# Software Architecture WS 13/14

Tim Krämer

4. Februar 2014

# Seminararbeit

**Len Bass**

*Software Architecture in Practice (3rd Edition)*

Kapitel 9: Security

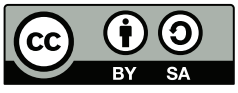
Fachbereich Informatik

Universität Hamburg

# Inhaltsverzeichnis







# 1. Einleitung

Len Bass beschreibt in seinem Buch „Software Architecture in Practice, Third Edition“ die Aspekte und Annahmen bei der Planung von Softwarearchitekturen. Die Softwarearchitektur stellt eines der wichtigsten Punkte in der Entwicklung einer Software dar. Sie enthält unter anderem Softwarequalitätsattribute, wie Perfomanz, Sicherheit, Testbarkeit und Benutzbarkeit. Über die Ausprägung der einzelnen Qualitätsattribute wird häufig schon zu beginn anhand von Anforderungen und Qualitätssicherung entschieden. Eine einmal eingerichtete Softwarearchitektur ist später nur mit hohem Aufwand veränderbar, deshalb müssen Softwarearchitekten in der Lage sein, Entscheidungen über die Ausrichtung der Softwaresysteme im Hinblick auf die Anforderungen an das zu erstellende oder das zu erweiternde System zu treffen. Diese Seminararbeit wurde im Rahmen des Seminars „Softwarearchitektur“ im WS 13/14 an der Universität Hamburg erstellt und behandelt das 9. Kapitel aus „Software Architecture in Practice, Third Edition“, welches sich mit dem Qualitätsattribut „Security“ beschäftigt, im Folgenden als „Softwaresicherheit“ bezeichnet.

# 2. Softwaresicherheit als Qualitätsattribut in der Softwarearchitektur

Im Kontext der Qualitätsattribute in der Softwarearchitektur finden wir häufig messbare Metriken oder greifbare Konzepte mit direktem Bezug zur Praxis. Im Fall der Softwaresicherheit geht es um die Eigenschaften des Systems die den autorisierten Zugriff auf Daten und Informationen des Systems verwalten. Im Gegengewicht dazu befinden sich die Angriffe. Dabei handelt es sich Handlungsweisen, die sich mit Absicht gegen eine Software oder das Computer System richten. Angriffe können in unterschiedlichsten Formen auftreten, beispielsweise ein unautorisierter Zugriffsversuch auf Daten oder Dienste.
Bei den grundlegenden Charakteristika von Softwaresicherheit handelt es sich um *Vertraulichkeit*, *Unversehrtheit*, *Verfügbarkeit* (im englischen Original: *Confidentiality*, *Integrity*, *Availability*) [Bass et al., 2012].

- Ein Computersystem oder eine Softwarearchitektur besitzt die Eigenschaft der *Vertraulichkeit*, wenn die Daten und das System selbst gegen unautorisierten Zugriff geschützt sind. Beispielsweise darf ein Angreifer nicht in der Lage sein die Steuerunterlagen fremder Personen einzusehen, die sich auf einem Computer des Finanzamts befinden.

- *Unversehrtheit* oder Integrität ist die Eigenschaft die besagt, dass Daten oder Dienste nicht Opfer von unautorisierten Veränderungen werden. So muss sichergestellt sein, dass sich eine vom Prüfer vergebene Note danach nicht verändert hat.

- Wenn die Daten oder das Computersystem, wie geplant verwendet werden können, spricht man von *Verfügbarkeit*. Sollte das System zu Beispiel auf Grund einer zu hohen Zahl von Anfragen nicht antworten können, ist es nicht verfügbar.

Um diese Charakteristika entscheidend zu stützen werden weitere Softwarequalitätsattribute eingeführt.

- **Authentisierung**: Durch eine erfolgreiche Authentisierung des Kommunikationspartners kann sichergestellt werden, dass es sich tatsächlich um die korrekte, berechtigte Person handelt. Aus Sicht der Softwarearchitektur wird üblicherweise der Benutzer authentisiert um die Vertraulichkeit des Systems zu wahren, das System also gegen unautorisierten Zugriff zu schützen.

