Betriebssystem-Sicherheitspolitiken formal analysieren

Martin Rabe

TU Ilmenau
martin.rabe@tu-ilmenau.de

Zusammenfassung

Je größer unsere Abhängigkeit von vernetzten IT-Systemen wird, desto wichtiger ist es diese Systeme vor unerlaubten Zugriffen zu schützen. Eine Möglichkeit die Sicherheit eines IT-Systems zu erhöhen ist die Umsetzung eines politikkontrollierten Systems wie SELinux. Um überprüfen zu können, ob die für SELinux erstellte Sicherheitspolitik auch die gewünschten Sicherheitseigenschaften erfüllt, ist eine formale Analyse von Vorteil. Diese ermöglicht es die Systeme zur Laufzeit auf Durchsetzung gewünschter Sicherheitseigenschaften zu überprüfen. Tools zur Durchführung einer solchen formale Analyse können die sich auf die Zugriffssteuerung beziehende wichtige Semantik von SELinux aber nicht direkt analysieren. Sie benötigen dafür ein Modell in dem die in SELinux verteilten Informationen zusammengetragen werden können. Um die auf Modellebene gewonnenen Erkenntnisse der Analyse auf Systemebene auswerten zu können ist es weiterhin von Nöten das Modell auf die zugrundeliegende SELinux-Instanz zurück abzubilden.
Um dieses Problem zu lösen wird in dieser Arbeit ein Verfahren für die Hintransformation von einer SELinux-Instanz zu einem Modell sowie dessen Rücktransformation entwickelt. Durch die Hintransformation wird eine modellbasierte Analyse einer SELinux-Instanz ermöglicht. Nach einer formalen Analyse des Modells wird durch die Rücktransformation ermöglicht, die Ergebnisse im Kontext der SELinux-Instanz auszuwerten.

1 Einleitung

Ob im Finanzwesen, bei der Strom- und Wasserversorgung oder im Gesundheitswesen – die Abhängigkeit von IT-Systemen hat in den letzten Jahren stark zugenommen. Je größer diese Abhängigkeit wird, desto wichtiger ist es, dass die IT-Systeme vor unberechtigten Zugriffen oder Manipulationen geschützt sind. Die geforderten Sicherheitseigenschaften an die Betriebssysteme, auf die die IT-Systeme aufbauen, werden daher immer stärker.

Eine Möglichkeit die Umsetzung dieser Eigenschaften zu gewährleisten ist die Nutzung eines politikkontrollierten Betriebssystems, da sich die Sicherheitseigenschaften anhand einer solchen Politik verifizieren lassen. Eine Politik, im Folgenden Sicherheitspolitik genannt, ist eine Menge von Regeln, die die Erfüllung der Sicherheitsanforderungen an ein IT-System verfolgt. Um verifizieren zu können, ob die für das Betriebssystem erstellte Sicherheitspolitik auch die gewünschten Sicherheitseigenschaften erfüllt, ist ein formales Modell nötig. Ein solches formales Modell enthält die für die Zugriffssteuerung relevante Semantik von Betriebssystemen und kann von Tools, wie zum Beispiel WorSE [AKP14], analysiert werden. Eines der momentan weit verbreitetsten politikkontrollierten Betriebssysteme ist SELinux [sel13].

Für eine formale Analyse müssen die sicherheitsrelevanten Komponenten eines SELinux-

Systems, wie zum Beispiel Objekte (Dateien, Sockets, etc.) und Subjekte (Prozesse) mit ihren sicherheitsrelevanten Attributen (Sicherheits-Kontexten) berücksichtigt werden. Darüber hinaus werden die Regeln der Sicherheitspolitik benötigt, welche die zulässigen Veränderungen dieser Systemkomponenten beschreiben. Diese sind zum Beispiel der Wechsel des Sicherheits-Kontextes eines Subjekts oder eines Objekts. Da diese Informationen jedoch an verschiedenen Stellen in der Implementierung und der Konfiguration des Betriebssystems (im Folgenden SELinux-Instanz) verteilt sind, eignet sich diese Repräsentation der sicherheitskritischen Informationen nicht für die modellbasierte Analyse. Deshalb wird eine Transformation dieser Informationen in ein formal analysierbares Modell benötigt, da in diesem die Informationen zusammengetragen werden können. Wenn ein geeignetes Modell gewählt wurde, kann dieses auf Sicherheitseigenschaften, wie zum Beispiel die Ausbreitung von Rechten, analysiert werden.

Nach der Analyse müssen die dadurch gewonnenen Erkenntnisse auf die zugrundeliegende SELinux-Instanz und die dazugehörige Sicherheitspolitik abgebildet werden, um somit die auf Modellebene erhaltenen Ergebnisse auf SELinux-Systemebene übertragen zu können. D.h., es ist notwendig die Hintransformation von der SELinux-Instanz mit ihrer Sicherheitspolitik zu einem Modell wieder rückzutransformieren. Hierfür ist es erforderlich Zusatzinformationen, die eine verlustfreie Rücktransformation möglich machen schon bei der Hintransformation zu wahren.

Somit ergeben sich folgende zwei Hauptziele für diese Arbeit:

1. Ein Verfahren erarbeiten, welches es ermöglicht eine SELinux-Instanz und deren Sicherheitspolitik in ein Modell zu transformieren (Hintransformation) sowie das Modell in eine SELinux-Instanz mit deren Sicherheitspolitik zu transformieren (Rücktransformation).
2. Ein geeignetes Modell finden in dem alle für die Rücktransformation benötigten Informationen enthalten sind.

Die Umsetzung der zwei Hauptziele ermöglicht die modellbasierte Analyse von SELinux-Instanzen mit ihren Sicherheitspolitiken auf Rechteausbreitung wie bei [AKP11] beschrieben.

