

# Systemsicherheit 1

WS 2017/18

**Prof. Dr.-Ing. Hans-Georg Eßer**

**Foliensatz MS:**

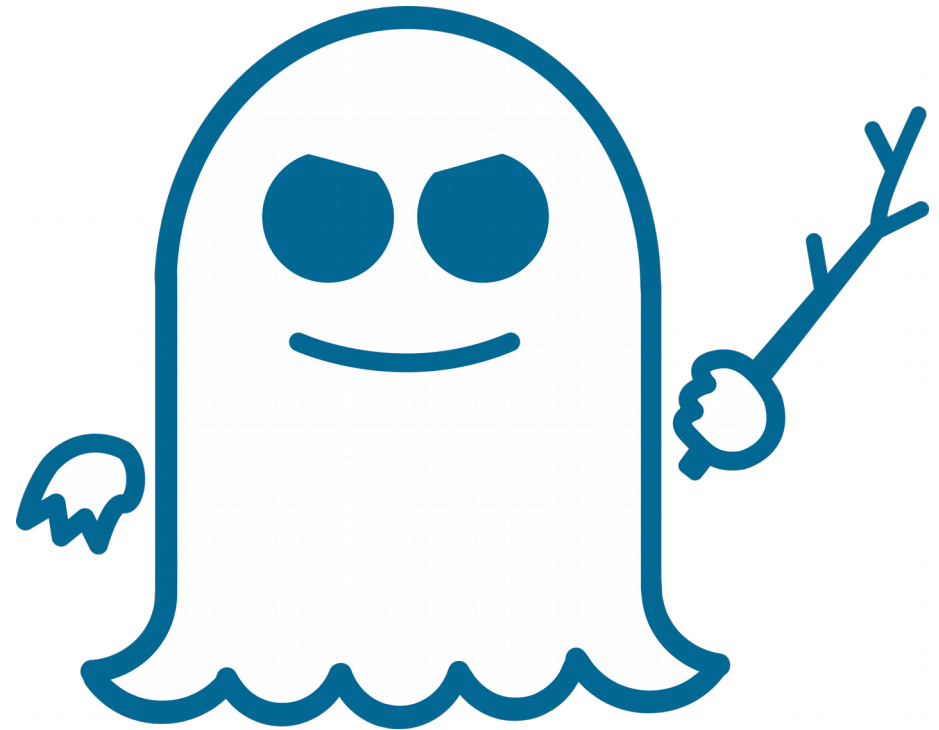
- Meltdown und Spectre



v1.0, 2018/01/19



**MELTDOWN**



**SPECTRE**

- Neue Angriffsmethoden, die Features moderner CPUs ausnutzen
- Out-of-order execution
  - **Speculative execution:** CPU füllt Pipeline mit Instruktionen, die *vermutlich* später ausgeführt werden sollen
  - **Branch Prediction:** CPU analysiert Sprungverhalten bei bedingten Sprüngen und nutzt Analyse für Sprungvorhersagen (und entsprechendes Befüllen der Pipeline)

- Unterscheidung Kernel / User Mode
  - Kernel Mode (Ring 0)
  - User Mode (Ring 3)
- Programme (Prozesse) laufen im User Mode