- **Unleugbarkeit** im englischen Original „nonrepudation“: Bei einer legitimen Kommunikation ist es im Interesse beider Kommunikationspartner, dass die Kommunikation später nicht geleugnet werden kann. Für eine Softwarearchitektur ist diese Eigenschaft wichtig, da im Rahmen der Implementation eine Authentisierung stattfindet mit der sich beweisen lässt, dass ein Datentransfer oder eine Kommunikation nur zwischen berechtigten Parteien stattgefunden hat. Dies ist die Grundlage für die Unversehrtheit von Daten.

- **Autorisierung** - Wenn Benutzern bestimmte Berechtigungen zugeordnet werden, die für das geplante Benutzen der Software benötigt werden, spricht man von autorisierten Benutzern. Die benötigten Berechtigungen der Benutzer müssen bei der Erstellung der Softwarearchitektur eingeplant werden und bei Veränderungen und Weiterentwicklungen stets gepflegt werden. So kann ein autorisierter Benutzer das System immer so nutzen, wie es vorgesehen ist. Das System bleibt also verfügbar.

In den allgemeinen Sicherheitsszenarien werden diese Charakteristika verwendet. Die Ziele, die durch diese Maßnahmen vor Angriffen beschützt werden sollen sind gespeicherte Daten, Daten auf dem Übertragungsweg, und laufende Prozesse.

# 3. Allgemeines Sicherheitsszenario

Im Bereich der IT-Sicherheit wird zumeist ein sog. „Threat Model“ eingesetzt. Ein Bestandteil des Threat Models ist ein sog. „Attack Tree“, der genuzt wird um mögliche Gefahren zu bestimmen. Die Wurzel dieses Baus ist ein erfolgreicher Angriff, bei den davon abgehenden Zweigen, handelt es sich um mögliche direkte Grüne für diesen Angriff. Weitere Verästelungen teilen die Gründe in Unterpunkte auf.
Bei einem Angriff handelt es sich um den Versuch die Charakteristika der Softwaresicherheit zu verletzen. Die Reaktion auf einen solchen Angriff muss die Absicherung dieser Charakteristika sein, oder die Abwehr der Angreifer durch Beobachtung ihrer Aktivitäten. Mit diesen Überlegungen werden nun die einzelnen Bestandteile eines allgemeinen Sicherheitsszenarios in Tabelle 3 dargestellt. Ein Beispiel Szenario wird in Abbildung 3 gegeben.

- *Quelle des Ereignisses (Source of stimulus).* Bei der Quelle eines Angriffs kann es sich entweder um einen Menschen oder ein anderes System handeln. Die Quelle könnte bereits vorher identifiziert worden sein, oder sie ist bisher noch unbekannt. Ein

| **Portion of Scenario** | **Possible Values** |
|---|---|
| Source | Human or another system which may have been previously identified (either correctly or incorrectly) or may be currently unknown. A human attacker may be from outside the organization or from inside the organization. |
| Stimulus | Unauthorized attempt is made to display data change or delete data, access system services, change the system's behavior, or reduce availability. |
| Artifact | System services, data within the system, a component or resources at the system, data produced or consumed by the system. |
| Artifact | The system is either online or offline: either connected to or disconnected from a network: either behind a firewall or open in a network; fully operational, partially operational, or not operational. |
| Response | Transactions are carried out in a fashion such that<br>• Data or services are protected from unauthorized access.<br>• Data or services are not being manipulated without authorization.<br>• Parties to a transaction are identified with assurance.<br>• The parties to me transaction cannot repudiate their involvements.<br>• The data, resources, and system services will be available for legitimate use.<br><br>The system tracks activities within it by<br>• Recording access or modification<br>• Recording attempts to access data. resources, or services<br>• Notifying appropriate entities (people or systems) when an apparent attack is occurring |
| Response Measure | One or more of the following:<br>• How much of a system is compromised when a particular component or data value is compromised<br>• How much time passed before an attack was detected<br>• How many attacks were resisted<br>• How long does it take to recover from a successful attack<br>• How much data IS vulnerable to a particular attack |

Tabelle 1: Allgemeines Sicherheitsszenario

menschlicher Angreifer kann von außen oder im inneren der betroffenen Organisation agieren.

