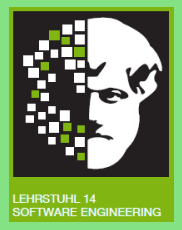
Analysetechniken für Informationssicherheit: Einleitung & Grundlagen

Alexander Schäferdiek

Technische Universität Dortmund, Fakultät für Informatik,
Lehrstuhl XIV – Software Engineering,
Arbeitsgruppe Prof. Dr. Jan Jürjens

12. Februar 2014



Inhaltsverzeichnis

Inhaltsverzeichnis

Einleitung

Heutige IT:

- steigende Komplexität in Programmen
- Sicherheit wichtige Anforderung
- Schutz eigener Infrastrukturen vor Zugriff von Dritten

$\rightarrow$ theoretische Analyse von Code nötig

2 Arten der theoretischen Analyse:

- statische Analyse (ohne Codeausführung)
- **Datenflussanalyse** (Abstrahierung der möglichen Zustände zur Laufzeit)

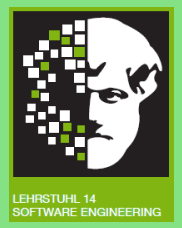
Inhaltsverzeichnis

Ziele

Gegeben: Programmcode
Gesucht:

- **automatisiertes Analyseverfahren**
- Wissen über alle möglichen Zustände
- Wissen über Zustandsänderungen innerhalb des Programms
- Darstellung des Wissens

$\rightarrow$ Speicherzustände (Wissen, welches ein Angreifer haben könnte)

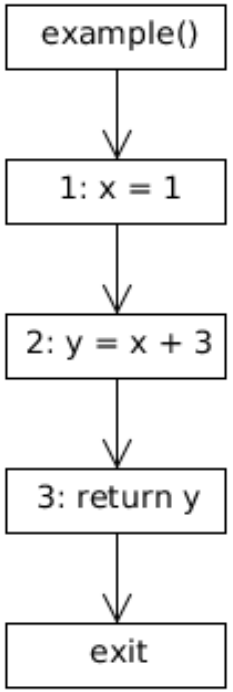


Inhaltsverzeichnis

Grundlagen (CFG)

Kontrollflussgraph zu einem Beispielprogramm

```
int x,y;
int example() {
    x = 1;
    y = x + 3;
    return y;
}
```



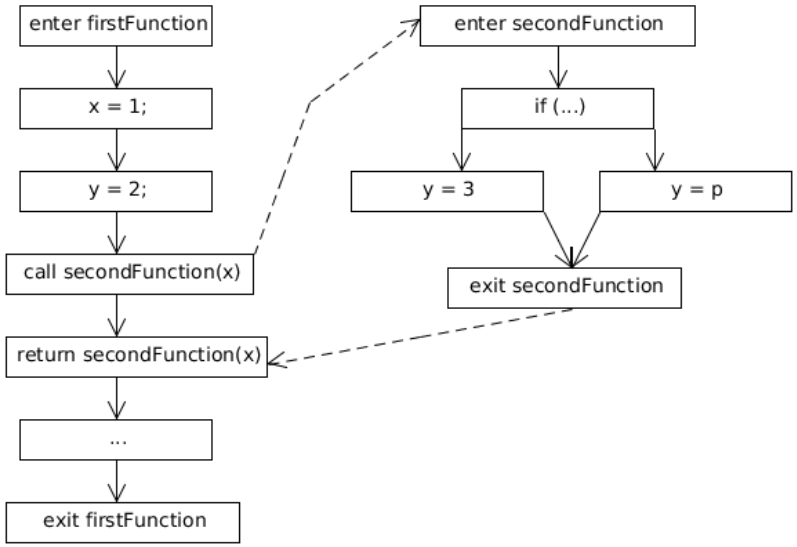
Grundlagen (ICFG I)

Interprozeduraler Kontrollflussgraph zu einem erweiterten Beispielprogramm

```
int y;

void firstFunction() {
int x = 1;
int y = 2;

secondFunction(x);
...
}
void secondFunction(int p) {
if (...)
y = 3;
else
y = p;
}
```



Grundlagen (ICFG II)

- zusätzliche Knoten für Ein- und Austritt einer Funktion
- jede Funktion stellt eigene Prozedur dar
- call mit enter Knoten verbunden
- return mit exit Knoten verbunden
- Graphenalgorithmen zur Pfadsuche

Welche Knoten (Speicherzustände) sind von einem Startknoten v_0 erreichbar?

