Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Fuzzing

Phillip Raffeck, Tim Rheinfels, Simon Schuster, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
https://sys.cs.fau.de

Wintersemester 2022

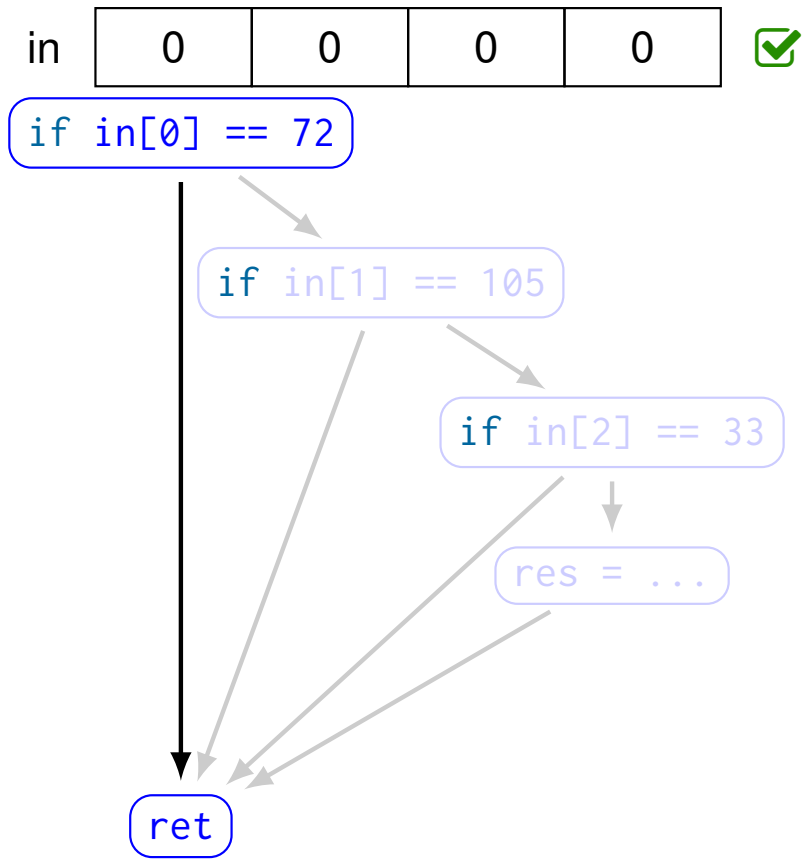
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



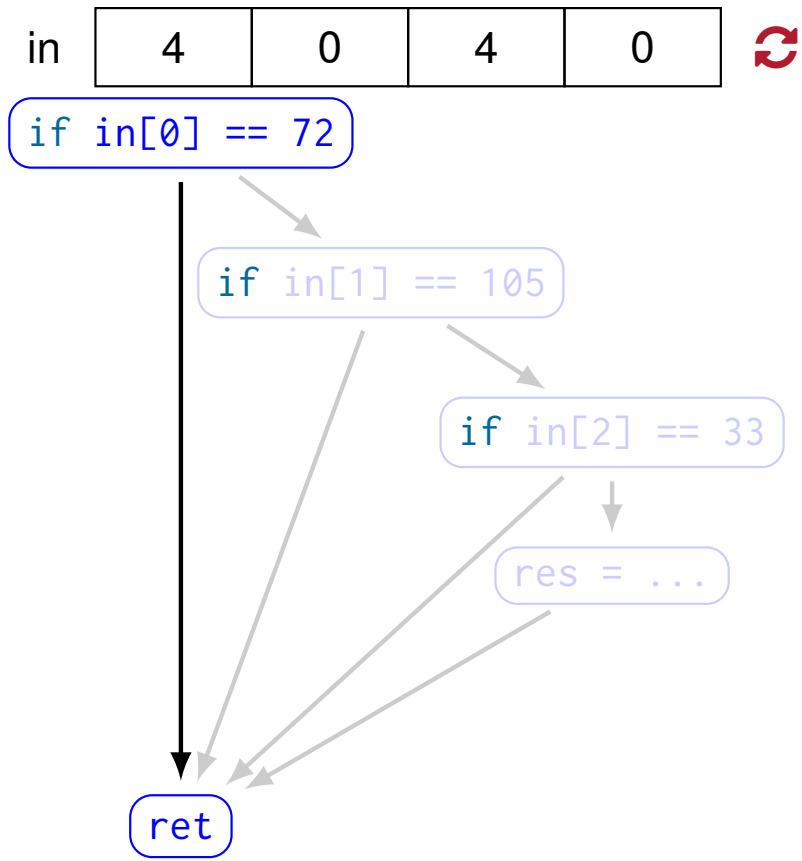
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



