

**Seminar:
Sicherheit und technischer Datenschutz
in Informationssystemen - SoSe 2006**

Architekturen und
Grundbegriffe
sicherer Systeme

Ausarbeitung

Jan Parchmann

Ausarbeitung zum Vortrag vom 12.06.2006

jan.uka@online.de, Tel.: 0721/56 044 02

Betreuung: Jutta Mülle, IPD - Lehrstuhl Prof. Böhm

Universität Karlsruhe

Inhaltsverzeichnis

1	Einleitung	3
2	Security Life Cycle – Sicherheits-Lebenszyklus	3
2.1	Begriffe	4
2.2	Zusammenfassung	7
3	Policies – Richtlinien	8
3.1	Security Policies – Sicherheits-Richtlinien	8
3.2	Confidentiality Policies – Vertraulichkeits-Richtlinie (auch: information flow policy – Informations-Fluss-Richtlinie) ..	10
3.3	Integrity Policies – Integritäts-Richtlinien	12
3.4	Hybrid Policies – Gemischte Richtlinien	15
3.5	Zusammenfassung	19
4	User Security – Benutzersicherheit	20
4.1	Zugriff	20
4.2	Dateien	21
4.3	Prozesse	21
4.4	Elektronische Kommunikation	22
4.5	Zusammenfassung	23
5	Security Life-Cycle im verteilten Umfeld	23
5.1	EISM Tool Suite	23
5.2	EISM Tool Suite als Web-Services	24

1 Einleitung

Die folgende Arbeit stellt eine Einführung in den Bereich der Computersicherheit dar.

Der im folgenden Kapitel beschriebene **Security Life Cycle** ist eine Basis für die Erstellung und Pflege von Computersystemen. Weiterhin werden hier wichtige Terminologien und Definitionen eingeführt.

Das dritte Kapitel beschäftigt sich mit verschiedenen **Richtlinien** (*policies*). Einleitend werden Sicherheitsrichtlinien im allgemeinem eingeführt und behandelt, während in den weiteren Unterabschnitten Vertraulichkeits-Richtlinien (*confidentiality policies*), Integritäts-Richtlinien (*integrity policies*) und gemischte Richtlinien (*hybrid policys*) eingeführt und anhand von wichtigen Vertretern vorgestellt werden.

Das Kapitel „**User Security – Benutzersicherheit**“ greift einige Aspekte auf, die Gefahren für die Sicherheit eines Systems darstellen, wenn Nutzer in einem System interagieren. Diese Gefahren müssen beim Entwurf sicherer Systeme berücksichtigt werden und auch sollten die Nutzer von Computersystemen diesbezüglich aufgeklärt werden, um einen sicheren Betrieb zu gewährleisten.

Das letzte Kapitel dieser Arbeit greift noch einmal den **Security Life Cycle** auf und stellt eine aktuelle wissenschaftliche Arbeit vor. Diese beschreibt, wie in einem **verteilten Umfeld** mit Hilfe von **Web-Services** eine Unterstützung realisiert werden kann, um beim Entwurf sicherer Systeme die Aspekte des Security Life Cycles berücksichtigen zu können.

Kapitelabschließende Zusammenfassungen greifen jeweils noch einmal kurz die wichtigsten Aspekte eines Kapitels auf.

Die in dieser Arbeit verwendeten Informationen sind [Pfleger, 2003] entnommen. Sollten an einigen Stellen zur Verdeutlichung oder Ergänzung weitere Quellen mit einbezogen sein, so sind diese gesondert vermerkt.

2 Security Life Cycle – Sicherheits-Lebenszyklus

Der Security Life Cycle bildet eine elementare Basis bei der Entwicklung von sicheren Systemen und beschreibt das schrittweise Vorgehen bei der Entwicklung von Systemen auf verschiedenen Abstraktionsebenen.

Zunächst werden wichtige Begriffe beschrieben, die am Ende des Kapitels zum life cycle zusammengefügt werden.

2.1 Begriffe

Es gibt drei grundlegende Begriffe auf der Computersicherheit basiert. Diese Begriffe spannen ein Feld auf, welches durch Sicherheitsmaßnahmen es zu füllen gilt.

Die Begriffe confidentiality (Vertraulichkeit), integrity (Integrität) und availability (Verfügbarkeit) variieren in ihrer Bedeutung – abhängig vom Kontext in dem sie genutzt werden. Jedoch lassen sie sich grundlegend wie folgt definieren:

Confidentiality beschreibt die Vertraulichkeit von Ressourcen (wobei in diesem Zusammenhang auch Informationen als Ressourcen zu verstehen sind) und besagt, dass eine Ressource genau dann vertraulich gegenüber einer Gruppe ist, wenn keine Person aus dieser Gruppe auf die Ressource zugreifen kann. Vertraulich sind also die Ressourcen, auf die nur die dafür bestimmten Personen und Objekte zugreifen können.

Integrity ist Integrität und nimmt auf die Vertrauenswürdigkeit einer Ressource Bezug. Sie besagt, dass alle Mitglieder einer Gruppe einer Ressource vertrauen. Das bedeutet, dass die Gruppe sicher ist, dass keine nicht autorisierten Personen oder Objekte Inhalte an der Ressource verändert haben.

Availability (Verfügbarkeit) ist ein wichtiger Aspekt der sich auf zwei Bereiche bezieht. Er beschreibt die Fähigkeit auf eine benötigte Ressource zugreifen zu können und ist sowohl als Zuverlässigkeit während des Betriebes, als auch bei Erreichbarkeitsüberlegungen während des Systementwurfes zu beachten.

Threats – Bedrohungen eines Systems. Eine Bedrohung ist eine potentielle Gefährdung der Sicherheit eines Systems. Sie werden durch Attacken (attacks) von Angreifern (attackers) durchgeführt.

Es lassen sich vier umfassende Klassen bilden mit denen eine gute Klassifikation sämtlicher Bedrohungen möglich ist.

- disclosure – Auskunft: unautorisierter Zugang zu Informationen
- deception – Täuschung: Akzeptanz von falschen Daten
- disruption – Störung: korrekte Operationen unterbrechen oder verhindern
- usurpation – Usurpation: unautorisierte Kontrolle über Teile des Systems

Des Weiteren lassen sich verschiedene Angriffsformen aufzählen, die sich direkt in die Klassen einordnen lassen, zum Teil jedoch auch Überschneidungen der Klassen darstellen.

- snooping – schnüffeln: Kommunikation passiv mithören/-lesen gehört in die Klasse Auskunft.

- modification/alteration – modifizieren/ändern: Änderung von Informationen. Das Ziel ist mittels Täuschung die Möglichkeit zu Störungen und Usurpation zu erlangen)
- masquerading/spoofing – maskieren/reinlegen: betrügerisches Auftreten als jemand anderes als Täuschung oder Usurpation)
- repudiation of origin – leugnen des Ursprungs: betrügerisches Leugnen des Sendens oder Erstellens eines „Dokumentes“. Auch dies gehört in die Klasse Täuschung.
- denial of receipt – leugnen der Annahme: betrügerisches Leugnen der Annahme einer „Ware“ (ebenso Täuschung)
- delay – verzögern: Zeitweilige Hemmung eines Services wird durch Usurpation erreicht. Täuschung dient hierbei als Unterstützung.
- denial of service – versagen eines Services: Langfristiges Blockieren eines Services wird durch Usurpation erreicht.

Policy – Richtlinie. Um den oben genannten Bedrohungen zu begegnen ist es zunächst wichtig zu ermitteln, welche Bedrohungen für das eigene System relevant sind. Hieraus lässt sich dann eine spezielle Sicherheitsrichtlinie ermitteln, die das entstehende System beschreibt.

