Modellbasierte Softwareentwicklung mit Sicherheitseigenschaften und UMLsec

Patrik Elfert

Fakultät für Informatik
TU Dortmund

5. Februar 2014

Inhalt

Einleitung

Fehlerquellen bei der Entwicklung sicherer Systeme

- Entwickler mit geringen Kenntnissen in Computer-Sicherheit
- Anpassung vorhandener Sicherheitsmechanismen an Anforderungen des Systems

Traditionelle Strategie: *penetrate and patch*

- Versuch in System einzudringen
- Aufdecken von Schwachstellen
- Beheben der Schwachstellen
- **Aber:** Für sicherheitskritische Systeme unbrauchbar

Unified Modeling Language

Prinzip: Modellbasierte Softwareentwicklung

- Informationen über Softwareprojekt in (graphischem) Modell erfassen
- Direkte Verwendung oder Generierung von Artefakten

UML Allgemein

- Spezifikation objektorientierter / komponentenorientierter Software-Systeme
- Verschiedene Diagrammtypen:
  - unterscheiden sich in Anwendungsbereich und Abstraktionsgrad

Unified Modeling Language: Erweiterung

UML-Profile

- Erweiterungsmöglichkeit seit UML 2.0
- Leichtgewichtige Änderungen an Modellen
- Regeln:
  - Semantik des Profils darf Semantik des Referenzmodells nicht widersprechen
  - Austauschbarkeit zwischen UML-Werkzeugen
  - Auflistung bestehender, relevanter Elemente
  - Möglichkeit zur Kombination mehrerer Profile

Unified Modeling Language: Erweiterung

UML-Profile: Struktur

- Stereotype
  - definieren neue Typen von Modellelementen
  - erweitern die Semantik bestehender Typen
  - Notation: *«stereotyp»*
- Attribute *(Tagged Values)*
  - Schlüssel-Wert-Paare
  - weisen Modellelementen Eigenschaften zu
  - Notation: $\{tag = value\}$
- Beschränkungen *(Constraints)*
  - Einschränkungen für Stereotype
  - Informelle Sprache: z.B. Pseudocode

Das UMLsec-Profil

UMLsec-Profil

- Profil kann auf gesamte UML angewendet werden
- Profil setzt keine weiteren Profile voraus

UMLsec-Stereotype (Auszug)

Stereotype	Base Class	Tags	Constraints	Description
Internet	link			Internet connection
encrypted	link			encrypted connection
LAN	link, node			LAN connection
wire	link			wire
fair exchange	subsystem	start, stop, adversary	after start eventually reach stop	enforce fair exchange
secrecy	dependency			assumes secrecy
integrity	dependency			assumes integrity
high	dependency			high sensitivity
secure links	subsystem	adversary	dependency security matched by links	enforces secure communication links

UMLsec-Attribute (Auszug)

Tag	Stereotype	Type	Multip.	Description
start	fair exchange	state	*	start states
stop	fair exchange	state	*	stop states
adversary	fair exchange	adversary model	1	adversary type

Wohldefinierte Regeln

Stereotype *«Internet»* , *«encrypted»* , *«LAN»* und *«wire»*

- Nutzung in Deployment-Diagrammen
- Definieren Art der Kommunikationsverbindung
  - Mindestens ein Stereotyp für jede Verbindung
- Verschiedene Angreifertypen$\rightarrow$ verschiedene Angriffsszenarios
  - Für jeden Angreifer-Typ A
  - und jedes Stereotyp
    $s \in \{$ *«encrypted»* , *«LAN»* , *«Internet»* , *«wire»* $\}$
  - $\rightarrow$ $Threats_A(s) \subseteq \{delete, read, insert\}$
- Dient der Analyse von Gefahren, die von verschiedenen Angreifern ausgehen

Wohldefinierte Regeln

Beispiel

- *default*-Angreifer: Außenstehender Angreifer mit moderaten Fähigkeiten

Stereotype	$Threats_{default}()$
Internet	$\{delete, read, insert\}$
encrypted	$\{delete\}$
LAN	$\emptyset$
wire	$\emptyset$

Wohldefinierte Regeln

Stereotyp *«fair exchange»* in Anwendungsfalldiagrammen

- Art und Weise der Durchführung einer Transaktion muss verhindern, dass eine der beiden Parteien betrügen kann.
- Veränderung eines Subsystems mit *«fair exchange»* nur durch Subsystem mit *«fair exchange»*