$\rightarrow$ approximierende Graphenalgorithmen zur Erstellung von autonomen Systemen bei der Pfadfindung

Grundlagen (Pfade)

Formale Vorgehensweise zur Mengenbestimmung der erreichbaren Knoten

Definition

Sei ein Pfad p = $[e_1,..,e_k]$ eine Hintereinanderreihung von k Kanten $e_i \in E$ eines ICFG, wobei der Endknoten der Kante e_i der Startknoten der Kante e_{i+1} ist, die Bedingung $k >= 1$ erfüllt und ein Pfad von Knoten v nach v die Länge 0 hat, dann lässt sich mithilfe einer Zuweisung $M(e) \in V \rightarrow V$, einer Datenflussfunktion zu jedem verbundenen Knotenpaar, eine **Pfadfunktion** bestimmen:
$pf_q = M(e_k) \circ ... \circ M(e_1)$.
Diese Funktion pf_q ist für den Pfad der Länge $k = 0$ eine leere Hintereinanderreihung [] (die Identitätsfunktion am Knoten v).

Grundlagen (gültige Pfade & JOVP)

- nicht alle Pfade ausgeführt
- Konzentration auf gültige Pfade
- Bestimmung mit kontextfreier Grammatik
  - *exit-return* Kanten mit *i)* notieren
  - *call-enter* Kanten mit *(i* notieren
  - Hintereinanderreihung von v_0 nach n

valid $\rightarrow$ matched valid
| (i valid
| ϵ

matched $\rightarrow$ matched matched
| (i matched i)
| Kantenname
| ϵ

Definition

Join-Over-All-Valid-Paths (JOVP). Für einen Startknoten $v_0 \in V$ zu einem Zielknoten n, seien alle Pfade definiert durch: $JOVP_n = \cup_{q \in GueltigePfade(enter,n)} pf_q(v_0)$

Inhaltsverzeichnis

WPDS (1)

Definition

Weighted Pushdown System (WPDS). Ein WPDS ist ein Tripel W = (P, S, f), wobei P = (Q, Γ, Δ) ein Pushdown System (PDS) ist und Q eine endliche Menge an Zuständen (Kontrollzustände), Γ eine endliche Menge an Kellersymbolen (Kelleralphabet) und Δ, mit $\Delta \subseteq P \times \Gamma \times Q \times \Gamma^*$, eine endliche Menge an Regeln ist. Eine Regel wird definiert als $r \in \Delta$, geschrieben $(q, \gamma) \rightarrow (q', u)$, wobei $q, q' \in Q, \gamma \in \Gamma$ und $u \in \Gamma^*$. Die Funktion f bildet ein Gewicht auf jede Regel aus P ab. S = (d $\in$ {Gewichte}, $\oplus$, $\otimes$, 0', 1'), ist ein idempotenter (mit sich selbst verknüpfter) Halbring, welcher unterschiedliche Datenabstraktionen (= **Domänen**) darstellt.

WPDS (2)

Definition

Domäne. Eine Domäne ist ein idempotenter Halbring S = (d $\in$ {Gewichte}, $\oplus$, $\otimes$, 0', 1'), der folgende Eigenschaft einhält:

a) (D, $\oplus$) ist Monoid (e = $0'$); (D, $\otimes$) ist Monoid (e = $1'$)
b) Distributivität für $\oplus$
c) neutrales Element 0', wobei Verknüpfung mit $\otimes$ ist $0'$
d) Halbordnung für $\oplus$

- Boolean-Domäne ({true, false}, $\wedge$, $\vee$, F, T) $\rightarrow$
- Relationale Gewichts-Domäne ($2^{G \times G}$, $\cup$, ;,$\emptyset$, id)
- uvm. (ARA) [01]