2 Typischer SELinux Anwendungsfall

Unix-basierte Betriebssysteme, wie Linux, benutzen diskrete Zugriffssteuerungsmechanismen (DAC) um zu entscheiden, ob ein Benutzer Rechte für eine Ressource, zum Beispiel eine Datei, besitzt oder nicht. Bei DAC kann ein Nutzer einer Ressource eigenverantwortlich bestimmen, welche Zugriffsrechte er selbst und andere für diese Ressource zugeteilt bekommen. Dieses Verfahren führt dazu, dass jeder Nutzer eines IT-Systems für seine Ressourcen die Sicherheitsrichtlinien selbst festlegen kann. Da der Nutzer im Allgemeinen aber beschränkte Einsicht in das Gesamtsystem hat, können dadurch Sicherheitslücken entstehen. Des Weiteren sind die DAC-Zugriffsregeln in Linux nicht dazu geeignet feingranulare Sicherheitseigenschaften durchzusetzen, es ist somit anfälliger für privilege escalation (vgl. [MMC06] Kapitel 2) und unterstützt auch keine Möglichkeit Zugriffsregeln basierend auf Attributen umzusetzen.

Um diese Probleme zu beheben, gibt es in SELinux die Möglichkeit der Umsetzung einer obligatorischen Zugriffssteuerung (MAC) [LS01]. Diese basiert auf der Idee, dass nicht jeder einzelne Nutzer bei der Umsetzung einer systemumfassenden Sicherheitspolitik beteiligt ist. Stattdessen gibt es eine zentrale systemumfassende Politik, welche die erlaubten Zugriffe für alle Nutzer festlegt. Eine solche Politik, im Folgenden SELinux-Sicherheitspolitik genannt, ist eine

Menge von Regeln, die die Erfüllung der Sicherheitsanforderungen an ein IT-System verfolgt. Diese SELinux-Sicherheitspolitik kann auch nur von autorisierten Nutzern verändert werden.

Ein Zugriff wird bei SELinux immer von einem Subjekt auf ein Objekt ausgeführt. Dabei werden die Objekte in Klassen wie Datei, Ordner, etc. unterteilt. Die Entscheidung, ob der Zugriff erlaubt ist basiert aber nicht auf den Subjekten bzw. den Objekten, sondern auf deren Sicherheits-Kontext und der Klasse.

Um eine möglichst hohe Flexibilität zu bieten und damit viele Einsatzgebiete abzudecken unterstützt SELinux drei Sicherheitskonzepte. Diese können je nach Bedarf kombiniert werden. Im Folgenden sind nur die zwei zum Verständnis der Arbeit wichtigen Konzepte erklärt.

Type Enforcement
Type Enforcement (TE) ist die Grundlage jeder SELinux-Sicherheitspolitik. Es bildet eine Abbildung der Objekte in den Kontext von SELinux. Anhand dieser werden daraufhin die Regeln der SELinux-Sicherheitspolitik durchgesetzt.

In SELinux ordnet das TE jedem Subjekt und jedem Objekt einen Type Identifier zu. Dieser wird unterschieden in Domänen für Subjekte und Typen für Objekte. In der SELinux-Sicherheitspolitik kann einem Paar von Domäne und Typ durch eine allow-Regel ein Recht zugewiesen werden. Dies ermöglicht es den Zugriff von Subjekten auf Objekte zu erlauben. Wenn für einen Zugriff keine allow-Regel definiert ist, wird dieser verweigert.

Eine allow-Regel könnte zum Beispiel so aussehen:

```
allow dnsmasq_t dnsmasq_conf_t : file {getattr read}
```

Diese Regel würde einem Subjekt, das in der Domäne `dnsmasq_t` ist, auf ein Objekt der Klasse `file`, welches vom Typ `dnsmasq_conf_t` ist, die Rechte `getattr` und `read` geben.

Role-based Access Control
Bei Role-based Access Control (RBAC) werden die vom TE festgelegten Rechte für Prozesse auf die Nutzer abstrahiert. Dies wird ermöglicht indem in der SELinux-Sicherheitspolitik dem Nutzer Rollen zugewiesen werden. Diese Rollen sind als Mengen von Domänen spezifiziert.

Wenn ein Prozess die Domäne wechselt, um zum Beispiel andere Rechte zu erlangen, wird dies nur erlaubt, falls die Zieldomäne in der selben Rolle wie die ursprüngliche Domäne ist und eine TE-Regel existiert, welche den Wechsel gestattet. Diesen Übergang von einer Domäne in eine andere nennt sich Transition und ist ein wichtiger Bestandteil von SELinux.

In Abb. 1 werden die Zusammenhänge zwischen einem Nutzer (Nutzer$_1$), den ihm zugeordneten Rollen (Rolle$_1$ ... Rolle$_n$) und den Rollen zugeordneten Domänen (Domäne$_{11}$ ... Domäne$_{1m}$, Domäne$_{n1}$ ... Domäne$_{nk}$) für einen Prozess dargestellt. Wie zu erkennen ist, gibt es in diesem Beispiel nur einen Nutzer. Das liegt daran, dass in SELinux der Wechsel des Nutzers nicht vorgesehen ist. Die Rollenwechsel und Transitionen müssen jeweils in der SELinux-Sicherheitspolitik erlaubt werden. Die durchgezogenen Pfeile stellen den Sicherheits-Kontext dar.

Die Umsetzung der vorangegangenen Sicherheitskonzepte basiert auf Sicherheitsattributen, welche im sogenannten Sicherheits-Kontext gespeichert sind. Dieser Sicherheits-Kontext wird den Subjekten und Objekten zugeordnet. Der Aufbau ist wie folgt:

```
user : role : type
```

Dabei ist `user` der SELinux-Nutzer, `role` die SELinux-Rolle und `type` der SELinux-Typ.

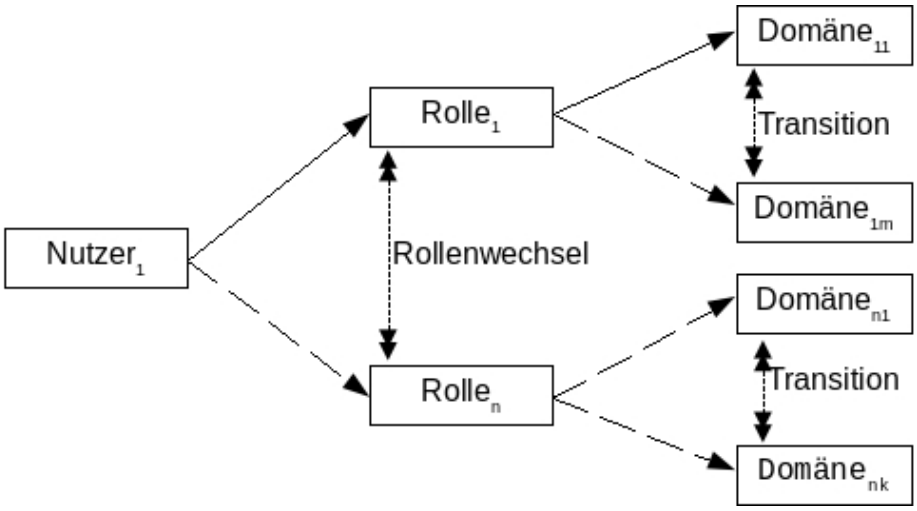

Abb. 1: Mögliche Zuordnung von Nutzer zu Rollen und Rollen zu Domänen.

