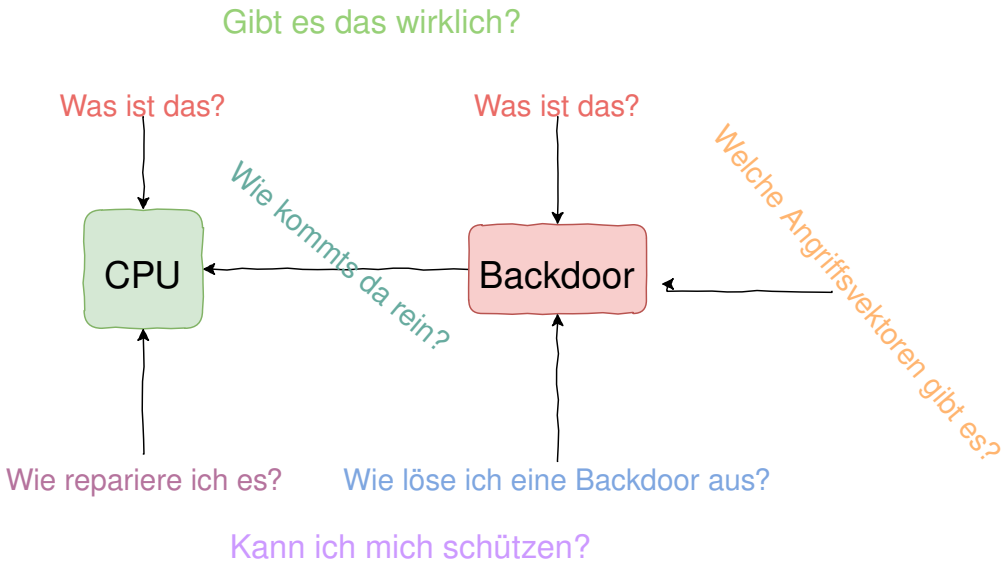
CPU Backdoors

because sometimes its becoming real

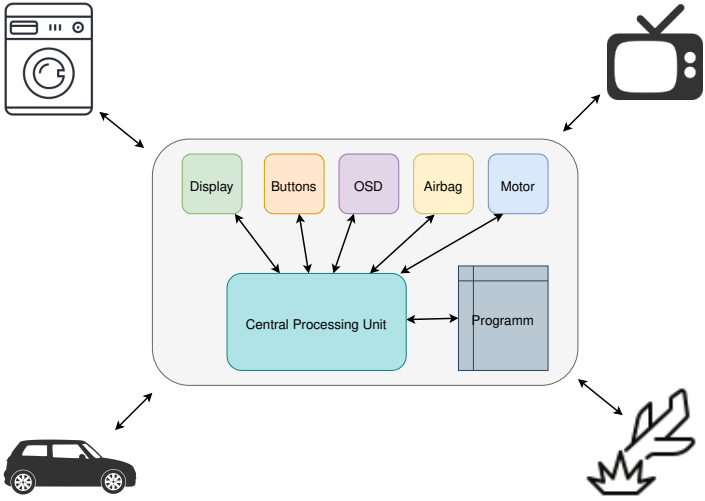
Dominik Meyer (byterazor)

Geraffel

26. März 2021


Gibt es das wirklich?
Was ist das?
Was ist das?
Welche Angriffsvektoren gibt es?
Wie kommts da rein?
CPU
Backdoor
Wie repariere ich es?
Wie löse ich eine Backdoor aus?
Kann ich mich schützen?

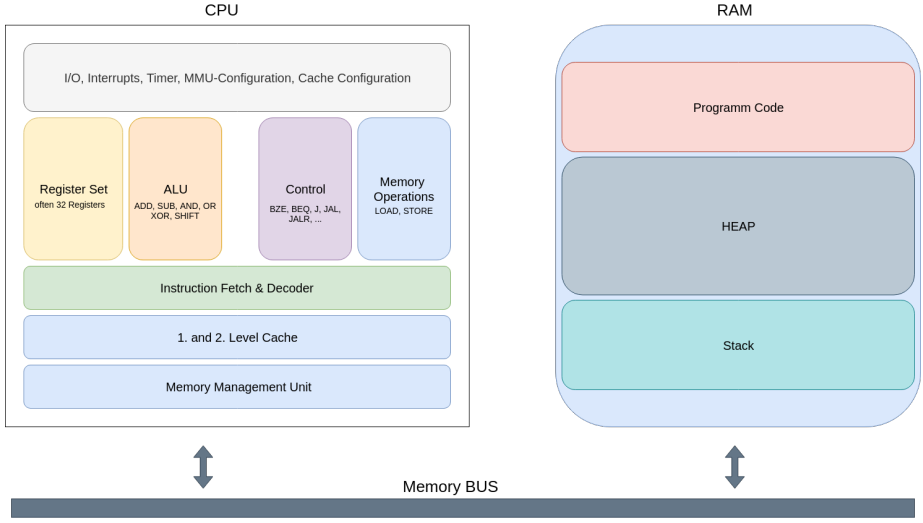
Was ist eine CPU?