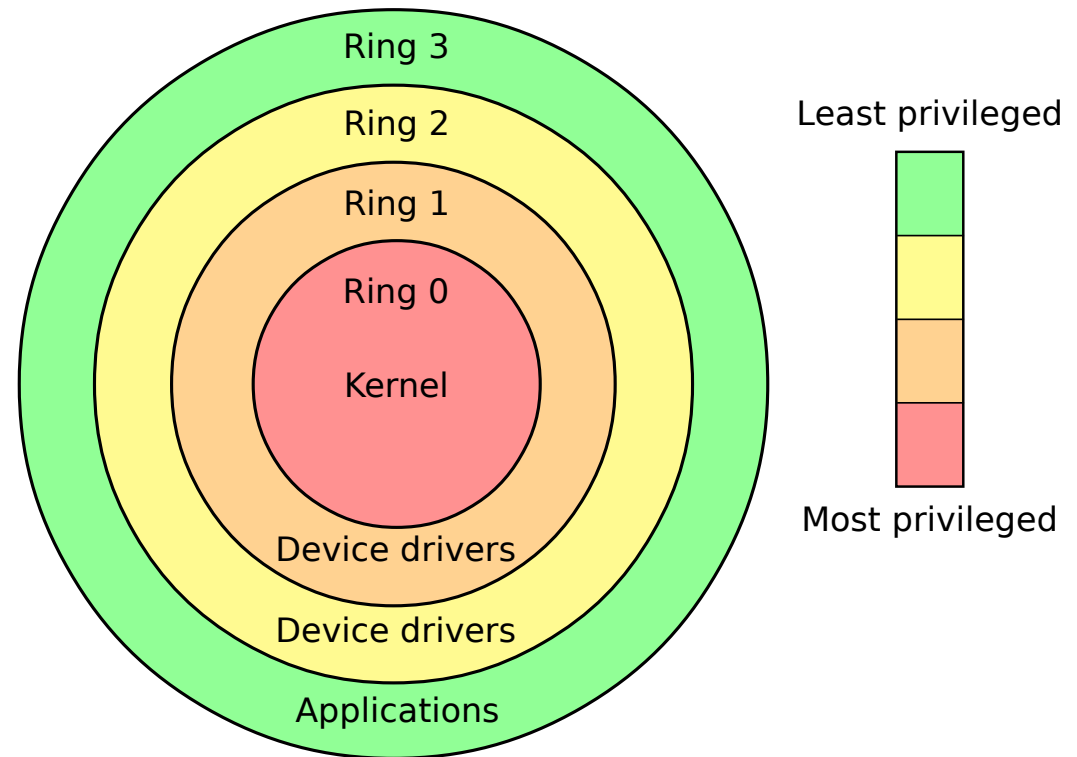


Bild: [https://en.wikipedia.org/wiki/Protection\\_ring](https://en.wikipedia.org/wiki/Protection_ring)

- Moderne Betriebssysteme nutzen Paging zur Speicherverwaltung
  - RAM in Seitenrahmen (frames) unterteilt, Größe 4K
  - Für Prozesse wird ein virtueller Adressraum definiert, der aus Seiten (pages) besteht, Größe 4K
  - BS verwaltet Seitentabelle, die i.W. eine Zuordnung von Seiten zu Seitenrahmen bewirkt – Tabelle definiert aber auch **Zugriffsrechte**
  - MMU übersetzt on-the-fly Speicheradressen (z. B. beim Ausführen von mov- und jmp-Befehlen)

