Vorlesung Sicherheit

Dennis Hofheinz

ITI, KIT

07.07.2014

Überblick

Szenario positionsbasierte Kryptographie

- **Szenario:** *Erzeugen* von großen, „breitfrequenten“ (d.h. über viele Frequenzen verteilten) Datenmengen einfach, vollständiges *Speichern* (oder Weiterleiten) schwierig
- Insbesondere: sogar ehrliche V_i können viele Daten gleichzeitig senden, von dem jedes $\mathcal{A}_i$ nur einen kleinen Teil (etwa auf einer bestimmten Frequenz) speichern oder weiterleiten kann

Beispielprotokoll

- **Beispiel:** V_1 sendet gleichzeitig auf verschiedene Frequenzen verteilte Daten $(X_1, \ldots, X_n)$, und V_2 sendet $i^* \in \{1, \ldots, n\}$

$$\mathrm{V}_1 \xrightarrow{(X_1,\ldots,X_n)} \mathrm{P} \xleftarrow{i^*} \mathrm{V}_2$$

- V_1 und V_2 können sich vorher über i^* abstimmen
- Zweiter Schritt (eigentliche Authentifikation):

$$\mathrm{V}_1 \xleftarrow{X_{i^*}} \mathrm{P} \qquad \mathrm{V}_2$$

V_1 überzeugt, wenn das richtige X_{i^*} frühestmöglich ankommt

Beispielprotokoll

- Ehrliches Protokoll:

$$\mathrm{V}_1 \xrightarrow{(X_1,\ldots,X_n)} \mathrm{P} \xleftarrow{i^*} \mathrm{V}_2$$

$$\mathrm{V}_1 \xleftarrow{X_{i^*}} \mathrm{P} \qquad \mathrm{V}_2$$

- Angriffssituation (Beispiel: zwei $\mathcal{A}_i$, um Ps Position verteilt):

$$\mathrm{V}_1 \xrightarrow{(X_1,\ldots,X_n)} \mathcal{A}_1 \underbrace{\longleftrightarrow}_{\text{Abstand} \geq 2\Delta} \mathcal{A}_2 \xleftarrow{i^*} \mathrm{V}_2$$

- **Intuition:** $\mathcal{A}_1$ und $\mathcal{A}_2$ können X_{i^*} nicht an V_1 liefern
  - $\mathcal{A}_1$ erfährt i^* zu spät, hat das richtige X_{i^*} nicht gespeichert
  - $\mathcal{A}_2$ wird X_{i^*} erfahren, kann aber nicht rechtzeitig antworten

Überblick

Überblick

Erinnerung

- **Ziel:** *dynamische* Zugriffsverwaltung mit angepassten Sicherheitsleveln
- **Formal:** System besteht aus Subjekten (Nutzern) $\mathcal{S}$, Objekten (Dateien) $\mathcal{O}$, Zugriffsoperationen (`read`, `write`, ...) $\mathcal{A}$
- Zugriffe haben die Form $b \in \mathcal{S} \times \mathcal{O} \times \mathcal{A}$ und können gültig oder ungültig sein
- Zugriffskontrollmatrix M legt generelle Berechtigungen fest, Funktionen (f_s, f_c, f_o) Sicherheitslevel von Subjekten und Objekten

Sicherer Systemzustand

- Bell-LaPadula-Zustand sicher, wenn alle Zugriffe
  - die Zugriffskontrollmatrix respektieren (ds-Eigenschaft)
  - von hinreichend privilegierten Subjekten kommen (ss-Eigenschaft)
  - keine privilegierten Informationen freigeben ($\star$-Eigenschaft)
- Zugriff nur genehmigt, wenn er Systemsicherheit erhält (d.h. die obigen Eigenschaften hat)
- **Wichtig:** nach (genehmigten) `read`-Zugriffen wird *aktueller* Sicherheitslevel von Subjekten angepasst (genauer: galt vor Zugriff $f_c(s) < f_o(o)$, setzen wir nachher $f_c(s) := f_o(o)$)

Beispiel

- Subjekte $\mathcal{S} = \{\texttt{smith}, \texttt{jones}, \texttt{spock}\}$
- Objekte $\mathcal{O} = \{\texttt{salary.txt}, \texttt{mail}, \texttt{fstab}\}$
- Zugriffskontrollmatrix M gegeben durch

	salary.txt	mail	fstab
smith	{read}	{execute}	$\emptyset$
jones	{read, write}	{execute, read, write}	$\emptyset$
spock	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$

- Maximale/aktuelle Sicherheitslevel:

	$f_s(\cdot)$	$f_c(\cdot)$
smith	unclass.	unclass.
jones	secret	unclass.
spock	topsecret	unclass.