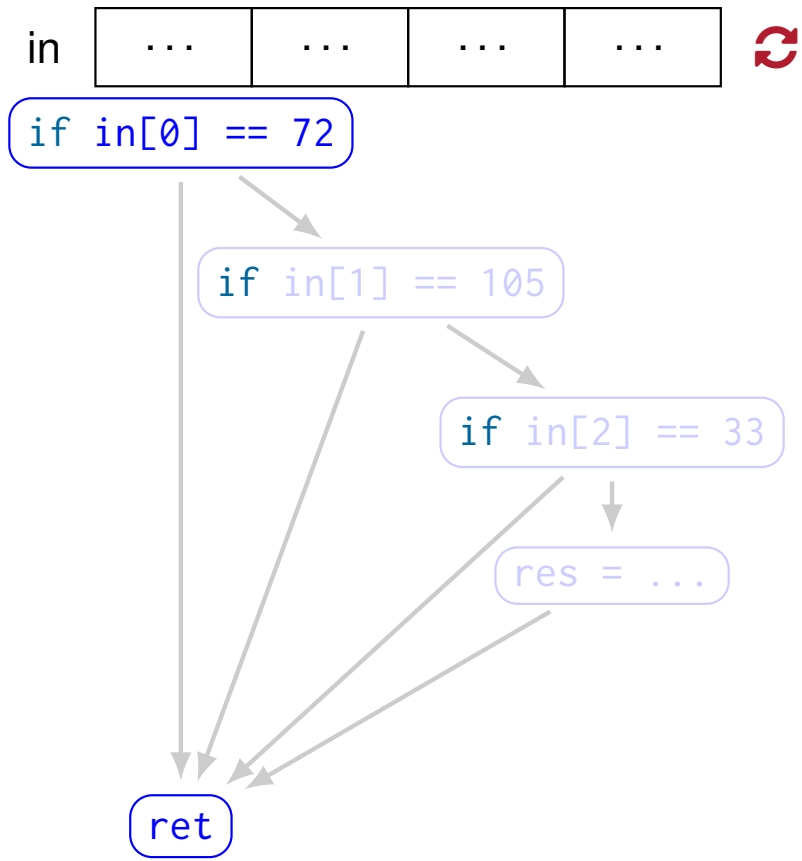
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



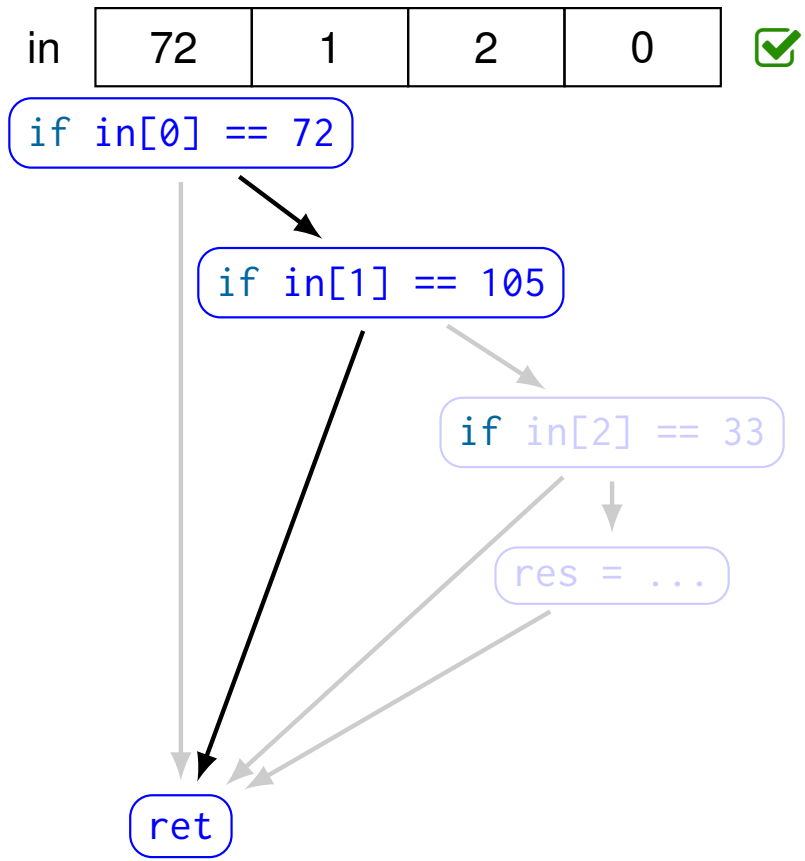
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