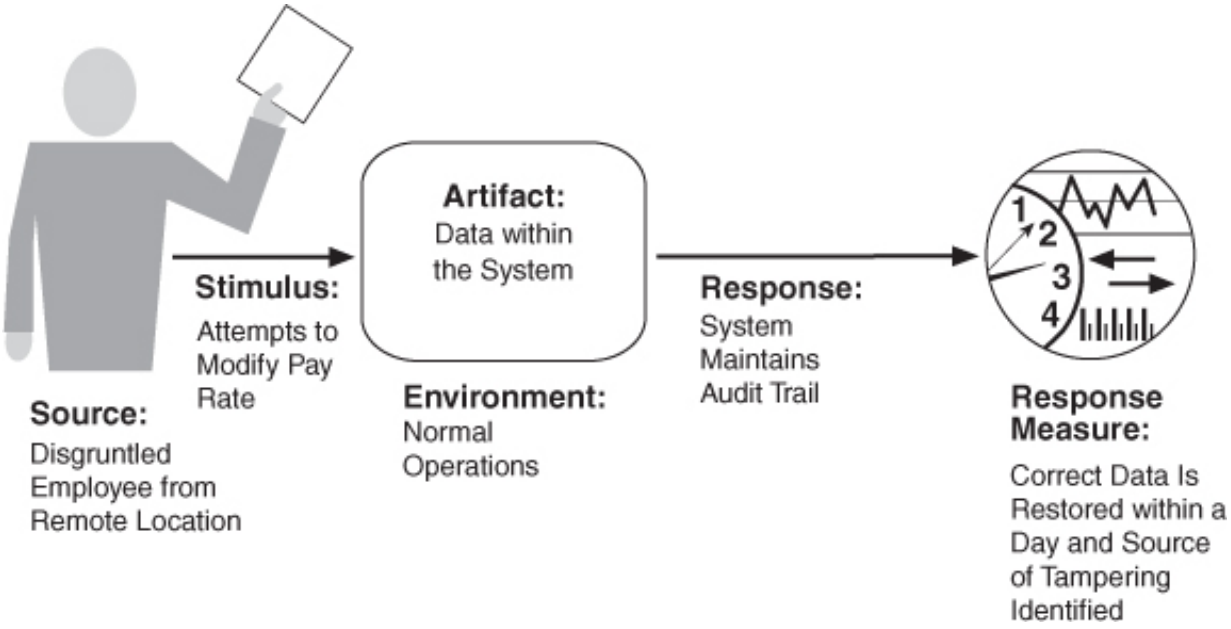



Abbildung 1: Beispiel für ein konkretes Sicherheitsszenario

- *Ereignis (Stimulus.* Bei dem Ereignis handelt es sich um einen Angriff. Es wurde als unautorisierter Zugriff auf Daten innerhalb des betroffenen Systems erkannt. Der Angriff könnte Daten anzeigen, verändern oder löschen, auf Dienste zugreifen, das Verhalten des Systems verändern oder die Verfügbarkeit reduzieren wollen.

- *Ziel (Artifact).* Ziele des Angriffs sind entweder die Dienste des Systems, die darin enthaltenen Daten oder die Daten, die von dem System erzeugt oder gesammelt wurden. Manche Angriffe werden auf ganz bestimmte verwundbare Komponenten des Systems gestartet.

- *Umgebung (Environment).* Ein Angriff kann erfolgen, wenn das betroffene System online oder offline ist, eine Verbindung zum Internet hat oder nicht, hinter einer Firewall betrieben wird oder in einem offenem Netzwerk, vollständig funktionsfähig, halbwegs funktionsfähig oder nicht funktionsfähig ist.

