16 Informationsfluss

Bereits gesehen: Bell Lapadula soll Informationsfluss kontrollieren
Zwei Fragen:

- Wie setzt man das in Programmen um?
- Wie geht man mit verdeckten Kanälen um?
  Kanäle, die nicht für Informationsfluss vorgesehen sind

Compiler-based Mechanismen

Wie kann der Compiler testen, ob Informationsflüsse erlaubt sind?

Beispiel (expliziter Informationsfluss). Betrachte folgende Anweisung

$y := x;$

Informationenfluss von x nach y (in Zeichen $y \overset{flow}{\longleftarrow} x$).

Beispiel (impiziter Informationsfluss). Betrachte folgende Programme

if $x = 0$ then $y := a$;

else $y := b$;

Nicht nur Fluss $y \overset{flow}{\longleftarrow} a$ oder $y \overset{flow}{\longleftarrow} b$, sondern auch von $y \overset{flow}{\longleftarrow} x$
(aus $y = a$ folgt $x = 0$)

Im Folgenden: Sicherheitsmodell basierend auf einen Verband:

- Sicherheitsklassen SK mit partieller Ordnung $\leq$.
- Zwei Funktionen: glb (greates lower bound) und
  lub (least upper bound)
- Informationen fließen nur entlang der partiellen Ordnung
  $o \overset{flow}{\longleftarrow} o_1, \ldots, o_n$, wenn $\text{lub}(o_1, \ldots, o_n) \leq o$.

Jeder Variablen x ist eine Sicherheitsklasse $\underline{x}$ zugeordnet
Beispiel Deklaration: x : integer class A, dann gilt $\underline{x} = A$

Einige Beispiele für Anweisungen (Statements):

Assignment Statements: $y := f(x_1, \ldots, x_n)$
Informationen fließen von $x_1, \ldots, x_n$ nach y: $\text{lub}(\underline{x}_1, \ldots, \underline{x}_n) \leq \underline{y}$

Compound Statements: $S_1; S_2; \ldots S_n;$
Jede Anweisung $S_1, \ldots, S_n$ muss sicher sein

Conditional Statements: if $f(x_1, \ldots, x_n)$ then S_1; else S_2; end if

- S_1, S_2 müssen sicher sein
- $\text{lub}(\underline{x}_1, \ldots, \underline{x}_n) \leq \text{glb}\{\underline{y}; y$ Zielvariable einer Zuweisung in S_1 oder $S_2\}$

Beispiel siehe oben.

Hinweis: Hier kann es zu einer falschen Zurückweisung kommen (Übung)

Iterative Statements: while $f(x_1, \ldots, x_n)$ do S;

Wie Conditional Statements

- S muss sicher sein
- $\text{lub}(\underline{x}_1, \ldots, \underline{x}_n) \leq \text{glb}\{\underline{y}; y$ Zielvariable einer Zuweisung in $S\}$

Wissen darüber, ob Anweisung terminiert, liefert Informationen über $x_1, \ldots, x_n$
Daher zusätzlich: Loop muss terminieren.

Verdeckte Kanäle

Beispiel siehe oben (While-Schleife): Laufzeit liefert Informationen über $x_1, \ldots, x_n$.

Konkretes Beispiel:

Beispiel. Erinnerung RSA:

- Verschlüsselung: $x \mapsto x^e \bmod n$, e öffentlicher Exponent
- Entschlüsselung $c \mapsto c^d \bmod n$, d geheimer Exponent

- $n = p \cdot q$ Modul, p, q Primzahlen, $\phi(n) = (p-1)(q-1)$ Eulerzahl
  $e \in \mathbb{Z}^*_{\phi(n)}$ und $e \cdot d = 1 \bmod \phi(n)$.

Umsetzung der Entschlüsselung über binäre Exponentiation:
Eingabe c und $d = (d_{k-1} \cdots d_0)$ (in Binärdarstellung)

$x := 1$; $tmp := c$;

for $i := 0$ to $k-1$ do

if $d_i = 1$ then

$x := (x \cdot tmp) \bmod n$;

$tmp := (tmp \cdot tmp) \bmod n$;

end;

return x;

Zeit hängt von d ab (für $d_i = 1$ zwei Multiplikationen, für $d_i = 0$ eine)
Gibt also Informationen über den geheimen Schlüssel d

Weiteres Beispiel: Genking, Shamir, Tromer 2013
RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis

- Beispiele für Informationsquellen: Zeit, Energie, Akustik, Speicher
- Grund für diese Kanäle: gemeinsam genutzte Ressourcen.

