungen für den Zugriff erteilt werden oder nicht. Der PDP (Policy Decision Point) trifft alleinig die Entscheidung, ob ein bestimmter Zugriff erlaubt werden soll oder nicht. Geht ein Zugriffswunsch ein, so fragt der PEP (Policy Enforcement Point) den PDP, ob der Zugriff gewährt werden soll oder nicht. Fehlen dem PEP Informationen, sodass nicht eindeutig über den Zugriff entschieden werden kann, wird der PIP (Policy Information Point) in den Entscheidungsprozess einbezogen.

Listing 1: XACLM Request im JSON-Format

```
{
  "Request": {
    "AccessSubject": {
      "Attribute": [      {
        "AttributeId": "de.oc.user.employeeId",
        "Value": "SKU_1871"
      }]
    },
    "Resource": {
      "Attribute": [{
        "AttributeId": "de.oc.record.recordId",
        "Value": "9257" },
      {
        "AttributeId": "de.oc.object.objectType",
        "Value": "record"}]
    },
    "Action": {
      "Attribute": [      {
        "AttributeId": "de.oc.action.actionId",
        "Value": "view"}]
    },
    "Environment": {
      "Attribute": []
    }
  }
}
```

Der Context Handler arbeitet sehr eng mit dem PDP zusammen. Die Aufgabe des Context Handlers besteht darin, die Anfragen mit Kontextinformationen aufzubereiten, sodass diese durch den PDP beantwortet werden können. **Abbildung 1** zeigt sowohl die Rollen als auch den Datenfluss nach dem XACML-Standard [4].

Listing 1 zeigt einen XACML-Request im JSON-Format. Hierbei fordert der Mitarbeiter (Subject) mit der ID SKU_1871 einen lesenden Zugriff auf den Typ *Record* mit der ID 9257 an.

Eine starke Authentifizierung reicht jedoch nicht aus, um personenbezogene Daten adäquat vor unbefugtem Zugriff zu schützen. Ein Risiko besteht beispielsweise darin, die Sitzung eines authentifizierten Benutzers durch die Analyse von Datenströmen in den Netzwerken zu identifizieren. Aus diesem Grund sind zum Schutz von personenbezogenen Daten weitere Sicherheitsmaßnahmen nötig, wie die Verwendung einer sicheren Kommunikation sowie die Verschlüsselung.

Sichere Kommunikation und Verschlüsselung

Eine wichtige Technologie, um Daten zu schützen, ist die Verschlüsselung. Verschlüsselung wird für gespeicherte Daten (Encryption at Rest) und für Bewegungsdaten (Encryption in Transit) gefordert. Hier gibt es mehrere anerkannte Technologien, symmetrische, asymmetrische oder ID-basierte Verfahren, die je nach Verwendungszweck eingesetzt werden. Bei gespeicherten Daten kommt häufig AES 256 zum Einsatz, bei Bewegungsdaten nutzen viele TLS oder SSL. Wichtig ist, dass für den jeweiligen Zweck die richtige Verschlüsselungsmethode verwendet wird, und dass beim Systementwurf von Anfang an identifiziert wird, wo welche Methoden eingesetzt werden sollen.

Verschlüsselte Kommunikation mit TLS/SSL

Um die Übertragung von personenbezogenen Daten vor unbefugtem Zugriff zu schützen, müssen diese verschlüsselt übertragen werden. TLS/SSL erfüllt sowohl die Anforderungen an den Datenschutz als auch an die Datenintegrität beim Austausch zwischen verschiedenen Anwendungen bzw. Microservices über das Netzwerk. Für den Aufbau einer sicheren Kommunikation zwischen zwei Endpunkten werden ein privater und ein öffentlicher Schlüssel benötigt. Zusätzlich kann TLS/SSL noch Zertifikate verwenden, um die Vertrauenswürdigkeit der Schlüssel zu überprüfen. Auf diesem Weg können Spoofing und andere potenzielle Sicherheitsprobleme verhindert werden.