Bei der Formulierung einer Richtlinie ist es wichtig diese von möglichen Sicherheitsmechanismen zu unterscheiden. Es sollten also folgende Begriffe klar getrennt werden:

Definition 1. Security policy – Sicherheits-Richtlinie: *Eine Sicherheitsrichtlinie ist eine Auflistung der Dinge die erlaubt, beziehungsweise nicht erlaubt sind.*

Sie ist gewöhnlich in „gesprochener Sprache“ verfasst, aber auch formale Ausführungen sind möglich.

Definition 2. Security mechanism – Sicherheits-Mechanismus: *Ein Sicherheitsmechanismus ist eine Methode, ein Werkzeug oder eine Maßnahme, die Sicherheitsrichtlinie zu erzwingen.*

Ein Beispiel soll die Definitionen verdeutlichen:

In einer Sicherheitspolicy könnte es beispielsweise heißen, dass nur der Benutzer selbst sein Zugangspasswort ändern darf (von Systemadministratoren einmal abgesehen). Dies würde bedeuten, dass sichergestellt werden müsste, dass niemand anderes das Passwort ändern kann – auch dann nicht, wenn der Angreifer als eigentlicher Benutzer eingeloggt an dessen Rechner säße.

Ein Mechanismus setzt diese Richtlinie nun um und könnte beispielsweise die Identität der Person vor der Änderung eines Passwortes durch Abfragen des bisherigen Passwortes sicherstellen.

Das Erstellen von Sicherheitsrichtlinien muss gut überlegt werden und Erfahrungen sind hier eine gute Basis. Das wohl größte Problem ist zwei oder mehrere verschiedene Policies miteinander zu vereinen, wie es bei Fusionen, Umstrukturierungen oder anderen Gegebenheiten in der Praxis immer wieder der Fall ist.

Bei der Entwicklung von Sicherheitsmechanismen sind drei grundsätzliche Strategien zu unterscheiden, die sich auch kombiniert einsetzen lassen. Diese Strategien sind:

- prevention – Prevention: Vorbeugende Maßnahmen, die dafür Sorge tragen, dass ein Angriff scheitern wird.
- detection – Erkennung: Erkennungs-Mechanismen erlauben zwar Attacken, aber identifizieren diese und melden sie.
- recovery – Rettung: Diese Strategie besteht aus zwei Teilen. Zum einen wird die Attacke gestoppt, aufgedeckt und der mögliche entstandene Schaden repariert. Darüber hinaus wird jedoch die korrekte Funktionalität des Systems trotz laufender Attacke garantiert. Dies ist vor allem bei sicherheitskritischen Systemen eine notwendige Strategie.

Specification – Spezifikation/Anforderung.

Definition 3. *Ein System genügt (satisfies) einer Spezifikation, wenn die Spezifikation die Funktionalität des Systems korrekt darstellt.*

Die Spezifikation ist eine formale oder weiterhin informale Beschreibung der gewünschten Funktionalität des zu konstruierenden Systems.

Der Großteil der Spezifikation leitet sich aus der Festlegung der relevanten Anforderungen ab. Diese entstehen durch den Zweck der späteren Nutzung des Systems.

Soll das zu konstruierende System beispielsweise nicht aus dem Internet angreifbar sein, so hieße eine mögliche Spezifikation: „Das System kann nicht über das Internet angegriffen werden.“

Design – Entwurf. Beim Entwurf eines Systems werden die Spezifikationen in Komponenten übersetzt, die sich implementieren lassen.

Ein Entwurf *genügt (satisfies)* einer Spezifikation, wenn (unter allen relevanten Umständen) der Entwurf es nicht zulässt, dass das System die Spezifikation verletzt.

Das oben beispielhaft erwähnte System könnte also nun in der Entwurfsphase ohne Netzwerkkarten und Modems und ohne Netzwerktreiber im Kernel entworfen werden. Dieses System würde der geforderten Spezifikation genügen.

Implementation – Implementierung. Eine Implementierung bringt ein System hervor, welches dem Entwurf genügt.

Definition 4. *Ein Programm ist korrekt, wenn sich die Implementierung wie spezifiziert verhält.*

Die Korrektheit muss mathematisch für jede Codezeile überprüft werden. Hierbei wird jede Zeile als mathematische Funktion betrachtet, die eine Eingabe in eine Ausgabe verwandelt.

In diesem Verifikationsprozess treten drei Probleme auf. Zum einen macht die Komplexität von Programmen die mathematische Prüfung sehr schwierig. Hierbei müssen auch schwer erfassbare Rahmenbedingungen mit in die Beweisführung einfließen, da das System ansonsten möglicherweise nicht als sicher verifiziert werden kann. Ein weiteres Problem ist die Annahme, dass als sicher verifizierte Programme auch als sicher angesehen werden. Jedoch können falsche Compilerübersetzungen, Hardwarefehler oder Fehler in anderen Tools zu Problemen führen, die im einzelnen mathematisch nicht erkannt werden. Als letztes kann die Sicherheit eines Programmes auf diese Weise auch nur auf bestimmte Eingaben überprüft werden. Alle anderen Eingaben müssen zur Laufzeit aus Sicherheitsgründen abgelehnt werden.

Operation and Maintenance – Inbetriebnahme/Betrieb und Instandhaltung. Die Betriebsphase birgt meist neue Herausforderungen an ein System. Zuvor schwer zu erfassende oder sich ändernde Rahmenbedingungen, neue Anforderungen an ein System oder wechselnde Benutzerrechte, um nur ein paar Beispiele zu nennen, sind Situationen, die während des Betriebes die Sicherheit eines Systems nicht gefährden dürfen.

Der gesamte „Security Live Cycle“. Die einzelnen bereits aufgeworfenen Punkte bilden Phasen beim Erstellen eines Systems, welche direkt aufeinander aufbauen. Hierbei ist ebenso zu bedenken, dass es auch eine Rückkopplung in die andere Richtung gibt.

Threats ⇔ Policy ⇔ Specification ⇔ Design ⇔ Implementation ⇔ Operation and Maintenance

Selbst ein getestetes Programm führt während des Betriebes immer wieder zu Schwierigkeiten, die Auswirkungen auf die übergeordneten Entwurfsebenen haben können. Auch neue Anforderungen an ein System, die sich während des Betriebes ergeben, erfordern eine Reorganisation der einzelnen Ebenen.

2.2 Zusammenfassung

Computersicherheit hängt von vielen Aspekten eines Computersystemes ab. Die Bedrohungen und die Qualität der Gegenmaßnahmen bilden eine Mischung, die durch eine sorgfältige Analyse zu einer Sicherheitsrichtlinie führen.

Unter Voraussetzung beliebiger Ressourcen ist es einem Angreifer meist möglich, gegebene Sicherheitsmechanismen zu umgehen, ist jedoch die Regenerierung von Daten weniger kostenaufwändig als ein Angriff, so ist dies die erste Wahl und ein System kann als sicher angesehen werden.

3 Policies – Richtlinien

Dieses Kapitel beschreibt zwei verschiedene Arten von Richtlinien und wie sich diese miteinander kombinieren lassen.

Zunächst soll mit einer allgemeinen Beschreibung von **Sicherheitsrichtlinien** begonnen werden, bevor auf die speziellen Einzelheiten von **Vertraulichkeits-Richtlinien (confidentiality policies)** und **Integritäts-Richtlinien (integrity policies)** eingegangen wird. Abschließend befasst sich das Kapitel mit so genannten **hybriden Richtlinien**, welche eine Vereinigung der eben genannten darstellen.

Durch wichtige Vertreter, die mathematisch auf ihre Korrektheit überprüft sind, sollen die einzelnen Klassen veranschaulicht werden.

3.1 Security Policies – Sicherheits-Richtlinien

Zunächst sollen einige Begriffe definiert werden, welche für die Beschreibung von Policies im allgemeinen wesentlich sind.

