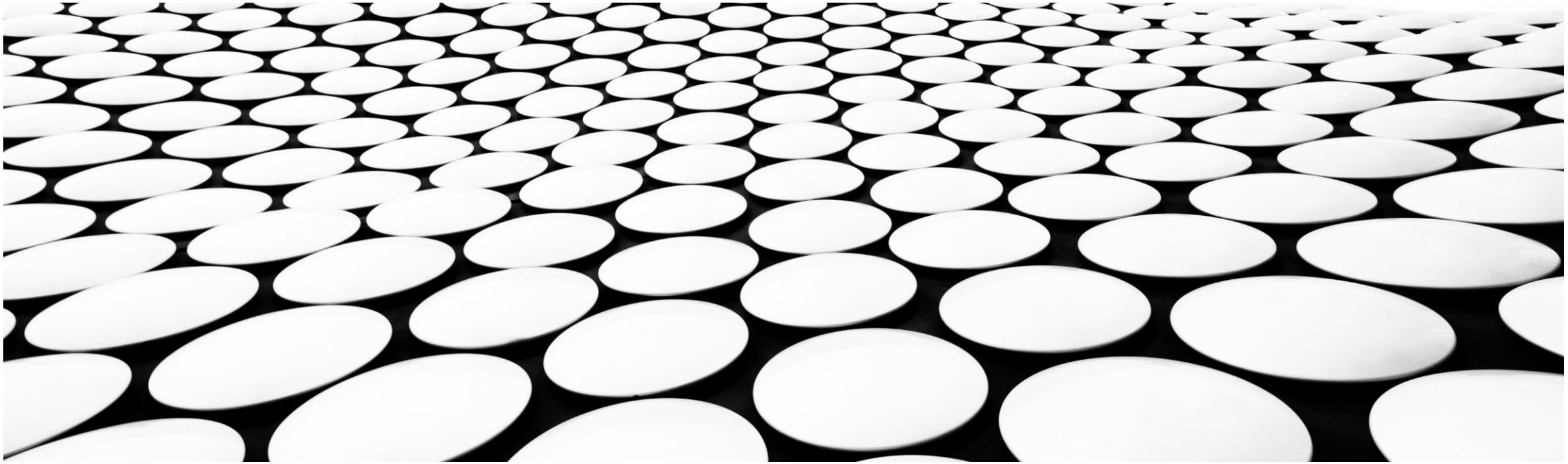
SICHERE SOFTWAREENTWICKLUNG

SICHERHEITSASPEKTE BEI DER ENTWICKLUNG VON SOFTWAREANWENDUNGEN BERÜCKSICHTIGEN



SCHUTZZIELE DER INFORMATIONSSICHERHEIT

Ziel:

- Informationen jeglicher Art und Herkunft zu schützen

Dies umfasst:

- Informationen, die sich in den Köpfen der Beteiligten befinden
- Informationen in Papierform
- Informationen, die elektronisch in Systemen und Netzwerken verarbeitet werden.

Informationssicherheit verfolgt somit einen breiteren Ansatz als die IT-Sicherheit, die sich vorrangig auf elektronisch gespeicherte Informationen und die verarbeitenden IT-Systeme bezieht.

SCHUTZZIELE DER INFORMATIONSSICHERHEIT

Die drei primären Schutzziele der Informationssicherheit:

- Vertraulichkeit
- Integrität
- Verfügbarkeit von Informationen

VERTRAULICHKEIT

Vertraulichkeit ist der Schutz vor unbefugter Preisgabe von Informationen. Vertrauliche Daten und Informationen dürfen ausschließlich Befugten in der zulässigen Weise zugänglich sein.

z.B. Kundendaten, Patente oder Forschungsdaten

In der Praxis besteht die Gefahr von Verstößen gegen die Vertraulichkeit, wenn z.B. Flipcharts und Whiteboards nach Strategiebesprechungen nicht gereinigt werden, oder Kundendaten in Büros mit Publikumsverkehr offen zugänglich sind. Um die Vertraulichkeit von Informationen sicherzustellen, muss klar festgelegt sein, wer in welcher Art und Weise berechtigt ist, auf diese Daten zuzugreifen.

