Karlsruhe Institute of Technology

Seminar Softwaresicherheit, SS2011

Gregor Snelting, Martin Hecker, Jürgen Graf

IPD, PROGRAMMING PARADIGMS GROUP, COMPUTER SCIENCE DEPARTMENT

```
theorem nonInterferenceSecurity:
  assumes "[cf1] ≈L [cf2]" and "(-High-) ∉ ⌊HRB-slice (CFG-node (-Low-))⌋CFG" and "valid-edge a"
  and "sourcenode a = (-High-)" and "targetnode a = n" and "kind a = (λs. True)√" and "n ≙ c"
  and "final c'" and "⟨c,[cf1]⟩ ⇒ ⟨c',s1⟩" and "⟨c,[cf2]⟩ ⇒ ⟨c',s2⟩"
  shows "s1 ≈L s2"
proof –
  from High-target-Entry-edge obtain ax where "valid-edge ax" and "sourcenode ax = (-Entry-)"
    and "targetnode ax = (-High-)" and "kind ax = (λs. True)√" by blast
  from `n ≙ c` `⟨c,[cf1]⟩ ⇒ ⟨c',s1⟩` obtain n1 as1 cfs1 where "n –as1→√* n1" and "n1 ≙ c'" and "preds (kinds as1) [(cf1,undefined)]"
    and "transfers (kinds as1) [(cf1,undefined)] = cfs1" and "map fst cfs1 = s1" by(fastsimp dest:fundamental-property)
  from `n –as1→√* n1` `valid-edge a` `sourcenode a = (-High-)` `targetnode a = n` `kind a = (λs. True)√`
  have "(-High-) –a#as1→√* n1" by(fastsimp intro:Cons-path simp:vp-def valid-path-def)
  from `final c'` `n1 ≙ c'` obtain a1 where "valid-edge a1" and "sourcenode a1 = n1" and "targetnode a1 = (-Low-)" and "kind a1 = ↑id"
    by(fastsimp dest:final-edge-Low)
```

KIT - University of the State of Baden-Wuerttemberg and
National Research Center of the Helmholtz Association

www.kit.edu

Softwaresicherheit

Was versteht man unter Sicherheit?

- Verfügbarkeit (Availability)
- Vertraulichkeit (Confidentiality)
- Integrität (Integrity)

Focus: Sicherheit von Programmen

- keine Netzwerke (Protokolle, Kryptographie...) - Prof. Müller-Quade
- keine Hardware

Genauer: Sprachbasierte Sicherheit von Programmen

- Analyse des Quelltextes (oder Bytecode)

Ablauf eines Vortrags

- Thema: 2 Paper (Grundlage + Vertiefung) zu einem Gebiet
  - weitergehende Recherchen erwünscht
- ca. 45min Vortrag + 15min Diskussion
- 15-20 Seiten Ausarbeitung
- Inhalt
  - Eingrenzen und Einordnen + Motivation
  - Grundlagen des jeweiligen Verfahrens (Paper 1)
  - Aktuelle Verbesserungen und Forschungsrichtungen (Paper 2)
  - Vor- und Nachteile des vorgestellten Verfahrens
  - Eigene substantielle Beispiele
  - Eigene Einschätzung, Alternativen
- Master-Studenten anwesend? => Blaue Zettel
  - [IN4INSEMSWT] Modul: Seminar Softwaretechnik
    - VF 6: Softwaretechnik und Übersetzerbau
  - [SWTSem] Lehrveranstaltung: Seminar Softwaretechnik

Forschung am KIT

- RS3: Realiably Secure Software Systems
  - DFG SchwerPunkt Programm (SPP)
  - DeduSec (KeY) - *Prof. Beckert, Prof. Schmitt*
  - System-wide data-driven runtime usage control - *Prof. Pretschner*
  - IFC for Mobile Components - *Prof. Snelting*
- VALSOFT/Joana
- KASTEL - Kompetenzzentrum für Angewandte Sicherheits-Technologie - *Prof. Müller-Quade*
- Dependable Software for Critical Infrastructures (DSCI)
  - Antrag für Exzellenzcluster
- u.v.m.