Was ist eine CPU?

- benötigt man immer für General Purpose Computing (GPC)
- Program legt fest was die CPU genau machen soll
- CPU führt die Instruktionen des Programms aus (oft sequentiell)
- CPU stellt die Hardware zur Ausführung bereit

Was ist eine CPU?



Was ist eine Backdoor?

Backdoor bezeichnet einen Teil eines Systems, der es ermöglicht, unter Umgehung der normalen Zugriffssicherung, Zugang zum System oder einer sonst geschützten Funktion des Systems zu erlangen.

angepaßt von `https://de.wikipedia.org/wiki/Backdoor`

- in der Regel ist das Vorhandensein einer Backdoor unbekannt
- das Vorhandensein der Backdoor ist von jemandem gewollt
- bei Bekanntwerden, wird die Backdoor zum Zero-Day-Exploit

Aufteilbar in

Software

- Anwendung
- Betriebssystem
- Betriebssystemtreiber
- Firmware

Hardware

- Printed Circuit Board
  - Motherboard
  - Netzwerkkarte
  - ...
- Integrated Circuit
  - Network Phy
  - CPU
  - ...

Wie kommt die Backdoor in die CPU?

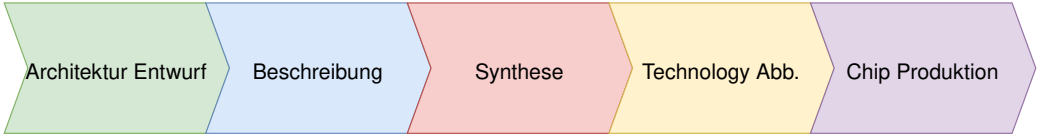


Abbildung: Produktions-Schritte einer CPU

Modus	Instruktionen	Speicherbereiche
Supervisor	alle	alle
User	eingeschränkt	eingeschränkt

Tabelle: generische CPU Modi

Backdoor erlaubt aus dem User Modus:

- Ausführung von nicht erlaubten Instruktionen
- Zugriff auf nicht erlaubte Speicherbereiche
- (Zugriff auf nicht erlaubte Register)
- Änderung des CPU Modus
- (Zugriff auf den Microcode der CPU)

NEIN

- Bugs durch Design Entscheidungen
- **aber:** ähnliche Auswirkungen!

Wie löse ich eine Backdoor aus? ⇒ durch eine bekannte Instruktion

```
int a = 5 + 6   li $a1, 0x0005        0x01010011011111100010110111000110
                li $a2, 0x0006        0x01010011011111100010110111000111
                add $a3, $a1, $a2     0x11110011011111100010110111000111
```

Nachteil:

- wird mehrfach und oft zufällig ausgelöst, da die meisten Programme die Instruktion benutzen
- fällt dadurch leicht auf

Wie löse ich eine Backdoor aus? ⇒ durch einen Parameter einer Instruktion

```
int a = 5 + 0xEFEF
```

```
li $a1, 0x0005
li $a2, 0xEFEF
add $a3, $a1, $a2
```

Nachteil bleibt bestehen

```
add $a3, $a1, $PC
```

besser, unwahrscheinlich dass jemand den PC aufaddiert

```
0x01011011011111010101101011111110
```

Vorteile

- keine Mnemonics
- können nicht zufällig benutzt werden

Nachteile

- im Source Code des Prozessors „leicht" identifizierbar

Wie löse ich eine Backdoor aus? ⇒ durch einen bestimmten Wert in einem Register

```
li $a1, 0xF4E2A511
```