	$f_o(\cdot)$
salary.txt	secret
mail	unclass.
fstab	topsecret

Beispiel

- Abfolge von Zugriffen (in Reihenfolge):

Anfrage	ds	ss	$\star$	Bemerkung
(spock,salary.txt,append)	✓	✓	✓	keine Änderung bei $f_c(\texttt{spock})$
(smith,mail,read)	X	✓	✓	$\texttt{read} \notin M_{\texttt{smith,mail}}$
(smith,mail,execute)	✓	✓	✓	keine Änderung bei $f_c(\texttt{smith})$
(smith,salary.txt,read)	✓	X	✓	$f_s(\texttt{smith}) < f_o(\texttt{salary.txt})$
(jones,salary.txt,read)	✓	✓	✓	setzt $f_c(\texttt{jones}) = \texttt{secret}$
(spock,fstab,read)	✓	✓	✓	setzt $f_c(\texttt{spock}) = \texttt{topsecret}$
(spock,salary.txt,append)	✓	✓	X	$f_c(\texttt{spock}) > f_o(\texttt{salary.txt})$

Nachteile von Bell-LaPadula

- Bell-LaPadula betrachtet nur Sicherheits- (aber z.B. keine Integritäts-)eigenschaften
  - Niedrigprivilegierte Subjekte dürfen „nach oben schreiben“
- Unhandlich, weil „irgendwann jeder zuviel weiß“
  - Formaler: $f_c(s)$ kann nur wachsen $\Rightarrow$ $\star$-Eigenschaft wird zunehmend einschränkender
  - Lösung: $f_c(s)$ nach Zugriff zurücksetzen, aber ist das realistisch?
  - Alternative: „Mittelsmänner“, die gezielt $\star$-Eigenschaft umgehen dürfen

Nachteile von Bell-LaPadula

- Verdeckte Kanäle werden nicht grundsätzlich verhindert
  - Beispiel: hochprivilegierter Prozess `spock` legt Menge von (hochprivilegierten) Dateien an
  - niedrigprivilegierter Prozess `smith` darf diese Dateien beschreiben, erhält aber Fehlermeldung, wenn Datei schon existiert
    $\Rightarrow$ `smith` kann feststellen, ob `spock` Datei angelegt hat
    $\Rightarrow$ Informationsfluss von `spock` zu `smith` möglich
- Bell-LaPadula immer noch vergleichsweise **statisch**: Zugriffsrechte selbst (d.h. M) unverändert

Überblick

Szenario

- Menge $\mathcal{C}$ von Firmen
- Menge $\mathcal{S}$ von Beratern
- Menge $\mathcal{O}$ von Objekten
- Jedes Objekt $o \in \mathcal{O}$ gehört zu einer eindeutigen Firma $y(o)$
- Jedes Objekt $o \in \mathcal{O}$ hat Konflikte mit Firmen $x(o) \subseteq \mathcal{C}$
- **Ziel:** konfliktfreie Zuordnung von Beratern zu Objekten
- **Frage:** welche Interessenkonflikte könnten auftreten?

Simple Security

- **Gegeben:** `read`- oder `write`-Zugriffsanfrage $(s, o) \in \mathcal{S} \times \mathcal{O}$
- **Naheliegend:** Konflikt, wenn s in Vergangenheit schon Zugriff auf Objekt hatte, das in Konflikt zu $y(o)$ steht
- **Formalisierung:**

Definition (Simple-Security-/ss-Eigenschaft)

Eine (Lese- oder Schreib-)Anfrage (s, o) hat die ss-Eigenschaft, wenn für alle $o' \in \mathcal{O}$, auf die s schon Zugriff hatte, gilt: $y(o) = y(o')$ oder $y(o) \notin x(o')$.