Interessante Konferenzen (unvollständig)

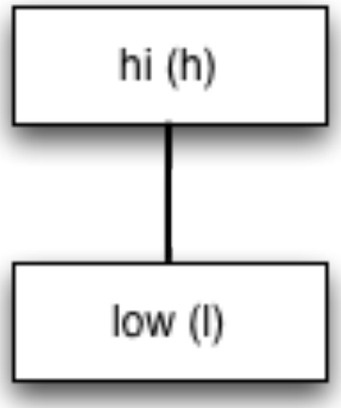
- Sicherheit
  JIS: International Journal of Information Security
  JCS: Journal of Computer Security
  ESORICS: European Symposium on Research in Computer Security
  ESSoS: International Symposium on Engineering Secure Software and Systems
  SEC: International Information Security Conference
  CSF: Computer Security Foundations Symposium/Workshop
  ICISS: International Conference on Information Systems Security
  FAST: International Workshop on Formal Aspects of Security & Trust
  SSP: IEEE Symposium on Security and Privacy
- Sicherheit, Programmanalyse & Programmiersprachen
  PLAS: Workshop on Programming Languages and Analysis for Security
- Programmanalyse, Programmiersprachen
  TOPLAS: Transactions on Programming Languages and Systems
  PLDI: Programming Language Design and Implementation
  POPL: Principles of Programming Languages
  ISSTA: International Symposium on Software Testing and Analysis
  SCAM: International Working Conference on Source Code Analysis and Manipulation

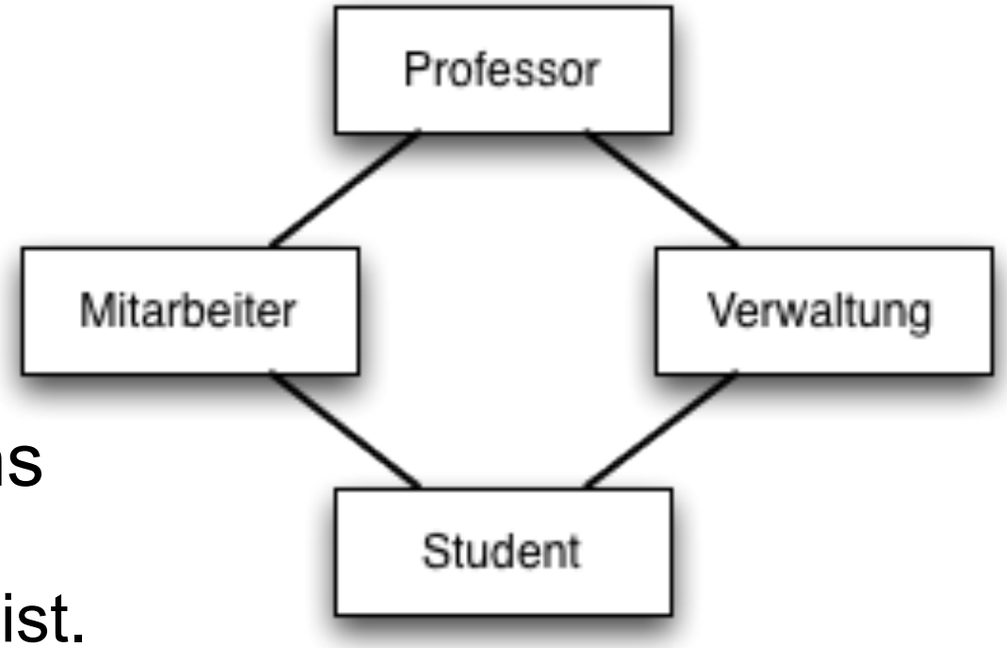
Information Flow Control (IFC)

- Was passiert mit meinen Daten?
- Zusätzlich zu Zugriffskontrolle, Protokollsicherheit, ...
- Problem: Wodurch fliesst Information?
  - Direkter Fluss `a = b`
  - Indirekter Fluss `if (b == 4) a = 6`
  - Timing `while (a > 0) a--`
  - Diverse Seitenkanäle
    - Stromverbrauch, Prozessorauslastung, Speicherverbrauch, ...
- Problem: Wieviel Information fliesst?
  - Wie misst man Information?
- Sicherheitskriterium: Was bedeutet Sicherheit?
- Was ist in der Praxis relevant?