INTEGRITÄT

Integrität bezeichnet die Sicherstellung der Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen.

Der Verlust der Integrität von Informationen kann bedeuten, dass diese unerlaubt verändert, Angaben zum Autor verfälscht oder Zeitangaben zur Erstellung manipuliert wurden.

Integritätsverlust von Informationen kann durch unautorisiertes Ändern, Löschen oder Einfügen von Daten eintreten.

VERFÜGBARKEIT

Die Verfügbarkeit von Dienstleistungen, Funktionen eines IT-Systems, IT- Anwendungen oder IT-Netzen oder auch von Informationen ist vorhanden, wenn diese von den Anwendern stets wie vorgesehen genutzt werden können.

z.B. die Anforderungen an Server, Netze und Rechenzentren, die oft vertraglich in Service Level Agreements geregelt sind.

ERWEITERTE SCHUTZZIELE DER INFORMATIONSSICHERHEIT

AUTHENTIZITÄT

„Mit dem Begriff Authentizität wird die Eigenschaft bezeichnet, die gewährleistet, dass ein Kommunikationspartner tatsächlich derjenige ist, der er vorgibt zu sein. Bei authentischen Informationen ist sichergestellt, dass sie von der angegebenen Quelle erstellt wurden. Der Begriff wird nicht nur verwendet, wenn die Identität von Personen geprüft wird, sondern auch bei IT-Komponenten oder Anwendungen.“

Dieses Schutzziel der Echtheit von Informationen ist wichtig, um die Vertrauenswürdigkeit des Ursprungs einer Information bewerten zu können.

NICHTABSTREITBARKEIT

„Bei der Nichtabstreitbarkeit liegt der Schwerpunkt auf der Nachweisbarkeit gegenüber Dritten. Ziel ist es zu gewährleisten, dass der Versand und Empfang von Daten und Informationen nicht in Abrede gestellt werden kann. Es wird unterschieden zwischen

- Nichtabstreitbarkeit der Herkunft: Es soll einem Absender einer Nachricht unmöglich sein, das Absenden einer bestimmten Nachricht nachträglich zu bestreiten.
- Nichtabstreitbarkeit des Erhalts: Es soll einem Empfänger einer Nachricht unmöglich sein, den Erhalt einer gesendeten Nachricht nachträglich zu bestreiten.“

VERBINDLICHKEIT

„Unter Verbindlichkeit werden die Sicherheitsziele Authentizität und Nichtabstreitbarkeit zusammengefasst. Bei der Übertragung von Informationen bedeutet dies, dass die Informationsquelle ihre Identität bewiesen hat und der Empfang der Nachricht nicht in Abrede gestellt werden kann.“

Ein Beispiel für die Schutzziele der Nichtabstreitbarkeit und der Verbindlichkeit ist das Identitätsmanagement von Organisationen. Dadurch können Handlungen immer eindeutigen Identitäten zugeordnet werden und nachvollziehbar und verbindlich Nutzern zugewiesen werden können, sodass sie auch nicht abgestritten werden können.

ZUVERLÄSSIGKEIT

Das Schutzziel der Zuverlässigkeit bezieht sich auf die technische Funktionsfähigkeit von IT-Systemen und Komponenten und kann daher in Szenarien hoher Abhängigkeit von IT-Systemen zusätzlich zum Schutzziel der Verfügbarkeit betrachtet werden.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 1: Injektionen sowie XSS- und CSRF-Schwachstellen verhindern!