Um verdeckte Kanäle zu entdecken (zu verhindern), Untersuchung:

- ob über einen Kanal Informationen fließen
- wie groß die Kapazität dieses Kanals ist

Sehr sehr kurze Einführung in Informationstheorie

$(W, \Pr)$: Wahrscheinlichkeitsraum, $\mathcal{X}$ Menge (Informationen).

$X : W \longrightarrow \mathcal{X}$: beschreibt Wkeit, mit der eine Nachricht x gewählt wird.
genauer $\Pr(X = x) := \Pr(X^{-1}(x))$: Wkeit, mit der $x \in \mathcal{X}$ gewählt wird.

Informationsgehalt einer Nachricht x: $I(x) := \log_2(1/\Pr(X = x)) = -\log_2 \Pr(X = x)$.

Entropie einer Verteilung X: $H(X) := -\sum_{x\in\mathcal{X}} \Pr(X = x) \log_2 \Pr(X = x)$
(mittlere Ungewissheit über Ausgang des Experimentes „Wähle $x \in \mathcal{X}$“)

Beispiel. Wir betrachten einen Münzwurf. Hier gilt $\Pr(1) = \Pr(0) = 1/2$.
Also $I(0), I(1) = \log_2 2 = 1$ und damit $H = 1$.

Übung: Betrachten Sie einen $n-$maligen Münzwurf.

Zurück zu verdeckten Kanälen:

- $(W, \Pr), \mathcal{X}$ und X wie oben
  (z.B. die Menge der eingesetzten kr. Schlüssel + Auswahlwkeit)
- Sei weiter $\mathcal{Y}$ eine Menge weitere Menge von Informationen
  (z.B. Laufzeiten eines Algorithmus)
- $Y : M \longrightarrow \mathcal{Y}$ eine Funktion (verdeckter Kanal)
  (z.B. Funktion, die einen kr. Schlüssel auf mögliche Laufzeiten abbildet)
- Wir definieren auf $\mathcal{Y}$ ebenfalls Wahrscheinlichkeitsmaß wie folgt

$$\Pr(Y = y) := \Pr(Y^{-1}(y))$$

Frage: Angreifer kennt $y \in \mathcal{Y}$. Was verrät das über $x \in \mathcal{X}$?

Bedingte Wahrscheinlichkeit: $\Pr(A|B) := \frac{\Pr(A\cap B)}{\Pr(B)}$.
Beispiel. $6-$seitiger Würfel, gerade Zahlen sind blau, ungerade rot.

- $M = \{1, \ldots, 6\}$, $\Pr_M(i) = 1/6$.
- $Z = \{\text{blau}, \text{rot}\}$
- $Y(1) = Y(3) = Y(5) = \text{rot}$ und $Y(2) = Y(4) = Y(6) = \text{blau}$

Es gilt $\Pr(2|Y = \text{blau}) = 1/3, \Pr(2|Y = \text{rot}) = 0$.

Bedingte Entropie:

- $H(X|y) := -\sum_{x\in\mathcal{X}} \Pr(X = x|Y = y) \log_2 \Pr(X = x|Y = y)$
  mittlere Unkenntnis über Ausgang des Experimentes wenn y bekannt

- $H(X|Y) := -\sum_{y\in Y} \Pr(Y = y) H(X|Y = y)$
  mittlere Unkenntnis über Ausgang bei Kenntnis der Verteilung Y

Offensichtlich gilt: $H(X|Y) \leq H(X)$

- Gilt $H(X|Y) = H(X)$ liegt kein verdeckter Kanal vor (X, Y unabhängig)
- Differenz gibt Kapazität des Kanals an.

Wie verhindert man Informationsfluss über verdeckte Kanäle?

- Isolation: Virtuelle Maschinen, Sandbox
  Immer noch gemeinsam genutzte Ressourcen
- Deployment-Problem:
  - Gegeben: Menge von Nutzern mit Restriktionen über einen Graph
    Kante zwischen Nutzern: Dürfen gemeinsam Ressourcen nutzen
    (z.B. Computer, VM, Programm)
  - Ziel: Minimiere Anzahl Computer, VM, Programme
  - Lange, Margraf, Ruehl: Problem ist NP-schwer.

Nächste Idee: Nutze Zufall:

- Für Beispiel oben: Rechne $c^{d+r\cdot\phi(n)}$ mod n, r Zufall
  Damit ist der Kanal ausgeschaltet
- Für Großrechner: Für zufällig ausgewählte Prozesse aus.