- *Reaktion (Response).* Das System sollte sicherstellen, dass Transaktionen so ausgeführt werden, dass Daten oder Dienste vor unautorisierten Zugriffen geschützt sind; Daten oder Dienste nicht ohne Autorisierung verändert werden können; die beteiligten Parteien einer Transaktion sicher identifiziert wurden; die beteiligten Parteien einer Transaktion nicht ihre Teilnahme verleugnen können; die Daten, Ressourcen und Dienste für die geplante Nutzung zur Verfügung stehen.
  Außerdem sollten die systemeigenen Aktivitäten so protokolliert werden, dass der Zugriff oder die Veränderung von Daten, Diensten oder Ressourcen erkannt werden

kann. Im Falle eines Angriffs sollten die zuständigen Stellen (Personen oder Systeme) benachrichtigt werden.

- *Reaktionsbewertung (Response measure).* Die Bewertung der Reaktion des Systems auf einen Angriff beinhaltet in wie fern das System kompromittiert ist, wenn eine bestimmte Komponente oder bestimmte Daten kompromittiert wurden, wie viel Zeit vergangen ist, bis ein Angriff als solcher erkannt wurde, wie viele Angriffe abgewehrt wurden, wie lange es dauert um sich von einem erfolgreichem Angriff zu erholen und wie viele Daten gegenüber einem bestimmten Angriff verwundbar waren.

Tabelle 3 zählt die Elemente eines allgemeinen Sicherheitsszenarios auf, die Sicherheit im Rahmen der Softwarearchitekturen charakterisieren. Abbildung 3 zeigt ein konkretes Beispiel aus [Bass et al., 2012, Seite 9999]: Ein verärgerter Angestellter versucht die Tabelle zu verändern, in der die Arbeitslöhne verwaltet werden. Dies passiert während der normalen Laufzeit und von einem externem Computer. Das System verwaltet einen sog. „Audit Trail“, eine Protokollierung aller Aktivitäten und stellt die korrekten Daten im Laufe eines Tages wieder her.

# 4. Taktiken für Softwaresicherheit

Analog zu den Sicherheitsmaßnahmen in der physischen Welt, können diese Konzepte in der Softwarearchitektur angewandt werden. Sichere Einrichtungen haben eine Zugangsbeschränkung, beispielsweise eine Sicherheitsschleuse. Die Aufgaben einer solchen Schleuse sind, Eindringlinge zu erkennen (zum Beispiel in dem legitimen Besuchern ein Besucherausweis ausgestellt wird) und Maßnahmen zum Aufhalten der Eindringlinge bereit zu stellen (automatisch schließende Türen und Wachpersonal).
Solche Überlegungen führen zu vier Taktiken der Softwaresicherheit:

- Erkennen von Angriffen
- Standhalten gegen Angriffe
- auf Angriffe reagieren
- von Angriffen erholen

Abbildung 4 zeigt diese Taktiken als Ziele von Softwaresicherheit.

## 4.1. Erkennen von Angriffen

Beim Erkennen von Angriffen gibt es vier weitere Unterpunkte, die jeweils spezielle Taktiken beschreiben um bestimmte Angriffe zu entdecken.