Zu den am meisten verbreiteten Sicherheitsschwachstellen in Web-Anwendungen gehören Injektions-, XSS- und CSRF-Verwundbarkeiten. Sie gelten als das am meisten genutzte Einfallstor für Cyber-Kriminelle. Schaffen Hacker aufgrund dieser Schwachstellen im System den Zugriff auf sensible Daten der Kunden, lassen sich diese manipulieren und schädigen im schlimmsten Fall dauerhaft die gesamte IT-Infrastruktur.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Injektionsschwachstellen

Injektionsschwachstellen entstehen, wenn Benutzereingaben zur Erstellung von Anweisungen oder Abfragen, z.B. SQL-Abfragen, verwendet und ohne ausreichende Überprüfung an einen Interpreter weitersendet werden. Durch Manipulation der Eingabesyntax kann ein Angreifer unerwünschte Befehle ausführen und unbefugten Zugriff auf sensible Daten erhalten. Zum Beispiel ermöglicht eine SQL-Injection in einer Web-Anwendung einen Datenbankzugriff im Backend.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Cross-Site-Scripting

Cross-Site-Scripting (XSS) ist eine Art HTML-Injection, die im Web-Browser des Benutzers stattfindet. XSS-Schwachstellen entstehen, wenn Web-Anwendungen Benutzereingaben annehmen und diese anschließend ohne Validierung an den Browser weiterleiten. Ein Beispiel dafür wäre, wenn ein Benutzer einen Suchbegriff in ein Formular eingibt und anschließend eine neue Seite mit Suchergebnissen im Browser angezeigt wird, auf der auch vom Benutzer gewählte Suchbegriff nochmals mit erscheint. Auf diese Weise können Angreifer Schadcode auf indirektem Wege im Browser des Opfers ausführen – häufig über bestimmte, gut vorbereitete URLs.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Cross-Site-Request-Forgery

Bei einem CSRF-Angriff (Cross-Site-Request-Forgery) führt ein Angreifer HTTP-Anfragen über den Web-Browser eines Anwenders aus. Aus Sicht der Anwendungen sehen die Anfragen legitim aus und werden im Kontext und mit den Rechten eines aktuell angemeldeten Benutzers bearbeitet. Die Anwendung unterscheidet nicht zwischen den Anfragen des rechtmäßigen Nutzers und den gefälschten Anfragen des Angreifers. Anwendungen ohne SSRF-Schutz stellen immer eine potentielle Bedrohung für Benutzer und deren Daten dar.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Tipps zur Programmierung

- Benutzereingaben nur im Bedarfsfall
- Eingaben filtern und prüfen

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 2: Trennung von Datenverarbeitung und Datendarstellung

Wichtig bei der Verarbeitung von Daten ist die Trennung von Darstellungslogik, z. B. Ausgabe im Web-Browser, und der internen Anwendungslogik. Dafür ist das Model-View-Controller-Pattern (MVC) prädestiniert, das vielen Java-Entwicklern noch von der GUI-Programmierung mit Java-Swing bekannt sein dürfte.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 3: Daten vor der Verarbeitung immer validieren

Die konsequente Validierung, also die Verifizierung von Eingabe- und Ausgabedaten, stellt ein Grundprinzip für die Entwicklung von korrekt funktionierenden, stabilen und sicheren Anwendungen dar. Fehlerhafte Datenpools oder gar Sicherheitsschwachstellen entstehen oftmals aufgrund fehlerhafter Prüfung von Syntax und Semantik.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 4: Datenschutz zwischen Web-Browser und Anwendung vor externer Einsicht

Daten, die im Austausch stehen zwischen Servern und Web-Browsern, benötigen Schutz vor externer Einsicht. Eine sichere Datenübertragung zwischen Client und Server, z.B. über eine HTTP(S)-gesicherte Verbindung, ist wichtig, da die Bevölkerung zunehmend freie WLAN-Hotspots mit unverschlüsselten Verbindungen akzeptiert. Angreifer nutzen diese Lücke, um private und sensible Informationen wie Passwörter, Nutzersitzungen oder auch persönliche Daten einzusehen.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 5: Die Nutzer der Anwendung im Auge behalten

