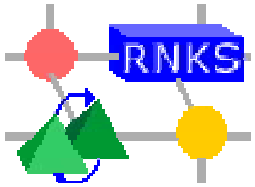


Brandenburgische Technische Universität Cottbus

Lehrstuhl Rechnernetze und Kommunikationssysteme

Validierung von Signaturen
–
Testmethoden zur Feststellung von Korrektheit und Präzision

Sebastian Schmerl

Brandenburg Technische Universität Cottbus

sbs@informatik.tu-cottbus.de

Überblick

- Motivation
- Signaturtests
- Strategien zum Test von komplexen, mehrschrittigen Signaturen
  - Signaturbeschreibung mit EDL
- Zusammenfassung

Motivation

- Signaturanalyse liefert per Definition keine Fehlalarme
  - Als Signatur beschriebene Muster werden erkannt
- Zu viele Alarme
  - 3 Alarme pro Minute
  - >99% Fehlalarme
- Ungeeignete Signaturen
- Keine systematische Vorgehensweise in der Signaturentwicklung

Motivation (2)

- Entwicklungsprozess einer Signatur für einen Angriff
  1. Ausführung und Aufzeichnung des Angriffs
  2. Identifizierung der wesentlichen Ereignisse in den Spuren
  3. Schrittweise Modellierung der neuen Signatur
     - Mit Hinblick auf die Angriffsstrategie
  4. Testphase, Überprüfung der Korrektheit und der Präzision

- Es existieren keine systematischen Ansätze zum Testen von Signaturen
  - Signaturen werden (wenn überhaupt) erst im realen Einsatz getestet

Elemente der Signaturanalyse

Audit-Trail

open, “...”, 12, 200, ...;
write, 2, 12, 200, ...;
open, “secret.txt”, 10, 100, ...;
read, 2, 12, 200, ...;
fork, 12, 200, 201, ...;
exec, 3, 12, 201, ...;
open, “...”, 10, 400, ...;
write, 4, 10, 400, ...;
read, 4, 10, 400, ...;
fork, 10, 400, 401, ...;
read, 1, 10, 100, ...;
exec, 5, 10, 401, ...;
…

- Welche Signatur identifiziert die Manifestierung dieser Attacke?
  - Welche Ereignisse sind relevant?
  - Welche Muster gibt es?
  - Charakteristische Zusammenhänge?
  - Typische Wertebelegungen?

Modellierungsfehler

Signaturtest

- Signaturtests sollen Spezifikationsfehler identifizieren
- *Ziel:* die ideale Signatur

- Ideale Signatur beschreibt genau die Menge M_I aller Manifestationen eines Angriffs
  - Beschreibt die Invariante eines Angriffs

- *Überspezifizierte* Signatur
  - beschreibt nicht alle Ausprägungen einer Attacke
  - $M_{Ü} \subset M_I$
- *Unterspezifzierte* Signatur
  - beschreibt auch Aktionsfolgen die legitimen Nutzer- bzw. Interaktionsverhalten entsprechen
  - $M_I \subset M_U$

