



One-Location Rowhammer Angriff

Gilian Henke, Dominik Mairhöfer

8. Februar 2018

Aktuelle Themen IT-Sicherheit und Zuverlässigkeit

Inhalt

1 Einführung

2 Hintergrund

- One-Location Rowhammer
- Memory Waylaying
- Prefetch Side-Channel Angriff

3 Ergebnisse

4 Fazit

Motivation

- Neuer Rowhammer Angriff
- Neue Technik zum Ausnutzen des Rowhammer Angriffs

Motivation

- Neuer Rowhammer Angriff
- Neue Technik zum Ausnutzen des Rowhammer Angriffs

Wie gut funktioniert der Angriff?

- Durchführbarkeit des Angriffs testen
 - schnell, zuverlässig, unauffällig, ...
 - Voraussetzungen
 - Gegenmaßnahmen

Der Angriff

- Ziel: Verändern einer ausführbaren Datei ohne Schreibrechte
 - z.B. sudo Datei für lokale Rechteauserweiterung
- Basieren auf drei Teilen
 - One-Location Rowhammer
 - ⇒ Bitflips im Speicher erzeugen
 - Memory Waylaining
 - ⇒ Dateien an bestimmten Stellen im Speicher platzieren
 - Prefetch Side-Channel Angriff
 - ⇒ Virtuelle Adressen in physikalische auflösen

Der Angriff - Ablauf

- 1 One-Location Rowhammer
⇒ Finde virtuelle Adresse bei der Bitflip möglich ist

Der Angriff - Ablauf

- ① One-Location Rowhammer
⇒ Finde virtuelle Adresse bei der Bitflip möglich ist
- ② Prefetch Side-Channel Angriff
⇒ Finde physikalische Adressen zu dieser

Der Angriff - Ablauf

- ① One-Location Rowhammer
⇒ Finde virtuelle Adresse bei der Bitflip möglich ist
- ② Prefetch Side-Channel Angriff
⇒ Finde physikalische Adressen zu dieser
- ③ Memory Waylaying & Prefetch Side-Channel Angriff
⇒ Platziere ausführbare Datei an dieser physikalischen Adresse

Der Angriff - Ablauf

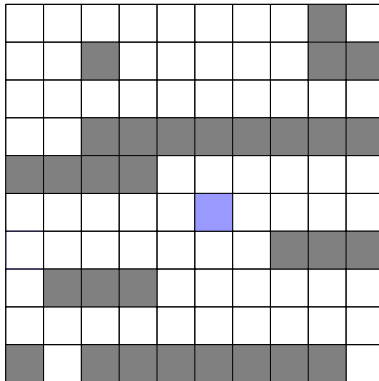
- ① One-Location Rowhammer
⇒ Finde virtuelle Adresse bei der Bitflip möglich ist
- ② Prefetch Side-Channel Angriff
⇒ Finde physikalische Adressen zu dieser
- ③ Memory Waylaying & Prefetch Side-Channel Angriff
⇒ Platziere ausführbare Datei an dieser physikalischen Adresse
- ④ One-Location Rowhammer
⇒ Erzeuge erneut Bitflip an gleicher physikalischer Adresse

One-Location Rowhammer

- Idee Rowhammer
 - Speicher benötigt immer weniger Spannung
 - Geringe Änderungen der Spannung führen zu Bitflip
 - Werden Beeinflusst durch benachbarten Speicher
 - ⇒ Manipulieren von Speicher durch Wiederholten Zugriff auf benachbarte Speicherzellen
- Idee One-Location Rowhammer
 - Nur eine Position wird „gehämmert“

One-Location Rowhammer - Beispiel

load



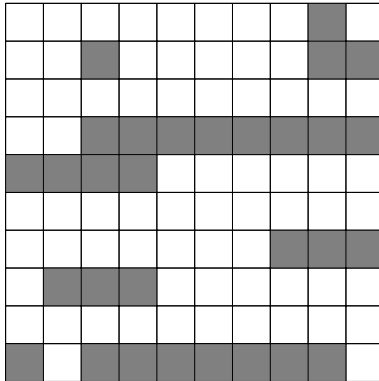
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

