

Microarchitectural Attacks: From the Basics to Arbitrary Read and Write Primitives without any Software Bugs

Daniel Gruss

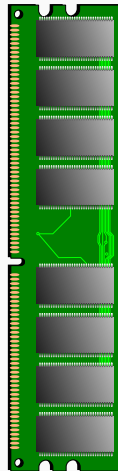
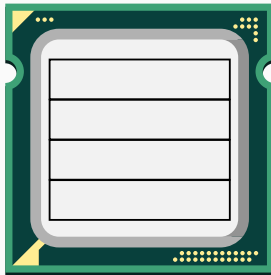
April 11, 2018

Graz University of Technology

- Daniel Gruss
- Post-Doc @ Graz University of Technology
- Twitter: @lavados
- Email: `daniel.gruss@iaik.tugraz.at`

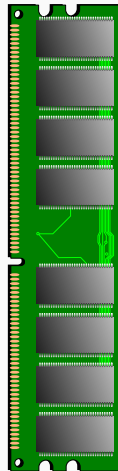
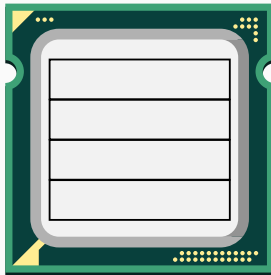


```
printf("%d", i);  
printf("%d", i);
```



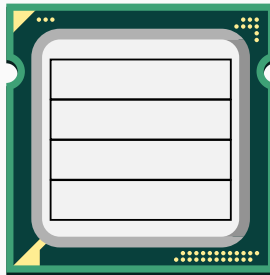
```
printf("%d", i);  
printf("%d", i);
```

Cache miss

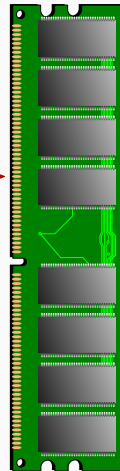


```
printf("%d", i);  
printf("%d", i);
```

Cache miss

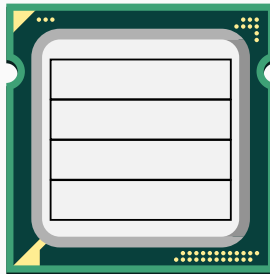


Request



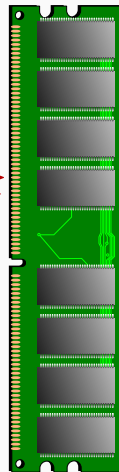
```
printf("%d", i);  
printf("%d", i);
```

Cache miss



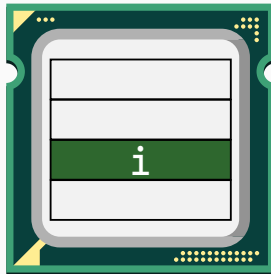
Request

Response



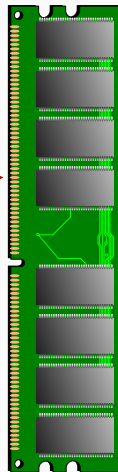
```
printf("%d", i);  
printf("%d", i);
```