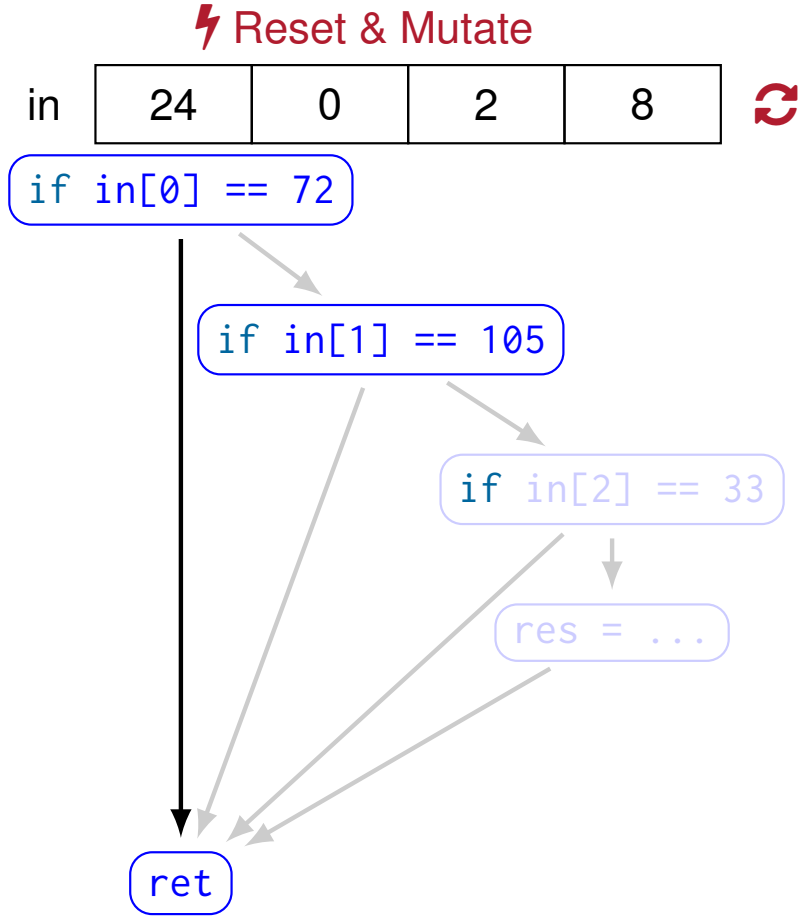
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen: möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



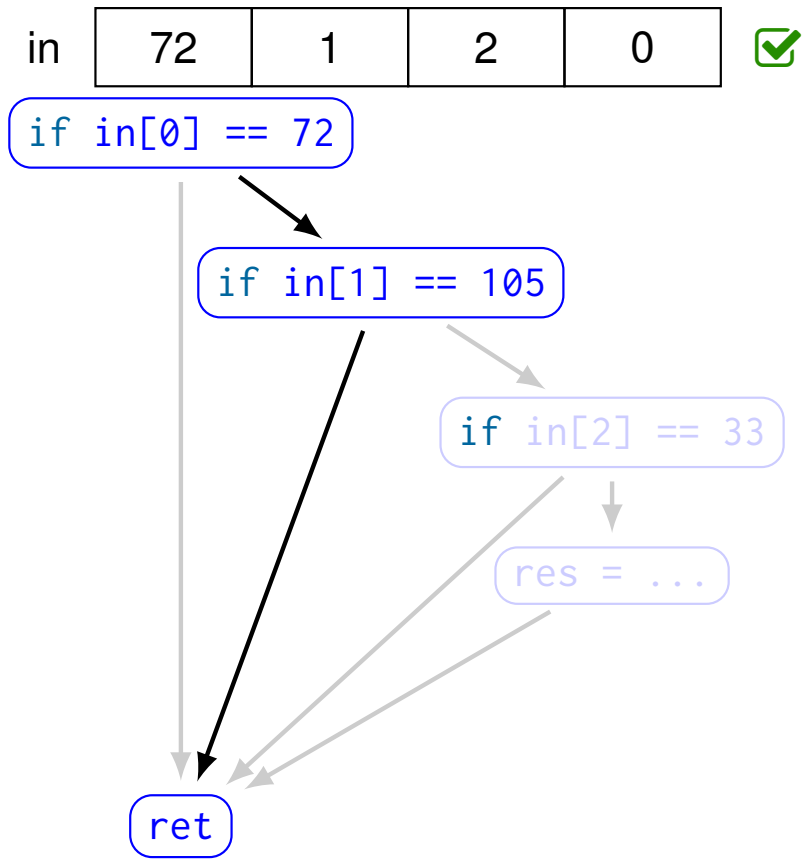
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