Teststrategien

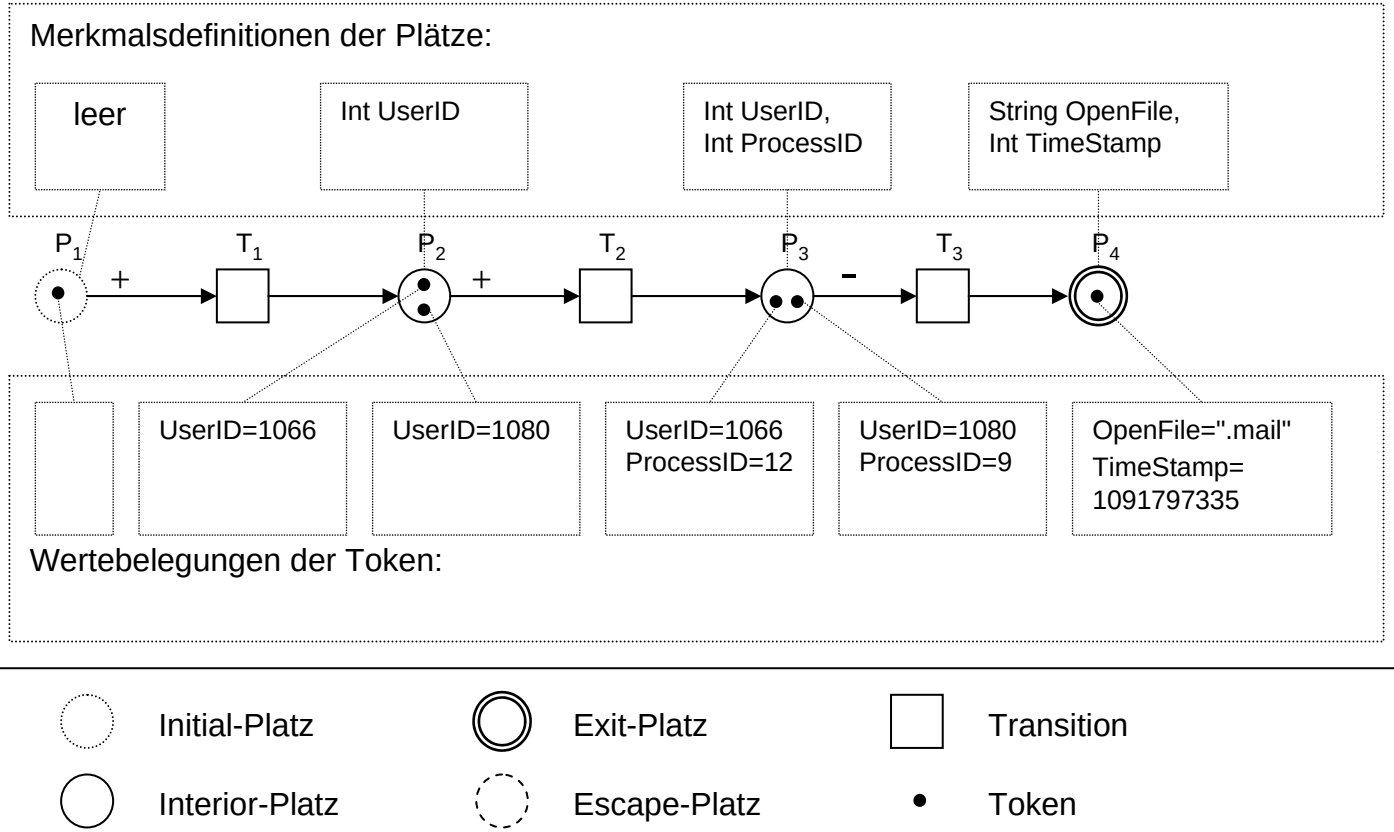
- Teststrategie 1: Grobe Ableitungsfehler
- Teststrategie 2: Pfadüberdeckung
- Teststrategie 3: Berücksichtigung von isomorphen Aktionsfolgen
- Teststrategie 4: Abbruchereignisse
- Teststrategie 5: Backward Slicing und Programm-Pfade

- Strategien 1-4 setzen auf komplexen mehrschrittigen Signaturen auf
- Alle Teststrategien haben Grenzen bzw. Probleme