Definition 5. *Eine Sicherheits-Richtlinie ist eine Beschreibung, welche die Zustände eines Systems in eine Menge von sicheren bzw. autorisierten Zuständen und eine Menge von unsicheren bzw. unautorisierten Zuständen unterteilt.*

Definition 6. *Ein sicheres System ist ein System, welches in einem sicheren Zustand startet und nicht in einen unsicheren Zustand wechseln kann.*

Definition 7. *Ein Verstoß gegen die Sicherheit besteht, wenn ein System in einen unsicheren Zustand wechselt.*

Auch die bereits im zweiten Kapitel beschriebenen grundlegenden Begriffe der Computersicherheit sollen nun formalisiert werden.

Definition 8. Confidentiality – Vertraulichkeit: *Eine Information I ist vertraulich gegenüber einer Gruppe X , falls kein Individuum von X Zugriff auf I hat.*

Definition 9. Integrity – Integrität: *Eine Information oder Ressource I ist integter für eine Gruppe X , falls alle Mitglieder von X I vertrauen.*

Definition 10. Availability – Verfügbarkeit: *Eine Ressource I ist für eine Gruppe X verfügbar, falls alle Mitglieder von X Zugriff auf I haben*

Hierbei ist insbesondere auch zu bedenken, dass die Zugriffszeit einen wesentlichen Faktor für diese Definition spielt. Eine Ressource auf die erst zugegriffen werden kann, wenn die Ressource nicht mehr benötigt wird, gilt als NICHT verfügbar.

Neben der bereits erwähnten Unterscheidung zwischen Sicherheitsrichtlinie und Sicherheitsmechanismus gibt es noch eine weitere Abgrenzung zum Begriff „Sicherheitsmechanismus“ – das Sicherheitsmodell.

Definition 11. *Ein security mechanism – Sicherheits-Mechanismus ist eine Person oder Maßnahme, welche Teile der Sicherheitsrichtlinie umsetzt.*

Definition 12. *Ein security model – Sicherheits-Modell ist ein Modell, welches eine spezielle Richtlinie oder eine Gruppe von Richtlinien repräsentiert. (Eine abstrakte Beschreibung einer Richtlinie, um Analysen zu ermöglichen.)*

Ein Sicherheitsmodell kann also als Teil einer Sicherheitsrichtlinie gesehen werden, so dass sich mehrere Modelle zu einer Richtlinie zusammensetzen.

Arten von Sicherheitsrichtlinien. Es gibt verschiedene Arten von Sicherheitsrichtlinien, die entsprechend ihres Einsatzzweckes entwickelt wurden.

Definition 13. *Eine military/governmental security policy – militärische / staatliche Sicherheitsrichtlinie ist primär entwickelt, um vertrauliche (confidential) Informationen zu schützen.*

Eine Policy dieser Art soll also vertrauliche Informationen vor dem Zugriff nicht autorisierter Personen schützen.

Definition 14. *Eine commercial security policy – kommerzielle Sicherheitsrichtlinie ist primär entwickelt, um Integrität zu wahren.*

Kommerzielle Datenbestände enthalten oft Informationen zu Kunden (wie zum Beispiel Kontodaten), die nicht unerlaubt verändert werden dürfen. Weder darf an Kontoständen noch an Rechnungsbuchungsdaten ohne Autorisierung etwas geändert werden, um Konsistenz sicherzustellen und Manipulationen zu verhindern.

Arten der Zugriffskontrolle. Um auf Ressourcen oder Informationen zugreifen zu können gibt es verschiedene Arten der Zugriffskontrolle. Diese unterscheiden sich im wesentlichen in der Möglichkeit der Rechtevergabe. Auf der einen Seite gibt es personen-basierte Modelle, welchen die automatisierten gegenüber stehen, bei denen keine individuelle Rechtevergabe möglich ist.

DAC/IBAC: Wenn ein individueller Benutzer durch einen Zugriffs-Kontroll-Mechanismus den Zugriff auf ein Objekt erlauben oder verbieten kann, wird von einer *discretionary access control (DAC)* – beliebige/treuhänderische Zugriffs-Kontrolle oder *identity-based access control (IBAC)* – identitäts-basierende Zugriffs-Kontrolle gesprochen. Hierbei ist zu beachten, dass der Benutzer nicht der Ersteller oder Eigentümer des Objektes sein muss.

MAC: Wenn ein Sicherheitsmechanismus den Zugriff auf ein Objekt kontrolliert und ein individueller Benutzer die Zugriffsrechte NICHT ändern kann, wird von einer *mandatory access control (MAC)* – verbindlichen Zugriffs-Kontrolle oder *rule-based access control – rollen-basierte Zugriffs-Kontrolle /RBAC)* gesprochen.

ORCON/ORGCON: Eine *originator controlled access control* (*ORCON* oder *ORGCON*) – *ersteller-kontrollierte Zugriffs-Kontrolle* ermöglicht dem Ersteller eines Objektes (oder der zugehörigen Information) den Zugriff zu regeln. Dies bedeutet, dass der Besitzer eines Objektes keine Rechte besitzt den Zugriff zu ändern, nur der Ersteller.

Richtlinien-Sprachen.

Definition 15. *Eine policy language - Richtlinien-Sprache ist eine Sprache, um Sicherheitsrichtlinien zu repräsentieren.*

Es wird zwischen High-Level und Low-Level-Sprachen unterschieden.

High-Level Sprachen. Eine Richtlinie ist unabhängig von ihren Mechanismen. Sie beschreibt Einschränkungen für Personen und Aktionen innerhalb eines Systems. Eine High-Level Sprache ist eine eindeutige Repräsentation der Richtlinie. Sie ist mathematischer oder programmatischer Natur.

Als Beispiel die Richtlinien-Sprache für Java von Pandey und Hashii [Pandey, Hashii, 1999]:

formal: `deny (s op c.f) when b`

Beispiel:

`deny(|-> file.read) when (file.getfilename() == '/etc/passwd')`

Das Beispiel sagt aus, dass jeglichem „Subjekt“ der Zugriff auf die Methode „read“ der Klasse „file“ verwehrt wird, wenn die folgende Bedingung erfüllt ist, also der Dateiname „/etc/passwd“ lautet.

Low-Level Sprachen. Eine Low-Level Sprache ist hingegen lediglich eine Menge von Eingaben oder Argumenten, welche die Einschränkungen eines Systems setzen oder testen.

Ein Beispiel aus dem UNIX-basierten Fenstersystem X11:

`xhost +groucho -chico` erlaubt Verbindungen vom Host groucho und verbietet sie vom Host chico.

3.2 Confidentiality Policies – Vertraulichkeits-Richtlinie (auch: information flow policy – Informations-Fluss-Richtlinie)

Zunächst zur Verdeutlichung folgende Definition:

Definition 16. *Eine confidentiality policy – Vertraulichkeits-Richtlinie ist eine Sicherheitsrichtlinie, welche sich NUR mit Vertraulichkeit befasst.*

Vertraulich bedeutet, dass auf Ressourcen nur die dafür bestimmten Personen und Objekte zugreifen können. Das primäre Ziel einer Vertraulichkeitsrichtlinie ist also den unautorisierten Zugriff auf Informationen zu verhindern. Unautorisiertes Ändern von Informationen ist ein zweitrangiger Bestandteil.

Das Bell-LaPadula Modell. Eine der ersten „military security policies“ ist das Bell-LaPadula Modell [Bell, LaPadula, 1975], welches die Entwicklung vieler weiterer Modelle und der Computer-Sicherheit stark beeinflusst hat.

Die einfachste Art der Vertraulichkeits-Klassifikation ist die Unterteilung in eine Menge von Sicherheits-Freigaben (*security clearances*), die in eine lineare (optional totale) Ordnung gebracht werden können. Die einzelnen Intervalle repräsentieren die entsprechende Höhe der Vertraulichkeit. Je sensitiver die Daten eingestuft werden können, desto höher ist das entsprechende Sicherheitslevel. Auch die Personen, die innerhalb eines Systems agieren, werden den entsprechenden Intervallen zugeordnet. Bei Personen (subjects) spricht man von Sicherheits-Freigaben (*security clearance*), bei Informationen und Daten (*objects*) von Sicherheits-Klassifikation (*security classification*).