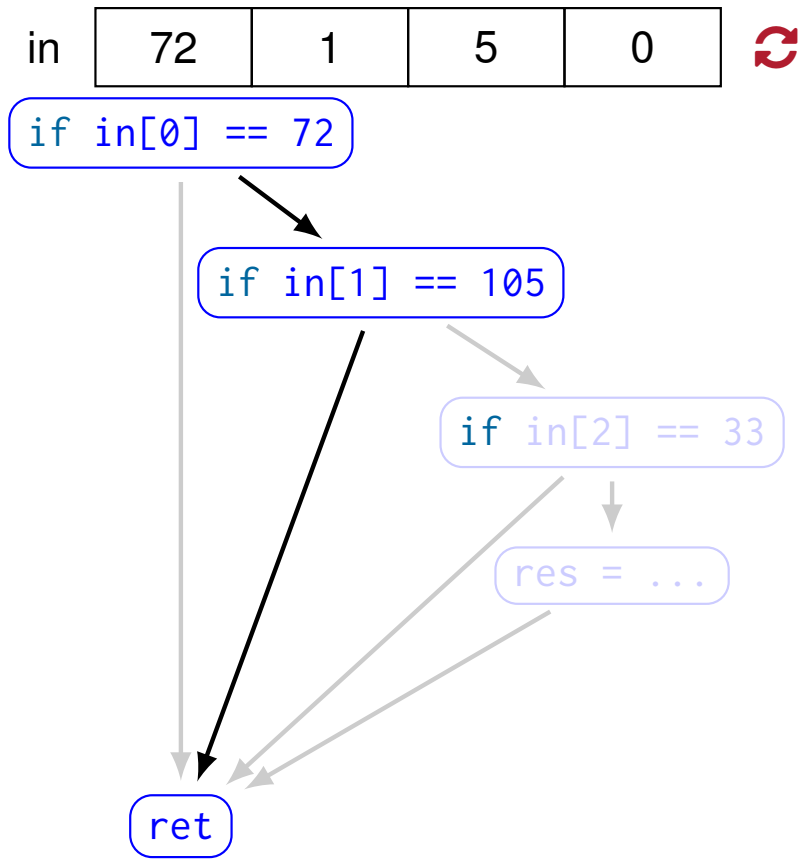
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



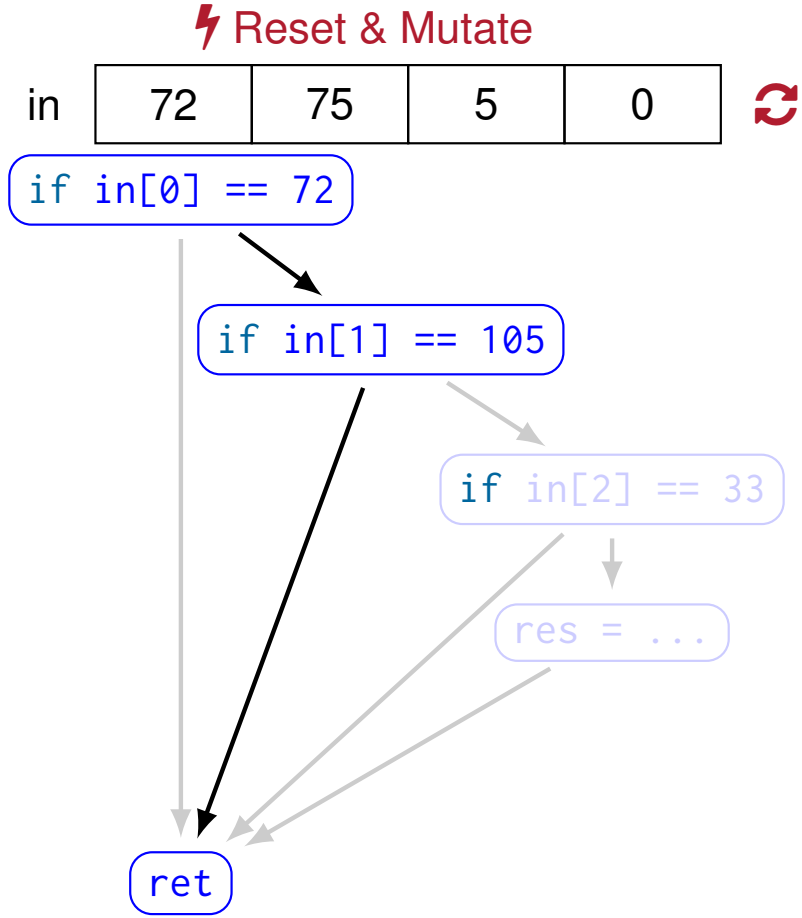
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen: möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