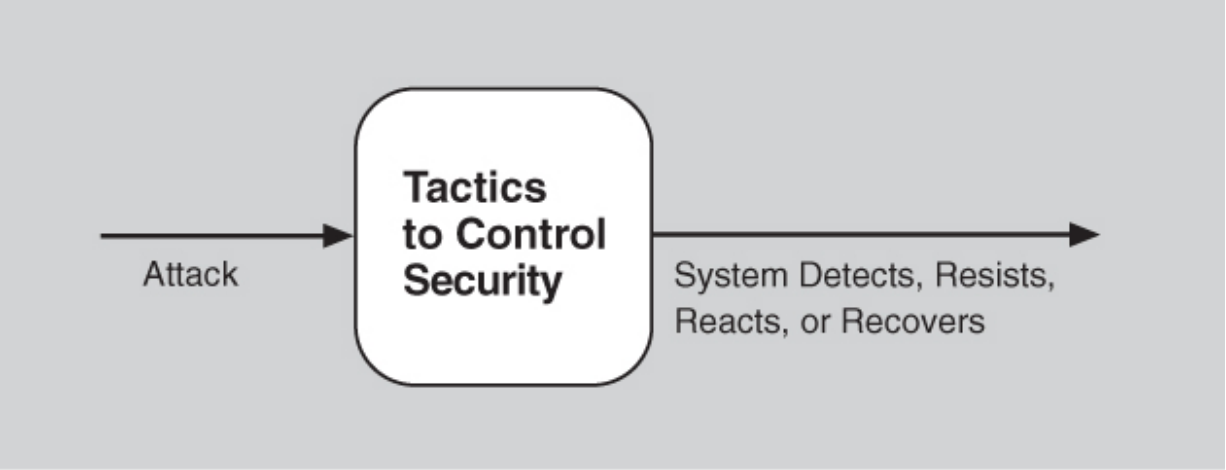



Abbildung 2: Taktiken für Softwaresicherheit

- *Eindringen erkennen (detect intrusion)*: Ähnlich zum Verfahren eines Antiviren Programms vergleicht ein sog. „Intrusion Detection System“ die Muster oder Signaturen von Netzwerkverkehr, die in das zu beschützende Syste eingehen, mit einer Datenbank von Signaturen oder bekannten Mustern von schädlichem Verhalten. Solche Signaturen können auf Protokollen, TCP-flags, Payloadgrößen, Anwendungen, Quell- oder Zieladressen oder Portnummern basieren.

- *Erkennen von Dienstblockaden (detect service denial)*: Auch hier werden Muster oder Signaturen im Netzwerkverkehr der in das System eingeht mit Profilen verglichen, die bereits als DOS (denial-of-service) Angriffe bekannt sind.

- *Unversehrtheit von Daten sicherstellen (verify message integrity)*: Checksummen oder Hashes können benutzt werden um die Integrität von Nachrichten, Ressourcen oder Konfigurationsdateien sicherzustellen. Eine Checksumme ist ein Verfahren bei dem das System eine bestimmte Menge von redundanten Informationen für die Konfigurationsdateien und Nachrichten speichert und diese Informationen benutzt um die betroffenen Dateien vor Gebraucht zu überprüfen. Sollte die Checksumme nicht stimmen, wurde die Datei höchstwahrscheinlich verändert. Ein Hash oder eine Hashsumme ist eine Kombination von Zahlen und Buchstaben die von einer Hashfunktion generiert wird. Die Hashfunktion ist so ausgelegt, dass selbst die kleinste Veränderung an einer Datei einen signifikant unterschiedliche Hashumme verursacht.

- *Erkennen von Verzögerter Übertragung (detect message delay)*: Bei dieser Taktik wird die Zeit gemessen, die eine Nachrichtenübertragung braucht. Sollte es bei der Übertragung eine ungewohnte Verzögerung geben, ist es möglich, dass die Nachrichten von einem Angreifer unterwegs abgefangen und abgehört werden. Besonders auffällig sind schwankende Übertragungszeiten bei einer sonst stabilen Verbindung.

## 4.2. Standhalten gegen Angriffe

Es gibt viele bekannte Methoden, um gegen einen Angriff standzuhalten:

- Akteure identifizieren: Beim identifizieren der „Akteure“ geht es eigentlich darum, die Quelle jeder externen Eingabe in das System zu identifizieren. Benutzer werden üblicherweise anhand von Benutzer-IDs identifiziert. Andere System könnten durch Passwörter, IP Adressen, Protokolle, Ports oder ähnlichem „identifiziert“ werden.

- Akteure authentisieren: Bei einer Authentisierung wird sichergestellt, dass ein Akteur (ein Benutzer oder ein fremder Computer) tatsächlich der ist, der es vorgibt zu sein. Passwörter, digitale Zertifikate und biometrische Identifikation bieten viele Möglichkeit für die Authentisierung.

- Akteure autorisieren: Die Autorisierung stellt sicher, dass ein authentisierter Benutzer die Möglichkeit hat auf Dienste und Daten zuzugreifen und bei Bedarf zu verändern. Dieses Verfahren wird üblicherweise durch Software-Zugangskontrollen in einem System realisiert.