flush



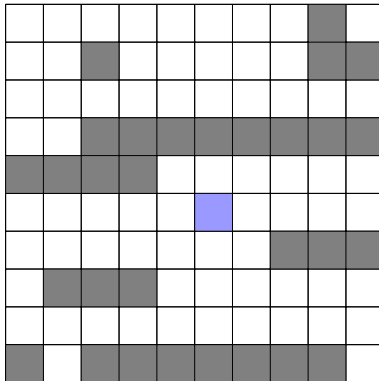
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

load



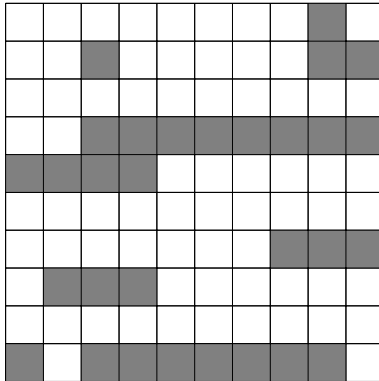
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

flush



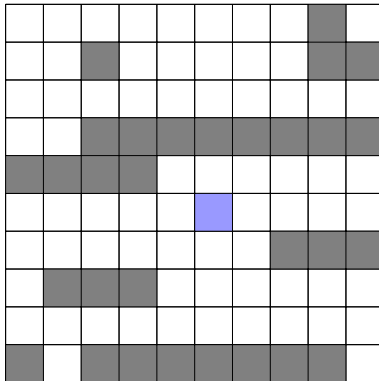
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

load



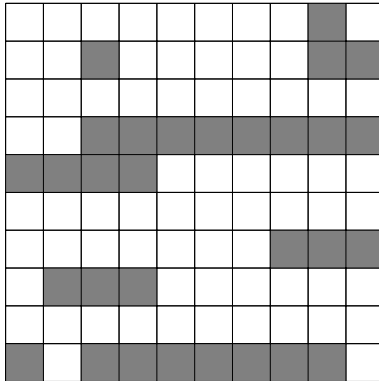
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

flush



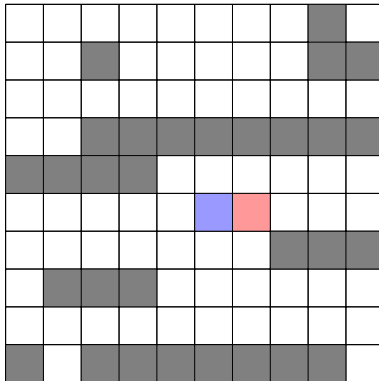
benutzter Speicher



Hämmern

One-Location Rowhammer - Beispiel

load



benutzter Speicher



Hämmern



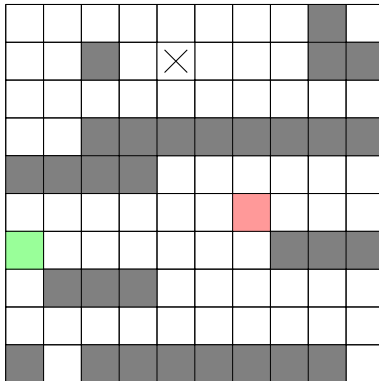
Bitflip





Memory Waylaying

- Datei an bestimmter Stelle im Speicher platzieren
 - Datei schreibbar laden \Rightarrow neue Kopie im Speicher
 - Datei aus Speicher entfernen und wieder neu laden
 - Datei an neuer Adresse
 - Mit Prefetch Side-Channel Angriff Adresse überprüfen
 - Wiederholen bis Datei an richtiger Adresse
- Speicher komplett mit eigenen Dateien füllen \Rightarrow original Datei wird entfernt
- Eigene Kopie aus Speicher entfernen und original Datei lesbar laden