Im Falle einer gemeinsamen Betrachtung der beiden Einteilungen spricht man nur von Klassifikation (*classification*).

Das Ziel des Bell-LaPadula Modells ist es, Personen den Zugriff auf Objekte eines „höheren“ Levels zu verwehren.

Das Modell soll anhand einer beispielhaften Einteilung veranschaulicht werden:

TOP SECRET (TS) für Thomas bzw. persönliche Dateien
 SECRET (S) für Samuel bzw. Emails
 CONFIDENTIAL (C) für Claire bzw. Log-Files
 UNCLASSIFIED (UC) für Ursula bzw. Telefonliste

Personen haben also Freigaben auf einem gewissen Sicherheitslevel und Objekte liegen innerhalb eines gewissen Sicherheitslevels.

Thomas hätte also beispielsweise das Recht auf die persönlichen Daten sämtlicher Leute im System zuzugreifen, während Claire zum Beispiel nur auf Log-Files und die Telefonliste zugreifen darf. Selbstverständlich bräuchte Claire auch Zugriff auf ihre eigenen Mails und persönlichen Dateien. Um dies zu ermöglichen wird das Modell erweitert.

Um die verschiedenen Niveaustufen noch zusätzlich zu verfeinern, wird das Modell um eine Menge von Kategorien (*categories*) erweitert.

Ein Beispiel für die Verwendung von Kategorien:

Die Kategorien EUR und US ergeben folgende Mengen von Kategorien:

{ EUR }, { US } und { EUR, US }.

Eine mögliche Vertraulichkeitsverteilung wäre nun:

William: (SECRET, { EUR, US })

George: (TOP SECRET, { US })

Dokument: (CONFIDENTIAL, { EUR })

William hätte in diesem Beispiel die Möglichkeit auf Dokumente bis zu der Vertraulichkeitsstufe „SECRET“ zuzugreifen. Weiterhin könnten diese Dokumente in den Kategorien „EUR“ oder „US“ liegen. Hiermit hätte William die Berechtigung auf das angegebene Dokument zuzugreifen.

Für George hingegen, der zwar auf der Vertrauensstufe „TOP SECRET“ steht, gilt diese jedoch nur für die Kategorie „US“. Da das Dokument aber in der Kategorie „EUR“ liegt, würde ihm – trotz des höheren eigenen Vertraulichkeitslevels – der Zugriff verwehrt werden.

Um eindeutig sagen zu können, ob ein Subjekt auf ein Objekt zugreifen kann oder nicht, wird die Relation „dominiert“ eingeführt:

Definition 17. *Das Sicherheitslevel (L, C) dominiert (dominates / dom) das Sicherheitslevel (L', C') gdw. $L' \leq L$ und $C' \subseteq C$.*

Um Personen eines niedrigeren Sicherheitslevels das Lesen von Dokumenten zu ermöglichen, die von einer Person eines höheren Levels erstellt wurden, werden die Begriffe *maximum security Level – maximales Sicherheitslevel* und *current security level – aktuelles Sicherheitslevel* eingeführt. Das maximale Sicherheitslevel muss das aktuelle dominieren.

Tranquility – Ruhe/Stille. Das *Prinzip der Stille (principle of tranquility)* besagt, dass einmal instantiierte Personen und Objekte ihr Sicherheitslevel nicht mehr ändern können.

Definition 18. *Das principle of strong tranquility – Prinzip der starken Stille besagt, dass Sicherheitslevel während der gesamten Lebenszeit eines Systems nicht geändert werden dürfen.*

Da dieses sehr strenge Kriterium in der Realität zu zu großer Inflexibilität führt, gibt es eine leicht abgeschwächte Variante:

Definition 19. *Das principle of weak tranquility – Prinzip der schwachen Stille besagt, dass Sicherheitslevel nicht geändert werden dürfen, wenn diese Änderungen gegen die Prinzipien der gegebenen Sicherheitsrichtlinie verstoßen.*

Das Verringern des Sicherheitsstatus eines Objektes führt zum „Herabstufungs-Problem“ (*declassification problem*). Eine gebräuchliche Lösung stellt hier das Ernennen von einer Gruppe von *vertrauensvollen Personen (trusted entities)* da. Diese entfernen sensitive Informationen bevor ein Objekt deklassifiziert wird.

3.3 Integrity Policies – Integritäts-Richtlinien

Auch hier zunächst zur klaren Verdeutlichung folgende Definition:

Definition 20. *Eine integrity policy – Integritäts-Richtlinie ist eine Sicherheitsrichtlinie, welche sich NUR mit Integrität/Unversehrtheit befasst.*

Diese Richtlinien wurden größtenteils für Banken und kommerzielle und industrielle Firmen entwickelt, wo Fehlerfreiheit wichtiger ist als „Verschlossenheit“.

Biba Integrity Modell. Das Biba Integrity Modell [Biba, 1977] ist mathematisch wie das Bell-LaPadula-Modell aufgebaut: Subjekte (S) und Objekte (O) – jeweils mit einem zugewiesenen Integritätslevel (I) (integrity-level). Die Levelhierarchie kann geordnet oder total geordnet sein. Als Kategorien gibt es die Möglichkeiten lesen (r), schreiben (w) und ausführen (x).

Low-Water-Mark Policy. Die Low-Water-Mark Policy regelt nun den Zugriff von Subjekten auf Objekte und wie verhindert werden kann, dass sich Informationen aus Quellen mit niedriger Vertraulichkeit unautorisiert in Quellen hoher Vertraulichkeit mischen.

Wenn eine Person auf ein Objekt zugreift, so ändert die Richtlinie die Integritätsstufe der Person auf das Minimum von Objekt und Person.

1. Wenn s aus S o aus O liest, dann $i'(s) = \min(i(s), i(o))$, wobei $i'(s)$ die Integritätsstufe der Person nach dem Lesen ist.
2. s aus S kann auf o aus O schreiben, gdw. $i(o)$ kleiner gleich $i(s)$.
3. s_1 aus S kann s_2 aus S ausführen, gdw. $i(s_2)$ kleiner gleich $i(s_1)$.

Ring Policy. Die Ring Policy ist eine weitere Variante, die sich von der Low-Water-Mark Policy in folgendem Punkten unterscheidet:

1. Jedes S kann alle O unabhängig des Integritätslevels lesen.

Strict Integrity Policy. Die Strict Integrity Policy, die meist einfach „Bibas Modell“ genannt wird, hält die gleichen Bedingungen beim Lesen, wie das Ausgangsmodell – nur wird die Integritätsstufe des Lesers nicht abgesenkt.

1. s aus S kann o aus O lesen, gdw. $i(s)$ kleiner gleich $i(o)$.

Lipners Integrity Matrix Modell. Ein weiteres Modell, um in einem System die Integrität zu wahren, ist Lipners Integrity Matrix Modell. Es ist eine Mischung aus dem Bell-LaPadula- und dem Biba-Modell, um den Bedürfnissen von kommerziellen Richtlinien entgegenzukommen.

In Lipners Modell gibt es zwei Sicherheitslevel, Audit Manager (AM) und System Low (SL) mit sinkender Integrität.

Des Weiteren gibt es fünf Kategorien:

Development (D),
 Production Code (PC),
 Production Data (PD),
 System Development (SD),
 Software Tools (T)

Folgende Gruppen für Subjekte werden nach Lipner definiert:

User: (SL, { PC, PD })
 Programm-Entwickler: (SL, { D, T })
 System-Programmierer: (SL, { SD, T })

System-Manager und Auditor: (AM, { D, PC, PD, SD, T })

System-Control: (SL, { D, PC, PD, SD, T }) und downgrade Privileg

und folgende Gruppen für Objekte:

Entwickler-Code: (SL, { D, T })

Produktions-Code: (SL, { PC })

Produktions-Daten: (SL, { PC, PD })

Software Tools: (SL, { T })

System-Programme: (SL,)

System-Programme in Änderung: (SL, { SD, T })

System- und Programm-Logs: (AM, { *entsprechende Kategorien* })

Nun werden Personen und Mitarbeiter entsprechend ihrer Aufgaben in der Firma in eine Gruppe eingeordnet und können dann auf Objekte zugreifen und mit ihnen arbeiten, wenn diese von der eigenen Integritätsstufe dominiert werden.