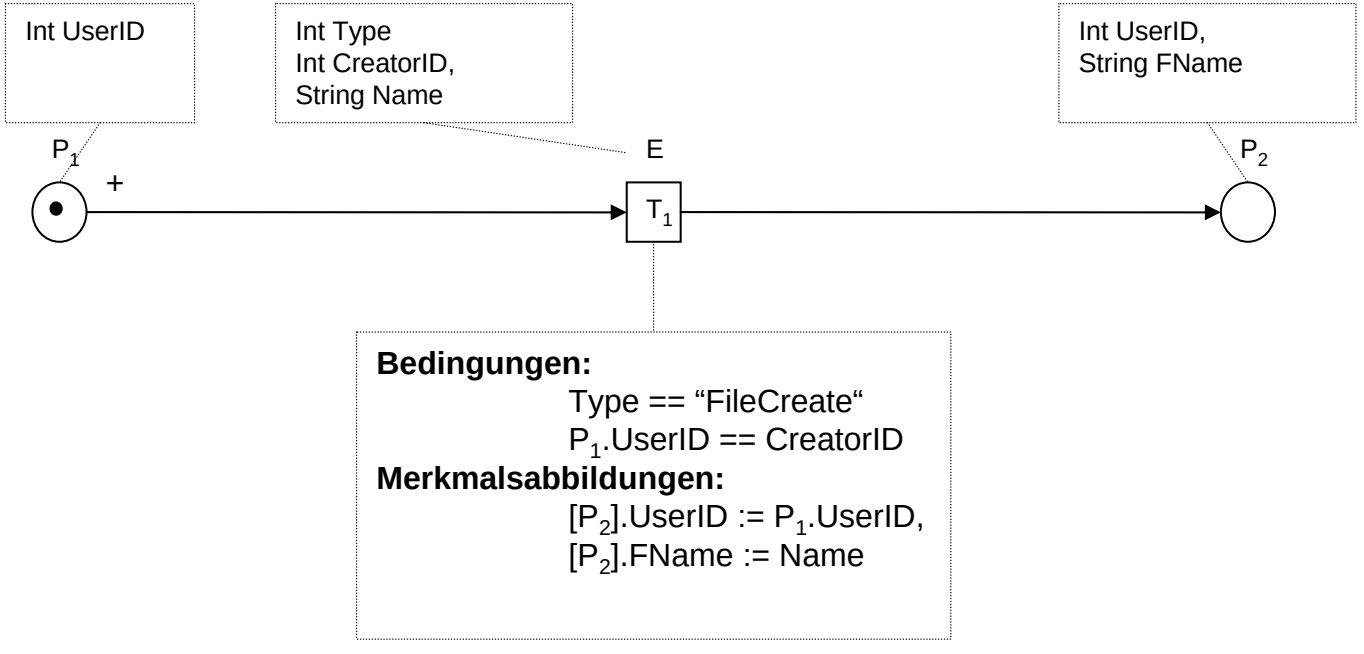
Beschreibungskonstrukte in EDL

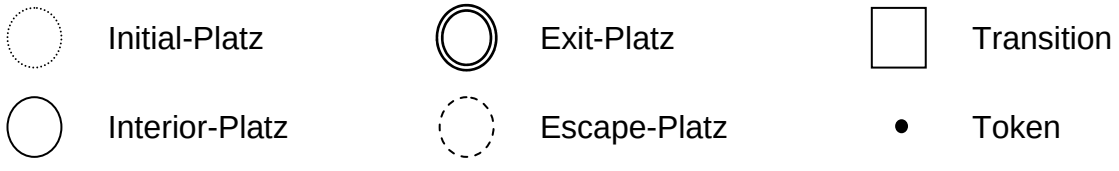
- Beschreibung von komplexen mehrschrittigen Angriffen
- Erfüllt semantischen Anforderungen einer Signatursprache [Meier04]
- Angelehnt an gefärbte Petri-Netze
- Hauptkonstrukte:
  - Platz:
    - Repräsentiert einen Systemzustand (Schnappschuß)
  - Transition:
    - Beschreibt ein Zustandübergang
  - Ereignis:
    - Charakterisiert eine sicherheitsrelevante Aktion
  - Token:
    - Kennzeichnet eine Signaturinstanz

Plätze und Token



Transitionen





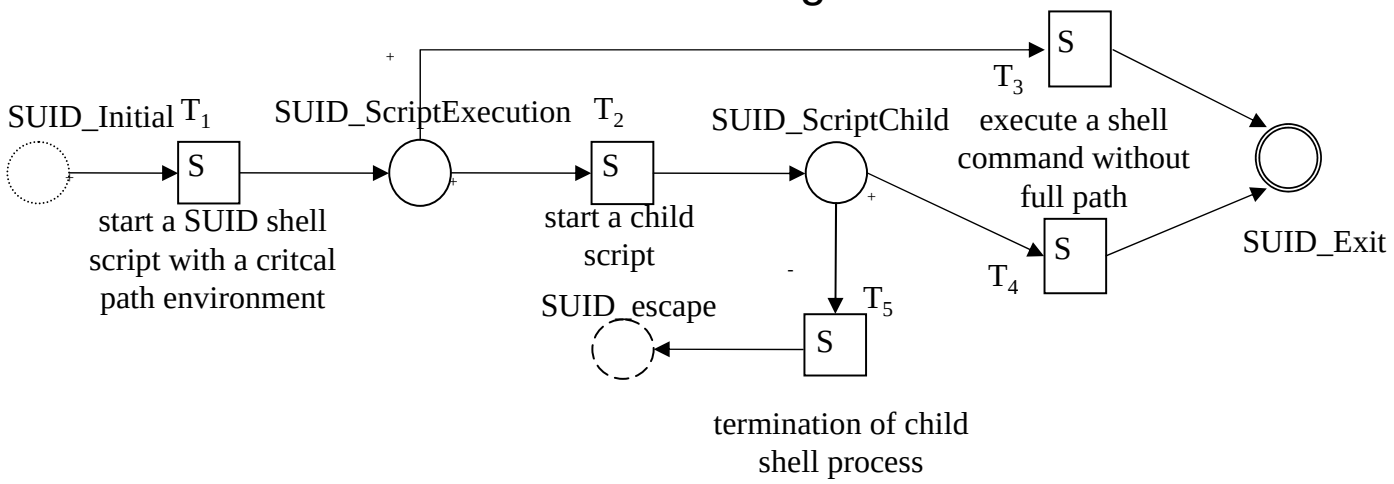
Strategie 1/5

- Erkennung grober Ableitungsfehler
  - Testdaten:
    - Bereinigte Audit-Trail der aufgezeichneten Exploitausführung
    - Gesamte Audit-Trail der aufgezeichneten Exploitausführung
  - Weiteres Anhaltskriterium ist die Anzahl der Signaturinstanzen