Memory Waylaying - Beispiel

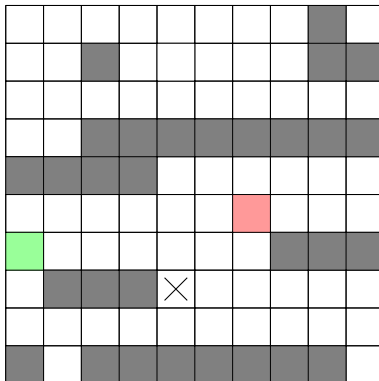
Datei laden







-  benutzter Speicher
-  Ziel
-  original Binary
-  geladene Binary

Memory Waylaying - Beispiel

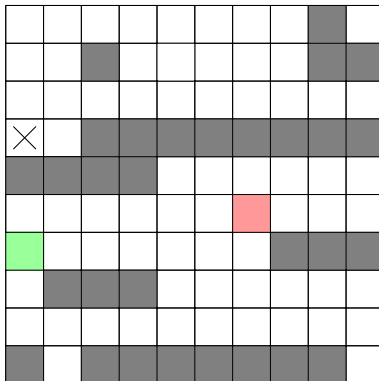
Datei neu laden



-  benutzter Speicher
-  Ziel
-  original Binary
-  geladene Binary

Memory Waylaying - Beispiel

Datei neu laden



benutzter Speicher

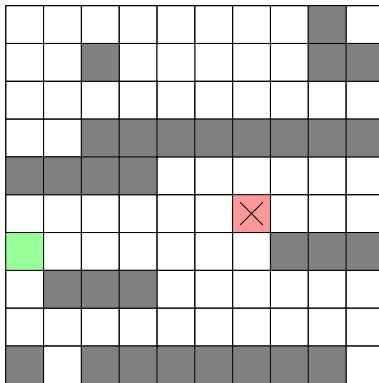
Ziel

original Binary

geladene Binary

Memory Waylaying - Beispiel

Datei neu laden - richtige Position



■ benutzter Speicher

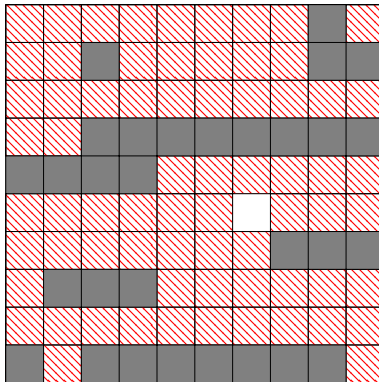
■ Ziel

■ original Binary

× geladene Binary

Memory Waylaying - Beispiel

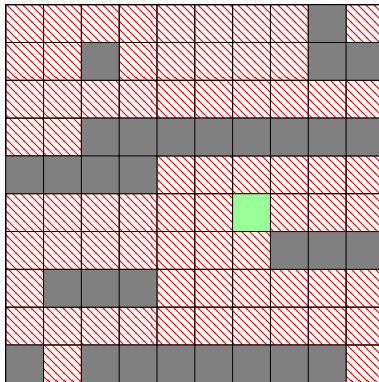
Speicher füllen, eigene Datei entfernen



- benutzter Speicher
- Ziel
- original Binary
- × geladene Binary

Memory Waylaying - Beispiel

Original Datei neu laden

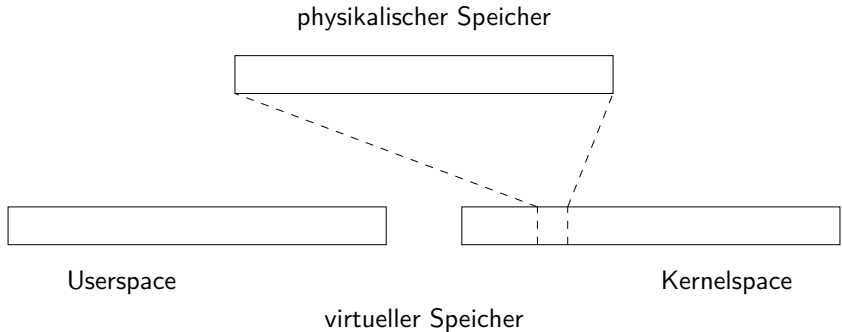