- Zugriff einschränken: Den Zugang zu einem Dienst oder zu Ressourcen einzuschränken, kann durch den Einsatz von Speicherschutzmechanismen, Sperren von bestimmten Computern, Schließen von Ports oder Verweigern von bestimmten Protokollen erreicht werden.

- Kleinstmögliche Angriffsflächen: Die Taktik möglichst wenige Bereiche des Systems von außen erreichbar zu machen, beinhaltet die Idee, dass die Wahrscheinlichkeit einer unbekannten Sicherheitslücke linear zu der Anzahl von erreichbaren Diensten wächst.

- Verschlüsselung von Daten: Daten sollten immer vor unautorisiertem Zugriff geschützt sein. Vertraulichkeit wird üblicherweise durch eine Form von Datenverschlüsselung erreicht. Besonders bei Datenübertragungswegen oder externen Speichermedien ist es besonders wichtig, dass geeignete verschlüsselte Kanäle und Abstraktionen eingesetzt werden. Verschlüsselung kann symmetrisch (mit einem Passwort) oder asymmetrisch (mit öffentlichen und geheimen Schlüsseln) implementiert werden.

## 4.3. auf Angriffe reagieren

Potentiellen Angriffen kann mit der richtigen Taktik entsprechend entgegnet werden:

- Zugang entziehen: Wenn das System oder der Systemadministrator glaubt, dass ein Angriff erfolgt, kann der Zugang zu sensiblen Ressourcen für alle Benutzer global eingeschränkt werden. Ist zum Beispiel der Rechner eines Mitarbeiters mit einem Virus infiziert, kann der Zugang für diesen Mitarbeiter solange eingeschränkt werden, bis der Virus vollständig von dessen Computer entfernt wurde.

- Computer sperren: Wiederholte fehlgeschlagene Logins deuten auf einen potentiellen Angreifer hin. Viele Systeme sperren den betroffenen Computer aus, nachdem eine festgelegte Anzahl von fehlgeschlagenen Logins erfolgt ist. Da es sich häufig schlicht um ein Versehen eines legitimen Mitarbeiters handelt, sind diese Sperren üblicherweise nur für einen kurzen Zeitraum aktiv.

- Akteure informieren: Noch laufende Angriffe erfordern möglicherweise den Eingriff durch Personal oder andere Systeme. Diese Personen oder Systeme sollten benachrichtigt werden, wenn ein Angriff identifiziert wurde.

### 4.4. von Angriffen erholen

Wurde eine System erfolgreich angegriffen und hat diesen Angriff erkannt, muss es sich von diesem Angriff erholen und einen vertrauenswürdigen, funktionsfähigen Zustand wiederherstellen. Ein Teil dieser Wiederherstellung ist der Restauration von Diensten. So könnten beispielsweise zusätzliche Server oder Netzwerkverbindungen in Reserve gehalten werden um in solchen Fällen angeschaltet zu werden. Außerdem sollte ein Audit Trail protokolliert werden um die Handlungsweisen des Angreifers oder den Angreifer selbst zu identifizieren. Ein solches Protokoll hilft zusätzlich dabei, künftige Angriffe dieser Art frühzeitig zu erkennen und zu verhindern.
In Abbildung 4.4 sind die vorgestellten Taktiken in hierarchischer Struktur angeordnet.

## 5. Fazit

Angriffe gegen ein Softwaresystem können als Angriffe gegen die Vertraulichkeit, die Unversehrtheit oder Verfügbarkeit des Systems oder dessen Daten betrachtet werden. Vertraulichkeit bedeutet Daten von Personen fernzuhalten, die auf diese keinen Zugriff erhalten sollten und gleichzeitig Zugriff für diejenigen zu bieten, die das auch dürfen. Unversehrtheit bedeutet, dass es keine unautorisierten Veränderungen oder Löschungen von Daten gibt. Verfügbarkeit heißt, dass das System und all seine Komponenten erreichbar ist, für alle die berechtigt sind es zu nutzen.
Es sollte immer davon ausgegangen werden, dass keine der beschriebenen Taktiken vollständig sicher ist und dass Systeme dennoch kompromittiert werden können. Deshalb ist es wichtig, dass Taktiken zum Erkennen von Angriffen, Eindämmung des Schadens und Wiederherstellung der kompromittierten Daten und Dienste existieren.

