M. Meier, D. Reinhardt, S. Wendzel (Hrsg.): Sicherheit 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 185

# Sicherheit im Softwareentwurf – Integrierte Modellierung von Security und Softwarearchitektur

Stefanie Jasser[1]

**Abstract:** Die Anforderungen an die Informationssicherheit heute eingesetzter Softwaresysteme steigen zunehmend. Dennoch werden sie während der Softwareentwicklung meist vernachlässigt. Softwarearchitekturen beeinflussen die Erfüllung nichtfunktionaler Eigenschaften wie Informationssicherheit wesentlich. Existierende Ansätze erlauben keine explizite Modellierung der Informationssicherheit eines Softwaresystems auf der Abstraktionsebene der Softwarearchitektur.
Der zu erarbeitende Ansatz ermöglicht die integrierte Modellierung von Aspekten der Informationssicherheit und den Architektureigenschaften eines Softwaresystems. Es ist geplant, die Evaluation dieses Ansatzes im Rahmen einer Feldstudie durchzuführen, d. h. die Anwendbarkeit wird durch den Einsatz in realen Softwareprojekten gezeigt.

**Keywords:** Secure Software, Security Engineering, Software Engineering, Softwarearchitektur, Modellierung, Informationssicherheit

## 1 Motivation

Softwaresysteme sind ein wesentlicher Bestandteil der heutigen Gesellschaft und Industrie. Viele dieser Systeme sind kritisch, d. h. sie sind etwa notwendig für die Fähigkeit eines Unternehmens, seine Dienstleistungen oder Produkte anzubieten. [MMR10] Zugleich werden Softwaresysteme offener und leichter angreifbar: Die Anforderungen an die Sicherheit dieser Softwaresysteme steigen. Dabei werden Anforderungen an den Schutz gegen Angriffe (Security) und den Schutz gegen unbeabsichtigte Ereignisse (Safety) unterschieden.

Nach [BSYJ15, SC09] muss die Sicherheit bereits während der initialen Entwicklung eines Softwaresystems berücksichtigt werden, um diese Anforderungen gerecht erfüllen zu können. Das bedeutet, Sicherheitsziele sind bereits in der Anforderungsanalyse zu erheben und zu formulieren und sollten den Softwareentwurf beeinflussen. Dies gilt besonders für den Entwurf der Softwarearchitektur, die für die Erfüllung nichtfunktionaler Eigenschaften eines Softwaresystems wesentlich ist. Zu diesen Eigenschaften zählen auch Security und Safety. Grundlegende Sicherheitsmängel (Security bzw. Safety Flaws) sind schon in der Softwarearchitektur erkennbar. Sie sind in späteren Entwicklungsphasen nur noch schwer zu beheben [FKK+14, SCH04] und müssen daher frühzeitig korrigiert werden. In den anschließenden Phasen Implementierung und Softwaretest muss die Einhaltung der Sicherheitsziele und der Softwarearchitektur sichergestellt und überprüft werden. Analysewerkzeuge auf Codeebene unterstützen die Durchführung von Penetrationstests oder Reviews.

[1] Universität Hamburg, Fachbereich Informatik, Arbeitsbereiche Sicherheit in verteilten Systemen und Softwareentwicklungs- und Konstruktionsmethoden, Vogt-Kölln-Straße 30, 22527 Hamburg, jasser@informatik.uni-hamburg.de

[NNY10] geben einen Überblick über Arbeiten, die die Erhebung und Formulierung von Sicherheitsanforderungen behandeln. Weniger Autoren befassen sich mit der Berücksichtigung von Sicherheitsaspekten im Softwareentwurf. Einige Ansätze unterstützen den Entwurf fehlertoleranter Softwaresysteme und beziehen Safety-Aspekte ein. [WK04, VKE12] Die meisten Ansätze, die Security im Softwareentwurf berücksichtigen, behandeln nur einzelne Security-Aspekte. Alle Arten von Security-Aspekten eines Softwaresystems können in keinem Ansatz modelliert werden. [BSYJ15]

Auch nach der initialen Entwicklung verändern sich die Umgebung und die Anforderungen an ein Softwaresystem. Es muss an diese neu entstehenden Anforderungen angepasst werden. Dieser Prozess heißt Softwareevolution und kann auch die Softwarearchitektur betreffen (Architekturevolution). [SB12, MMR10] Durch Änderungen des Systems und seiner Umgebung können wiederum Bedrohungen und Security Flaws entstehen. Forschungsarbeiten zur Berücksichtigung der Security während der Evolution eines Softwaresystems fehlen weitgehend. [NNY10] Dies gilt insbesondere für die Architekturevolution. [FKK+14, Ren06]