- benutzter Speicher
- Ziel
- original Binary
- × geladene Binary

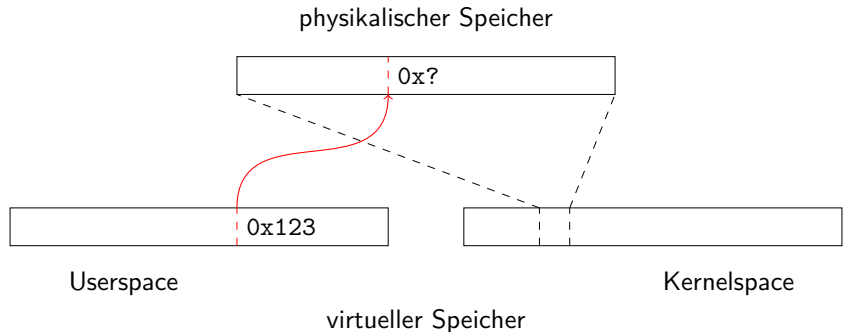
Prefetch Side-Channel Angriff

- Physikalische Adressen zu virtuellen Adressen finden
- Nutzt das direct mapping im Kernelspace
 - physikalischer Speicher ist im Kernelspace gemappt
 - virt: 0xffff880000000000 \Rightarrow phys. 0x00000
 - virt: 0xffff880000000001 \Rightarrow phys. 0x00001
 - ⋮
- Jede Adresse zweimal gemappt: Userspace und Kernelspace
- `prefetch` Befehl ermöglicht laden aller Adressen in den Cache
 \Rightarrow Timing Angriff

Prefetch Side-Channel Angriff - Beispiel

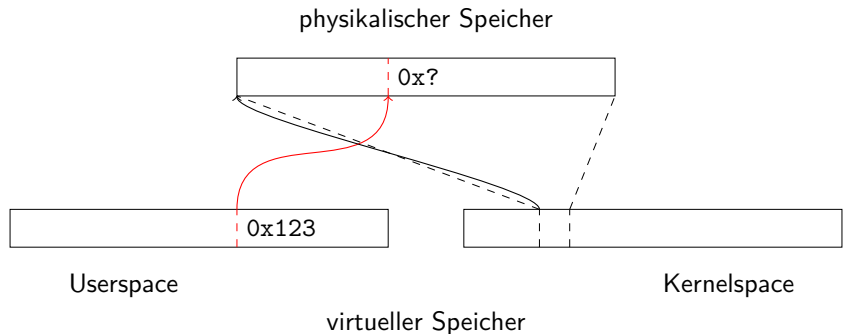


Prefetch Side-Channel Angriff - Beispiel



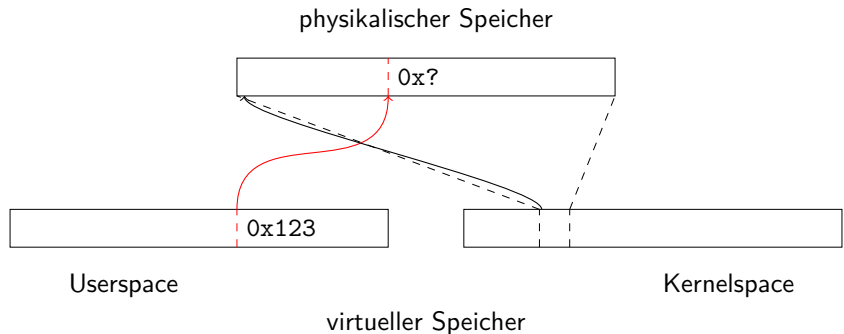
Prefetch Side-Channel Angriff - Beispiel

```
flush 0x123, prefetch 0xff...0000, reload 0x123
```



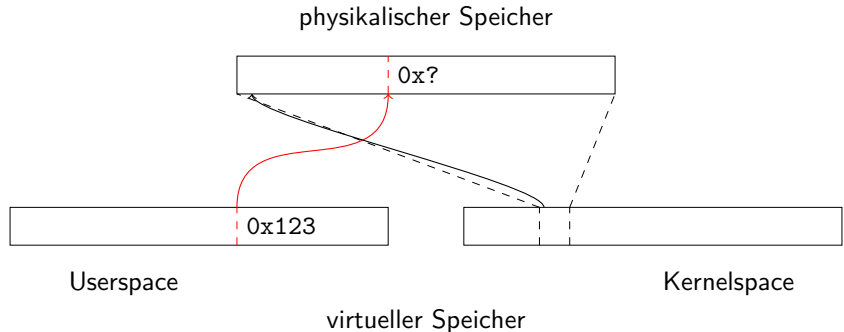
Prefetch Side-Channel Angriff - Beispiel