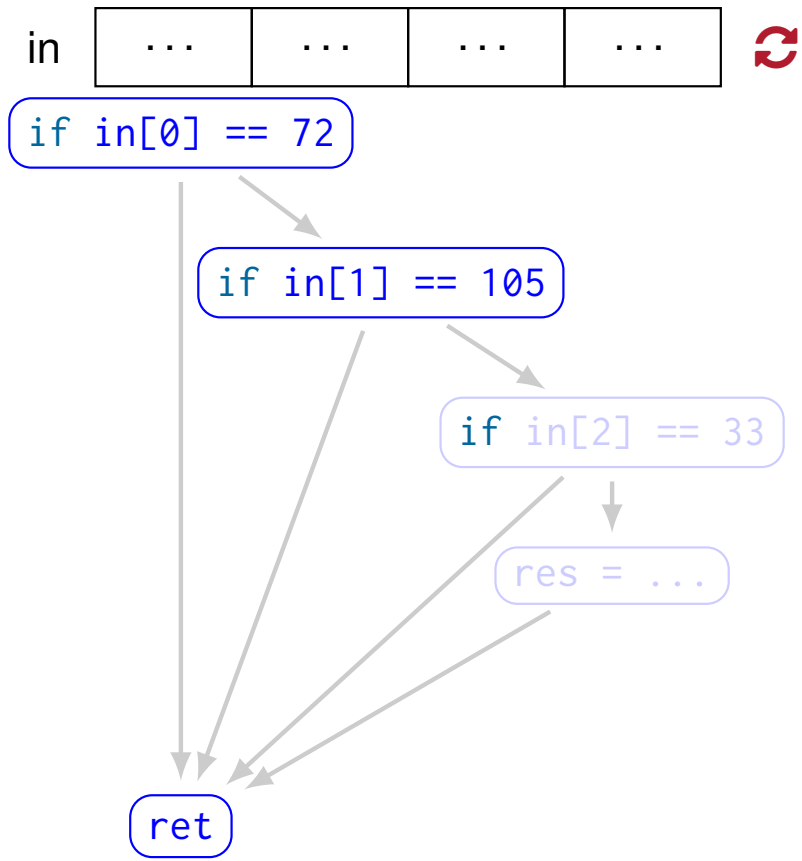
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