Clark-Wilson Integrity Model. David Clark und David Wilson entwickelten 1987 ein radikal neues Integritäts-Modell [Clark, Wilson, 1987]. Sie verwendeten Transaktionen als Basisoperatoren, da diese Annahme die Realität von kommerziellen Systemen besser widerspiegelt.

Das Modell definiert integere Daten als „*constrained data items*“ (CDIs). Daten, die nicht Bestandteil des Integritätsmodelles sind, werden als „*unconstrained data items*“ (UDIs) bezeichnet.

Das Modell definiert weiterhin zwei Gruppen von Transaktionen. „*Integrity verification procedures*“ (IVPs) testen, ob CDIs den Integritätsbedingungen zum aktuellen Zeitpunkt genügen. Ist dies der Fall, befindet sich das System in einem gültigen Zustand („*valid state*“). „*Transformation procedures*“ (TPs) verändern den Zustand der Daten von einem gültigen in einen anderen gültigen Zustand. Im Falle einer Bank wären beispielsweise die Kontoinformationen CDIs. Ein Test, ob alle Kontostände in Ordnung sind, wäre eine IVP und das Umbuchen von Geld eine TP.

Um eine korrekte Funktion des Systems zu gewährleisten, gelten folgende Regeln:

Zertifizierungsregel 1 (CR1): Wenn eine IVP läuft, so muss diese sicherstellen, dass alle CDIs in einem gültigen Zustand sind.

Zertifizierungsregel 2 (CR2): Eine, zu einer TP gehörenden, Menge von CDIs wird von dieser TP von einem gültigen in einen (anderen) gültigen Zustand transformiert. (Hiermit wird eine Relation definiert, die als „*certified*“ bezeichnet wird.)

Erzwingungsregel 1 (ER1): Das System verwaltet die „*certified*“-Relationen und stellt sicher, dass nur für die CDI zertifizierte TPs diese manipulieren.

Erzwingungsregel 2 (ER2): Das System muss einen Benutzer für jede TP zu einer Menge von CDIs assoziieren. Eine TP darf CDI nur im Namen des Benutzers verändern, wenn die Assoziation dieses erlaubt. (Hieraus entsteht eine Menge von Trippeln ($user, TP, \{Menge\ der\ CDIs\}$). Die Relationen wird *erlaubt* genannt.)

Zertifizierungsregel 3 (CR3): Erlaubte Relationen müssen die gegebenen Bedingungen des „seperation of duty“ erfüllen.

Erzwingungsregel 3 (ER3): Das System muss jeden Nutzer identifizieren, der versucht eine TP auszuführen die mit CDIs arbeitet.

Zertifizierungsregel 4 (CR4): Alle TPs müssen, im Sinne eines Logs, genügend Informationen in CDI schreiben, um ausgeführte Transaktionen rekonstruieren zu können.

Zertifizierungsregel 5 (CR5): Eine TP, die UDI als Eingabe akzeptiert, darf nur gültige Transformationen durchführen oder keine. Die Transformation lehnt die UDI entweder ab oder verwandelt sie in CDI.

Erzwingungsregel 4 (ER4): Lediglich der Zertifizierer einer TP kann die Liste der assoziierten Nutzer mit dieser TP ändern. Kein Zertifizierer einer TP darf jemals die Rechte bekommen, diese TP auszuführen.

3.4 Hybrid Policies – Gemischte Richtlinien

Hybride Richtlinien sind eine Mischung aus Vertraulichkeits- und Integritäts-Richtlinien. Sie sind notwendig, da die Praxis oft eine Kombination aus beiden erfordert.

Die hier vorgestellten Modelle wurden für Systeme des Aktienhandels und für Medizin-Systeme entwickelt. Hierbei gibt es verschiedene erzeuger-kontrollierte und rollen-basierte Ausprägungen.

Chinese Wall Modell. Das Chinese Wall Modell [Brewer, Nash, 1989] wurde für die Börse nach britischem Recht konzipiert und erfüllt die gesetzlichen Anforderungen. Diese besagen unter anderem, dass Investmentberater die internen Informationen, die sie zum Beraten eines Unternehmens zur Einsicht bekommen, nicht nutzen dürfen, um andere Unternehmen zu beraten. Um das Modell an einem Beispiel zu erläutern werden einige Begriffe eingeführt:

Die *Objekte* einer Datenbank sind Informationen, die eine bestimmte Firma betreffen.

Ein *Datensatz* ($CD - company\ dataset$) enthält Informationen zu lediglich einer Firma.

Eine *Klasse von Interessenkonflikten* ($COI - conflict\ of\ interest$) enthält die Datensätze von Firmen, die in Konkurrenz stehen.

Unter der Annahme, dass jedes Objekt zu lediglich einer COI-Klasse gehört, repräsentiert $COI(O)$ die COI-Klasse, die das Objekt O enthält. Des Weiteren

ist $CD(O)$ der Datensatz der Firma, in dem das Objekt O liegt.

Eine beispielhafte Anordnung von Firmen in Konkurrenz und weiteren Firmen wäre:
 „Bank of America“ und „Citibank“ in COI-Klasse „Banken“ und einige Ölfirmen in der „Gasoline Company“ COI-Klasse.

Die *Chinese Wall Sicherheitsbedingung* sagt, dass O von S gelesen werden kann, genau dann wenn eine der folgenden Bedingungen zutrifft:

1. Es existiert ein Objekt O' für das gilt: S hat Zugriff auf O' und $CD(O')=CD(O)$.
2. Für alle Objekte O' gilt: $O' \text{ aus } PR(S) \implies COI(O') \text{ ungleich } COI(O)$, mit $PR(S)$ als Menge aller von S gelesenen Objekte.
3. O ist ein bereinigtes Objekt.

$PR(S)$ wächst somit für ein Subjekt über die Zeit und verhindert, dass auf Informationen zugegriffen werden kann, die in der selben COI-Klasse liegen, jedoch zu einer anderen Firma gehören. Somit wäre zum Beispiel ein Investmentberater, der Daten der „Bank of America“ erhalten hat, nicht in der Lage, auch auf Informationen zuzugreifen, die zur „Citibank“ gehören. Hiermit ist sichergestellt, dass keine konkurrierenden Firmen gemeinsam beraten werden. Hingegen wäre der Berater weiterhin in der Lage eine Firma der „Gasoline Company COI Class“ zu beraten, da dieses nicht im Konflikt zu seinem bisherigen Kunden stünde. Bereinigte Objekte, die unter 3. aufgelistet sind, sind keine vertraulichen Informationen, sondern Informationen, die öffentlich zugänglich sind – zum Beispiel Börsenberichte.

Clinical Information Systems Security Policy. Die *Clinical Information Systems Security Policy* [Anderson, 1996] wurde speziell für Kliniken entwickelt. Der Schwerpunkt liegt auf sensiblen Patientendaten, für die sowohl Vertraulichkeit als auch Integrität sichergestellt werden muss – dies jedoch auf andere Weise wie bei börsennotierten Firmen. Nicht der Interessenkonflikt, sondern die Tatsache, dass Patientendaten auf der einen Seite vertraulich behandelt werden müssen, um Patientenrechte zu wahren, und zum anderen sichergestellt sein muss, dass die Daten nicht versehentlich geändert werden, um den Patienten zu schützen, steht hier im Mittelpunkt.

Das Modell enthält folgende Personen:

Ein *Patient* als Subjekt der Patientendaten (oder eine vertrauensvolle Person, die Daten bereitstellt).

Persönliche Gesundheitsinformation ist Information über die Gesundheit oder die gewählten Behandlungsverfahren des Patienten.