IFC - Notation und Grundlagen

- **Sicherheitsstufen** für Information (Variablen, Ausdrücke, ...)
  - Verband





- **Integrität**: Beeinflussen des Verhaltens
  - s1 -> s2 erlaubt, falls s1 >= s2
  - Bsp: Student sagt Professor was zu tun ist.
  - low -> hi? `h = l`
- **Vertraulichkeit**: Geheime Information bewahren
  - s1 -> s2 erlaubt, falls s1 <= s2
  - Bsp: Mitarbeiter sagt Student Note des Kommilitonen.
  - hi -> low? `l = h`
- Dualität: Verband umdrehen

Einordnen des Themas

- Zugriffskontrolle, IFC (integrity - confidentiality), ...
- Sicherheitskriterium
- Analyseverfahren
  - statisch - dynamisch
  - konservativ - nicht konservativ
  - whole program - modular - inkrementell
  - Skalierbarkeit
  - Präzision
- Elfenbeinturm - relevant in der Praxis
  - Programmiersprache eingeschränkt?
  - Sicherheitskriterium sinnvoll?
- Anwendung
  - einfach (Automatisiert, wenig Interaktion, Lernaufwand)
  - umständlich (hoher Lern- und Arbeitsaufwand)

Karlsruhe Institute of Technology

Die Themen

IFC mit Typsystemen

- Sicherheitstypsysteme
  - Programm ist typkorrekt => Programm hat kein Informationsleck
- Grundlagen: Typsysteme und Sprachdesign

$$\frac{a : T \qquad l : List[T]}{a * l : List(T)} \qquad \frac{a : T \qquad l : List[T]}{h(a * l) * t(a * l) = a * l}$$

- JFlow / Jif: Erweiterung für Java

```
int{Alice→Bob} x;
int{Alice→Chuck} y;
if (Bob actsfor Chuck) {
    x = y; // OK, since {Alice→Chuck} ⊑ {Alice→Bob} is
           // deducible from the label environment at this point
}
```

IFC mit Abhängigkeitsgraphen

- Programmanalyseverfahren
- Grundlage: Abhängigkeitsgraphen (PDG/SDG)
- Joana/jSDG: IFC für bestehende Javaprogramme
  - Eclipse Plug-in