3 SELX-Modell

Modelle welche sich für die Analyse einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik eignen sind z.B. [SSS04], [ZM04] und [XSA13]. Diese Modelle sind allerdings immer unter Berücksichtigung eines speziellen Analyseziels entwickelt worden und somit nicht allgemein genug um in dieser Arbeit genutzt werden zu können. Deshalb wurde sich im fogenden für das SELX-Modell [Amt16] als Ziel der Hintransformation entschieden. Es basiert auf dem Konzept der Core-basierten Modelle [Pöl14]. An dieser Stelle werden nur die für das Verständnis dieser Arbeit wichtigen Komponenten des SELX-Modells erklärt. Für weitere Informationen siehe [Amt16].

Der Aufbau des Modells ist ein Tupel $SELX = \langle Q, \Sigma, \delta, \lambda, q_0, EXT \rangle$ und beschreibt einen deterministischen Automaten, dessen Zustand die sicherheitsrelevanten Komponenten einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik repräsentiert.

Jeder Zustand $q \in Q$ des Tupels ist ein Tripel $\langle E_q, cl_q, con_q \rangle$, wobei

- die Menge $E_q \subseteq E$ die Entitäten in Zustand q (Subjekte und Objekte),
- die Menge $CL = \{cl_q | cl_q : E_q \to C\}$ alle zustandspezifischen Klassenzuweisungen und
- die Menge $CON = \{con_q | con_q : E_q \to SC\}$ alle zustandspezifischen Sicherheits-Kontext-Zuweisungen enthält.

Die Eingabemenge Σ setzt sich aus Befehlen Σ_{CMD} sowie der Menge von möglichen Parametern Σ_X (Aufrufer, Objekte, etc.) zusammen. Die Befehle Σ_{CMD} können die in SELinux möglichen Systemaufrufe oder Operationen auf Anwendungsebene sein.

δ ist die Zustandsüberführungsfunktion des Automaten. Mit deren Hilfe kann die Dynamik einer SELinux-Instanz beschrieben werden. Sie ermöglicht es somit z.B. eine Transition zu modellieren. Die Funktionen von δ wird über die Bedingungen PRE und POST gesteuert. Für jeden Befehl $cmd \in \Sigma_{CMD}$ mit seinem Vektor von Parametern $x_{cmd} \in \Sigma_x$ beschreiben diese beiden Bedingungen eine logische Vorbedingung PRE(cmd), welche ein Zustand q erfüllen muss, sowie eine Nachbedingung POST(cmd), welche der einzige mögliche Folgezustand $\delta(q)$ erfüllen muss.

Die Eingaben für den Automaten sind die sicherheitsrelevanten Befehle einer SELinux-Instanz. Um die Spezifikation dieser zu vereinfachen wird ein zweistufiges Verfahren angewendet. Zuerst werden grundlegende Befehle, im Folgenden Basis-Befehle genannt spezifiziert. Diese können für jede SELinux-Implementierung genutzt werden, da sie elementare Zugriffsmechanismen modellieren. Im Beispiel von SELinux sind diese z.B. **`access`**, **`create`** und **`relabel`**. Danach kann ein Abgeleiteter-Befehl angegeben werden. Dieser repräsentiert eine Zusammensetzung mehrere Basis-Befehle zu einem Befehl, z.B. einem Linux-Systemaufruf. Um die Zusammensetzung der Abgeleiteten-Befehle aus Basis-Befehlen zu modellieren wird der Kompositions-Operator $\circ : \Sigma \times \Sigma \cup \{\epsilon\} \rightarrow \Sigma$ wie folgt definiert:

$$\begin{aligned} \langle c_1, x_{c_1} \rangle \circ \epsilon &::= \langle c_1, x_{c_1} \rangle \\ \langle c_1, x_{c_1} \rangle \circ \langle c_2, x_{c_2} \rangle &::= \langle c_{12}, x_{c_1} x_{c_2} \rangle \end{aligned}$$

wobei $x_{c_1} x_{c_2} \in \Sigma_X$ eine Verkettung der Parameter $x_{c_1} \in \Sigma_X$ und $x_{c_2} \in \Sigma_X$ und $c_{12} \in \Sigma_{CMD}$ ein zusammengesetzter Befehl mit $\text{PRE}(c_{12}) = \text{PRE}(c_1) \wedge \text{PRE}(c_2)$ und $\text{POST}(c_{12}) = \text{POST}(c_1) \wedge \text{POST}(c_2)$ ist.

4 Hin- und Rücktransformation

Bei der Hintransformation müssen Informationen, die ein eindeutiges Zuordnen der Komponenten des SELX-Modells zu den zugrundeliegenden Komponenten der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik ermöglichen, gesichert werden. Um diese zu identifizieren wird zunächst mit der Rücktransformation begonnen.

Der Grundgedanke der Rücktransformation ist, dass Operationen des SELX-Modells auf Operationen der SELinux-Instanz abgebildet werden. Diese Vorgehensweise wird der sich anbietenden Herangehensweise, den Zustand eines SELX-Modells direkt auf einen Zustand der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abzubilden vorgezogen. Der Grund dafür ist, dass bei der Rücktransformation der Folge der Operationen diese Folge in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik nachvollzogen werden kann. Dies ermöglicht es nicht nur die Ergebnisse der Analyse in die zugrundeliegende SELinux-Instanz zu überführen, sondern auch die Folge von Operationen, welcher dahin geführt hat, nachzuvollziehen. Um den Endzustand der Analyse auch auf SELinux-Systemebene zu erreichen ist es dann nötig diese Folge von Operationen auszuführen.