Schlüsselrotation

Moderne Verschlüsselungstechnologien bieten eine sehr starke Vertraulichkeitsgarantie. Diese Garantie kann jedoch nur gewährleistet werden, wenn der Zugang

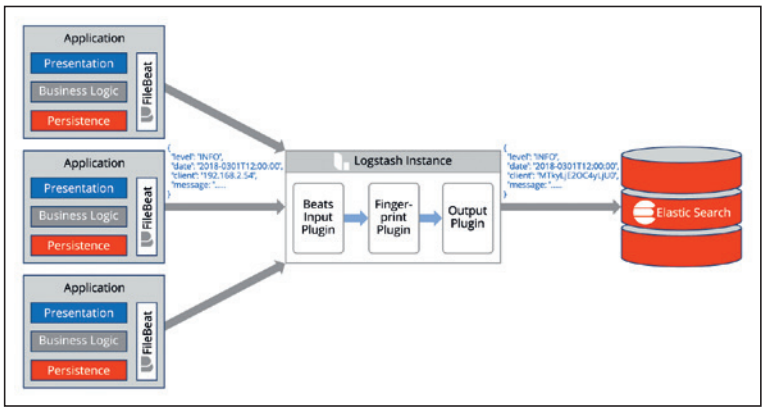

Abb. 2: Via Filter-Plug-in Fingerprint die IP-Adresse pseudonymisieren

zu den Schlüsseln entsprechend geschützt ist und diese nicht kompromittiert werden können. Um das Risiko der Kompromittierung zu minimieren, ist es zwingend notwendig, die Schlüssel in regelmäßigen Abständen zu rotieren. Moderne kryptografische Systeme, einschließlich einiger Konfigurationen von TLS, verwenden Forward Secrecy. Mit Forward Secrecy wird für jede Session ein neuer geheimer Sitzungsschlüssel generiert und nach deren Beendigung wieder verworfen. Mithilfe dieses Verfahrens wird sichergestellt, dass im Fall des Bekanntwerdens des geheimen Sitzungsschlüssels die ausgetauschten Nachrichten von anderen Sessions nicht entschlüsselt werden können.

Listing 2: IP-Adresse pseudonymisieren

```
filter {
  ruby {
    code => "event.set('identities',[])"
  }
  # pseudonymize client field
    #fingerprint client
    fingerprint {
      method => "SHA256"
      source => ["client"]
      key => "${FINGERPRINT_KEY}"
    }
    #create sub document under identities field
    mutate { add_field => { '[identities][0][key]' => "%{fingerprint}"
                            '[identities][0][value]' => "%{client}" } }
    #overwrite client field with fingerprint
    mutate { replace => { "client" => "%{fingerprint}" } }
  #extract sub documents and yield a new document for each one into the
                                                          LS pipeline.
  ruby {
    code => "event.get(identities).each { |p| e=LogStash::Event.new(p);
e.tag(identities); new_event_block.call(e); } "
  }
  #remove fields on original doc
  mutate { remove_field => ["fingerprint","identities"] }
}
```

Verschlüsselung in der Cloud

Verschlüsselung in der Cloud ist in diesem Zusammenhang sehr wichtig. Genau deswegen stellen Cloud-Anbieter Verschlüsselungsverfahren mit ihren Services bereit. Aktuell ist es so, dass Anwender noch entscheiden können, ob sie die Daten verschlüsselt speichern wollen oder nicht. Die Umsetzung ist aber bei vielen Anbietern denkbar einfach: Oft genügt es, einen Haken zu setzen und die Daten werden verschlüsselt – sei es bei Objektspeichern oder Datenbanken. Ebenso sind Verschlüsselungsmethoden bei der Kommunikation integriert. Die Zugriffe auf die Cloud, die typischerweise über APIs erfolgen, sind per se SSL- oder TLS-verschlüsselt.

Anonymisierung und Pseudonymisierung

Anonymisierung und Pseudonymisierung werden gerne verwechselt, doch sind das zwei verschiedene Vorgehen mit einem alles entscheidenden Unterschied: Bei anonymisierten Daten müssen die Datenschutzgrundsätze nicht angewendet werden, bei pseudonymisierten Daten allerdings schon. Wenn die Identifizierung einer Person ausgeschlossen werden kann, dann sind die Daten anonymisiert, wenn sie nicht ausgeschlossen werden kann, dann sind sie pseudonymisiert.

Eine Pseudonymisierung ist einfach zu erreichen. Daten sind pseudonymisiert, wenn der Personenbezug nicht auf den ersten Blick besteht, aber über Identifier oder Datenbestände aus dritter Hand wiederhergestellt werden kann. Werden beispielsweise die Daten in unterschiedliche Container aufgeteilt und mit Referenzen versehen, dann sind sie pseudonymisiert. Um eine Anonymisierung zu erreichen, muss mehr getan werden. Ein Kriterium ist zum Beispiel der Verlust der Eindeutigkeit. Wenn von mehreren Datensätzen der Personenbezug entfernt wird, sodass redundante Daten übrig bleiben, es danach also mehrere identische Datensätze gibt, ist es nicht mehr möglich, Rückschlüsse auf den Ursprungsdatensatz zu ziehen. Das kann auch über eine Aggregation (etwa Summenbildung) oder eine Einteilung in Cluster (etwa Altersklassen) erfolgen.