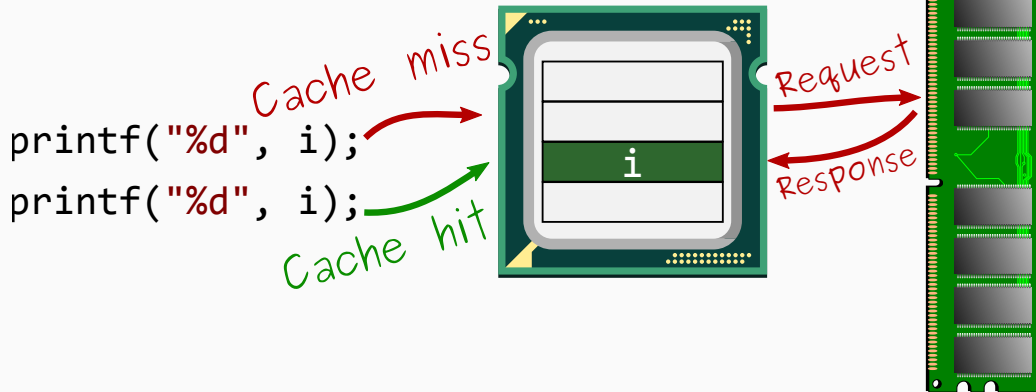
Cache miss

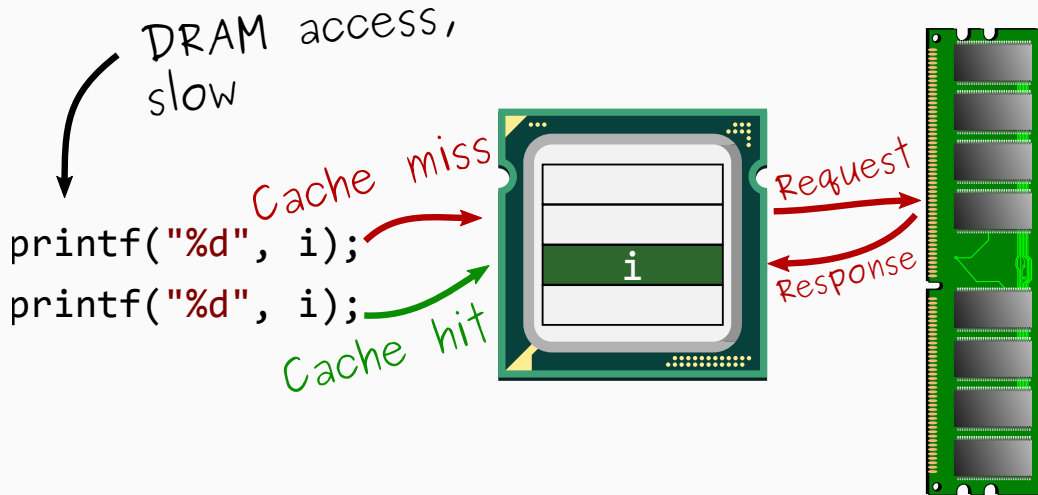


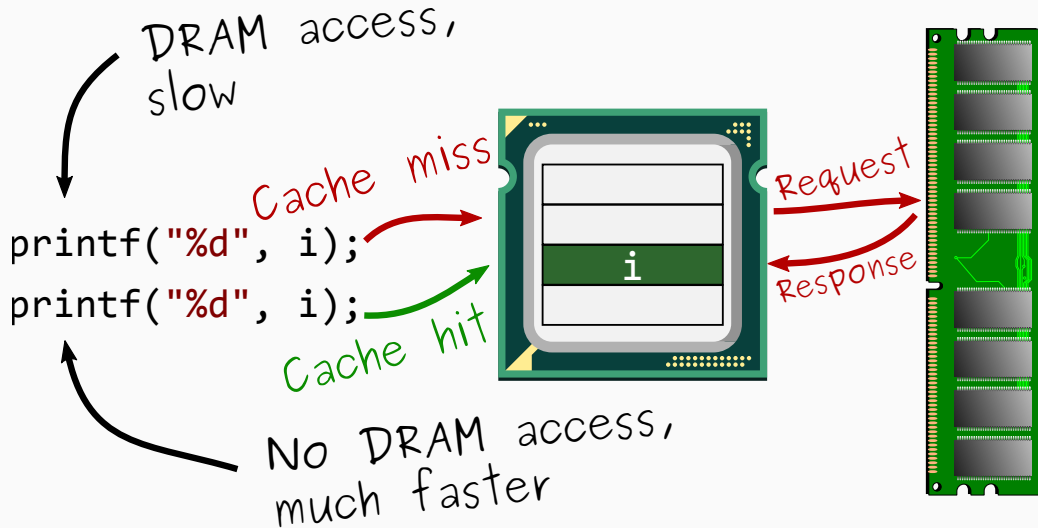
Request

Response





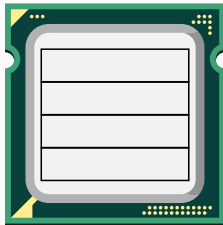




Shared Memory

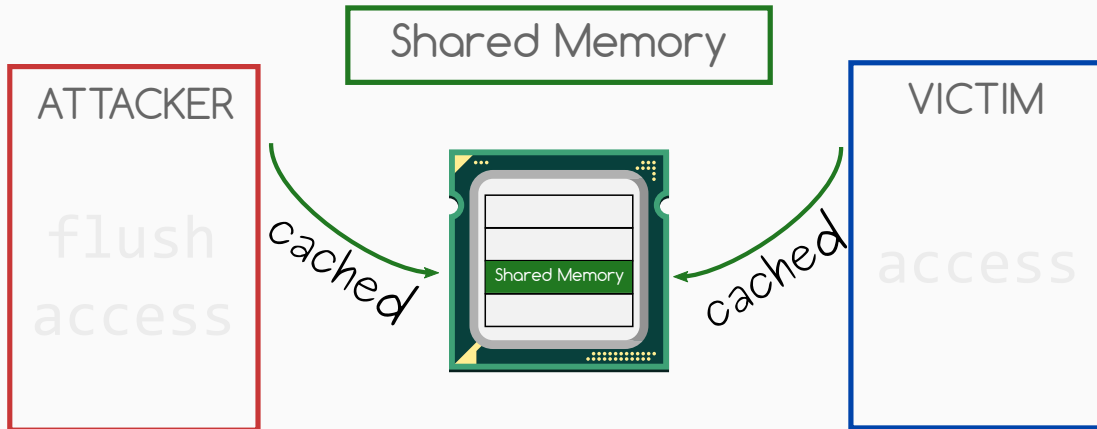
ATTACKER

flush
access



VICTIM

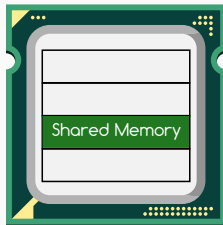
access



Shared Memory

ATTACKER

flush
access