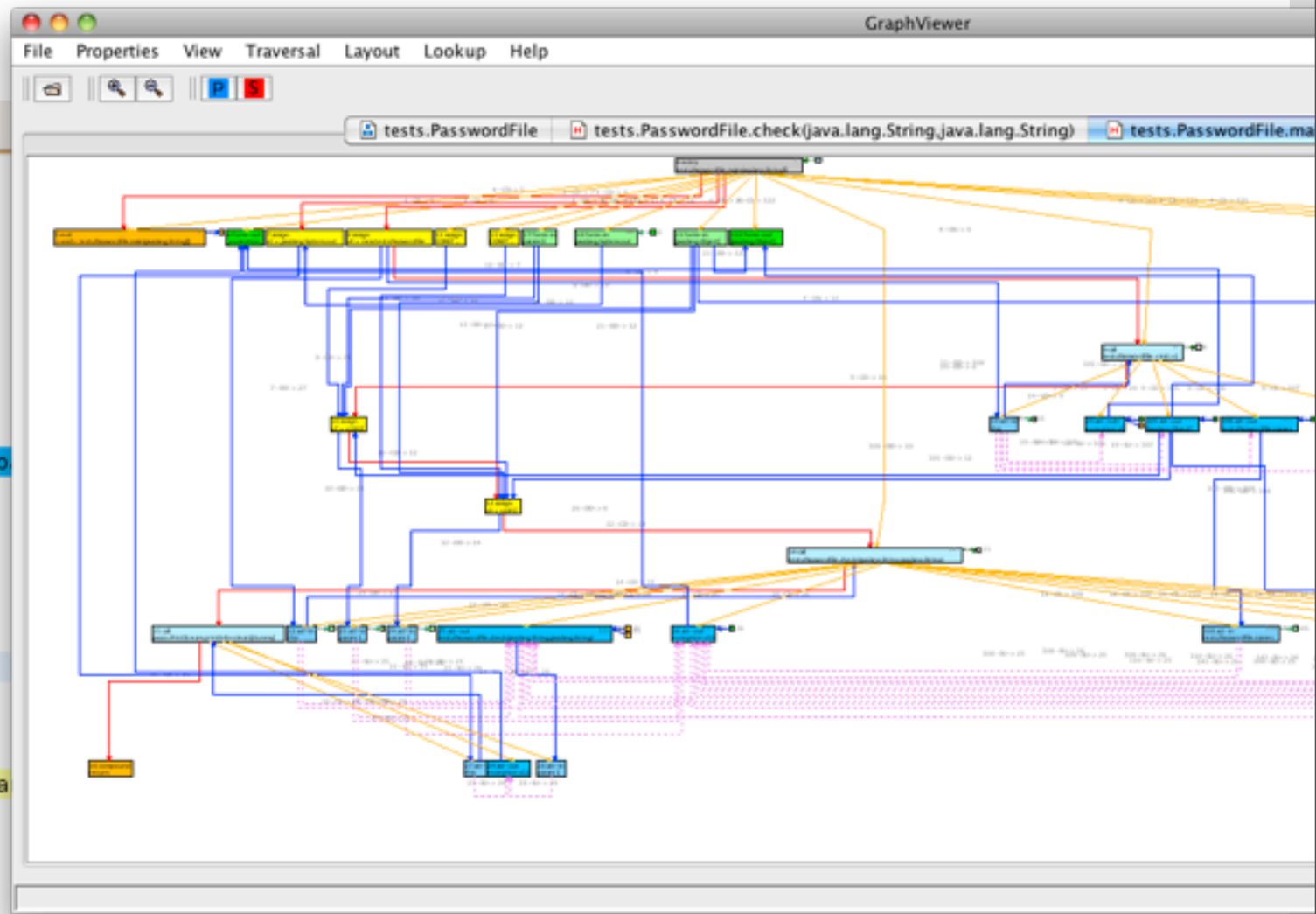
PasswordFile.java

```
package tests;

public class PasswordFile {
  private String[] names = { "A", "B"};
  private String[] passwords = { "x", "y"};

  public boolean check(String user, String password) {
    boolean match = false;
    try {
      for (int i=0; i<names.length; i++) {
        if (names[i].equals(user) && passwords[i].equals(p
          match = true;
          break;
        }
      }
    }
    catch (Throwable t) {};
    return match;
  }

  public static void main(String[] args) {
    System.out.println(new PasswordFile().check(args[0], a
  }
}
```



Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

```
if (password == secret) {
  print("login ok");
}
```

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

```
print(password.length);
```

```
if (password == secret) {
  print("login ok");
}
```

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

```
print(password.length);
```

```
print(password.charAt(0));
```

```
if (password == secret) {
  print("login ok");
}
```

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

```
print(password.length);
```

```
print(password.charAt(0));
```

```
for (i < password.length) {
  print(password.charAt(i));
}
```

```
if (password == secret) {
  print("login ok");
}
```

Nichtinterferenz und Deklassifikation

"*Das Programm ist sicher*" - Formale Bedeutung

- **Nichtinterferenz**
  - Keinerlei Beeinflussung
  - `h = l` verboten
  - Praxis: Zu strikt
- **Deklassifikation**: Ausnahmen
  - Wann ist `h = l` evtl. doch ok?
  - Was muss man beachten?
  - Wie kann man es umsetzen?

```
print(password);
```

```
print(password.length);
```

```
print(password.charAt(0));
```