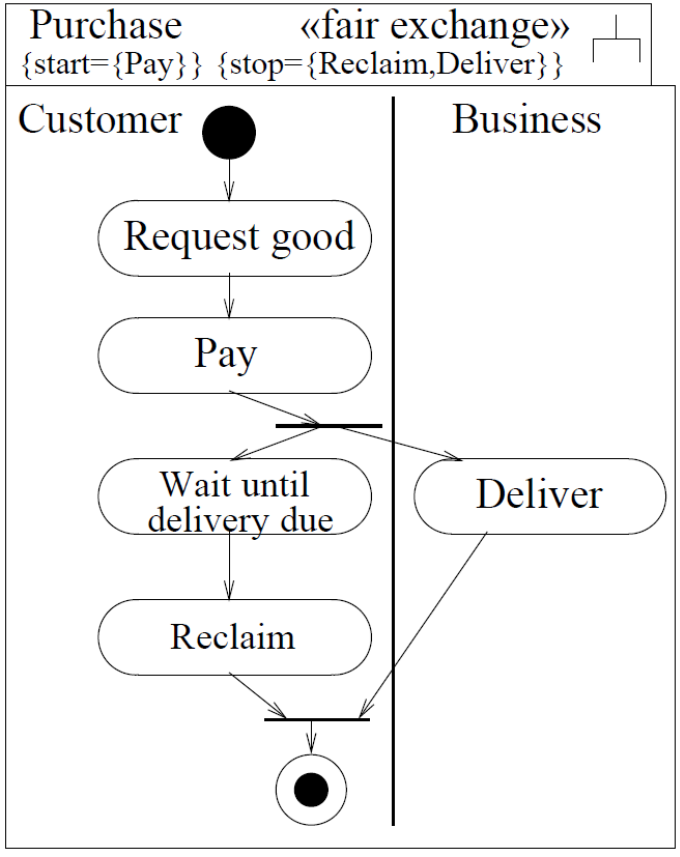
Stereotyp *«fair exchange»* in Aktivitätsdiagrammen

- Zusätzliche Attribute: $\{start\}$, $\{stop\}$, $\{adversary\}$
  - $\{start\}$ und $\{stop\}$: Nehmen Wertepaare mit $\{good, state\}$ an
  - $\{adversary\}$: Spezifiziert den Angreifer-Typ

Wohldefinierte Regeln

Beispiel für *«fair exchange»*

- ...in einem Aktivitätsdiagramm:



Wohldefinierte Regeln

Stereotype *«secrecy»* , *«integrity»* , *«high»*

- In Bedingung für Stereotyp *«secure links»*
- Angegebene Sicherheitsanforderung muss eingehalten werden