Ein *Arzt* als Zuständiger, der Zugriff auf die persönlichen Gesundheitsinformationen hat, während er seine Arbeit ausübt.

Vier Zugriffsprinzipien und fünf weitere Regeln stellen den gesicherten Zugriff und die Bearbeitung der persönlichen Patienteninformationen sicher.

Erstes Zugriffsprinzip: Jede persönliche Gesundheitsinformation hat eine *Zugriffskontrollliste*, welche die Individuen und Gruppen nennt, die lesend und anhängend Zugriff auf die Daten haben. Das System muss allen anderen den Zugriff verwehren.

Zweites Zugriffsprinzip: Ein Arzt auf der Zugriffskontrollliste wird *zuständiger Arzt* genannt. Er hat das Recht weitere Ärzte der Liste hinzuzufügen.

Drittes Zugriffsprinzip: Der zuständige Arzt muss den Patienten jedes mal wenn die persönliche Akte geöffnet wird darüber informieren, welche Namen auf der Zugriffsliste stehen. Außer in gesetzlich festgelegten Ausnahmesituationen oder in Notfällen, braucht der zuständige Arzt KEINE Genehmigung des Patienten einzuholen.

Viertes Zugriffsprinzip: Der Name des Arztes, das Datum und die Zeit eines Zugriffes müssen aufgezeichnet werden.

Erstellungsregel: Angenommen ein Arzt erstellt eine Akte mit sich und dem Patienten auf der Zugriffsliste. Wenn die Akte aufgrund einer Überweisung erstellt wurde, so ist der überweisende Arzt mit aufzunehmen.

Löschregel: Informationen können erst nach einer angemessenen Zeit wieder gelöscht werden.

Einschränkungsregel: Informationen aus einer Akte dürfen nur dann in eine andere Akte übernommen werden, wenn die Zugriffsliste der zweiten Akte eine Untermenge der ersten ist.

Aggregationsregel: Aggregation von Patienteninformationen muss verhindert werden. Patienten müssen benachrichtigt werden, wenn jemand auf die Zugriffsliste hinzugenommen werden soll, der Zugriff auf eine große Menge von Patienteninformationen hat.

Durchsetzungsregel: Jedes Computersystem, das mit medizinischen Daten agiert, muss ein Teilsystem haben, welches die geforderten Prinzipien und Regeln einhält. Die Effektivität des Systems muss durch unabhängige Auditoren evaluiert werden.

Originator Controlled Access Control. Die Zugriffskontrollen MAC und DAC sind nicht geeignet, wenn nur der Ersteller eines Dokumentes die Kontrolle über die Zugriffsrechte haben soll. „Originator Controlled Access Control“ (ORGCON bzw. ORCON) fordert, dass ein Subjekt nur mit Einverständnis des Erstellers einem anderen Subjekt Rechte vergeben darf. In der Praxis werden Objekte in der Regel im Namen einer Firma als ORCON markiert.

Ein Mitarbeiter markiert eine Datei als ORCON im Namen der Firma X. Die Firma erlaubt Mitarbeitern einer weiteren Firma, Y, diese Datei zu lesen. Hierbei gelten folgende Bedingungen:

- a. Die Datei darf nicht von Mitarbeitern einer weiteren Firma gelesen werden, ohne die Erlaubnis von Firma X zu haben.
- b. Jede Kopie, die von der Datei erstellt wird, muss die selben Restriktionen wie

die Datei selbst besitzen.

ORCON wird durch ein dezentrales System der Zugriffskontrolle realisiert, bei dem nur die Ersteller von Dateien Einfluss auf die Zugriffsrechte haben. Eine Lösung, diese Anforderungen zu realisieren besteht aus der Möglichkeit, einige Funktionen aus dem MAC und dem DAC Modell zu kombinieren. Hierbei ergeben sich folgende Regeln:

1. Der Besitzer eines Objektes kann die Zugriffsrechte NICHT ändern.
2. Wenn ein Objekt kopiert wird, so werden auch die Zugriffsrechte übernommen.
3. Der Ersteller eines Objektes kann die Zugriffsrechte auf Subjekt- oder Objektebene ändern.

Regel 1 und 2 – aus dem MAC Modell – besagen, dass das System den Zugriff auf Dateien regelt und dass kein Nutzer die Zugriffsrechte ändern kann.

Regel 3 – aus dem DAC Modell – gibt dem Ersteller die Möglichkeit über die Zugriffsrechte zu entscheiden.

Somit ist dieses Modell weder dem MAC noch dem DAC Modell zuzuordnen.

Role-Based Access Control. In der Praxis liegt die Notwendigkeit auf Daten zugreifen zu müssen oft in der Aufgabe beziehungsweise Rolle eines Mitarbeiters oder Prozesses begründet. Diese Tatsache legt ein Modell nahe, welches die Zugriffsrechte nach Arbeitsfunktionen und Rollen im Betrieb regelt.

Definition 21. *Eine Rolle ist eine Zusammenstellung von Funktionen und Aufgaben eines Bereiches im Betrieb. Jede Rolle ist autorisiert eine oder mehrere Transaktionen auszuführen. $\text{trans}(r)$ sei die Menge der autorisierten Transaktionen.*

Definition 22. *Die aktive Rolle eines Subjektes ist die Rolle, die ein Subjekt gerade ausübt. $\text{actr}(s)$*

Definition 23. *Die autorisierten Rollen eines Subjektes sind die Menge der Rollen für die das Subjekt autorisiert ist. $\text{authr}(s)$*

Definition 24. *Das Prädikat $\text{canexec}(s, t)$ ist wahr, genau dann wenn das Subjekt die Transaktion zu dem gegebenen Zeitpunkt ausführen kann.*

Drei Axiome regeln die Ausführbarkeit von Transaktionen.

Axiom 1. Sei S eine Menge von Subjekten und T eine Menge von Transaktionen. Die „rule of role assignment“ lautet: Für alle s aus S und für alle t aus T gilt: $\text{canexec}(s, t) \rightarrow \text{actr}(s)$ ungleich leere Menge.

Diese Regel besagt lediglich, dass ein Subjekt, welches irgendeine Transaktion ausführen kann, eine aktive Rolle hat.

Axiom 2. Sei S eine Menge von Subjekten. Die „*rule of role authorization*“ lautet: Für alle s aus S gilt: $\text{actr}(s)$ ist Untermenge von $\text{authr}(s)$. Diese Regel besagt, dass ein Subjekt autorisiert sein muss, seine aktive Rolle auszuführen.

Axiom 3. Sei S eine Menge von Subjekten und T eine Menge von Transaktionen. Die „*rule of transaction authorisation*“ lautet: Für alle s aus S und für alle t aus T gilt: $\text{canexec}(s,t) \rightarrow t \text{ aus } \text{trans}(\text{actr}(s))$. Diese Regel besagt, dass ein Subjekt eine Transaktion nicht ausführen kann, wenn ihre aktive Rolle hierzu nicht autorisiert ist.

Die Axiome begrenzen den Zugriff auf unautorisierte Transaktionen, stellen jedoch nicht sicher, dass eine erlaubte Transaktion wirklich ausgeführt werden kann. „Role-Based Access Control“ (RBAC) ist eine Form des MAC.

3.5 Zusammenfassung

Sicherheitsrichtlinien beschreiben die „Sicherheit“ eines Systems, wobei sie diese meist nicht eindeutig definieren. Formale, mathematische Modelle von Richtlinien ermöglichen es Analysten, diese konkret zu definieren, doch ist der Durchschnittsnutzer eines Systems meist nicht in der Lage, diese mathematischen Modelle korrekt zu interpretieren.

„Vertrauen“ ist die Basis für alle Richtlinien und Mechanismen. Während Richtlinien Annahmen über das Verhalten von System, Software, Hardware und Benutzer treffen, übernehmen Mechanismen die selben Annahmen auf eine niedrigere Ebene.