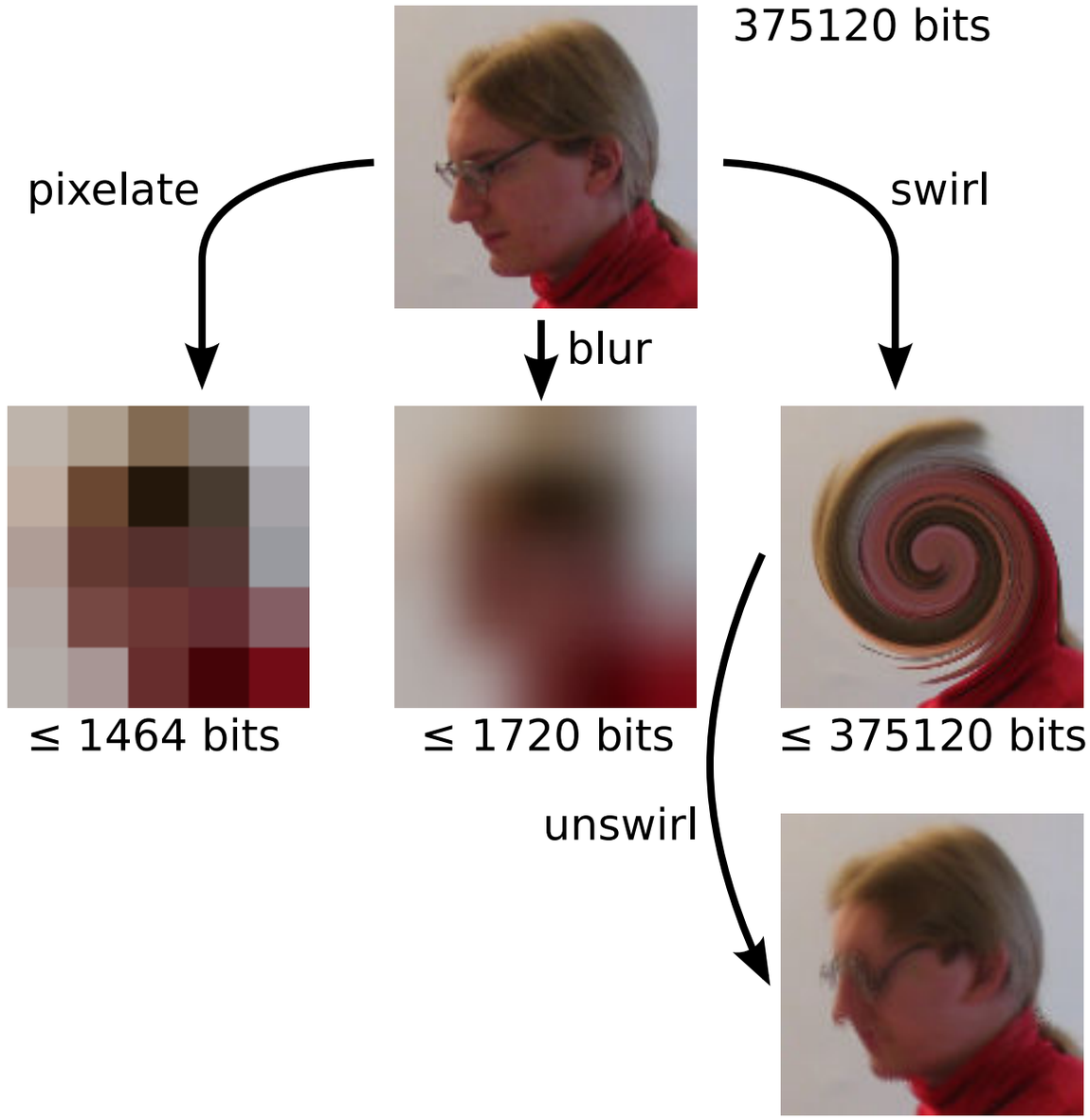
```
for (i < password.length) {
  print(password.charAt(i));
}
```

```
if (password == secret) {
  print("login ok");
}
```

```
for (i < password.length) {
  char ch = password.charAt(i);
  if (ch > 'a' && ch < 'A')
    print('x')
  else
    print('X')
}
```