- Grenzen
  - Nur die Erkennung des aufgezeichneten Exploits wird sichergestellt

Strategie 2/5

- Ableitung von Testdaten aus der Signatur
  - Sensor realisiert eine Funktion *δ*
    - Abbildung sicherheitsrelevanten Aktionen in Audit-Daten (Spuren) bzw. Ereignisse
  - Umkehrfunktion *δ^{-1} notwendig*
  - Test ob die Verwundbarkeit ausgenutzt wird



Strategie 2/5 - Grenzen

- Grenzen bzw. Probleme:
  - Unterteilung der Signaturen bzgl. der Schwachstellenausnutzung
    - Deterministisch erfolgreiche Attacken
    - Attacken mit Race-Condition
    - Anomalie/Brute Force-Attacken
  - Umkehrfunktion δ^{-1} nicht immer möglich
  - Nicht jede Aktion hinterlässt Spuren → Signatur vernachlässig diese
  - Feststellen der Ausnutzung einer Verwundbarkeit
    - Instrumentierung des Exploits
    - Instrumentierung der Schwachstelle
    - Instrumentierung des gesamten Systems
    - Einsatz eines „Orakels“

Strategie 3/5

- Isomorphe Aktionsfolgen
  - semantisch gleichwertige Aktionsfolgen die ebenfalls zur Ausnutzung der Verwundbarkeit führen
  - eigentliche Angriffstrategie bleibt erhalten
  - Ableitung von isomorphen Aktionen bzw. Aktionsfolgen aus der Signatur (oder dem Exploit)
    - z.B. mv() und cp()+del()
- Grenzen bzw. Probleme:
  - Quertest mit Strategie 2 notwendig

Strategie 4/5

- Wird die Attackenverfolgung abgebrochen, wenn die Verwundbarkeit nicht mehr ausgenutzt werden kann?
  - Abbruchereignisse
  - konträre Ereignisse
    - z.B. fork() und exit()

- Grenzen:
  - Meist nur auf Skript- und Shell-Attacken anwendbar
  - Konträre Ereignisse müssen nicht zum Attackenabbruch führen

Strategie 5/5

- Program Slicing
  - Quellcode-Analyse zur Identifikation:
    - welche Anweisungen eines Programms eine bestimmte Anweisung in einem bestimmten Programmpunkt beeinflussen
  - Backward Slicing

- Programm-Pfade zwischen zwei Programmpunkten
  - Liegen auf den Pfaden Anweisungen, die Spuren verursachen?
  - Werden diese Anweisungen von der Attacke beeinflusst?

Strategie 5/5 - Beispiel

```
main()
{   …
      red = fetch();
      blue = fetch();
      green = fetch();
      yelllow = fetch();
      red = 2*red;
      sweet = red*green;
      sour = 0;
      for (i = 0; i < red; i++)
            sour += green;
      salty = blue + yellow;
      green = green + 1;
      bitter = yellow + green;
      printf ("%d %d %d %d\n", sweet, sour, salty, bitter);
…
}
```

Strategie 5/5 - Grenzen

- Grenzen und Probleme:
  - Quellcode notwendig
  - Verwundbarkeit muss bekannt sein
  - Für große Projekte meist nicht durchführbar
  - Pointerarithmetik
    - möglicherweise Wechsel zum dynamischen Slicing
- Diese Strategie ist eher eine Signaturableitungsstrategie

Zusammenfassung und Ausblick

- Signaturtest-Strategien sind stark abhängig von der Verwundbarkeit, den Signaturen und den Sensoren …
- Es existieren Querbezüge zur Softwaretechnik
  - Pfadüberdeckung und Grenzwerttests
  - Extreme Programing
    - Ableitung von Testfällen während der Spezifikation?
- Evaluierung von Teststrategien ist aufwendig
  - Meist nur über Fallstudien
- Ziel ist die automatisierte oder wenigsten halbautomatische Ableitung von Testfällen durch diese Strategien

Ende

Fragen, Bemerkungen?