## 2 Stand der Forschung

Die Mehrheit der existierenden Ansätze erlauben nur die Modellierung ausgewählter Sicherheitsaspekte. Meist ist dies die Zugriffskontrolle (Access Control). Als Basissprache nutzen viele Autoren die *Unified Modeling Language* (UML) [BSYJ15]: Ein Ansatz, der die Modellierung verschiedener Sicherheitsaspekte erlaubt, ist UMLsec. UMLsec nutzt den vorgesehenen Erweiterungsmechanismus von UML, indem es ein Profil definiert. [Jür02, JW02] UML-Diagramme können so durch sicherheitsspezifische Stereotypen und Tags annotiert werden. Die meisten dieser Sicherheitsaspekte zählen zur Safety und die Modellierung erfolgt nicht explizit auf der Architekturebene. Dies gilt auch für andere auf UML basierende Ansätze wie SecureUML, das die Modellierung von rollenbasierten Zugriffskontrollen ermöglicht. [LBD02, BSYJ15, BDL06].

Mit MASC (Modelling Architectural Security Concerns) erweitern [SYB+15] UML, um Architekturentscheidungen für ausgewählte Security-Konzepte wie Kryptographie oder Interception zu dokumentieren. Eine automatisierte Validierung der Modelle findet nicht statt. MASC wurde bisher nicht anhand realer Projekte evaluiert.

Für die Erweiterung von UML um Sicherheitskonzepte spricht, dass UML als de-facto Standard der Industrie gilt, um Modelle in der Softwareentwicklung zu erstellen. Softwareentwickler und -architekten sind den Umgang mit UML-Modellen gewohnt. Die Notation wird in der Industrie jedoch häufig nur informell genutzt, da formal korrekte UML-Modelle leicht sehr komplex werden. Diese Komplexität wird durch Erweiterungen wie zusätzliche Elemente oder Annotationen noch erhöht, [SYB+15] sodass eine Nutzung außerhalb des wissenschaftlichen Kontextes unwahrscheinlich ist. Dies gilt insbesondere für eine automatisierte Validierung, die eine formal korrekte Modellierung voraussetzt. Meist werden UML-Diagramme nicht für die Modellierung auf Architekturebene genutzt.

Auch andere Architekturbeschreibungssprachen (Architecture Description Languages, ADLs) wurden um Sicherheitsaspekte erweitert. Diese Erweiterungen beziehen sich wiederum nur auf einzelne Security-Aspekte und erlauben deren ganzheitliche Modellierung nicht. Der in [Ren06, RTDR05] vorgestellte Ansatz Secure xADL beschreibt beispielsweise Aspekte der Zugriffskontrolle auf Architekturebene.

Das *Threat Modeling* ist ein Mittel zur Analyse von Bedrohungen für ein Softwaresystem. Eine verbreitete Threat Modeling Technik ist der iterativ angelegte STRIDE-Ansatz von [HL06], dessen Name ein Akronym der sechs Bedrohungskategorien: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. Er wird im Microsoft Secure Development Lifecycle eingesetzt, der Bedrohungsanalysen anhand eines Datenfluss-Diagramms vorsieht. Um identifizierte Bedrohungen zu mindern, werden Gegenmaßnahmen in Form von Änderungen des Softwaresystems ergriffen. Das neue Modell kann wiederum auf Bedrohungen analysiert werden. Weder diese Bedrohungen noch die ergriffenen Gegenmaßnahmen sind explizit im Datenfluss-Diagramm sichtbar und ein direkter Abgleich mit üblichen Softwarearchitektur-Modellen ist schwierig: Die initiale und evolutionäre Entwicklung des Softwaresystems wird unzureichend unterstützt.
Bedrohungen können auch mithilfe anderer Techniken modelliert werden. Eine solche Technik sind die von [Sch99] vorgestellten Attack Trees, deren Wurzelknoten ein Angriffsziel beschreiben. Die Kindknoten stellen Möglichkeiten dar, dieses Angriffsziel zu erreichen. [SO05] nutzen Misuse Cases oder Abuse Cases zur Analyse von Bedrohungen während der Anforderungsanalyse. Sie modellieren angelehnt an UML Use Case-Diagramme missbräuchliche Verwendungen eines Softwaresystems. Diese Techniken zur Bedrohungsanalyse beschreiben die Nutzung separater Modelle, die während der Anforderungsanalyse oder des Softwareentwurfs erstellt werden. Sie erlauben keine integrierte Modellierung und Validierung der Sicherheitsziele und Architektureigenschaften eines Softwaresystems.

Zusammenfassend ist eine umfassende, leichtgewichtige Notation für die Modellierung von Security-Aspekten im Softwareentwurf notwendig. Insbesondere fehlt die Möglichkeit zur integrierten Modellierung von Security-Aspekten und der Softwarearchitektur. Dies stellen auch andere Autoren fest: [BSYJ15, FKK+14, NNY10].