WPDS (Kodierung)

Vom ICFG zum (Weighted) Pushdown System [01]:

Regel	ICFG Modell
$(p, u) \rightarrow (p, v)$	interprozedurale Kante $u \rightarrow v$
$(p, c) \rightarrow (p, e_f\, r)$	ruft f von c auf und geht zu r
$(p, x_f) \rightarrow (p, \epsilon)$	return von f zu exit Knoten x_f

kodierter PDS aus Beispiel ICFG (Abbildung 9):

(p, enter FirstFunc) → (p, x=1)
(p, x=1) → (p, y=2)
(p, call SecFunc) → (p, e_f return SecFunc)
(p, ...) → (p, exit FirstFunc)
(p, exit FirstFunc) → (p, ϵ)
(p, enter SecFunc) → (p, if (...))

(p, if(...)) → (p, y=3)
(p, y=3) → (p, exit SecFunc)
(p, if (...)) → (p, y=p)
(p, y=p) → (p, exit SecFunc)
(p, exit SecFunc) → (p, ϵ)

WPDS (Beispiel)

Beispiel WPDS mit Boolean-Domäne [01] ({true, false}, $\wedge$, $\vee$, F, T) zur Konstruktion eines ω-Automaten :

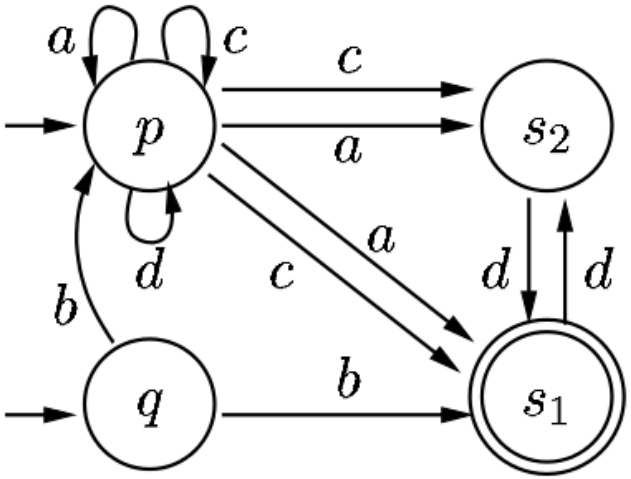
$$\mathrm{Q} = \{\mathrm{p,q}\},\ \Gamma = \{\mathrm{a,b,c,d}\}$$
$$r_1 = (p, a) \rightarrow (q, b)$$
$$r_2 = (p, a) \rightarrow (p, c)$$
$$r_3 = (q, b) \rightarrow (p, d)$$
$$r_4 = (p, c) \rightarrow (p, ad)$$
$$r_5 = (p, d) \rightarrow (p, \epsilon)$$

WPDS (pre*-Algorithmus I)

1. Für jedes Paar (p', γ'), sodass der PDS minimal eine Regel der Form (p, γ) $\rightarrow$ (p', $\gamma'\gamma''$) enhält, füge einen neuen Zustand $q_{p',\gamma'}$ hinzu.
2. Wenn eine Regel der Form (p, γ) $\rightarrow$ (p', ϵ) existiert und γ einen (auch ϵ) Übergang zu q beschreibt, dann füge eine Transition (p', ϵ, q) hinzu.
3. Wenn eine Regel der Form (p, γ) $\rightarrow$ (p', γ') existiert und γ einen (auch ϵ) Übergang zu q beschreibt, dann füge eine Transition (p', γ', q) hinzu.
4. Wenn eine Regel der Form (p, γ) $\rightarrow$ (p', $\gamma'\gamma''$) existiert und γ einen (auch ϵ) Übergang zu q beschreibt, dann füge eine Transition (p', γ', $q_{p',\gamma'}$) und ($q_{p',\gamma'}$, γ'',q) hinzu.