Das Bell-LaPadula Modell, welches als erstes mathematisches Modell die Attribute eines realen Systems in seinen Regeln aufgreift, stellt die Basis für viele Standards dar. Im praktischen Kontext sind weitere Vertraulichkeitsmodelle entstanden.

Integritätsmodelle variieren stark in ihrer Vielfalt und Beliebtheit. Gerade die Bedeutung dieses Bereiches wächst stark, da immer mehr kommerzielle Firmen Modelle und Richtlinien entwickeln, um ihre Daten zu schützen.

Typischerweise kombinieren Sicherheitsrichtlinien die Funktionalitäten sowohl von Vertraulichkeits- als auch von Integritätsrichtlinien. Während das „Chinese Wall model“ die Bedingungen eines speziellen Bereiches (Börsenhandel) unter gegebenen Aspekten (das britische Recht) aufgreift, tut dies das „Clinical Information Systems model“ für persönliche medizinische Daten. Beide Modelle basieren auf geschäftliche beziehungsweise klinische Wirklichkeit.

ORCON und RBAC sind zwei unterschiedliche Ansätze auf welche Art und Weise Nutzer und Prozesse auf Daten zugreifen. Sie beschäftigen sich weniger

mit der Frage, wie darauf zugegriffen werden *sollte*.

ORCON erlaubt dem Autor, den Zugriff zu kontrollieren; RBAC erlaubt durch spezielle Funktionen den Zugriff für Individuen auf Gruppenbasis.

4 User Security – Benutzersicherheit

In der gesamten Arbeit wurden die Benutzer eines Systemes bisher weder erwähnt noch berücksichtigt. Da sie jedoch großen Einfluss auf die Sicherheit eines Systemes haben, müssen sie stets berücksichtigt werden. Die meisten Benutzer haben eine informale Vorstellung einer persönlichen Sicherheitsrichtlinie, aber selten wird diese festgehalten oder gar analysiert.

Im folgenden wird eine Nutzer-Richtlinie vorgestellt und die Möglichkeiten, mit denen diese Richtlinie umsetzen werden kann, aufgezeigt.

U1. Nur Nutzer haben Zugang zu den eigenen Accounts.

U2. Kein anderer Nutzer kann ohne Berechtigung des Besitzers Dateien lesen oder ändern.

U3. Nutzer sollten Integrität, Vertraulichkeit und Verfügbarkeit ihrer Dateien sichern.

U4. Nutzer sollen sich aller eingegebenen Kommandos bewusst sein – auch derer, die in ihrem Auftrag eingegeben werden.

4.1 Zugriff

U1 erfordert einen geschützten Zugriff auf Accounts. Gerade hier entsteht eine ideale Schnittstelle für Angreifer.

Folgende Mechanismen sollten beachtet werden.

Passwörter. Sichere Passwörter sollten per Zufallsgenerator kreiert werden. Jedoch sind diese Passwörter nicht leicht zu merken. Je nach Umgebung des Systems und Art des Niederschreibens des Passwortes ergibt sich die tatsächliche Gefährdung.

Steht zum Beispiel ein Teilsystem, welches wichtige Konfigurationsmaßnahmen des Gesamtsystems ermöglicht, in einem eigenen verschlossenen Raum, und haben nur berechtigte Personen einen Zugang zu diesem Raum, so könnte das benötigte Passwort ohne Gefahr auf einem Zettel in diesem Raum stehen. U1 wäre erfüllt.

Eine Möglichkeit sich viele zufallsgenerierte Passwörter zu merken besteht darin, eine transformierte Form niederzuschreiben. Beispielsweise ließe sich die „Größe“ des dritten Buchstabens ändern und ein weiteres beliebiges Zeichen hinten anhängen.

Somit würde C04ceJxX5 das eigentliche Passwort C04cEJxX repräsentieren. Die Zeit zwischen Verlust der Liste und dem Bemerkenden, würde einem Angreifer nicht ausreichen, um die Passwörter zu erraten.

Besteht die Möglichkeit – wie bei Workstations meist der Fall – selbst Passwörter zu vergeben, so empfiehlt sich vor der Akzeptanz eines neuen Passwortes eine

Prüfung durchzuführen. Zu leicht zu erratende Passwörter werden verweigert. Eine gute Möglichkeit selbst Passwörter zu generieren besteht darin, zum Beispiel einen Vers oder Spruch zu nehmen und nur bestimmte Buchstaben daraus zu verwenden. Beispielsweise jeden ersten und jeden zweiten Buchstaben von „Ich kam, sah und siegte.“, verbunden mit einem „&“ ergäbe „Iksus&caani“.

Login. Bei der Loginprozedur muss sichergestellt werden, dass keine Trojaner eingegebene Passwörter abfangen.

Auch besteht die Gefahr, dass eingetippte Passwörter mitgelesen werden können. Eine sinnvolle Methode ist die Angabe des letzten Logins und der Anzahl der Fehlversuche seit dem letzten erfolgreichen Login. Somit kann ein Nutzer einen möglichen Einbruch entdecken oder wiederholte Attacken melden.

System verlassen. Beim Verlassen eines Systems ist es wichtig, auch den Zugang zum System zu trennen, damit keine unautorisierten Personen den offenen Zugang nutzen können, um sich selbst Zugang zum System zu verschaffen.

Bei Screen-Lock Methoden, die beim Verlassen des Arbeitsplatzes angewendet werden, muss berücksichtigt werden, dass diese teilweise durch ein Masterpasswort wieder zu öffnen sind.

4.2 Dateien

Um U2 zu gewährleisten müssen Nutzer ihre Informationen vertraulich und integer halten. Da zur Speicherung der Informationen meist Dateien verwendet werden, werden in der Regel die Datei-Schutzmechanismen des Systems genutzt, woraus sich einige Probleme ergeben.

Bei der *Erstellung von Dateien* werden meist entweder die Zugriffsrechte des aktuellen Ordners geerbt oder ein universelles Template verwendet. Dies kann zu ungewollten Freigaben führen.

Für *Gruppen* freigegebene Dateien können von jedem Mitglied der Gruppe gelesen oder verändert werden. Kommen neue Nutzer in die Gruppe hinzu, haben diese auch den für die Gruppe definierten Zugriff. Dies ist insbesondere ein Problem, wenn die Gruppen als Rollen definiert werden. Bekommen neue Nutzer eine neue oder erweiterte Rolle, so sind diese automatisch Mitglieder der entsprechenden Gruppe mit den entsprechenden Zugriffsrechten.

Beim *Löschen von Dateien* werden möglicherweise nur die Referenzen auf die Datei aus dem Verzeichnis der Festplatte gelöscht, nicht aber die Daten selbst, die dann wieder hergestellt werden könnten.

4.3 Prozesse

Prozesse bearbeiten Objekte – einschließlich Dateien. Um U3 zu gewähren, muss der Nutzer wissen, auf welche Weise Dateien von Prozessen bearbeitet werden. Hier sollen einige wichtige Aspekte genannt werden.

Kopieren und Verschieben von Dateien. Auf einigen UNIX-Systemen ist beim Kopieren von Dateien zu beachten, dass die Attribute der Ursprungsdatei nur dann für die Zieldatei übernommen werden, wenn die Zieldatei noch nicht existiert. Wird beispielsweise die Datei `xyzyz` mit dem Befehl `cp xyzyz plugh` kopiert, während die Datei `plugh` bereits existiert, dann enthält `plugh` die Daten von `xyzyz`, aber weiterhin die Zugriffsrechte von `plugh`. Eine ähnliche Situation kann es auch beim Verschieben-Kommando `mv` geben.

Versehentliches Löschen von Dateien. U3 fordert, dass der Nutzer die Verfügbarkeit seiner Daten sicherstellen muss. Hierfür muss der Nutzer gelegentlich vor sich selbst geschützt werden. Beispielsweise würde das versehentlich falsch eingegebene UNIX-Kommando `rm * .o` nicht alle Dateien auf `.o` endend löschen (wie eigentlich gewünscht, sondern alle Dateien des aktuellen Verzeichnisses und die Datei `„.o“`). Mit zusätzlichen Abfragemechanismen können Probleme dieser Art umgangen werden.