Vorteile

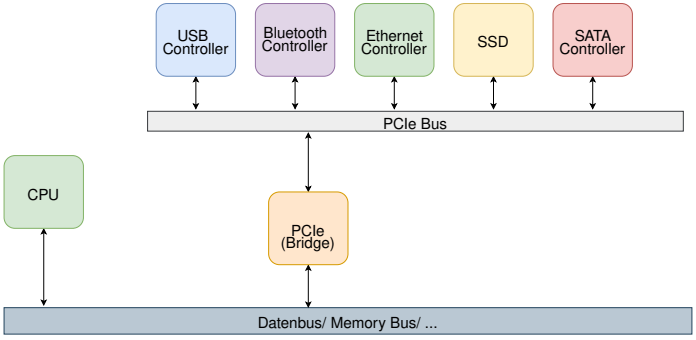
- einfach

Nachteile

- im Source Code des Prozessors „leicht" identifizierbar
- kann zufällig ausgelöst werden

Verbesserung: mehrere Register verwenden

Wie löse ich eine Backdoor aus? ⇒ durch einen bestimmten Wert auf dem Datenbus



Vorteile

- kann von überall ausglöst werden

Nachteile

- im Source Code des Prozessors „leicht" identifizierbar
- (kann zufällig ausgelöst werden)

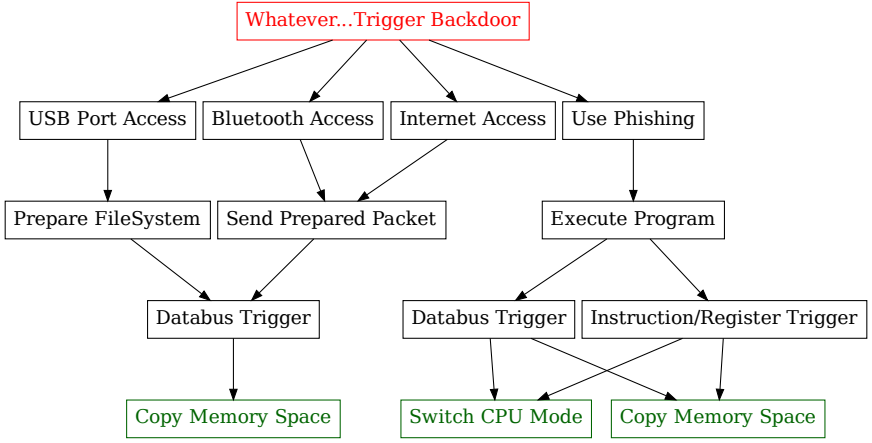


Abbildung: Attacktree for CPU Backdoor Triggering

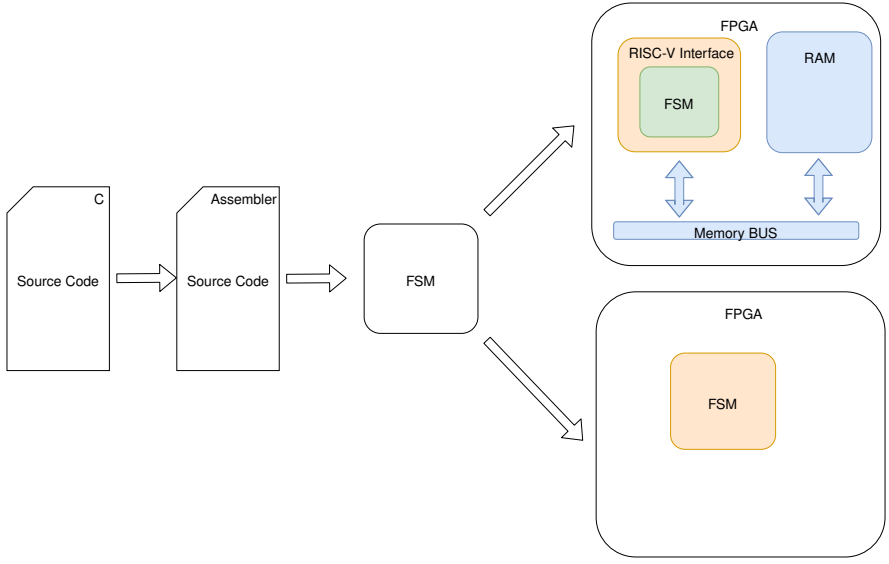
Wie reparier ich es?

Gar nicht! Ist Hardware!