Während der Analyse des Modells müssen die angewendeten Operationen und die betroffenen Entitäten gesichert werden. Die Rücktransformation nutzt diese Informationen, um aus der Folge der Operationen des SELX-Modells die zugrundeliegenden Linux-Systemaufrufe mit ihren Parametern zu rekonstruieren und die Prozesse, welche die Aufrufe ausführen sollen, zu identifizieren.

Die durch die Rücktransformation erstellte Folge von Operationen kann dann in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abgearbeitet werden. Dadurch wird die SELinux-Instanz in den äquivalenten Zustand gebracht, den das SELX-Modell nach der Analyse hat.

Zur bildlichen Darstellung des gewählten Verfahrens wird in Abb. 2 und 3 der Ablauf der Hin- und Rücktransformation dargestellt. Dabei stellt die Übertragung der statischen Regeln einer SELinux-Sicherheitspolitik (entnommen einer beliebigen Instanz des zu untersuchenden Systems) eine Vortransformation dar, welche nur einmal durchgeführt werden muss – formal bezeichnet durch die Transformationsfunktion g. Die eigentliche Hin- und Rücktransformation

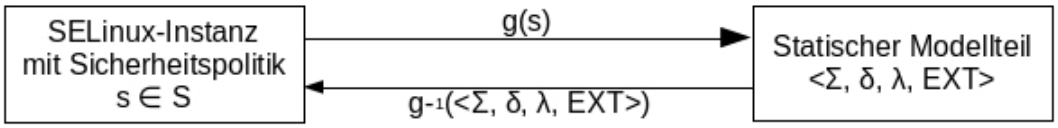

Abb. 2: Statische Transformation mittels g

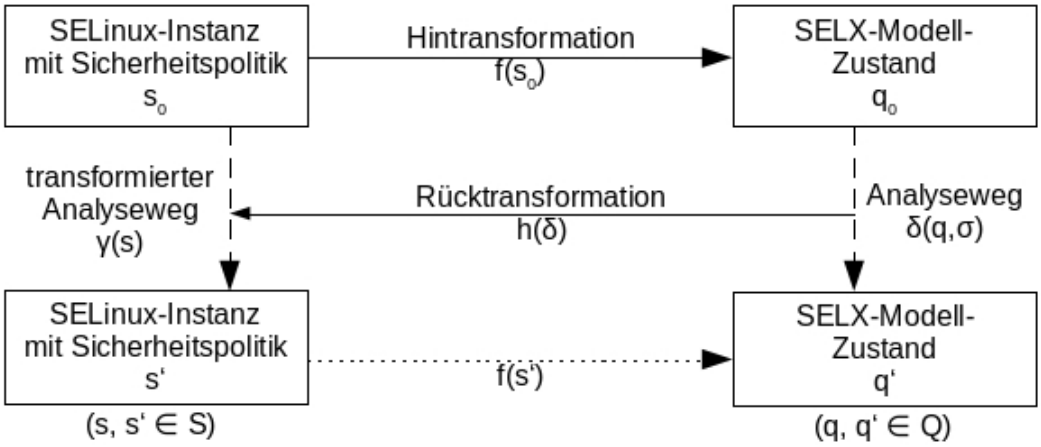

Abb. 3: Hin- und Rücktransformation des dynamischen Zustandes

des dynamischen Modellzustands, welcher später Gegenstand der modellbasierten Analyse sein soll, wird auf Basis der durch g bestimmten statischen Modellkomponenten mittels einer Transformationsfunktion f durchgeführt.

Der nach der Anwendung der rücktransformierten Folge von Operationen entstandene Zustand der SELinux-Instanz mit zugehöriger SELinux-Sicherheitspolitik muss nach erneuter Hintransformation den Zustand des SELX-Modells ergeben, der entsteht, wenn der Analyseweg auf das ursprüngliche SELX-Modell angewendet wird.

Um Verifizieren zu können, ob die durchgeführten Transformationen auch fehlerfrei wahren, bietet sich die Definition einer formalen Korrektheitsanforderung an. Für deren Definition werden zunächst folgende Funktionen benötigt:

- $f : S \to Q$ – ist die Funktion, welche den Zustand einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik auf einen Zustand eines SELX-Modells abbildet, also die Hintransformation repräsentiert
- $\delta : Q \times \Sigma \to Q$ – ist die Funktion, welche einen Zustand eines SELX-Modells auf einen anderen Zustand abbildet, also die Folge der Operationen von einem Zustand in einen anderen repräsentiert
- $\gamma : S \to S$ – ist die Funktion, welche einen Zustand einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik auf einen anderen Zustand abbildet, also die Folge der Operationen von einem Zustand in einen anderen repräsentiert
- $h : \Gamma \to \Delta$ – ist die Funktion, welche die Folge der Operationen von einem SELX-Modell-Zustand zu einem anderen auf eine Folge von Operationen von einem SELinux-Instanz-Zustand auf einen anderen repräsentiert

wobei

- S – ist die Menge aller SELinux-Instanzen mit der zu analysierenden SELinux-Sicherheitspolitik
- $\Delta = \{\delta | \delta : Q \times \Sigma \rightarrow Q\}$ – ist die Menge aller Funktionen δ
- $\Gamma = \{\gamma | \gamma : S \rightarrow S\}$ – ist die Menge aller Funktionen γ

Mit diesen Funktionen kann die Korrektheitsanforderung wie folgt beschrieben werden:

Der Zustand der SELinux-Instanz $s \in S$ eingesetzt in f ergibt den Zustand q des SELX-Modells.

$$f(s) = q \text{ mit } s \in S \text{ und } q \in Q$$

Wenn auf einen Zustand q des SELX-Modells eine Folge von Operationen angewendet wird, also die Eingabe $\langle q, \sigma \rangle$ in δ eingesetzt wird, entsteht der Zustand q' des SELX-Modells.

$$\delta(q, \sigma) = q' \text{ mit } q, q' \in Q$$

δ in die Funktion h eingesetzt ergibt γ, so dass:

$$\gamma(s) = s', \text{ wobei } \gamma = h(\delta) \text{ mit } s, s' \in S$$

Somit modelliert γ als abstrakte Übergangsfunktion die Folge der zu δ äquivalenten SELinux-Zustandsänderungen.