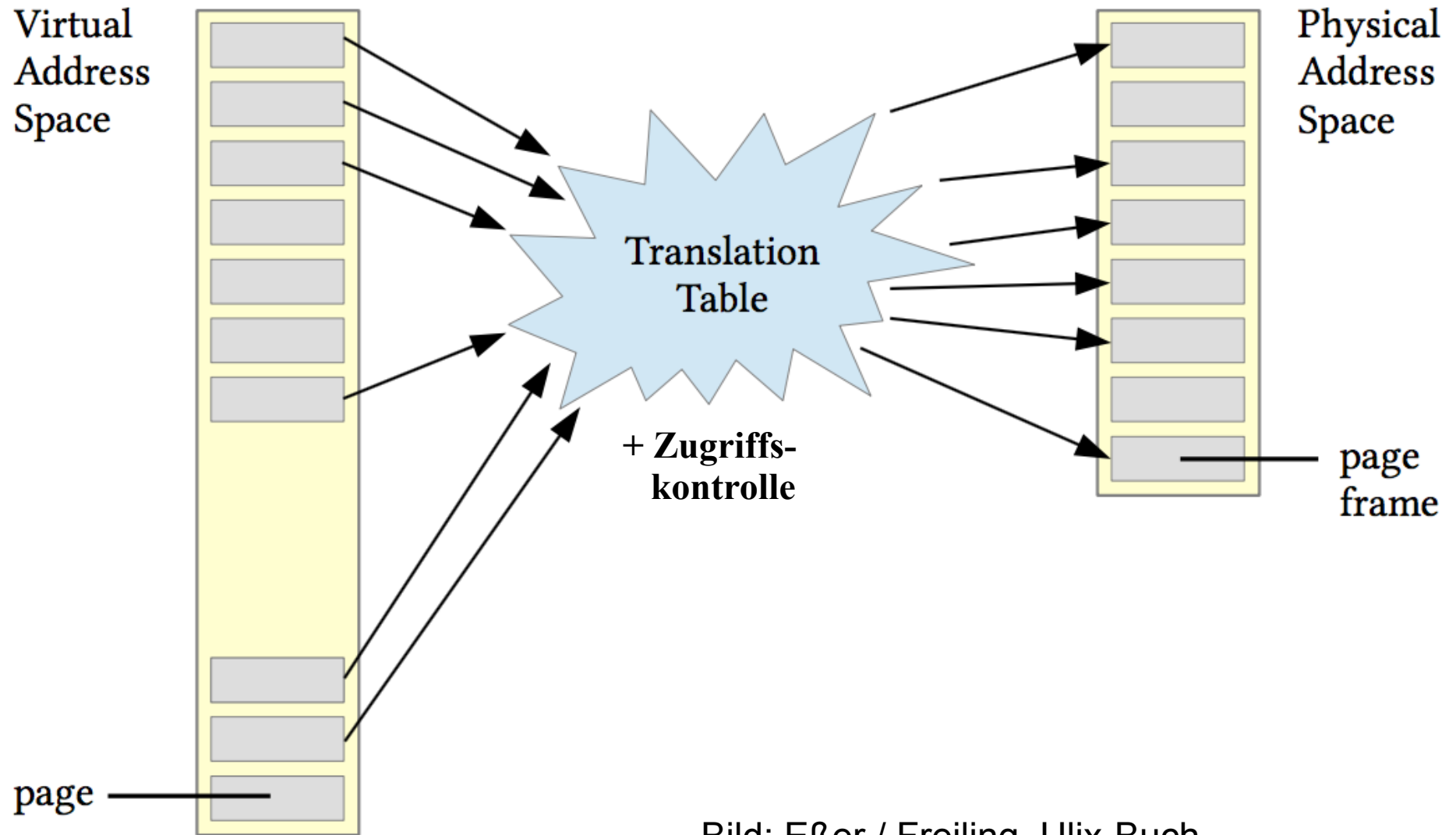
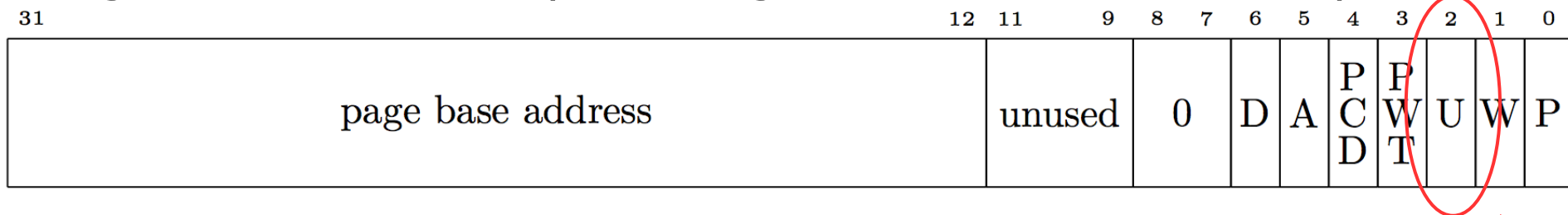


Bild: Eßer / Freiling, Ulix-Buch

- Page Descriptor (Eintrag der Seitentabelle)



mit Flags:

- Bit 2 = U = User Accessible Flag
    - für Seiten, die dem Prozess gehören: gesetzt
    - für Seiten, die dem BS gehören: nicht gesetzt
- was sind das für Seiten?

- typisches virtuelles Speicher-Layout bei 32 Bit:
  - untere 3 GB für den Prozess (Code, Daten, Heap, Stack)
  - oberes 1 GB für das BS
  - beim Aufruf eines System Calls (Sprung ins BS) kein Umschalten der Seitentabelle nötig, denn:
    - alle BS-Inhalte schon da, nur im User Mode „versteckt“
    - nach Sprung in Kernel Mode paralleler Zugriff auf Prozess- und BS-Speicher, praktisch für Transfers



- Zugriff auf Speicher fremder Prozesse:  
unmöglich, weil gar nicht über Seitentabelle  
„verlinkt“  
(prinzipiell unerreichbar)
- Zugriff auf BS-Speicher: verboten durch  
fehlende U-Flags  
(prinzipiell erreichbar, Zugriff im User Mode  
erzeugt aber Fault → Programmabbruch)

- Seitenkanal-Angriff (covert channel):  
timing channel
- Versuche Zugriff auf geschützten Bereich und beobachte zeitliche Auffälligkeiten

## Literatur:

[Wik] Wikipedia, Covert channel, [https://en.wikipedia.org/wiki/Covert\\_channel](https://en.wikipedia.org/wiki/Covert_channel)

[Eße05] Hans-Georg Eßer: Ausnutzung verdeckter Kanäle am Beispiel eines Web-Servers, Diplomarbeit, RWTH Aachen, Feb. 2005, <http://privat.hgesser.com/docs/esser-info-diplom.pdf>