In vielen IT-Systemen und Anwendungen wird zur Fehleranalyse in den Logdateien eine Reihe von Daten gesammelt und in einer zentralen Datenbank gespeichert. Insbesondere für eine Microservices-Architektur ist dies von enormer Bedeutung, denn nur so ist eine erfolgreiche Analyse im Fehlerfall möglich. Dabei stellt sich jedoch die Frage, wie personenbezogene Daten – hierzu zählt auch die IP-Adresse des Benutzers – sicher pseudonymisiert bzw. anonymisiert werden können. Inzwischen bieten die gängigsten Logging-Plattformen die unterschiedlichsten Plug-ins an, um die Anforderungen der DSGVO bei der Speicherung von personenbezogenen Daten zu erfüllen. Schauen wir uns exemplarisch an, wie die Umsetzung von Pseudonymi-

sierung unter Verwendung der entsprechenden Plug-ins erfolgt.

Pseudonymisieren mit Logstash

Logstash ist ein Produkt von Elastic und eine serverseitige Open-Source-Software zur simultanen Verarbeitung von Datenströmen aus den unterschiedlichsten Quellen [5]. Logstash kann sowohl einzelne Teile aus den Datenströmen extrahieren als auch einzelne Teile umwandeln. Hierzu sind eine Reihe von Filtern in Logstash vorhanden, um Daten dynamisch und unabhängig von Format oder Komplexität zu analysieren und zu transformieren. Für die Umsetzung der Anforderungen bezüglich Pseudonymisierung personenbezogener Daten eignet sich vor allem das Filter-Plug-in Fingerprint [6]. Damit können personenbezogene Daten mit dem Hash-Wert der entsprechenden Felder ersetzt werden. Zusätzlich wird für den Hash-Wert und die ursprünglichen Daten ein neuer Datensatz erzeugt, der zu Nachschlagezwecken zusätzlich in der Datenbank gespeichert werden kann (**Abb. 2,** https://www.elastic.co/brand). Listing 2 beschreibt, wie mit Fingerprint die IP-Adresse pseudonymisiert wird.

Das vollständige Beispiel, wie diese Konfiguration unter Verwendung von Rubby noch vereinfacht werden kann, ist unter [7] zu finden.

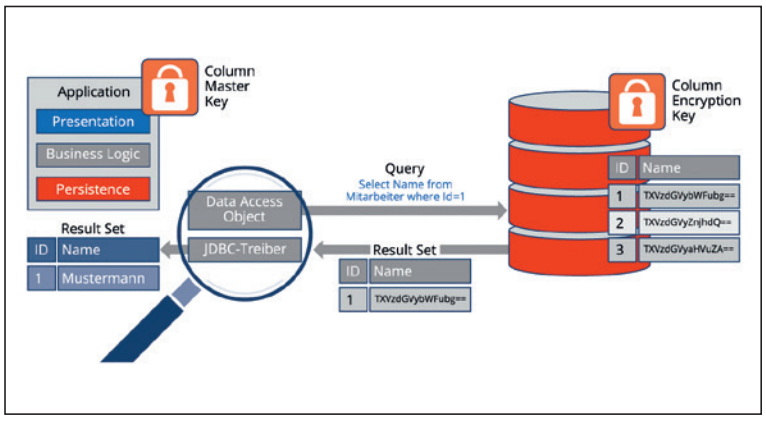

Abb. 3: Zu jedem Column Encryption Key existiert ein zugehöriger Column Master Key

Privacy in Database

Der Schutz von personenbezogenen Daten, die in einer Datenbank gespeichert werden, hängt weitestgehend vom Kontext ab, in dem diese verwendet werden. Auch wenn Datenbanken in separaten Netzwerksegmenten betrieben werden und somit physikalisch abgesichert sind, gibt es doch eine Reihe von Sicherheitsrisiken, die den Schutz von personenbezogenen Daten gefährden bzw. verletzen. Hierzu zählen beispielsweise nicht autorisierte oder unbeabsichtigte Aktivitäten durch autorisierte Datenbankbenutzer und -administratoren sowie nicht autorisierte Benutzer oder Hacker, die so unangemessene Änderungen an Daten und Strukturen vornehmen können.