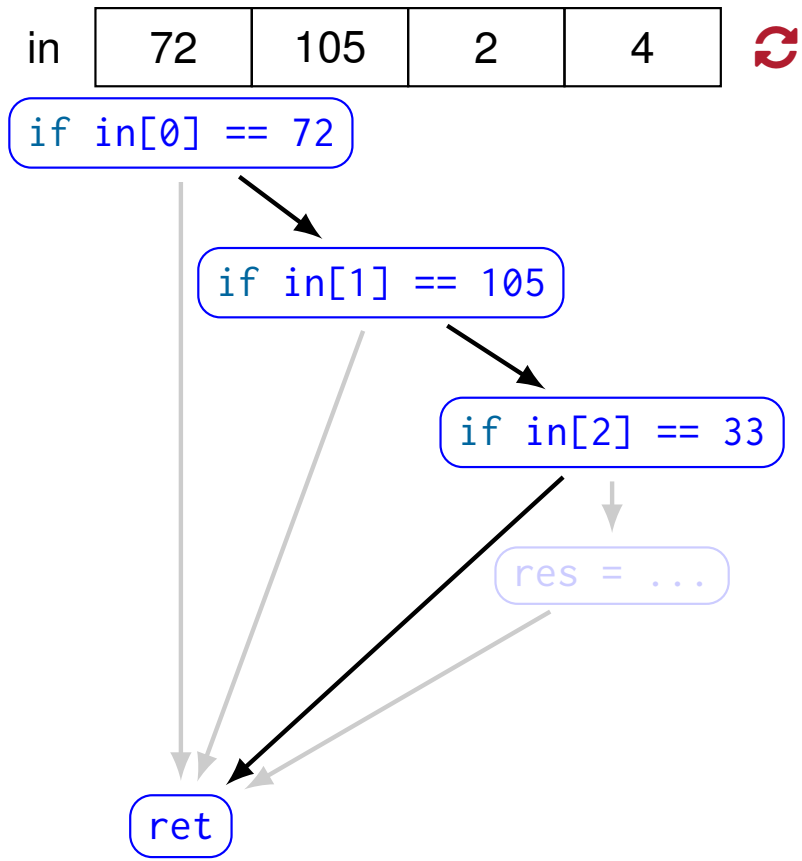
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



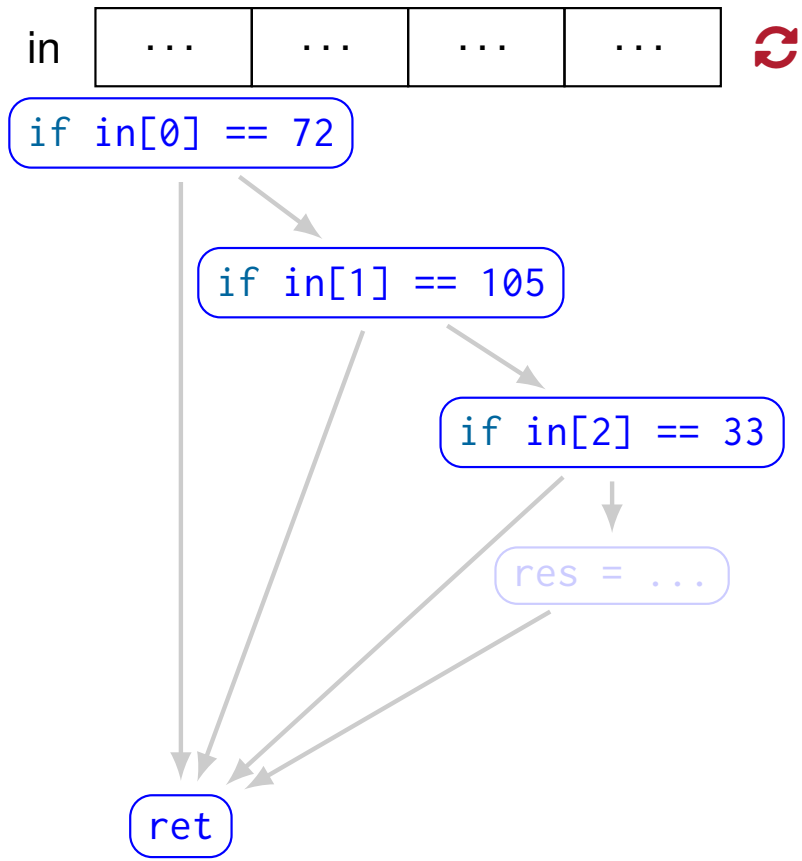
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