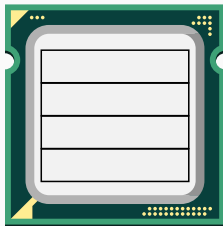
VICTIM

access

Shared Memory

ATTACKER

flush
access



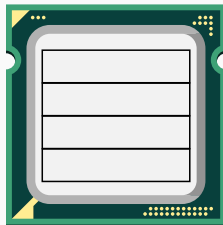
VICTIM

access

Shared Memory

ATTACKER

flush
access



VICTIM

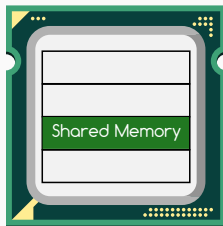
access



Shared Memory

ATTACKER

flush
access



VICTIM

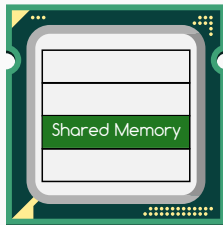
access



Shared Memory

ATTACKER

flush
access



VICTIM

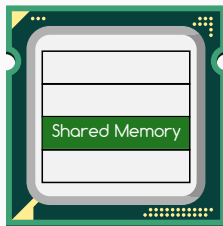
access

Shared Memory

ATTACKER

flush

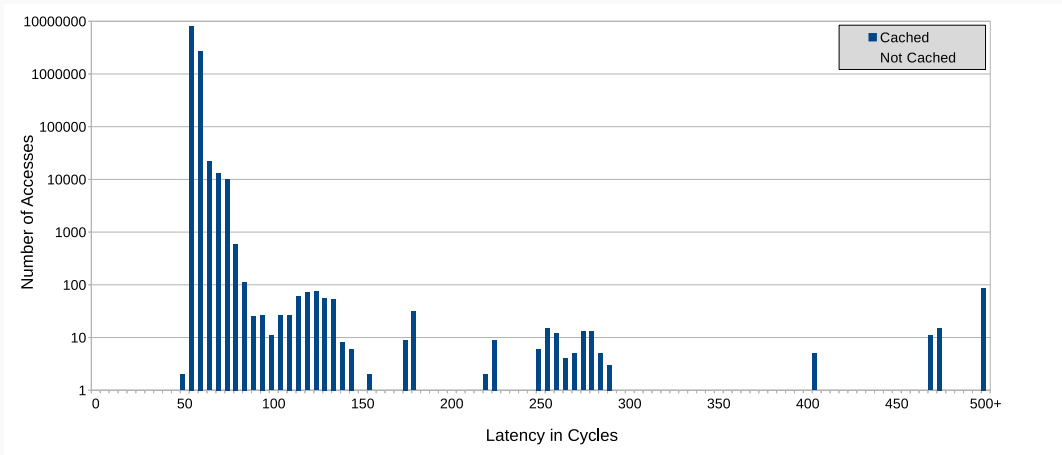
access

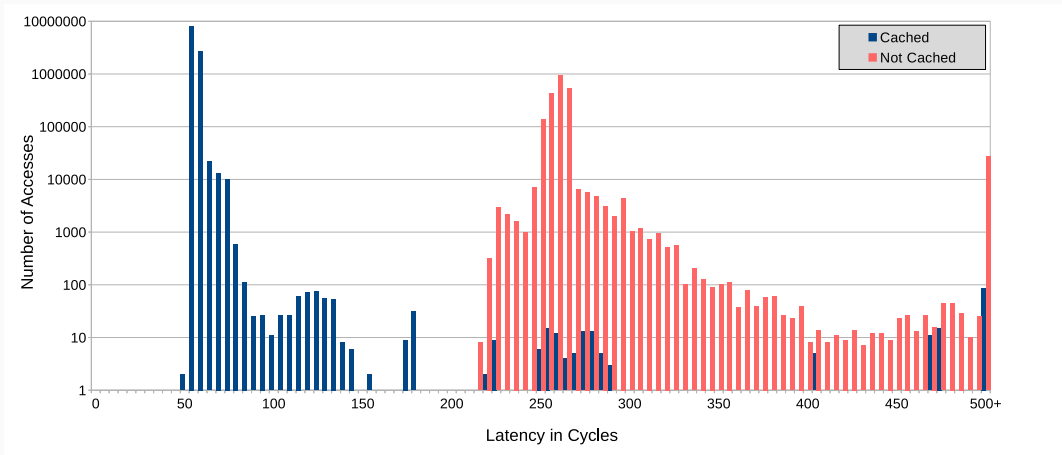


VICTIM

access

fast if victim accessed data,
slow otherwise





Terminal

File Edit View Search Terminal Help

```
% sleep 2; ./spy 300 7f05140a4000-7f051417b000 r-xp 0x20000 08:02 26
8050 /usr/lib/x86_64-linux-gnu/gedit/libgedit.so
```