Star Property

- **Subtiler:** betrachte *Schreibzugriff* (s, o)
- Konflikt, wenn s (bewusst oder unbewusst) Informationen über andere Objekte auf o überträgt; Beispiel:
  1. s_1 liest o_1 und schreibt auf o_2
  2. s_2 liest o_2 und schreibt auf o_3

  - Denkbar: $y(o_3) \in x(o_1)$ (d.h. o_1 in Konflikt mit Firma von o_3)
  - Problem: Information über o_1 *indirekt* nach o_3 geflossen

Star Property

- **Formalisierung:**

Definition (Star Property/$\star$-Eigenschaft)

Eine `write`-Anfrage (s, o) hat die $\star$-Eigenschaft, falls für alle Objekte o', auf die s schon lesend zugreift, gilt: $y(o') = y(o)$ oder $x(o') = \emptyset$.

- **Intuition:** Verhindert Informationsfluss aus der Firma heraus (außer, wenn unkritisch, weil Objekt in keinem Konflikt steht)

Beispiel

- $\mathcal{C} = \{c_1, c_2, c_3\}$, $\mathcal{O} = \{o_1, o_2, o_3\}$, wobei $y(o_i) = c_i$ und

$$x(o_1) = \{c_2, c_3\} \qquad x(o_2) = \{c_1\} \qquad x(o_3) = \emptyset$$

- Abfolge von Zugriffen (in Reihenfolge):

Anfrage	ss	$\star$	Bemerkung
(s_1, o_1) (`read`)	✓	✓	
(s_1, o_2) (`read`)	X	✓	s_1 hat schon o_1 mit $y(o_2) \in x(o_1)$ gelesen
(s_2, o_2) (`read`)	✓	✓	
(s_2, o_3) (`write`)	✓	X	s_2 liest schon o_2 (und $x(o_2) \neq \emptyset$)
(s_3, o_3) (`read`)	✓	✓	
(s_3, o_1) (`write`)	✓	✓	s_3 liest zwar schon o_3, aber $x(o_3) = \emptyset$

Überblick

Zusammenfassung

- Dateisysteme ermöglichen statische Zugriffskontrolle
- Bell-LaPadula-Modell verhindert unprivilegierten Zugriff auf privilegierte Objekte
  - Zentral: Hierarchisierung von Sicherheitsleveln
  - Berücksichtigt Vergangenheit (f_c)
  - Mechanismen zur Informationsflusskontrolle ($\star$-Eigenschaft)
- Chinese-Wall-Modell deckt potentielle Interessenkonflikte auf
  - Vollständig dynamisch, berücksichtigt Vergangenheit
  - Auch hier Informationsflusskontrolle ($\star$-Eigenschaft)

Beispiel aktueller Forschung

- Informationsflusskontrolle in komplexeren Systemen
  - Beispiel: Informationsfluss in Quellcode (z.B. JOANA, JIF)
  - Mögliche Aussage: Variable `secret` verlässt nie System über öffentliche, externe Ausgabe
  - Kann auch verborgene Informationskanäle aufdecken

Überblick

Überblick

Motivation

- Bislang Bausteine betrachtet
  - Verschlüsselung, Hashfunktionen, Authentifikation, ...
- Gesamtsystem erfordert üblicherweise mehrere Bausteine
  - Online-Banking benötigt Authentifikation und Verschlüsselung
- Welche Bausteine sollten wie eingesetzt werden?
- **Oder:** wie baut man größeres System modular auf?

Motivation

- Zwei Zugänge: (Information) Security und kryptographisch
- **Security-Zugang:**
  - Prüfe gezielt Eigenschaften des Gesamtsystems
  - Üblich: CIA-Paradigma (Confidentiality, Integrity, Availability)
- **Kryptographischer Zugang:**
  - Vergleiche reales System mit vereinfachtem, idealem System
  - Reales System sicher wenn „so sicher wie“ Idealisierung

Überblick

Das CIA-Paradigma

- Grundidee: überprüfe gezielt Eigenschaften des Gesamtsystems
- Üblicherweise drei entscheidende Eigenschaften (CIA):
  - Confidentiality: Geheimhaltung der Daten im System
    (Beispiel: Kontostand bleibt bei Online-Banking geheim)
  - Integrity: Integrität/Konsistenz von Daten im System
    (Beispiel: Angreifer kann Überweisungsdaten nicht ändern)
  - Availability: Verfügbarkeit des Systems
    (Beispiel: Angreifer kann Online-Banking-System nicht lahmlegen/Überweisungen kommen an)
- Manchmal auch weitere Eigenschaften betrachtet
  (Beispiel: Nicht-Abstreitbarkeit)