Quantitatives IFC

- Wie misst man Information?
- Wann ist zuviel geflossen?
- Ansatz: Anzahl geflossener Bits.
- Beispiel: Filter für ImageMagick
- Grundlagen: Anderer Ansatz von David Clark



Taint-Analyse

- Information wird im laufenden Programm markiert
  - “Daten bei Adresse 0x16a4 werden auf öffentlichen Kanal gesendet“
  - “Adresse 0x14fed enthält geheime Daten“
- Keine Garatien vorab möglich
- Technik für grosse Programme
- Teilweise schon im Einsatz (Perl Taint mode)
- 2 Tools im Focus:
  - Taint Analyse für privilegierten Java-Bytecode
  - TaintCheck für Binärcode

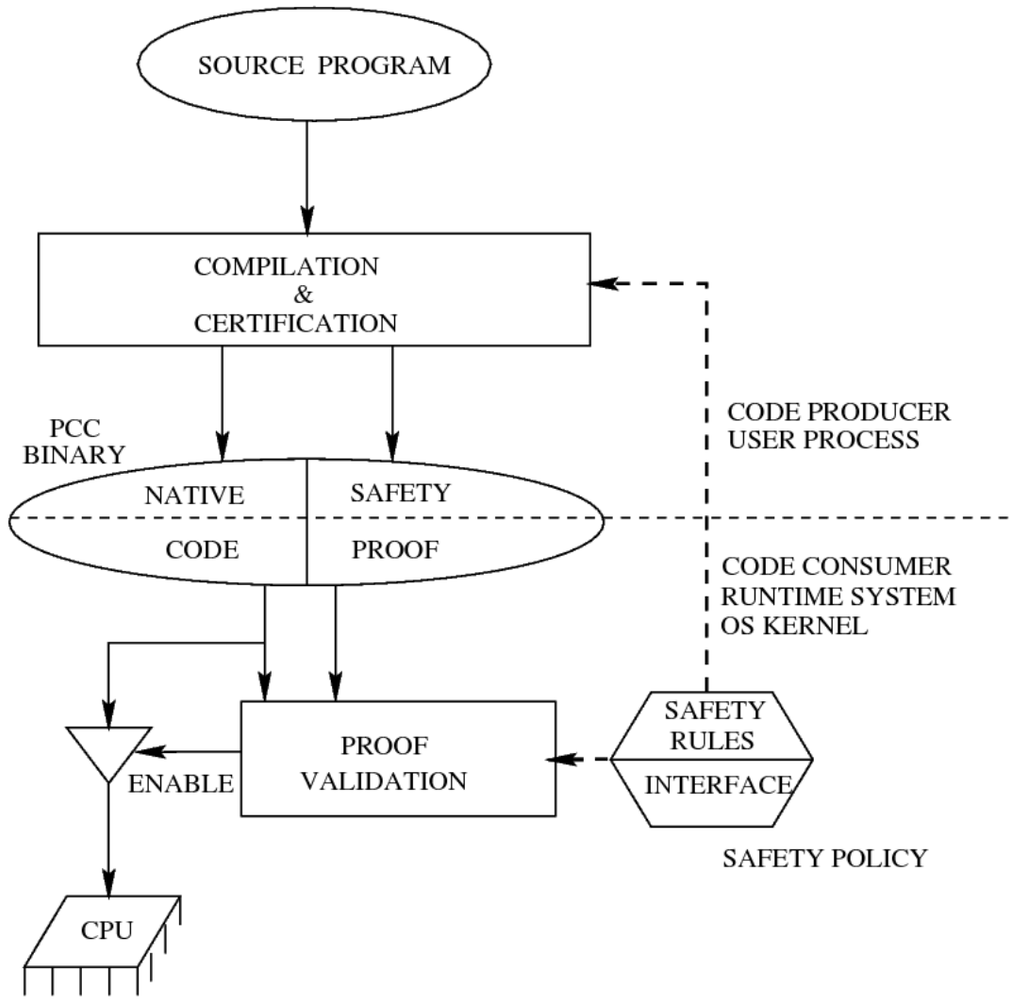
Proof-Carrying Code (PCC)