Aber auch Fehler, die bei der Erstellung von Anwendungen entstehen, können ein Sicherheitsrisiko für personenbezogene Daten darstellen. So ist beispielsweise bei der Erstellung von Webanwendungen darauf zu achten, dass Angreifer mittels SQL Injection keinen uneingeschränkten Zugriff auf die gesamte Datenbank erhalten. Um die Daten dennoch entsprechend zu schützen, sollten die Daten in der Datenbank verschlüsselt abgelegt werden. Um Daten zum Beispiel in einer Azure-SQL-Datenbank oder SQL-Server-Datenbank in-transit als auch at-rest zu verschlüsseln, bietet Microsoft als eine Möglichkeit Always Encrypted [7]. Damit wird sichergestellt, dass vertrauliche Daten niemals im Klartext übertragen werden. Die mit Always Encrypted verschlüsselten Daten können nur Clientanwendungen oder Applikationsserver verarbeiten, die berechtigt sind, den notwendigen Schlüssel zu beziehen.

Für die Verschlüsselung bzw. Entschlüsselung der jeweiligen Spalten verwaltet der SQL Server sogenannte Column Encryption Keys. Diese sind als verschlüsselte Metadaten in der Datenbank gespeichert. Zu jedem Column Encryption Key existiert ein zugehöriger Column Master Key. Dieser wird nicht in der Datenbank gespeichert, sondern ausschließlich vom Client bzw. Applikationsserver verwaltet (**Abb. 3**).

Zur Verwaltung des Master Keys beinhaltet der Microsoft JDBC-Treiber des SQL Servers einen Key Store. Für den Zugriff auf die unterschiedlichsten Provider beinhaltet der Microsoft-JDBC-Treiber die in Tabelle 1 aufgeführten Klassen.

Zur Verwendung der bereits registrierten Key-Store-Provider müssen keine Änderungen am Anwendungscode vorgenommen werden. Um Always Encrypted in einer Java-Anwendung verwenden zu können, müssen drei Voraussetzungen erfüllt sein: die Konfiguration von Always Encryted in der Datenbank, die Installation des Microsoft JDBC Driver 6.0 für SQL Server und die Installation der Java Cryptography Extension (JCE)

Class	Beschreibung	Provider	Bereits registriert
SQLServerColumnEncryptionAzureKeyVaultProvider	Provider für den Key Store von Azure	AZURE_KEY_VAULT	nein
SQLServerColumnEncryptionCertificateStoreProvider	Provider für den Windows-Zertifikatsspeicher	MSSQL_CERTIFICATE_STORE	ja
SQLServerColumnEncryptionJavaKeyStoreProvider	Provider für den Java-Key-Store	MSSQL_JAVA_KEYSTORE	ja

Tabelle 1: Klassen für den Zugriff auf die unterschiedlichen Provider

Unlimited Strength Jurisdiction Policy Files. Um aus einer Java-Anwendung auf die verschlüsselten Werte einer mit Always Encrypted verschlüsselten Tabelle zuzugreifen, ist es ausreichend, das Attribut *cumnEncryptionSetting* beim Aufbau einer Verbindung zur Datenbank auf *true* zu setzen. Das folgende Listing zeigt den Aufbau zu einer SQL-Server-Datenbank, in der Always Encrypted aktiviert ist.

```
String connectionString = "jdbc:sqlserver://localhost;user=<user>;password=<
    password>;databaseName=<database>;columnEncryptionSetting=Enabled;";
                SQLServerConnection connection = (SQLServerConnection)
                    DriverManager.getConnection(connectionString);
```

Natürlich bieten auch alle anderen Datenbankprovider entsprechende Produkte an, mit denen Daten verschlüsselt abgespeichert und vor unbefugtem Zugriff geschützt werden können. Oracle bietet beispielsweise mit dem Oracle Database Vault ein entsprechendes Produkt an, mit dem sich sicherstellen lässt, dass die behördlichen Auflagen bezüglich des Datenschutzes und der Integrität proaktiv erfüllt werden.