## 3 Forschungsfragen und methodisches Vorgehen

Um Entwickler und Architekten in der Entwicklung sicherer Softwaresysteme zu unterstützen, muss in der Entwurfsphase des Softwareentwicklungsprozesses angesetzt und die Modelle bis zur Abschaltung des Systems aktuell gehalten werden. Im Promotionsvorhaben sollen daher folgende Forschungsfragen bearbeitet werden:

*Wie lassen sich Security-Aspekte und Softwarearchitektur integriert modellieren?*

Es wird eine integrierte Modellierung der Softwarearchitektur und ihrer Eigenschaften angestrebt. Die erarbeitete Notation soll eine leichtgewichtig Modellierung der Security-Aspekte eines Softwaresystems ermöglichen. Je höher die Zahl der benötigten Modelle

ist, desto aufwendiger wird deren Synchronisation. Diese Herausforderung tritt besonders während der Evolution der Modelle im Rahmen der Softwareevolution auf. Werden Security-Aspekte und Architektureigenschaften eines Softwaresystems in wenigen Modellen dargestellt, kann dies eine erhöhte Modellkomplexität verursachen. Zwischen diesen Herausforderungen muss eine Abwägung getroffen werden. Die Erfüllung modellierter Security-Ziele auf Architekturebene sowie deren Konsistenz sind im angestrebten Konzept automatisiert validierbar. Im Rahmen dieser Validierung soll auch untersucht werden, inwiefern Softwareentwickler und -architekten ohne Expertenwissen in Security im Entwurf sicherer Systeme unterstützt werden können.

*Welche Wechselwirkungen bestehen zwischen Security-Zielen und Architektureigenschaften im Rahmen der Evolution?*

Es sollen die Auswirkung veränderter Security-Anforderungen auf die Softwarearchitektur und vice versa untersucht werden. Basierend auf den Ergebnissen, ist ein Konzept zu erarbeiten, das die Evolution der integrierten Modelle unterstützt. Dies kann durch eine Entscheidungsunterstützung realisiert werden, die Eigenschaften und Entscheidungshistorie des Legacy-Systems einbezieht. Für Modelländerungen ist zu untersuchen, wie eine erneute Validierung effizient gestaltet werden kann.

*Wie erodieren Security-Funktionen in Softwaresystemen?*

Während der Entwicklung kommt es zu Abweichungen zwischen der geplanten Soll-Architektur und der tatsächlich implementierten Ist-Architektur eines Softwaresystems. Es soll untersucht werden, inwieweit sich die Erosion von Security-Funktionen von der Erosion anderer Software-Funktionen unterscheidet. Darauf aufbauend werden Gegenmaßnahmen wie Security-Refactorings erarbeitet.

Die Basis des Promotionsvorhabens bildet die Analyse existierender Arbeiten zur Modellierung und Dokumentation von Softwarearchitekturen und Security-Aspekten sowie anerkannter Security-Lösungen auf Architekturebene. Solche Lösungen können Security-Muster, -Taktiken oder Best Practices aus Wissenschaft und Industrie sein. Auf den Ergebnissen aufbauend, wird ein Konzept erarbeitet, das die leichtgewichtige Modellierung der Security-Aspekte eines Softwaresystems erlaubt. Die Anwendbarkeit dieses Konzepts wird durch Feldstudien [Pre01] in realen Softwareprojekten evaluiert.
Zur Evolution und Erosion von Security-Funktionen in Softwarearchitekturen existieren nur wenige Arbeiten. [FKK+14] Daher werden die Auswirkungen evolutionärer Änderungen mittels einer Diskussion und anhand realer Softwareprojekte untersucht. Dies umfasst Änderungen der Security-Ziele und der Softwarearchitektur. Die Betrachtung der Erosion der Security-Funktionen erfolgt im Rahmen dieser Untersuchung. Die Erkenntnisse werden genutzt, um das Modellierungskonzept um Mechanismen zu erweitern, die evolutionäre Änderungen und Verbesserungen unterstützen.

## 4 Zusammenfassung

Die Anforderungen an die Security von Softwaresystemen steigen kontinuierlich. Existierende Ansätze ermöglichen es nur, ausgewählte Security-Aspekte im Softwareentwicklungsprozess zu berücksichtigen. Für die Erfüllung der Anforderungen ist ihre frühzeitige Einbeziehung jedoch notwendig. Das Promotionsvorhaben strebt die Integration von Security-Aspekten und der Softwarearchitektur in der Entwurfsphase an. Für die tatsächliche Anwendbarkeit ist ein leichtgewichtiges Konzept wesentlich, das die Modellierung aller benötigten Informationen erlaubt. Dieses Modellierungskonzept wird im Kontext des Softwarelebenszyklus entwickelt: Im ersten Schritt wird die initiale Entwicklung eines Softwaresystems betrachtet, in der die Softwarearchitektur erstmalig entworfen wird. Im zweiten Schritt werden evolutionäre Änderungen der Softwarearchitektur und der Security-Ziele betrachtet. Da Softwaresysteme während ihrer Entwicklung erodieren, erfolgt eine Untersuchung der Erosion von Security-Funktionen und möglicher Gegenmaßnahmen. Die Ergebnisse werden argumentativ und in realen Softwareprojekten evaluiert.