Sicherheitsanalyse: Suche nach Beweis für Sicherheit eines Programms

- Aufwändig, unmöglich auf eingebetteten/mobilen Systemen
  ⇒ Anbieter liefert Programm + **Sicherheitsbeweis**

  Anwender überprüft Beweis auf Korrektheit (weniger Aufwand!)

- **Anwendungsbeispiele**:
  - Sicherheit von eingebettetem Maschinencode
  - Informationsfluss Kontrolle
- **Herausforderungen**:
  - Übersetze Beweise von Quellcode-Ebene ⇒ Bytecode-Ebene



IFC mit abstrakter Interpretation

Konkrete Programmsemantik: Berechne Werte von Variablen

- Zustand: Variablenbelegung, z.B. (`a = 5, b = 10, x = 7`)

Abstrakte IFC-Semantik: Berechne Informationsfluss

- Zustand: möglicher Informationsfluss, z.B. {`x -> a`}

Programm P	Semantik		
`a = b`	$[\![P]\!](\mathtt{a}=5,\mathtt{b}=10,\mathtt{x}=7)$	$=$	$(\mathtt{a}=10,\mathtt{b}=10,\mathtt{x}=7)$
	$[\![P]\!]^{\alpha}(\{\mathtt{x} \rightsquigarrow \mathtt{b}\})$	$=$	$\{\mathtt{x} \rightsquigarrow \mathtt{b}, \mathtt{b} \rightsquigarrow \mathtt{a}, \mathtt{x} \rightsquigarrow \mathtt{a}\}$
`if (b==0) a = 42`	$[\![P]\!](\mathtt{a}=5,\mathtt{b}=10,\mathtt{x}=7)$	$=$	$(\mathtt{a}=5,\mathtt{b}=10,\mathtt{x}=7)$
	$[\![P]\!]^{\alpha}(\{\mathtt{x} \rightsquigarrow \mathtt{b}\})$	$=$	$\{\mathtt{x} \rightsquigarrow \mathtt{b}, \mathtt{b} \rightsquigarrow \mathtt{a}, \mathtt{x} \rightsquigarrow \mathtt{a}\}$