## Prozess B

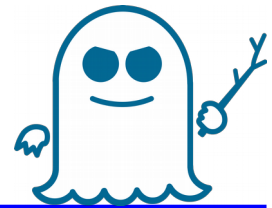
- legt 256 Speicherseiten an (je 4K) und leert Cache
- Zugriffe auf jede der Seiten sollten gleich lang dauern

## Prozess A

- `a = get_byte (verbotene_adresse);`  
`access (page[a]);`
- Durch spekulative Ausführung: Zugriff auf Speicherseite *a* findet *parallel zu Exception* statt → *a* landet im Cache

## Prozess B

- testet alle 256 Seiten und stellt fest, welche im Cache ist

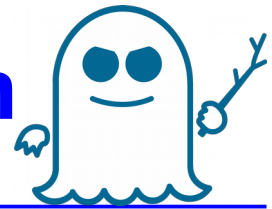


- Überwinden von Überprüfungen, ob ein Zugriff innerhalb gültiger Array-Grenzen erfolgt

```
if (index < array1_size)
    y = array2[array1[index] * 256];
```

→ interessant z. B. bei System Calls oder JavaScript-Interpretern

- Gleiche Idee wie bei Meltdown: Durch spekulative Ausführung erst Zugriff auf das gesuchte Byte, dann auf darüber berechneten Speicherbereich; anschließend Messen der Zugriffsgeschwindigkeiten



- CPU nutzt Branch Target Buffer (BTB) für Vorhersagen zu bedingten Sprüngen
- Angriff auf indirekte Sprünge (`jmp eax`):
  - Lokalisieren einer Code-Adresse im *anzugreifenden* Prozess
  - Training des BTB durch Ausführen indirekter Sprünge zu dieser Adresse im *eigenen* Prozess
  - Führt beim angegriffenen Prozess wieder dazu, dass Code spekulativ ausgeführt wird

- Auslese-Geschwindigkeit in Tests:
  - Meltdown: ~ 500 KByte/s
  - Spectre 1: ~ 2 KByte/s
  - Spectre 2: ~ 1,5 KByte/s

- Meltdown: Page Table Isolation (PTI)
- Spectre 1: BS- und Programm-Patches
- Spectre 2: CPU-**Microcode-Updates** (→ neue CPU-Instruktionen für Sprungvorhersage) und BS-Updates

Alternativ: Technik namens „Retpoline“ bzw. „Return Trampoline“, siehe z. B.

<https://lkml.org/lkml/2018/1/3/780> und

<https://support.google.com/faqs/answer/7625886>

- CPU-Instruktionen werden durch CPU interpretiert: Für jede Instruktion gibt es ein Microcode-Programm.
- Microcode ist aktualisierbar; BIOS oder BS kann alternativen Microcode laden
- Microcode-Update:
  - Windows: BIOS-Update
  - Linux: Firmware-Paket-Update



- Gepatchte Systeme werden langsamer:

BAPCo SYSmark 2014 SE [Punkte]	ohne	1808
BAPCo SYSmark 2014 SE [Punkte]	mit	1723
SYSmark Responsiveness [Punkte]	ohne	1431
SYSmark Responsiveness [Punkte]	mit	1331
PC Mark 10 [Punkte]	ohne	4806
PC Mark 10 [Punkte]	mit	4733
PCIe-SSD, 1 Worker [IOPS]	ohne	168045/197949
PCIe-SSD, 1 Worker [IOPS]	mit	79313/105986

Auszug aus Tabelle „Leistungseinbußen durch Windows- und BIOS-Updates gegen Meltdown & Spectre“, Spalte zu Intel Core-i8700K (2017) in C. Windeck: Die Riesenlücken, c't 3/2018, S. 58 ff.

- Meltdown- und Spectre-Seite: <https://meltdownattack.com/>
- Intel-Informationen: <https://www.intel.com/content/www/us/en/architecture-and-technology/facts-about-side-channel-analysis-and-intel-products.html>
- M. Lipp, M. Schwarz, Michael et al.: Meltdown, 01/2018, <https://meltdownattack.com/meltdown.pdf>
- P. Kocher, D. Genkin et al.: Spectre Attacks: Exploiting Speculative Execution, 01/2018, <https://spectreattack.com/spectre.pdf>
- C. Windeck: Die Riesenlücken, c't 3/2018, S. 58 ff.
- O. von Westernhagen: Verspekuliert: Meltdown & Spectre-Angriffe im Detail, c't 3/2018, S. 62 ff.
- ... und weitere Artikel im aktuellen c't-Schwerpunkt