Der Zustand s' der SELinux-Instanz eingesetzt in f ergibt den Zustand q' des SELX-Modells.

$$f(s') = q' \text{ mit } s' \in S \text{ und } q' \in Q$$

Aus diesen Bedingungen kann die Gleichung zur Erfüllung der Korrektheit abgeleitet werden:

$$f(\gamma(s)) = f(s') = \delta(f(s), \sigma) \text{ mit } s, s' \in S \text{ und } \sigma \in \Sigma \quad (1)$$

Um die Hintransformation zu veranschaulichen wird im Folgenden eine beispielhafte Transformation des Linux-Systemaufrufs **`fork`** durchgeführt, so dass dieser im SELX-Modell als Operation benutzt werden kann. Dazu muss der Systemaufruf in geeigneter Weise aus der SELinux-Instanz in das SELX-Modell übertragen werden. Hierzu wird die bereits beschriebene Herangehensweise zum Erstellen von Abgeleiteten-Befehlen genutzt.

Um einen solchen Abgeleiteten-Befehl erstellen zu können, ist es notwendig, die Implementierung des Linux-Systemaufrufs auf Quelltextebene nachzuvollziehen. Dies entspricht der Transformationsfunktion g. Dadurch können die benötigten Zugriffsüberprüfungen und damit die benötigten Basis-Befehle identifiziert werden.

Beim Aufruf des Linux-Systemaufrufs **`fork`** wird lediglich überprüft, ob der Prozess das Recht hat sich selbst zu forken, d.h. es wird überprüft, ob es eine TE-allow-Regel gibt, welche der Domäne in der der Prozess läuft das Recht `fork` für einen Prozess in derselben Domäne gibt. Weitere Rechteüberprüfungen finden während des Linux-Systemaufrufs **`fork`** im Zusammenhang mit SELinux nicht statt.

Mit diesen Erkenntnissen können nun die für den Linux-Systemaufruf **`fork`** benötigten Basis-Befehle spezifiziert werden. Da nur eine Rechteüberprüfung stattfindet, wird nur ein **`access`**

Befehl der folgenden Form benötigt:

acccess(*caller*, *caller*, fork) ,
wobei

- *caller* – ist der Prozesses, der den **fork** aufruft
- fork – ist das Recht, welches überprüft werden soll

Durch den **fork** Aufruf wird ein neues Subjekt angelegt, d.h. es wird noch der Basis-Befehl **create** der folgenden Form benötigt:

create(*caller*, *child*, process) ,
wobei

- *caller* – ist der Prozess, der den **fork** aufruft
- *child* – ist der neue Prozess, welcher durch den **fork** entsteht
- process – legt fest, dass es sich bei der neu angelegten Entität um einen Prozess handelt

Es ergibt sich bei der Spezifizierung des Linux-Systemaufrufs **fork** folgender Abgeleiteter-Befehl für das SELX-Modell:

fork-SELX(*caller*, *child*)::=

access(*caller*, *caller*, fork)

- **create**(*caller*, *child*, process)

Während der Analyse ist es notwendig die Folge der Operationen zu wahren. Dies ermöglicht es der Rücktransformation diese Folge aus dem Kontext des SELX-Modells in den Kontext der zugrundeliegenden SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abzubilden. Dies erlaubt es die Folge der Operationen in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik nachzuvollziehen und die Parameter, welche die Linux-Systemaufrufe benötigen zu rekonstruieren. Diese Herangehensweise wird benötigt um die Korrektheitsanforderung (1) zu erfüllen.

Um die Folge der Operationen des SELX-Modells zu wahren und in die ursprüngliche SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abbilden zu können, müssen bei jeder Anwendung der Operation **fork-SELX** zwei wesentliche Informationen vermerkt werden. Zum einen muss der Name der Operation, in diesem Fall **fork-SELX**, und zum anderen der Parameter, in diesem Fall der *caller*, gewahrt werden. Damit ist das Subjekt gemeint, auf welches die Operation **fork-SELX** angewendet wird.

Der Name der Operation muss gewahrt werden um bei der Rücktransformation diese Operation eindeutig identifizieren zu können. Der Name des Subjektes, auf welches die Operation **fork-SELX** angewendet wird, muss gewahrt werden, damit in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik der Prozess, welcher den Linux-Systemaufruf **fork** aufrufen soll, identifiziert werden kann.

Das Subjekt, welches von der Operation **fork-SELX** erstellt wird, besitzt noch keinen korrespondierenden Prozess in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik. Daher muss das neue Subjekt einen Vermerk erhalten, von welchem Subjekt es erzeugt wurde. So ist es bei der Abarbeitung der Folge von Operationen von der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik möglich diesen Prozess für weitere Aufrufe eindeutig zu identifizieren.

Da bei der Rücktransformation die Operation und das Subjekt des SELX-Modells auf die Operation und den Prozess der SELinux-Instanz abgebildet werden müssen, ist es notwendig diese

eindeutige Abbildung bei der Hintransformation zu sichern. Diese Informationen sind ausreichend, um in der ursprünglichen SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik den Prozess eindeutig zu identifizieren und diesen den Linux-Systemaufruf **`fork`** ausführen zu lassen, was zur Erfüllung der Korrektheitsanforderung (1) benötigt wird.

5 Fallbeispiel Transition

Um das Konzept der Hin- und Rücktransformation einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik in ein SELX-Modell weiter zu veranschaulichen, soll im Folgenden mit der Transition eines Prozesses ein sicherheitskritischer Aspekt von SELinux behandelt werden.

Durch eine Transition eines Prozesses von einer Domäne in eine andere werden mögliche Rechte der Prozesse gewechselt. Deshalb ist dies ein wichtiger und komplexer Bestandteil von SELinux und wird als Fallbeispiel für die weitere Veranschaulichung der Machbarkeit der Hin- und Rücktransformation und die Validierung des Konzeptes gewählt.