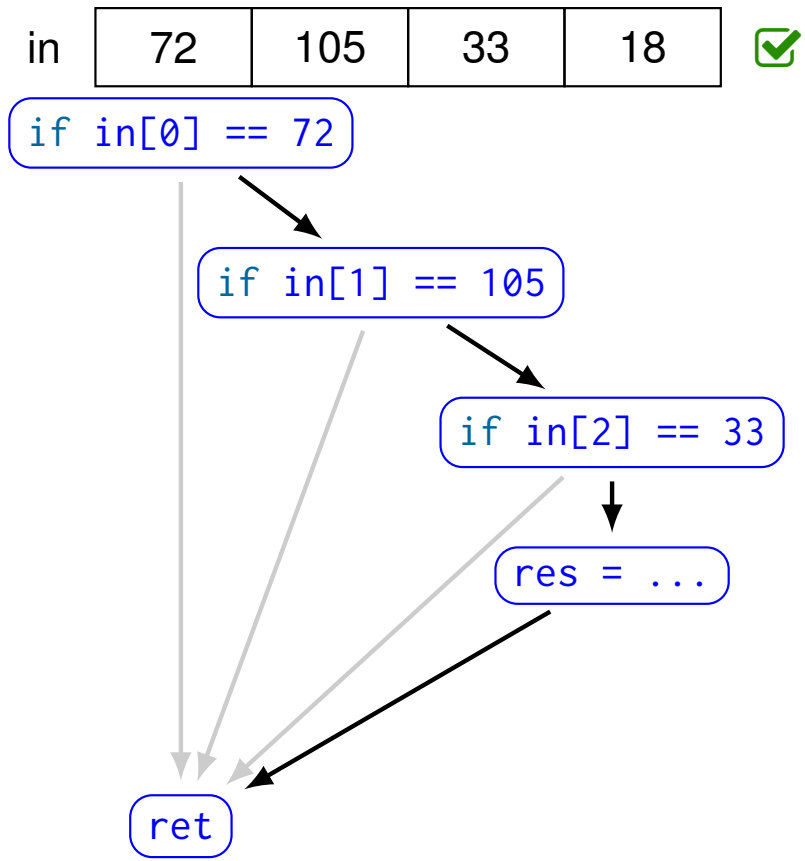
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



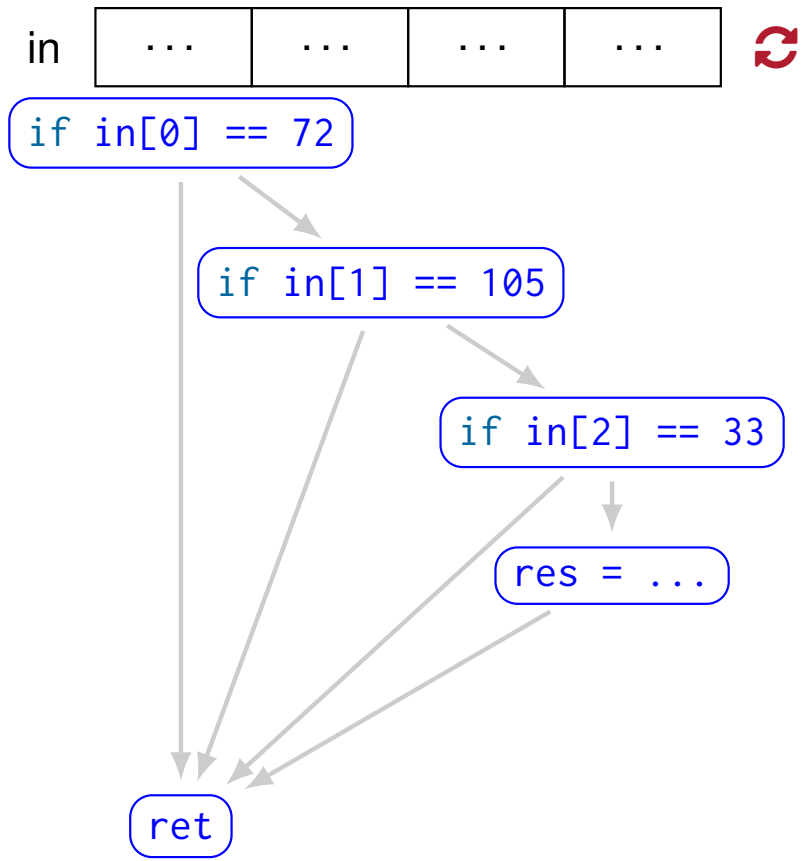
Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```



Überdeckungsgesteuertes Wuscheln (engl. *Fuzzing*)

- Raten möglicher Eingaben
- Strukturiertes Vorgehen:
  möglichst alle Blöcke gleichmäßig abdecken

  ⇝ Aus Modifikation von Eingaben die zum Vorgänger führten

  ⇝ *Mutation*: Zufällige Veränderung
  Crossover: Kombination existierender Eingaben
- Beispiel:

```
if (size < 4) return 0;
if(in[0] == 72)
  if(in[1] == 105)
    if(in[2] == 33)
      res = 42 / in[3];
return res;
```