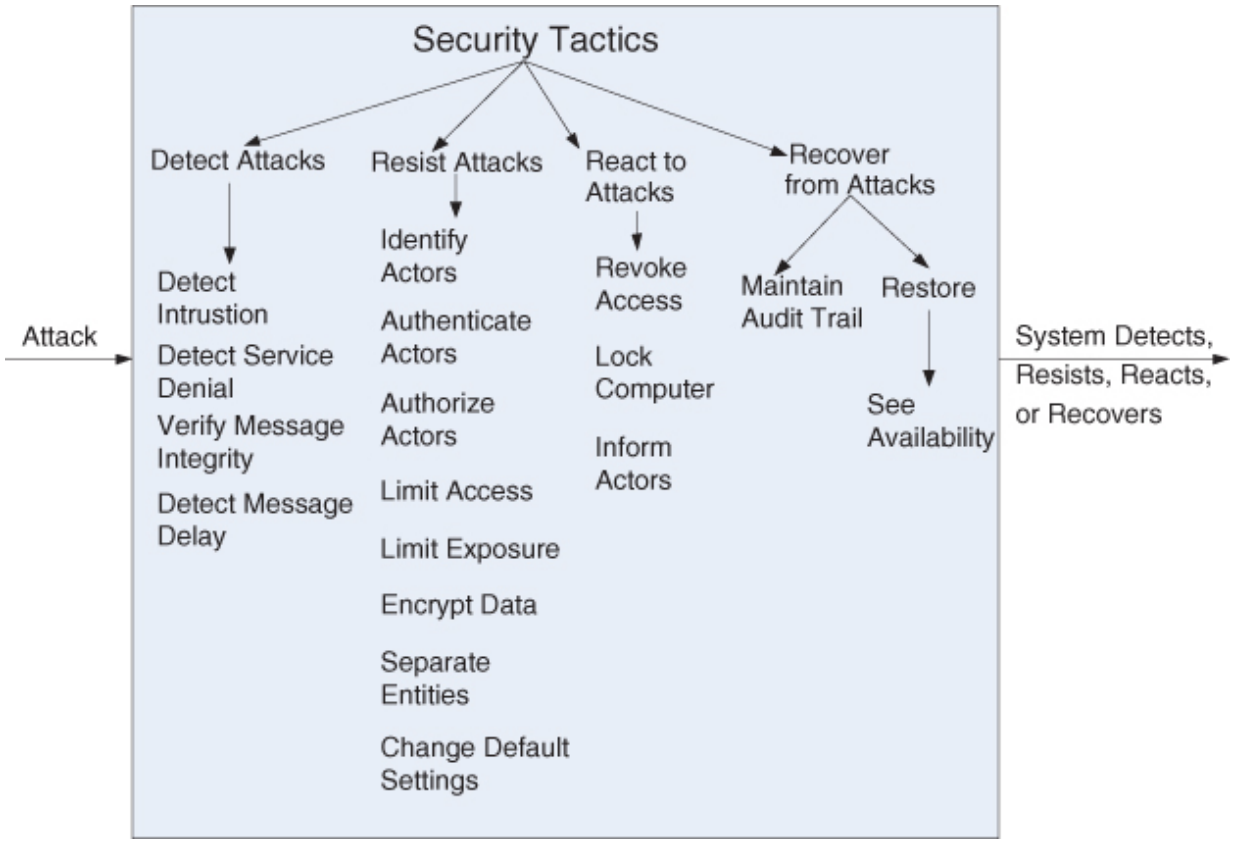



Abbildung 3: Taktiken für Softwaresicherheit

# A. Abbildungsverzeichnis



# Literatur

[Bass et al., 2012] Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice (3rd Edition) (SEI Series in Software Engineering)*. Addison-Wesley Professional.