
SE2006 Projekt

SecFlow – Automatische Ermittlung sicherheitskritischer Datenflüsse in Quellcode

Dr. Holger Peine

Fraunhofer IESE, Kaiserslautern

Software-Security



„We wouldn't need so much network security if we didn't have such bad software security“

- Bruce Schneier

(wahrscheinlich bekanntester IT-Sicherheitsexperte der Welt)

- Viren & Würmer: Sasser, Blaster, SQL-Slammer, ...
- Der Großteil der heutigen Sicherheitsschwachstellen in IT-Systemen beruht letztlich auf Programmierfehlern
 - Hauptfehler: Ungenügende Eingabedatenprüfung
- Firewalls, Virens Scanner etc. versuchen nur, vorhandene Löcher in der Software zu verdecken
 - unvollkommen; ständiger „Rüstungswettlauf“
- Besser: Sicherheitsschwachstellen schon während der Software-Entwicklung vermeiden
 - bessere Entwicklungsmethoden
 - Werkzeuge zur autom. Schwachstellenerkennung

Beispiel einer Sicherheitslücke durch fehlende Datenvalidierung

- Ein Java-Servlet zum Eintragen von Benutzer-Eingaben in eine Datenbank könnte folgenden Quelltext enthalten:

```
String email = req.getParameter( "email" );
```

```
String company = req.getParameter( "company" );
```

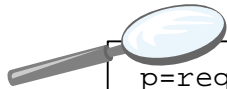
```
stmt.execute( "INSERT INTO customers ( email,  
company ) VALUES ( '" + email + "', '" + company +  
");" );
```

- Alles bestens bei email-Argumenten wie *mike@some.com* :
`INSERT ... VALUES('mike@some.com', 'Mike Inc.')`
- Was passiert, wenn ein Angreifer folgendes als email angibt:
`x', 'x'); DELETE FROM customers ... --`
- `INSERT ... VALUES('x', 'x'); DELETE FROM.. -- ');`
... löscht Kundeneinträge aus der Datenbank ☹



Folie 3/9

Ziel: Unvalidierte Datenflüsse erkennen



```
p=req.getParameter();  
query=buildQuery(p);  
stmt.execute(query);
```

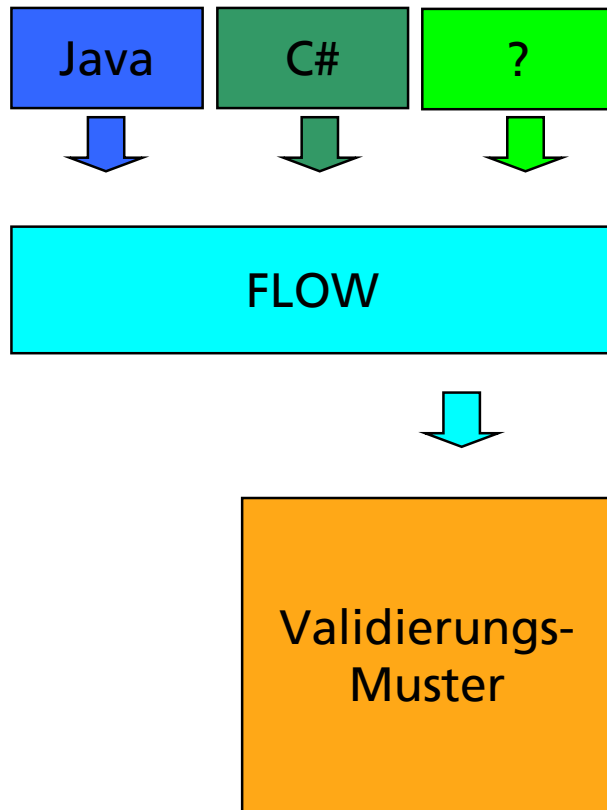
*nur heuristisch
erkennbar*

*schwierig, aber
machbar*

- Größte Klasse von Software-Sicherheitschwachstellen
 - SQL Injection, Cross-Site-Scripting, Command Injection, ...
- Finde alle Daten, die ...
 - aus nicht vertrauenswürdiger Quelle stammen (z.B. aus dem Internet)
 - ohne Prüfung durch die Variablen des Programms fließen (auch in Form von Verarbeitung durch Funktionen etc.)
 - an ein verwundbares externes System weitergereicht werden (z.B. eine Datenbank, Shell, ...)
- Für verschiedene Quellsprachen: Java, C#, ...(?)