In diesem Beispiel ist der Ablauf der Transition eines Prozesses wie folgt:

- Erstellung eines neuen Prozesses (Linux-Systemaufruf **`fork`**)
- Ersetzung dieses Prozesses mit einem neuen Prozess (Linux-Systemaufruf **`execve`**)

Der erste Teil wurde bereits im oben angegebenen Beispiel umgesetzt. Der zweite Teil, in dem auch die Transition des Prozesses aus einer Domäne in eine andere stattfindet, ist Gegenstand dieses Abschnittes.

Der Linux-Systemaufruf **`execve`** wird benutzt um einen Prozess mit einem anderen zu überschreiben. Diese Funktion an sich verlangt nicht zwingend eine Transition des Prozesses in eine andere Domäne. Da in diesem Beispiel aber genau dieser Aspekt der Domänen-Transition betrachtet werden soll, wird im Folgenden bei der Analyse des Linux-Systemaufrufs **`execve`** der Teil, der sich nicht mit einer Domänen-Transition beschäftigt, weggelassen. Es wird des Weiteren davon ausgegangen, dass die Transition durch die SELinux-Sicherheitspolitik erzwungen wird. Außerdem wurden aus Gründen der Übersichtlichkeit einige implementierungsspezifische Sicherheitsüberprüfungen weggelassen.

Zunächst ist es notwendig den Linux-Systemaufruf **`execve`** von der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik auf das SELX-Modell abzubilden. Dazu muss der Linux-Systemaufruf auf Quelltextebene analysiert und die notwendigen, für SELinux relevanten, sicherheitskritischen Aspekte extrahiert werden.

Beim Linux-Systemaufruf **`execve`** sind die sich auf SELinux beziehenden Sicherheitsüberprüfungen zunächst die Überprüfung, ob der Prozess, welcher den **`exevce`** aufruft, die Datei, welche ausgeführt werden soll, auch ausführen darf. Das wird im SELX-Modell mit folgendem Basis-Befehl repräsentiert:

`access`(*`caller`*, *`exec_file`*, `execute`) ,
wobei

- *`caller`* – ist der Prozess, der den **`execve`** aufruft
- *`exec_file`* – ist die Datei, welche ausgeführt werden soll
- `execute` – ist das Recht, welches überprüft werden soll

In diesem Zusammenhang wird auch überprüft ob der Prozess die Erlaubnis hat die Attribute

der Datei zu lesen. Dies wird im SELX-Modell durch folgenden Basis-Befehl repräsentiert:

access(*caller*, *exec_file*, getattr) ,
wobei

- *caller* – ist der Prozess, der den **execve** aufruft
- *exec_file* – ist die Datei, welche ausgeführt werden soll
- getattr – ist das Recht, welches überprüft werden soll

Wenn der Prozess alle vorangegangenen Rechte hat, kann überprüft werden, ob der Prozess die Rechte hat die Transition durchzuführen. Die dafür benötigten Rechte sind transition und entrypoint. Falls dies so ist, darf der Prozess den **execve** Aufruf ausführen. Dies hat zur Folge, dass sein Programmcode überschrieben wird und der Prozess eine Transition in eine neue Domäne durchführt. Dies wird in einem SELX-Modell durch folgenden Basis-Befehl repräsentiert:

relabel(*caller*, *exec_file*, *post_r*, *post_t*) ,
wobei

- *caller* – ist der Prozess, der den **execve** aufruft
- *exec_file* – ist die Datei, welche ausgeführt werden soll
- *post_r* – ist die Rolle, welche der Prozess nach der Transition annehmen soll (kann hier vernachlässigt werden, da nur die Domänen-Transition berücksichtigt wird)
- *post_t* – ist die Domäne, in welcher der Prozess nach der Transition laufen soll

Um nun eine Operation des SELX-Modells zu erhalten, werden die Basis-Befehle zu einem Abgeleiteten-Befehl zusammengefügt. Dieser hat die folgende Form:

execve-SELX(*caller*, *exec_file*, *post_r*, *post_t*)::=

- **access**(*caller*, *exec_file*, execute)
- **access**(*caller*, *exec_file*, getattr)
- **relabel**(*caller*, *exec_file*, *post_r*, *post_t*)

Um die Rückabbildbarkeit der Operation in die SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik umzusetzen, muss bei der Anwendung dieser Operation der Name dieser, also in diesem Beispiel **execve-SELX**, das Subjekt, auf welches die Operation angewendet wird, also *caller* und das Objekt, also *exec_file*, gewahrt werden. Die Rolle muss nicht gewahrt werden, da dieses Beispiel der Veranschaulichung einer Domänen-Transition dient somit keine Rolle gewechselt wird. Das gleiche gilt für die neue Domäne, welche in diesem Beispiel nicht gewahrt werden muss, da die Standard-Transition, wie sie in der SELinux-Sicherheitspolitik definiert ist, genutzt wird.

Diese Herangehensweise ermöglicht es die Korrektheitsanforderung (1) zu erfüllen, da daraus der zugrundeliegende Linux-Systemaufruf mit seinem Parameter sowie der Prozess, welcher den Aufruf durchführen soll, rekonstruiert werden kann.

Bei der Rücktransformation müssen die Operation, das Subjekt und das Objekt auf ihre korrespondierenden Bestandteile in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abgebildet werden. D.h., der Name der Operation muss auf den Namen des Linux-Systemaufrufs, der Name des Subjektes auf den aufrufenden Prozess, repräsentiert durch seine PID oder Verweis auf den Eltern-Prozess, und das Objekt auf seinen vollständigen Pfad abgebildet werden.

Dadurch kann in der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik der Prozess, welcher den **execve** ausführen soll, eindeutig identifiziert werden. Des Weiteren ist durch das Wahren des vollständigen Dateipfades eindeutig festgelegt, welche Datei ausgeführt werden soll. Durch diese Herangehensweise können die gewahrte Operation und die gewahrten Parameter auf die ursprüngliche SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik abgebildet werden, was nötig ist um die Korrektheitsanforderung (1) erfüllen zu können.

6 Evaluation

Um die Hin- und Rücktransformation auf ihre konzeptionelle Korrektheit zu untersuchen, d.h. ob das Verfahren grundlegend richtig funktioniert, soll im Folgenden eine exemplarische Hintransformation durchgeführt werden. Anschließend soll die oben spezifizierte Operation **execve-SELX** im Modell ausgeführt werden. Es wird bei der Ausführung der **execve-SELX**-Operation eine Transition stattfinden. Diese wird in der SELinux-Sicherheitspolitik als Standard definierte Transition aus der Domäne `init_t` in die Domäne `apache_t` über den entrypoint `apache_exec_t` angenommen.