WPDS (pre*-Algorithmus II)

ω-Automat mithilfe des **pre*-Algorithmus'**:



Inhaltsverzeichnis

Datalog (1)

- alternative Wissensrepräsentation
- Anwendung bei
  - Regeln für Sicherheitszugriffe
  - Datenbanken
  - uvm. [01]
- zu Prolog ähnliche Syntax und Semantik (:- durch $\rightarrow$ ersetzt)

Beispiel (Datalogprogramm P):

```
connected(pete, oscar).
connected(oscar, mary).
connected(X, Y) <- connected(X, Y), connected(Y, Z).
```

Datalog (2)

- vor $\leftarrow$ heißt *head*
- nach $\leftarrow$ heißt *body*
- *Regel* r ist $X \leftarrow Y$
- falls $body(r) = \emptyset$ heißt r Faktum

Gegeben: P, F $= \exists\, x_1, ..x_n(connected(X, mary) \wedge X \neq oscar)$
Gesucht: Antwort zur Anfrage F. Gilt P $\models$ F?
Verfahren:

- Fixpunktberechnung mit Herbranduniversen (bottom-up)
- Resolution (top-down)

Datalog (Fixpunktberechnung I)

Grundlagen:

- Herbranduniversum U_A: gegrundete Elemente (enthält keine Variablen)
- Herbrandbasis H_B: Konstantenkombination eingesetzt in Prädikate
- Herbrandmodell: Teilmenge von H_B, durch Regelanwendung bestimmt

Beispiel:

$U_A = \{\text{pete, oscar, mary}\}$

$H_B = \{\text{c(pete, oscar), c(oscar, pete), c(pete, mary), c(mary, pete, c(pete, oscar), c(oscar, mary), c(mary, oscar)}\}$

Datalog (Fixpunktberechnung II)

- H_B, U_A endlich
- Anwendung auf gegrundete Anfangsinstanz
- Berechnung mit Konsequenzoperator [01]
- wachsend monoton (Wissen bleibt erhalten)
- Ergebnis $\in H_B$

$T_P \uparrow^{\omega} = \cup_{i=0}^{\infty}(T_P \uparrow^{i})$, wobei $T_P \uparrow^{0} = \emptyset, T_P \uparrow^{i+1} = T_P(T_P \uparrow^{i})$ und $i \geq 0$

Beispiel:

1. {c(pete, oscar), c(oscar, mary)}
2. {c(pete, oscar), c(oscar, mary), c(pete, mary)}

$\rightarrow$ connected(pete, mary) $\in H_B$

Datalog (Resolution)

- ähnlich zur SLD-Resolution [02]
- erweitert mit SLG-Resolution (immer terminierend [01])
- $P \cup \neg F$ *kein* Modell, F wahr

Beispiel:
F = $\exists\, x_1, ..x_n(connected(X, mary))$
$\neg F = \forall\, x_1, ..x_n(\neg\, connected(X, mary))$
↯Widerspruch mit Fakt connected(oscar, mary)
→ kein Modell
→ F wahr

Datalogprogramme meist komplexer (z.B. Wissen in Datenbanken)

Inhaltsverzeichnis

Zusammenfassung & Ausblick

- WPDS/ω-Automaten effizientes Analyseverfahren
  - Konstruktion autonomer Systeme (pre*/post*)
  - intelligente Pfadsuche
  - Datentypen als Domänen
  - Gewichtungen als Informationsgehalt von Zuständen
  - (Speicher/Angreifer) Wissenbeschreibung
- alternative Wissenrepräsentations Datalog
  - Anwendung in vielen Gebieten (z.B. Datenbanken, *analyzing security policies* [01])
  - Ähnlichkeit zu Prolog
  - garantierte Terminierung

$\rightarrow$ stärkt Sicherheitsverständnis

$\rightarrow$ Vermeidung von Planungs- und Sicherheitsfehlern

Literatur

[01] Anupam Datta, Somesh Jha, Ninghui Li, David Melski, Thomas Reps: Analysis Techniques for Information Security. 2010.

[02] Uwe Schöning: Logik für Informatiker (5. Ausgabe). 2000.

[03] Thomas Reps, Stefan Schwoon, Somesh Jha, David Melski: Weighted Pushdown Systems and their Application to Interprocedural Dataflow Analysis. 2005.