Kryptographie und Schlüssel. Die Basis von Kryptographie ist Vertrauen. Eine entschlüsselte Nachricht kann jedoch möglicherweise von jedem gelesen werden, der Zugriff auf den Speicher hat. Auch wenn private Schlüssel in die Hände unautorisierter Besitzer gelangen, haben diese die Möglichkeit bei Angriffen, die Täuschung zur Absicht haben, ein hohes Vertrauen für ihre gefälschten Daten zu erlangen. Hier ist also Aufmerksamkeit geboten.

4.4 Elektronische Kommunikation

Elektronische Kommunikation in Form von Email ist eine riskante Schnittstelle, die mehrere Gefahren birgt. Emails gehen durch Firewalls hindurch (sofern dies nicht unterbunden wird). Auch wenn sie auf schadhafte Software überprüft werden, so kann man nicht sicher sein, alles zu finden. Drei Gesichtspunkte werden im folgenden näher betrachtet.

Automatisierte Abläufe. Viele Benutzer eines Systems nutzen die Möglichkeit, Emails automatisiert zu verarbeiten. Ein Programm entscheidet, ob die Mail gespeichert werden soll oder ob sie als eine Sequenz von Kommandos für ein Programm dienen soll oder vielleicht sogar Programmteile selbst enthält. Hierbei sind nicht alle Seiteneffekte im Vorfeld abzusehen, denn gerade Emails können von unbekanntem Quellen kommen.

Zertifikate. Das größte Problem bei Zertifikaten ist die Gefahr, ein Zertifikat nicht überprüfen zu können. Ein Zertifikat suggeriert Vertrauenswürdigkeit, doch ist die Zertifizierungsstelle nicht erreichbar, so ist nicht festzustellen, ob die Zertifikate möglicherweise zurückgezogen wurden.

Erstellt jemand beispielsweise Zertifikate, die es ihm ermöglichen Software im Namen der Microsoft Corporation zu signieren, dann werden diese Zertifikate

sofort zurückgewiesen und auf entsprechenden Listen veröffentlicht. Wenn jedoch ein Host diese Zurückweisungen noch nicht empfangen hat, dann werden hier diese Zertifikate für gültig erklärt und die schadhafte Software kann in der Email oder zum Beispiel auf Webseiten ausgeführt werden.

Auto-Speichern und versehentliches Senden. Die Funktion des automatischen Speicherns speichert unbemerkt einen Stand ab, der möglicherweise in dieser Form nicht an die Empfänger gehen soll. Was jedoch einmal gespeichert ist, ist möglicherweise zu rekonstruieren, auch wenn vertrauliche Informationen später wieder gelöscht wurden.

Weiterhin besteht die Gefahr, versehentlich Informationen zu senden, derer man sich nicht bewusst ist. Beispielsweise das Versenden einer Tabellenkalkulations-Datei mit Informationen zu einem aktuellen Projekt. Möglicherweise befindet sich in der Datei ein Datenblatt mit vertraulichen Informationen, das übersehen worden sein kann.

4.5 Zusammenfassung

Dieses Kapitel hat einige Aspekte aufgegriffen, wie Nutzer ihre Daten und Programme schützen können. Die gegebene Sicherheitsrichtlinie auf der einen und die persönlichen Anforderungen auf der anderen Seite kombinieren sich zu einer persönlichen Sicherheitsrichtlinie.

Gut gewählte Passwörter und elementare Dateizugriffs-Mechanismen waren zwei der besprochenen Themen.

5 Security Life-Cycle im verteilten Umfeld

Als interessanter, aktueller Forschungspunkt, soll auf eine Arbeit zum Security Life-Cycle im verteilten Umfeld [Sengupta, 2005] eingegangen werden. Sie zeigt, dass die einzelnen hier vorgestellten Modelle trotz ihres „Alters“ noch immer relevante Themengebiete im Bereich der Computersicherheit sind und dass sie als Basis dienen, um darauf aufbauend den Sicherheitsproblemen der heutigen vernetzten Computerwelt zu begegnen.

5.1 EISM Tool Suite

Die „*Enterprise Information Security Management (EISM) Tool Suite*“ soll verschiedene Stufen und Bereiche der Sicherheitsentwicklung und -analyse, wie zum Beispiel Anforderungen und Gefahrenanalyse, Richtlinienentwicklung und Testen des Security Life Cycles, vereinen.

Die hier vorgestellte Forschungsarbeit ist es nun, diese Komponenten in Form von Web-Services zur Verfügung zu stellen.

5.2 EISM Tool Suite als Web-Services

Die steigende Komplexität von Systemen macht es unmöglich, diese ohne formale Modelle zu managen.

Das Hauptanliegen des Projektes ist es, für diese Modelle ein vereinheitlichtes Modell von „Enterprise Information Security“ zu entwickeln. Als weiteres Anliegen besteht der Ansatz, für dieses vereinheitlichte Modell, ein Bündel von Web-Services zu entwickeln, welche in den unterschiedlichen Phasen des Security Life Cycles unterstützend zur Verfügung stehen.

Das Modell besteht aus folgenden sechs Komponenten:

Enterprise Security Requirement Analysis Component zur Eingabe der Spezifikation. Drei Ausgabeberichte ermöglichen die Einschätzung der Qualität der Spezifikation.

Risk Analysis and Mitigation Component zur Gefahrenanalyse in drei Phasen.

Policy Development Component zur automatischen Erstellung von individuellen Anleitungen für Sicherheitsrichtlinien (basierend auf der Spezifikation) auf drei Ebenen – unternehmensweit, aufgabenbezogen und systembezogen.

Security Architecture and Infrastructure Advisory Generation Component zur Ermittlung der benötigten Technologien und Tools – mit zusätzlicher Identifikation von verfügbaren Produkten.

Security Testing Component ermittelt auf Basis der Spezifikation und der Gefahrenanalyse die möglichen Schwachstellen des Systems und wie diesen begegnet werden kann. Zusätzlich wird eine Liste von Testfällen erstellt, mit denen das System überprüft werden kann.

Training Component bietet webbasiertes Onlinetraining für verschiedene Mitarbeitergruppen an.

Die EISM Tool Suite wird bereits in verschiedenen Unternehmen als Web-Services eingesetzt.

Literatur

- [Pfleger, 2003] Pfleeger, Charles P., Pfleeger, Shari Lawrence. Security in computing, Upper Saddle River, NJ : Prentice Hall, 2003.
- [Pandey, Hashii, 1999] Raju Pandey and Brant Hashii. Providing Fine-Grained Access Control for Java Programs. In The 13th Conference on Object-Oriented Programming, ECOOP'99. Springer-Verlag, June 1999.
- [Bell, LaPadula, 1975] D. Bell and L. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, Mitre Corporation, Belford, MA, 1975.
- [Biba, 1977] Biba, K.J.: Integrity considerations for secure computer systems. In: MITRE Technical Report TR-3153. (1977)
- [Clark, Wilson, 1987] D. Clark and D. Wilson. A Comparison of Commercial and Military Computer Security Policies. In Proc. of the IEEE Symposium on Security and Privacy, April 1987.
- [Brewer, Nash, 1989] Brewer, D., Nash, M.J. The Chinese wall security policy, in 1989 IEEE Symposium on Security and Privacy. IEEE Computer Society, pp. 206-214, May 1989 Buczak, A.L., Zimmerman, J., Kurapati, K.
- [Anderson, 1996] Anderson, Ross J., A Security Policy Model for Clinical Information Systems, 1996
- [Sengupta, 2005] A. Sengupta et al, „A Web-Enabled Enterprise Security Management Framework“, in S. Jajodia and C. Mazumdar (Eds.): ICISS 2005, LNCS 3803, S. 328–331, 2005.