Um nach der Rücktransformation die Korrektheitsanforderung (1) überprüfen zu können, wird zum Schuss noch einmal eine Hintransformation durchgeführt und ein komponentenweiser Vergleich des durch die erneute Hintransformation entstandenen Zustandes des SELX-Modells mit dem durch die Analyse entstandenen Zustandes des SELX-Modells durchgeführt. Eine formalere und genauere Beweisführung ist im Kontext dieser Arbeit leider nicht durchführbar.

Zunächst wird der Zustand einer minimalen SELinux-Instanz angegeben:

- Prozess:
  - `PID 42` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, init\_t \rangle$
- Datei/Ordner:
  - `/` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, root\_t \rangle$
  - `/apache.sh` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, apache\_exec\_t \rangle$

Es wird vorausgesetzt, dass die SELinux-Sicherheitspolitik die benötigten allow-Regeln enthält um eine Ausführung der **execve-SELX**-Operation zu gestatten.

Um nun aus diesem Zustand der SELinux-Instanz ein SELX-Modell zu erhalten wird für die oben genannte SELinux-Instanz der Prozess zu dem folgenden Subjekt:

- `sub1` mit $cl(\texttt{sub1}) = \texttt{process}$ und $con(\texttt{sub1}) = \langle user\_u, role\_r, init\_t \rangle$

die Datei und der Ordner werden zu den Objekten:

- `obj1` mit $cl(\texttt{obj1}) = \texttt{dir}$ und $con(\texttt{obj1}) = \langle user\_u, role\_r, root\_t \rangle$
- `obj2` mit $cl(\texttt{obj2}) = \texttt{file}$ und $con(\texttt{obj2}) = \langle user\_u, role\_r, apache\_exec\_t \rangle$

der Linux-Systemaufrufe **execve** wird auf die SELX-Operation **execve-SELX** abgebildet.

Bei der Hintransformation der SELinux-Sicherheitspolitik werden die benötigten Regeln zum Erlauben der Transition in die Datenstrukturen des SELX-Modells übertragen.

Um die Rückabbildbarkeit zu gewährleisten werden folgende Informationen gesichert:
`sub1`: `PID 42`; `obj1`: `/`; `obj2`: `/apache.sh` und **execve-SELX**: **execve**

Bei der Ausführung der **execve-SELX**-Operation werden folgende Parameter übergeben:

- `sub1` – das Subjekt, auf das die Operation angewendet wird
- `obj2` – das Objekt, auf das die Operation angewendet wird
- `role_r` – die Rolle nach der erfolgreichen Anwendung der Operation (ist in diesem Beispiel die gleiche wie vor der Operatione, da nur eine Domänen-Transition stattfindet)
- `apache_t` – die Domäne nach der erfolgreichen Anwendung der Operation

Da die benötigten Berechtigungen vorhanden sind um diese Transition durchzuführen, wird nach der Anwendung der Operation **execve-SELX** der Sicherheits-Kontext von `sub1` von $\langle user\_u, role\_r, init\_t\rangle$ auf $\langle user\_u, role\_r, apache\_t\rangle$ geändert, d.h. es gilt im neuen Zustand $con(\texttt{sub1}) = \langle user\_u, role\_r, apache\_t\rangle$. Zur Wahrung der Folge der Operationen werden folgende Informationen gesichert: **execve-SELX**; `sub1`; `obj2`

Um die Rücktransformation durchzuführen, werden die gesammelten Informationen zusammengetragen und in den Kontext der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik gebracht. Es wird also die Operation **execve-SELX** auf den Linux-Systemaufruf **execve**, das Subjekt `sub1` auf den Prozess `PID 42`, das Objekt `obj2` auf die Datei `/apache.sh` abgebildet. Anschließend wird der Linux-Systemaufruf **execve** mit dem Übergabeparameter `/apache.sh` von dem Prozess `PID 42` ausgeführt, was dazu führt, dass der Prozess eine Transition von der Domäne `init_t` in die Domäne `apache_t` durchführt. Der Zustand der SELinux-Instanz ist jetzt folgendermaßen:

- Prozess:
  - `PID 42` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, apache\_t\rangle$
- Datei/Ordner
  - `/` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, root\_t\rangle$
  - `/apache.sh` mit dem Sicherheits-Kontext $\langle user\_u, role\_r, apache\_exec\_t\rangle$

Die SELinux-Sicherheitspolitik bleibt unverändert.

Wenn nun auf diesen Zustand der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik die Hintransformation, mit den gleichen Rahmenbedingungen wie bei der ersten, angewendet wird, entsteht das folgende SELX-Modell.

Subjekt:

- `sub1` mit $cl(\texttt{sub1}) = \texttt{process}$ und $con(\texttt{sub1}) = \langle user\_u, role\_r, apache\_t\rangle$

Objekte:

- `obj1` mit $cl(\texttt{obj1}) = \texttt{dir}$ und $con(\texttt{obj1}) = \langle user\_u, role\_r, root\_t\rangle$
- `obj2` mit $cl(\texttt{obj2}) = \texttt{file}$ und $con(\texttt{obj2}) = \langle user\_u, role\_r, apache\_exec\_t\rangle$

Um die Rücktransformationen eindeutig durchführen zu können müssen folgende Informationen gewahrt werden:
`sub1`: `PID 42`; `obj1`: `/`; `obj2`: `/apache.sh` und **execve-SELX**: **execve**

Zur Überprüfung, ob dieser Zustand des SELX-Modells der gleiche ist wie nach der Analyse, wird ein komponentenweiser Vergleich durchgeführt:

Subjekte:

- Zustand des SELX-Modells nach Analyse:
  - `sub1` mit $cl(\texttt{sub1}) = \texttt{process}$ und $con(\texttt{sub1}) = \langle user\_u, role\_r, apache\_t \rangle$
- Zustand nach erneuter Hintransformation:
  - `sub1` mit $cl(\texttt{sub1}) = \texttt{process}$ und $con(\texttt{sub1}) = \langle user\_u, role\_r, apache\_t \rangle$