Fehler in der Nutzer- und Sitzungsverwaltung von Web-Anwendungen führen oftmals zu Sicherheitsschwachstellen. Beispielsweise durch Authentifizierungsfehler, unzureichende Prüfroutinen während des Zurücksetzens des Passworts oder durch unvollständige Prüfungen der jeweiligen Berechtigungen bei der Ausführung von Aktionen innerhalb der Anwendung (Datenzugriff, Änderungen, Datensätze löschen). Daher ist es wichtig, den Nutzer und seine Berechtigungen innerhalb einer Sitzung in den Anwendungen jederzeit im Auge zu behalten. Dies gilt insbesondere für die Prüfung von Berechtigungen bei der Ausführung von CRUD-Operationen.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 6: Über die geeignete Protokollierung von Ereignissen nachdenken

Während des normalen Betriebs der Anwendungen kommen eine Vielzahl von Ereignissen und Aktionen von Nutzern vor. Treten in der Anwendung Fehler auf oder hat sich ein sicherheitsrelevanter Vorfall, wie ein Hackerangriff ereignet, ist es für eine erfolgreiche Post-Mortem-Analyse unerlässlich, die Ereignisse nachhaltig zu betrachten. Hierzu braucht es eine ausführliche Protokollierung (Logging) der Anwendungen, Ereignisse und Benutzerinteraktionen. Dabei sind insbesondere die Datenschutzbestimmungen zu berücksichtigen.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 7: APIs und REST-Schnittstellen sichern

Häufig stellen Web-Anwendungen auch APIs sowie Teile ihrer Funktionalität über sogenannte REST-Schnittstellen bereit. Diese Schnittstellen implementieren und sichern Entwickler genauso robust wie die übrige Anwendung. Dazu gehören das Filtern und Validieren von Ein- und Ausgaben sowie das sichere und zuverlässige Abrufen und Verifizieren von Nutzern und deren Rechten. Je nachdem, welche Datenformate diese Schnittstellen verwenden, z.B. JSON oder XML, benötigen sie Filterroutinen, die spezifische Sicherheitsfunktionen für das jeweilige Format ergänzen.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 8: Die Methoden selbst testen, bevor andere es tun

Bereits während der Entwicklung sollten Programmierer die Funktionen und Schnittstellen auf korrektes Funktionieren und ihr Verhalten bei fehlerhaften Parametern und Eingabewerten überprüfen. Ein bewährtes Verfahren für diesen Test von Modulen sind die sogenannten Unit-Tests. In der Prüfeinheit erstellt der Test für jede Funktion oder Methode eine Prüfroutine, die zunächst einen Ausgabezustand erzeugt, zum Beispiel Zuweisung von Variablenwerten, dann die zu testende Funktion/Methode aufruft und schließlich das Ergebnis, wie einen Ausgabewert mit dem erwarteten oder gewünschten Verhalten vergleicht.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 9: Keinen externen und nicht selbst kontrollierten Quellen vertrauen

Web-Anwendungen enthalten oft Inhalte aus anderen Quellen, insbesondere von Content-Providern. Der Inhalt wird meist im Quellcode referenziert und vom Web-Browser des Nutzers dynamisch geladen. Bei der Integration von Inhalten aus Drittquellen ist die Vertrauenswürdigkeit der Quelle zu klassifizieren und gegebenenfalls sind die von Dritten gelieferten Inhalte zu überprüfen oder zu filtern.

GRUNDSÄTZE FÜR SICHERE SOFTWAREENTWICKLUNG

Grundsatz 10: Bei der Auswahl eines Partners die Augen offen halten

Um Funktionalitäten in Anwendungen zu implementieren, verwenden Softwareentwickler häufig Softwarekomponenten von Drittanbietern. Bei dem Gebrauch von Programmbibliotheken, Plugins und Add-Ons anderer Anbieter oder von Open-Source-Projekten sind diese auf ihre Robustheit und Sicherheit zu testen sowie die entsprechenden Lizenzbestimmungen zu überprüfen. Module externer Anbieter weisen manchmal technische Schwachstellen auf, die bei ihrer Integration neue zusätzliche Angriffsvektoren liefern. Auch wenn die eigenen Entwicklungen keine Schwachstellen aufweisen, besteht die Möglichkeit einer Kompromittierung des Systems durch unsichere Module und damit einer Gefahr für die Kundendaten.