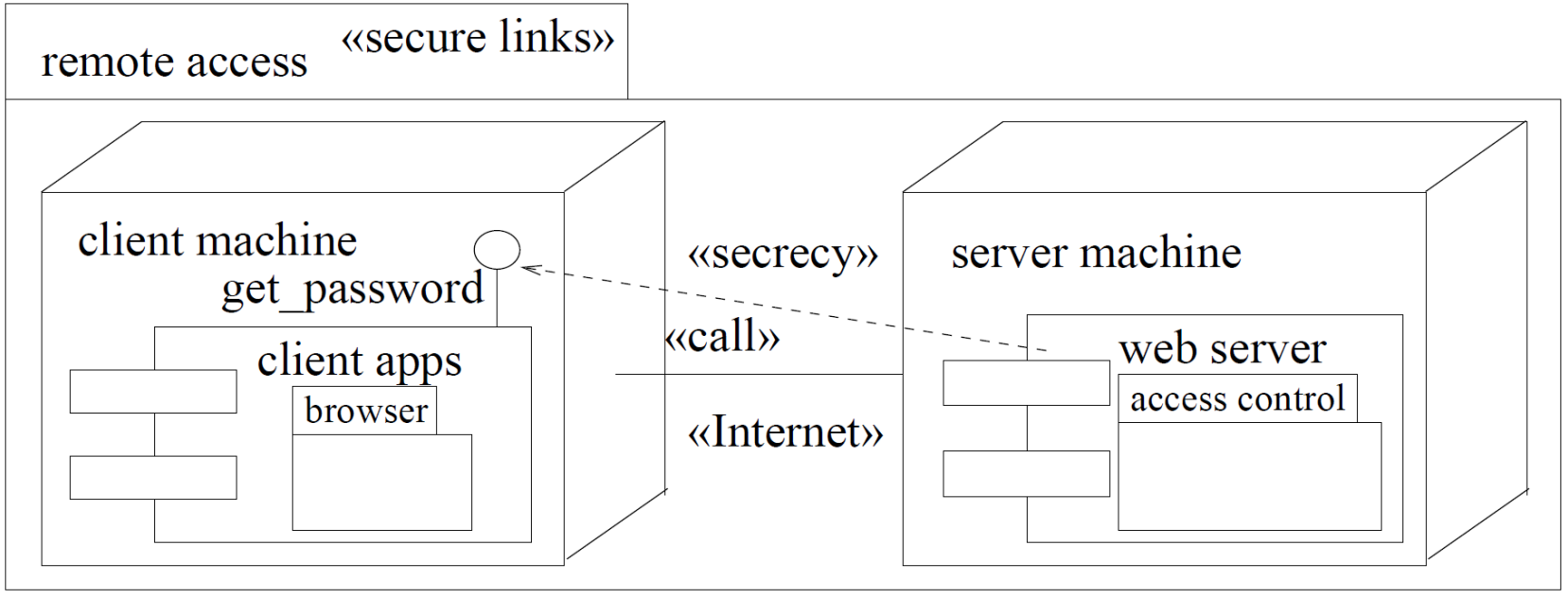
Stereotyp *«secure links»*

- Einhalten der Sicherheitsanforderungen durch physikalische Schicht
- Für jede Abhängigkeit mit Stereotyp $s \in \{$ *«secrecy»*, *«integrity»*, *«high»*$\}$ gilt für die Kommunikationsverbindung l zwischen zwei Objekten:
  - Wenn $s =$ *«high»*, dann gilt: $\mathit{threats}_A^S(l) = \emptyset$
  - Wenn $s =$ *«secrecy»*, dann gilt: $\mathit{read} \notin \mathit{threats}_A^S(l)$
  - Wenn $s =$ *«integrity»*, dann gilt: $\mathit{insert} \notin \mathit{threats}_A^S(l)$

Wohldefinierte Regeln

Beispiel für *«secure links»*

- ...mit Stereotyp *«secrecy»* an Kommunikationsverbindung



Anforderungen an die Erweiterung

Notwendige Anforderungen

Möglichkeit zur...

- ...präzisen Formulierung grundlegender Sicherheitsanforderungen
- ...Berücksichtigung verschiedener Gefährdungsszenarien
- ...Nutzung wichtiger Sicherheitskonzepte
- ...Einbeziehung von Sicherheitsmechanismen
- ...Betrachtung von Sicherheitsprimitiven
- ...Einbeziehung der Sicherheit der zugrundeliegenden physikalischen Schicht
- ...Nutzung des Sicherheits-Managements

Anforderungen an die Erweiterung

Optionale Anforderungen

- Domainspezifische Sicherheitskenntnisse, z.B.:
  - Java
  - Smartcards
  - CORBA
  - ...

Anwendungsmöglichkeiten von UMLsec

Webbasierte Bank-Applikation

- Ausfüllen und Signieren von Bestellformularen
- Sicherheitsanforderungen:
  - Vertrauliche Behandlung persönlicher Daten
  - Durchführung von Bestellungen nicht im Namen anderer
- Lösung:
  - Authentifikationsprotokoll beim Login
  - Verschlüsselte Verbindung

  $\Rightarrow$ Nutzung von UMLsec zur Formulierung der Sicherheitsanforderungen

Anwendungsmöglichkeiten von UMLsec

Smartcard-basiertes biometrisches Authentifizierungssystem

- Problem:
  - Kommunikationsverbindungen zwischen Sensoren, Hostsystem und Smartcard sind angreifbar
- Einsatz von UMLsec
  - zur Spezifizierung des Systems
  - zur automatischen Sicherheitsanalyse

Weitere Anwendungen

- Automatisches Notrufsystem eines Automobilherstellers
- Deutsche Gesundheitskarte
- Elektronische Geldbörse für das Oktoberfest
- ...

Vergleich mit alternativen Ansätzen

Vergleich mit SecureUML

Gemeinsamkeiten:

- Basis: Modellbasierte Softwareentwicklung
- Erweiterung der UML

Unterschiede:

- Grundlage: Erweitertes Modell für rollenbasierte Zugriffskontrolle (RBAC)
- Deutlich strikter gefasst
- Anwendungsgebiete
- Formulierung von Bedingungen (Constraints)
- Ziele der Modellierung

Vergleich mit alternativen Ansätzen

Weitere Ansätze

- Mit UML als Basis:
  - Konzentrieren sich häufig auf bestimmte Anwendungsgebiete
- Nicht modellbasierte Ansätze:
  - *penetrate and patch*-Strategie
  - Framework zur Anpassung von Sicherheitsrichtlinien an die Anforderungen eines Software-Systems

Fazit

Vorteile von UMLsec

- UML sehr bekannt unter Software-Entwicklern
  - → Leichter Einstieg für Software-Entwickler mit wenig Erfahrung
  - → Spart Zeit und Kosten
- Auch für erfahrene Entwickler interessant:
  - Vermeidet nachträgliche, langwierige Fehlersuche
  - Kann bei komplexen Systemen angewendet werden
- Tool-Support

Nachteile von UMLsec

- Lediglich grobe Sicherheitsanforderungen
  - → Ergänzung um spezifischere Ansätze möglich (z.B. SecureUML)