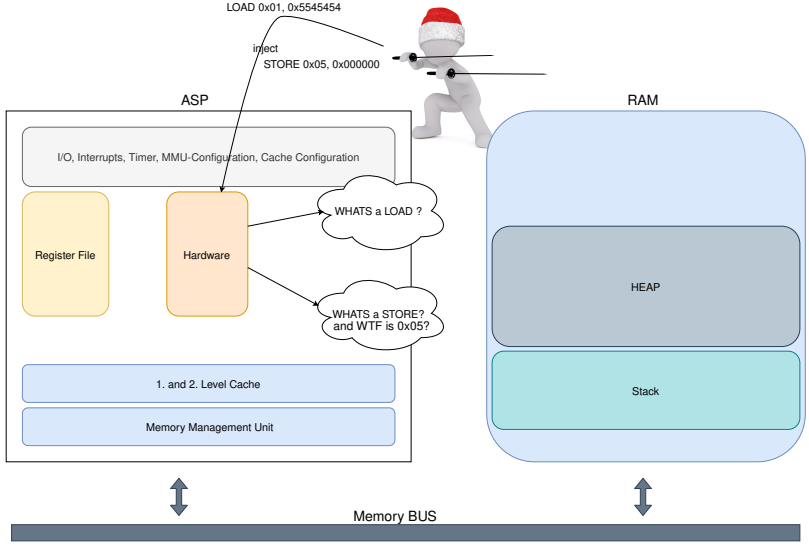
Wie schütze ich mich?

- vielleicht Virenscanner (bekannte nicht Datenbus basierte Backdoors)
- Betriebssystem (bekannte nicht Datenbus basierte Backdoors)
- alternative Konzepte für CPUs (FPGAs, Application Specific Processors)

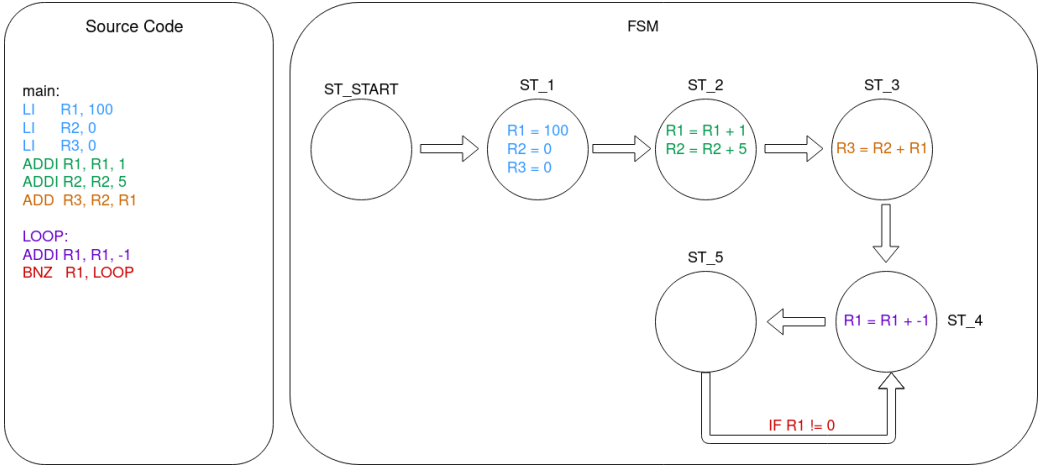
Application Specific Processor (ASP)



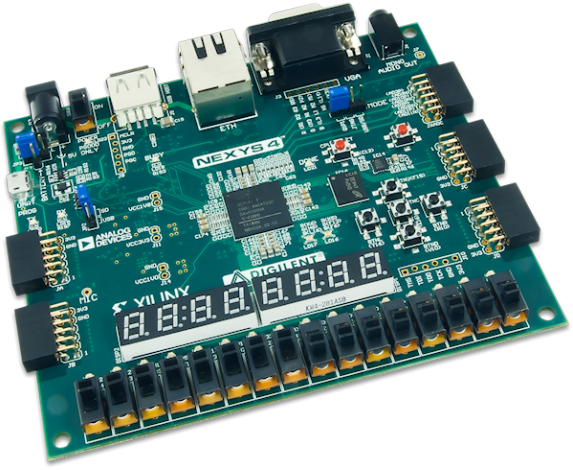
Application Specific Processor (ASP)



Application Specific Processor (ASP)



Processor Backdoor Implementation



- FPGA Evaluation Board, Prototyping for ASICs
- ARMv6 Softcore Processor
- Kernel Version 4.19