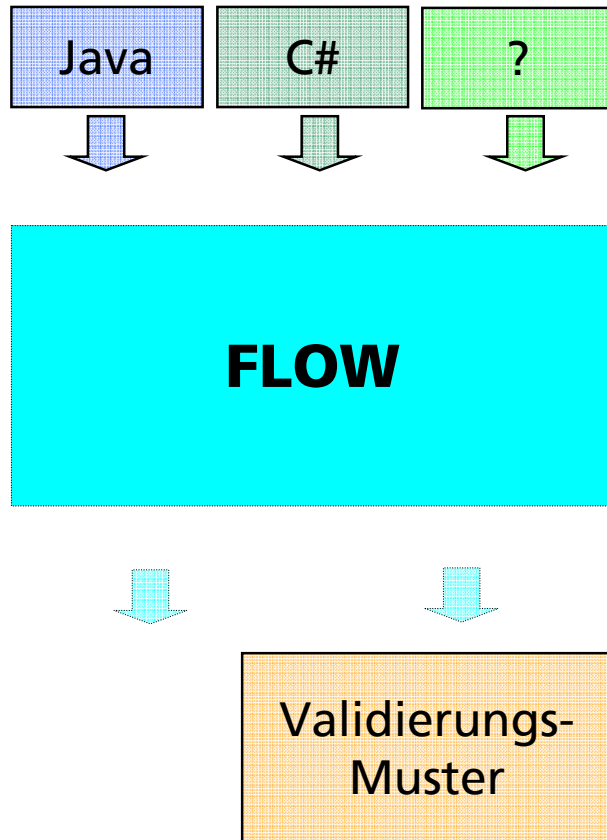
„nur“ Fleißarbeit

Werkzeug-Architektur



- Syntax parsen, Kontrollflussgraph generieren
 - quellsprachenabhängig
 - für verschiedene Sprachen
 - bis zum Zwischencode-Niveau (z.B. while-Programme mit SSA-Eigenschaft)
- Analyse
 - datenfluss- u. kontrollfluss-spezifisch (konservativ)
 - auf einer Inferenzmaschine ausgeführt
- Erkennung von Datenvalidierungen durch
 - Heuristiken zur Erkennung lokaler Struktur-Muster
 - musterunabhängige Analyse (globale Eigensch.)

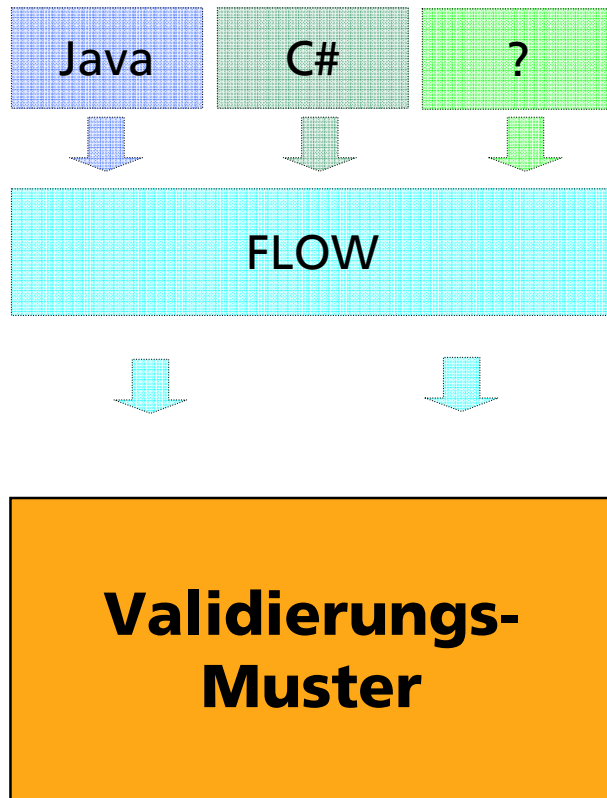
Programmbeschreibungssprache FLOW



- deklarative, logische Sprache (noch zu definieren)
- relationale Repräsentation des Kontrollflussgraphen und der Datenflussabhängigkeiten, z.B.
 - `call(f, g, Core.java:47)`
`assign(x, v, Core.java:112)`
- enthält auch Informationen über
 - statischen Prozeduraufrufstack
 - Daten-Constraints aus `if/while`-Bedingungen
- schwierig zu erzeugen, aber gut zu analysieren, z.B.
 - `redefinition(X, P, C) :-`
`call(_, P, C2), isDefined(X, C2),`
`assign(X, _, C).`

Folie 6/9

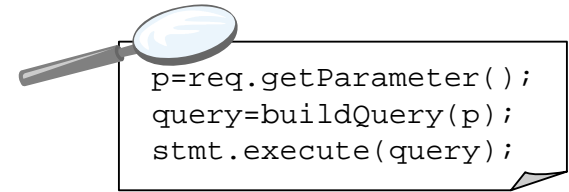
Heuristische Analyse der Programminformationen



- Strukturelle Heuristiken
 - Muster im Daten-/Kontrollfluss als Indizien für Sicherheitsschwachstellen bzw. Datenvalidierungen
 - `if (x > MAX) {return OUT_OF_RANGE;}`
 - `if (!someTest(x)) {
 x = someTransformation(x);}`
 - Muster werden in FLOW formuliert
- Heuristiken mehr oder weniger spezifisch
 - allgemein
 - plattformspezifisch (z.B. J2EE, .NET)
 - anwendungsspezifisch
- Information aus vielen Heuristiken kombinieren

Rahmendaten des Projekts SecFlow

- **Dauer:** 2,5 Jahre (06/2006 bis 11/2008)
- **Budget:** ca. 1750 K€ / 14 Personenjahre Gesamtaufwand
- **Partner:**
 - CC GmbH (Wiesbaden)
 - Implementierung, Koordinator
 - Fraunhofer IESE (Kaiserslautern)
 - Konzepte (Architektur, Konfig.-Sprache, Heuristiken), Java/J2EE-Arbeiten
 - ICT AG (Trier)
 - Java / J2EE-Arbeiten, empirische Erprobung
 - SHE AG (Ludwigshafen)
 - C# / .NET-Arbeiten , empirische Erprobung



Angestrebte Fähigkeiten des SecFlow-Werkzeugs



- Finden eines erheblichen Teils (50%?) der unsicheren Datenflüsse in Java, JSP, J2EE, C#, ASP.NET, .NET Programmen
 - Darunter möglichst selten (<50%) „falscher Alarm“
- Grafische Ausgabe als Pfad durch den Quelltext
- Konfigurierbar für neue Programmierplattformen und anwendungsspezifische Konventionen
- Vorkonfiguriert auch mit nur oberflächlichem Security-Wissen produktiv nutzbar
- Klare Schnittstelle zur Erweiterung um weitere Quellsprachen

Ergebnis: Software mit weniger Sicherheitslücken zu geringeren Kosten

Folie 9/9