Cloud-Security

Die Umsetzung der DSGVO ist an den Cloud-Anbietern nicht spurlos vorbeigezogen. Dies zeigt sich vor allem darin, dass die Cloud-Anbieter den Nutzern die Möglichkeit geben, wichtige Sicherheitseinstellung selbst zu konfigurieren. In vielen Fällen ist es jedoch sehr schwierig, die notwendigen Daten zu ermitteln, um den Schutz von personenbezogenen Daten zu gewährleisten. So ist es etwa unmöglich, Informationen über diverse Useraktivitäten, den Traffic auf dem Netzwerk und das Verhalten der verwendeten Systeme so zu sammeln, dass diese Daten nicht ausgewertet werden können. Sowohl Amazon als auch Microsoft bieten ihren Nutzern Dienste an, mit denen weitere Daten über die Cloud-Umgebung gesammelt und ausgewertet werden können. Dadurch können Nutzer Bedrohungen in Zukunft schneller identifizieren.

Amazon GuardDuty

Mit GuardDuty [8] stellt Amazon einen Managed Service zur Identifizierung von Bedrohungen auf den Konten und Instanzen der Nutzer bereit. Amazon GuardDuty überwacht fortlaufend auf böswillige und unbefugte Verhaltensweisen. GuardDuty konzertiert sich hierbei auf die wesentlichen API-Aufrufe innerhalb von AWS, insbesondere solche, die Systemänderungen vornehmen, das Anlegen von neuen Assets sowie den Zugriff auf Anmeldedaten.

Azure Advanced Threat Protection

Azure Advanced Thread Protection (ATP) [9] ist dem Service GuardDuty von Amazon sehr ähnlich. Im Prinzip handelt es sich bei ATP um eine Umsetzung des Diensts Advanced Threat Analytics (ATA) in der Azure-Cloud. Bei ATA handelt es sich um einen On-Premise-Service, der Daten aus den Logs und Events, die innerhalb des Windows Active Directory auftreten, sammelt und analysiert.

Fazit

Um das Datenschutzrisiko zu minimieren, sind swohl organisatorische als auch technische Maßnahmen zu ergreifen. Insbesondere im technischen Umfeld existiert bereits heute eine Reihe von etablierten Technologien, um die sensiblen personenbezogenen Daten entsprechend zu schützen. Hierbei ist jedoch darauf zu achten, dass die getroffenen Maßnahmen dem zu mitigierenden Risiko angemessen sind. Denn nur so können die getroffenen technischen Maßnahmen wirkungsvoll das Risiko minimieren.

Siegfried Höck arbeitet als Solution Architect für OPITZ CONSULTING und ist verantwortlich für Cloud-Architekturen und Migrationsstrategien für Kunden in die Cloud. Seine Leidenschaft ist es, Lösungsarchitekturen zu entwickeln, die den individuellen Bedürfnissen der Kunden entsprechen. Dabei muss ein breites Spektrum berücksichtigt werden: Es beginnt mit Top-Level-Architekturen und setzt sich bis in die Tiefen der Entwicklung fort. Siegfried ist Architekt, Projektleiter, zertifizierter Anforderungsingenieur und AWS Certified Solutions Architect.

Stefan Kühnlein arbeitet für OPITZ CONSULTING im Bereich Software-Engineering. In seiner Rolle als Solution Architect ist Stefan in verschiedenen Projekten tätig. Hierbei liegt sein Fokus aktuell vor allem auf dem Aufbau von robusten und zukunftsorientierten Architekturen mit Microservices bzw. nativen Cloud-Architekturen. Er betätigt sich zusätzlich als Referent auf verschiedenen Konferenzen und ist Autor von Fachartikeln.

Links & Literatur

[1] European Agency for Network and Information Security: https://www.enisa.europa.eu/topics/data-protection/privacy-enhancing-technologies

[2] Office of the Privacy Commissioner of Canada: „Privacy Enhancing Technologies – A Review of Tools and Techniques“: https://www.priv.gc.ca/en/opc-actions-and-decisions/research/explore-privacy-research/2017/pet_201711

[3] XACML: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.html

[4] Mezler-Andelberg, Christian: „Identity Management – eine Einführung“, dpunkt.verlag. https://books.google.de/books?id=WGj5DQAAQBAJ&printsec=frontcover&hl=de#v=onepage&q&f=false

[5] Logstash: https://www.elastic.co/blog/gdpr-personal-data-pseudonymization-part-1

[6] Fingerprint: https://www.elastic.co/guide/en/logstash/current/plugins-filters-fingerprint.html

[7] Always Encrypted: https://docs.microsoft.com/de-de/azure/sql-database/sql-database-always-encrypted

[8] Amazon GuardDuty: https://aws.amazon.com/de/guardduty/

[9] Azure Advanced Threat Protection Documentation: https://docs.microsoft.com/en-us/azure-advanced-threat-protection/