Sicherheitsanalysen für JavaScript

Problem: Dynamisches Laden fremden Codes

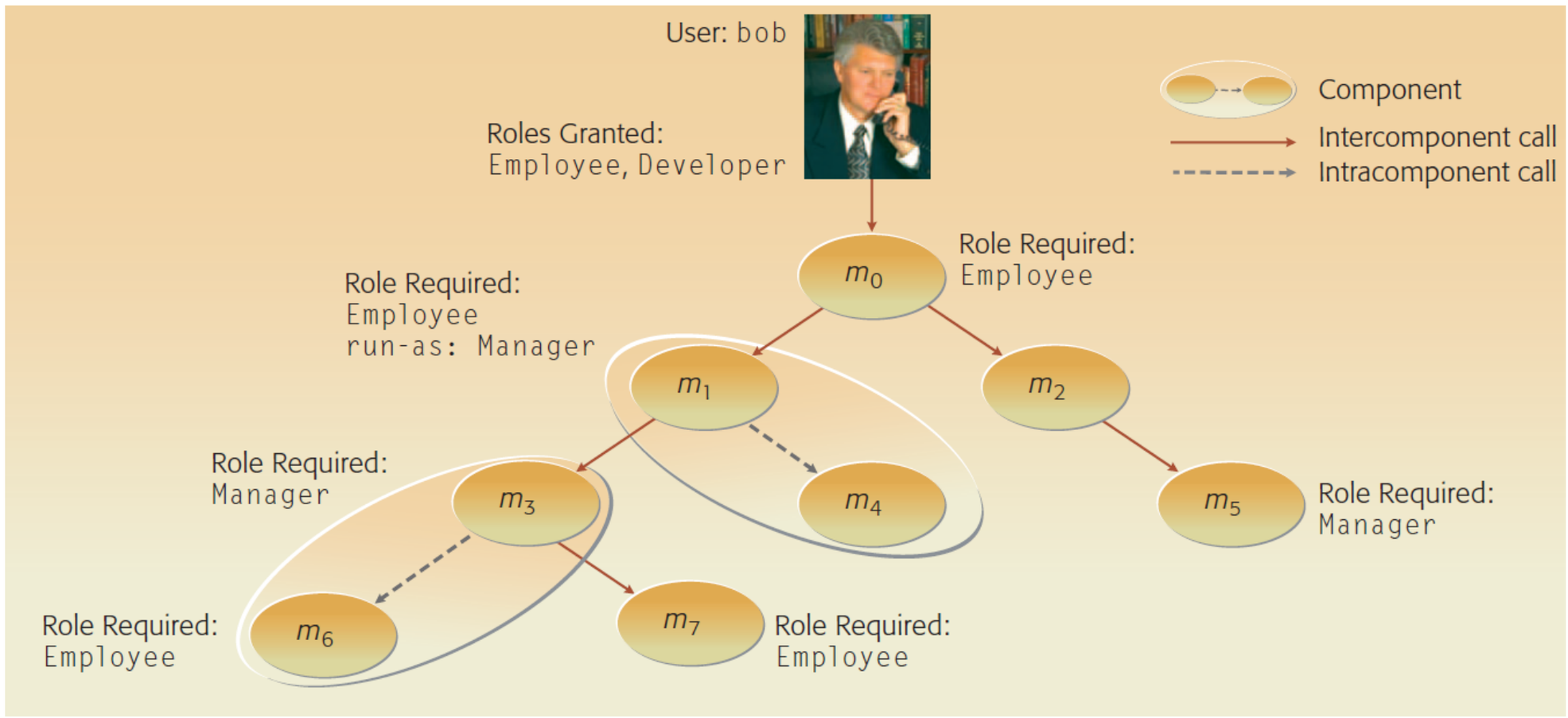
```
<script src="http://adnetwork.com/insert-ad.js">
<textbox id="SearchBox">
<button id="Search" onclick="doSearch()">
<script type="javascript">
var doSearch = function() {
  var searchBox = document.nodes.SearchBox.value;
  var searchStr = searchUrl + searchBox;
  document.location.set(searchStr);
}
</script>
```

- Modifiziert bösartiges insert-ad.js die Variable searchUrl?
- Code erst zur Laufzeit verfügbar
  ⇒ Informationsfluss-Analyse in mehreren Stages
- Aufwändige Analyse vorhandenen Codes anhand Security-Policies
- Ermöglicht effiziente Analyse dynamische nachgeladenen Codes

Zugriffskontrolle - Access Control

Role Based Access Control (RBAC)

- Zugriffssteuerung zur Laufzeit, anhand von ***Rollen***
- ***data***-based RBAC: Erlaube/Verbiete Zugriff auf Felder
- ***operation***-based RBAC: Erlaube/Verbiete Aufruf von Methoden
  Implementiert in JavaEE, .net, aber Policies oft komplex
- Herausforderung: operation-based Policy konsistent mit data-based?



Organisation

Themenvergabe & Termine

- Wer nimmt welches Thema?
- Wann machen wir die Vorträge?
  - Vorlesungszeit Ende: 16.07.2011

- 26.05.2011
- 09.06.2011
- 16.06.2011
- 30.06.2011
- 07.07.2011
- 14.07.2011

- Wo sind die Vorträge?
- Bei Nachfragen vorbeikommen, Email,...
  - Zur Sicherheit: Termin ausmachen
- Folien: 1 Woche vorher (graf@kit.edu, martin.hecker@kit.edu)
- Ausarbeitung: spätestens zum letzten Termin