`flush 0x123, prefetch 0xff...0001, reload 0x123`



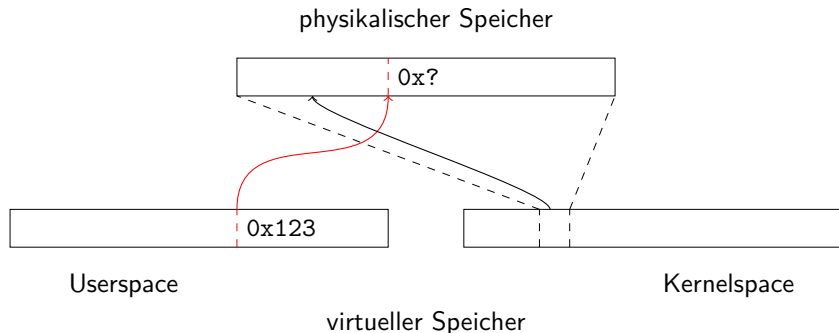
Prefetch Side-Channel Angriff - Beispiel

```
flush 0x123, prefetch 0xff...0002, reload 0x123
```



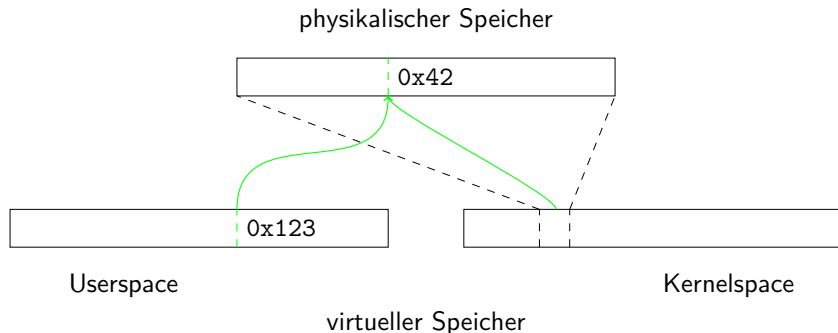
Prefetch Side-Channel Angriff - Beispiel

```
flush 0x123, prefetch 0xff...0010, reload 0x123
```



Prefetch Side-Channel Angriff - Beispiel

```
flush 0x123, prefetch 0xff...0042, reload 0x123
```



Testbeschreibung

- Durchführbarkeit
 - Funktioniert der Angriff überhaupt?
 - Gibt es Voraussetzungen in Hard- oder Software?
- Zuverlässigkeit
 - Funktioniert der Angriff immer?
 - Welche Bedingungen sind optimal? (CPU Last, Speicherauslastung...)
- Schnelligkeit
- Unauffälligkeit

Ergebnisse - One-Location Rowhammer

- Keine Erzeugung von Bitflips mit One-Location Rowhammer im Zeitrahmen möglich
- Getestete Systeme waren anfällig gegen Rowhammer-Angriffe
 - Erzeugung eines Bitflips mit Double-Sided Rowhammer in 30 Minuten
- Geringe CPU-Auslastung
- Hohe Speicherauslastung
- Verhinderung durch ECC-Speicher oder Überwachung des Systems

Ergebnisse - Memory Waylaing

- Einmaliges Ändern der physikalischen Adresse einer Seite: 3,5 Sekunden
- Platzieren einer Seite an gewünschter physikalischer Adresse: min. 40 Stunden
- Geringe CPU-Auslastung
- Wird nicht langsamer bei hoher Belastung des Rechners
- Verhinderung durch Überwachung des Systems

Ergebnisse - Prefetch Side-Channel Angriff

- Finden der physikalischen Adresse bei uns nicht möglich
 - Kein laden in den Cache durch `prefetch` im Kernelspace
 - Im Userspace funktioniert der Code
- Gepatcht? Durch KPTI (Meltdown) wird der Angriff verhindert
 - Funktionierte auch mit alten Kernen nicht
 - Ursache unklar
- Testen des direct mappings dauert sehr lange
- Sehr hohe CPU Last
- Teilweise viele false positives

Fazit

- Bitflips können erzeugt werden
- Seiten können an passende Stellen im Speicher positioniert werden
⇒ Angriff prinzipiell durchführbar
- Prefetch Side-Channel Angriff funktioniert bei uns nicht
- Unauffälligkeit des Angriffs durch SGX
- Bleibende Probleme
 - Langsame Laufzeit der Teilschritte
 - Prefetch Side-Channel Angriff
- Durch KPTI wird der Angriff komplett verhindert