Objekte:

- Zustand des SELX-Modells nach Analyse:
  - `obj1` mit $cl(\texttt{obj1}) = \texttt{dir}$ und $con(\texttt{obj1}) = \langle user\_u, role\_r, root\_t \rangle$
  - `obj2` mit $cl(\texttt{obj2}) = \texttt{file}$ und $con(\texttt{obj2}) = \langle user\_u, role\_r, apache\_exec\_t \rangle$
- Zustand nach erneuter Hintransformation:
  - `obj1` mit $cl(\texttt{obj1}) = \texttt{dir}$ und $con(\texttt{obj1}) = \langle user\_u, role\_r, root\_t \rangle$
  - `obj2` mit $cl(\texttt{obj2}) = \texttt{file}$ und $con(\texttt{obj2}) = \langle user\_u, role\_r, apache\_exec\_t \rangle$

Zusatzinformationen:

nach Analyse	nach erneuter Hintransformation
`sub1: PID 42`	`sub1: PID 42`
`obj1: /`	`obj1: /`
`obj2: /apache.sh`	`obj2: /apache.sh`
`execve-SELX`: `execve`	**`execve-SELX`: `execve`**

Wie zu erkennen ist sind die beiden Zustände gleich. Somit ist die Korrektheitsanforderung (1) erfüllt. Auf konzeptioneller Ebene wurden bei dieser exemplarischen Vorgehensweise keine Fehler entdeckt und es wurde die Korrektheit nachvollziehbar nahegelegt.

7 Zusammenfassung

In dieser Arbeit wurde ein Verfahren für die Hintransformation von einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik zu einem SELX-Modell sowie die Rücktransformation dieses SELX-Modells in eine SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik entwickelt.

Durch die Hintransformation wird eine modellbasierte Analyse einer SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik ermöglicht. Hierbei werden zu den für das SELX-Modell benötigten Komponenten zusätzliche Informationen gesammelt, welche für eine eindeutige Rücktransformation gebraucht werden.

Nach einer modellbasierten Analyse des SELX-Modells wird durch die Rücktransformation ermöglicht, die Ergebnisse im Kontext der SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik auszuwerten. Die Rücktransformation bildet die Folge der Operationen, die während der Analyse im SELX-Modell angewendet wurden, auf eine Folge von Linux-Systemaufrufen ab. Diese Linux-Systemaufrufe werden anschließend in der ursprünglichen SELinux-Instanz mit ihrer SELinux-Sicherheitspolitik ausgeführt. Dadurch wird diese in den Zustand gebracht, welcher äquivalent zu dem Zustand des SELX-Modells ist, der durch die Analyse erreicht wurde.

Die Spezifikation des Autorisierungsschemas benötigt ein tiefgreifendes Verständnis der Linux-

Systemaufrufe auf Quelltextebene und ist somit mit intellektueller und auch kreativer Arbeit verbunden.

Die Korrektheit des Verfahrens wurde anhand einer exemplarischen Durchführung einer Hin- und Rücktransformation und der Überprüfung anhand einer weiteren Hintransformation nahegelegt. Dabei wurde auf die Einhaltung der erstellten Korrektheitsanforderung geachtet.

Die theoretische Art der Erstellung der Hin- und Rücktransformation und der Rahmen einer Arbeit ziehen einige Limitationen mit sich, welche bei der Weiterentwicklung des Verfahrens berücksichtigt werden müssen. Durch die beabsichtigte Einfachheit des SELX-Modells wird eine echte Verhaltensäquivalenz zwischen der SELinux-Instanz und ihrer SELinux-Sicherheitspolitik und dem zugehörigen SELX-Modell nicht erreicht. Dies begründet sich darin, dass in einem SELX-Modell nur sicherheitskritische Aspekte bezüglich SELinux beachtet werden. In der SELinux-Instanz gibt es aber noch weitere Zugriffsverfahren und Hardwarebegrenzungen, welche nicht im SELX-Modell berücksichtigt werden.

Literatur

[AKP11] Peter Amthor, Winfried E. Kühnhauser, and Anja Pölck. Model-based Safety Analysis of SELinux Security Policies. In P. Samarati, S. Foresti, J. Hu, and G. Livraga, editors, *In Proc. of 5th Int. Conference on Network and System Security*, pages 208–215. IEEE, 2011.

[AKP14] Peter Amthor, Winfried E. Kühnhauser, and Anja Pölck. WorSE: A Workbench for Model-based Security Engineering. *Computers & Security*, 42(0):40 – 55, 2014.

[Amt16] Peter Amthor. *E-Business and Telecommunications: 12th International Joint Conference, ICETE 2015, Colmar, France, July 20–22, 2015, Revised Selected Papers*, chapter The Entity Labeling Pattern for Modeling Operating Systems Access Control, pages 270–292. Springer International Publishing, Cham, 2016.

[LS01] Peter Loscocco and Stephen Smalley. Integrating Flexible Support for Security Policies into the Linux Operating System. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 29–42, Berkeley, CA, USA, 2001. USENIX Association.

[MMC06] Frank Mayer, Karl MacMillan, and David Caplan. *SELinux by Example: Using Security Enhanced Linux (Prentice Hall Open Source Software Development Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[Pöl14] Anja Pölck. *Small TCBs of Policy-controlled Operating Systems*. Univ.-Verl. Ilmenau, Ilmenau, 2014. ISBN: 978-3-86360-090-7.

[sel13] SELinux Project. http://selinuxproject.org/page/Main_Page, 2013. [Online; Zugriff 14.04.2015].

[SSS04] Beata Sarna-Starosta and Scott D. Stoller. Policy Analysis for Security-Enhanced Linux. In *Proceedings of the 2004 Workshop on Issues in the Theory of Security (WITS)*, 2004.

[XSA13] Wenjuan Xu, Mohamed Shehab, and Gail-Joon Ahn. Visualization-based policy analysis for SELinux: framework and user study. *International Journal of Information Security*, 12(3):155–171, 2013.

[ZM04] Giorgio Zanin and Luigi Vincenzo Mancini. Towards a Formal Model for Security Policies Specification and Validation in the SELinux System. In *Proc. of the 9th ACM Symposium on Access Control Models and Technologies*, pages 136–145. ACM, 2004.