## Literaturverzeichnis

[BDL06] David Basin, Jürgen Doser und Torsten Lodderstedt. Model Driven Security: From UML Models to Access Control Infrastructures. *ACM Transactions on Software Engineering and Methodology*, 15(1):39–91, 2006.

[BSYJ15] Alexander van den Berghe, Riccardo Scandariato, Koen Yskout und Wouter Joosen. Design notations for secure software: A systematic literature review. *Software & Systems Modeling*, 2015.

[FKK+14] Michael Felderer, Basel Katt, Philipp Kalb, Jan Jürjens, Martín Ochoa, Federica Paci, Minh Sang Le Tran, Thein Than Tun, Koen Yskout, Riccardo Scandariato, Frank Piessens, Dries Vanoverberghe, Elizabeta Fourneret, Matthias Gander, Bjørnar Solhaug und Ruth Breu. Evolution of Security Engineering Artifacts: A State of the Art Survey. *International Journal of Secure Software Engineering*, 5(4):48–98, 2014.

[HL06] Michael Howard und Steve Lipner. *The security development lifecycle: SDL, a process for developing demonstrably more secure software*. Microsoft secure software development series. Microsoft Press, Redmond, Washington, 2006.

[Jür02] Jan Jürjens. *Principles for Secure Systems Design*. Dissertation, Oxford University, University of Oxford, 2002.

[JW02] Jan Jürjens und Guido Wimmel. *Security Modelling for Electronic Commerce: The Common Electronic Purse Specifications*, Jgg. 74 of *IFIP International Federation for Information Processing*, Seiten 489–505. Kluwer Academic Publishers, Boston, 2002.

[LBD02] Torsten Lodderstedt, David Basin und Jürgen Doser. *SecureUML: A UML-Based Modeling Language for Model-Driven Security*, Jgg. 2460 of *Lecture Notes in Computer Science*, Seiten 426–441. Springer, Berlin, Heidelberg, 2002.

[MMR10] Tom Mens, Jeff Magee und Bernhard Rumpe. Evolving Software Architecture Descriptions of Critical Systems. *Computer*, 43(5):42–48, 2010.

[NNY10] Armstrong Nhlabatsi, Bashar Nuseibeh und Yijun Yu. Security Requirements Engineering for Evolving Software Systems: A Survey. *IJSSE*, 1(1):54–73, 2010.

[Pre01] Lutz Prechelt. *Kontrollierte Experimente in der Softwaretechnik: Potentiale und Methodik*. Springer, Berlin Heidelberg, 1. Auflage, 2001.

[Ren06] Jie Ren. *A Connector-Centric Approach to Architectural Access Control*. Dissertation, California State University, Long Beach, California, USA, 2006.

[RTDR05] Jie Ren, Richard N. Taylor, Paul Dourish und David F. Redmiles. Towards An Architectural Treatment of Software Security: A Connector-Centric Approach. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005.

[SB12] Lakshitha de Silva und Dharini Balasubramaniam. Controlling software architecture erosion: A survey. *Journal of Systems and Software*, 85(1):132–151, 2012.

[SC09] Sarah Spiekermann und Lorrie Faith Cranor. Engineering Privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2009.

[Sch99] Bruce Schneier. Attack Trees: Modeling Security Threats, 1999.

[SCH04] Adam Sachitano, Richard O. Chapman und John A. Hamilton. Security in Software Architecture: A Case Study. In *from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004*, Seiten 370–376, 2004.

[SO05] Guttorm Sindre und Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.

[SYB+15] Laurens Sion, Koen Yskout, Alexander van den Berghe, Riccardo Scandariato und Wouter Joosen. MASC: Modelling Architectural Security Concerns. In *2015 IEEE/ACM 7th International Workshop on Modeling in Software Engineering (MiSE)*, Seiten 36–41, 2015.

[VKE12] Timo Vepsalainen, Seppo Kuikka und Veli-Pekka Eloranta. Software architecture knowledge management for safety systems. In *2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, Seiten 1–8, 2012.

[WK04] Weihang Wu und T. Kelly. Safety tactics for software architecture design. In *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004*, Seiten 368–375, 2004.