[nmap] <DIR> 08/02/2017 17:40:43
[nmap] <DIR> 14/03/2017 21:44:26

Terminal

File Edit View Search Terminal Help

```
shark% ./spy
```

[nmap] <DIR> 08/02/2017 17:40:43
[nmap] <DIR> 14/03/2017 21:44:26

Open +

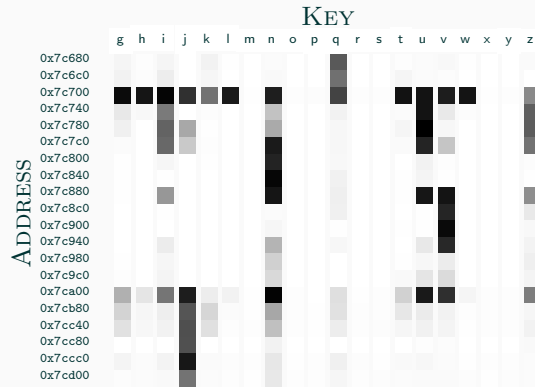
Untitled Document 1

Save - + x

1

I

Plain Text Tab Width: 2 Ln 1, Col 1 INS





MELTDOWN

*7. Serve with cooked
and peeled potatoes*





Wait for an hour





Wait for an hour

LATENCY

1. *Wash and cut
vegetables*

2. *Pick the basil leaves
and set aside*

3. *Heat 2 tablespoons of
oil in a pan*

4. *Fry vegetables until
golden and softened*



Dependency

1. Wash and cut vegetables

2. Pick the basil leaves and set aside

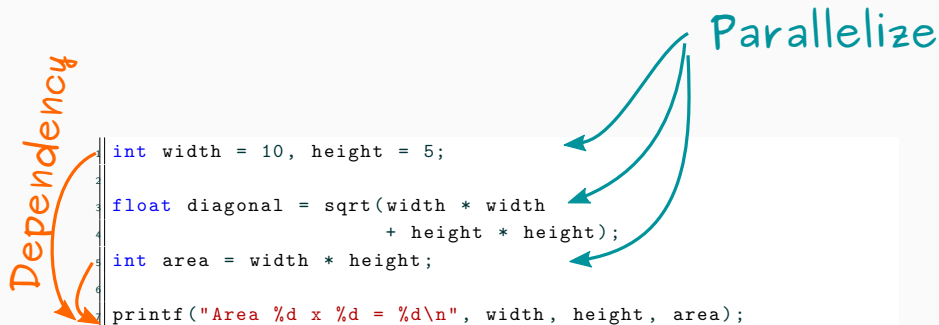
3. Heat 2 tablespoons of oil in a pan

4. Fry vegetables until golden and softened

Parallelize



```
1 int width = 10, height = 5;
2
3 float diagonal = sqrt(width * width
4                       + height * height);
5 int area = width * height;
6
7 printf("Area %d x %d = %d\n", width, height, area);
```




```
1 | char data = *(char*)0xffffffff81a000e0;  
2 | printf("%c\n", data);
```



```
1 char data = *(char*)0xffffffff81a000e0;  
2 printf("%c\n", data);
```



```
1 segfault at ffffffff81a000e0 ip 0000000000400535  
2          sp 00007ffce4a80610 error 5 in reader
```

```
1 char data = *(char*)0xffffffff81a000e0;  
2 printf("%c\n", data);
```



```
1 segfault at ffffffff81a000e0 ip 0000000000400535  
2          sp 00007ffce4a80610 error 5 in reader
```

- Kernel addresses are not accessible

```
1 char data = *(char*)0xffffffff81a000e0;  
2 printf("%c\n", data);
```



```
1 segfault at ffffffff81a000e0 ip 0000000000400535  
2          sp 00007ffce4a80610 error 5 in reader
```

- Kernel addresses are not accessible
- Are privilege checks also done when executing instructions out of order?

- Adapted code



```
1 | *(volatile char*)0;  
2 | array[84 * 4096] = 0; // unreachable
```



- Adapted code

```
1 | *(volatile char*)0;  
2 | array[84 * 4096] = 0; // unreachable
```

- Static code analyzer is not happy

```
1 | warning: Dereference of null pointer  
2 |     *(volatile char*)0;
```



- Flush+Reload over all pages of the array



- “Unreachable” code line was actually executed



- Flush+Reload over all pages of the array



- “Unreachable” code line was actually executed
- Exception was only thrown afterwards



- Maybe there is no permission check in transient instructions...



- Maybe there is no permission check in transient instructions...
- ...or it is only done when committing them



- Maybe there is no permission check in transient instructions...
- ...or it is only done when committing them
- Indirection through microarchitectural traces:

```
1 char data = *(char*)0xffffffff81a000e0;  
2 array[data * 4096] = 0;
```



- Maybe there is no permission check in transient instructions...
- ...or it is only done when committing them
- Indirection through microarchitectural traces:

```
1 char data = *(char*)0xffffffff81a000e0;  
2 array[data * 4096] = 0;
```

- Check whether any part of array is cached



- Flush+Reload over all pages of the array



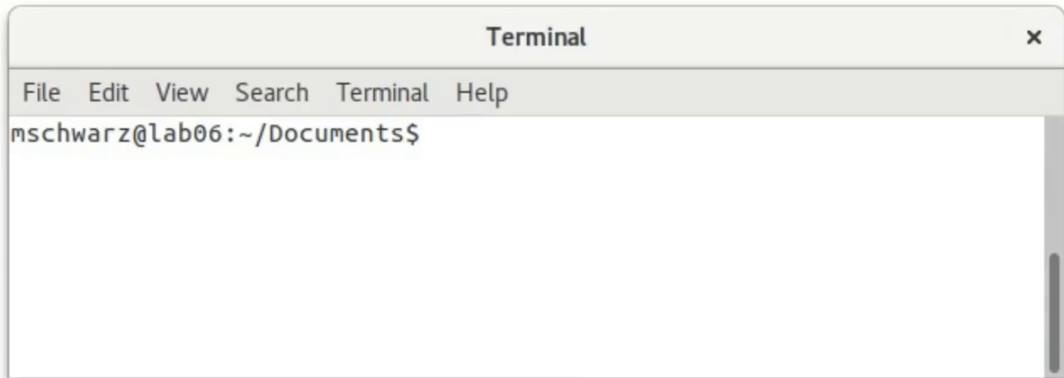
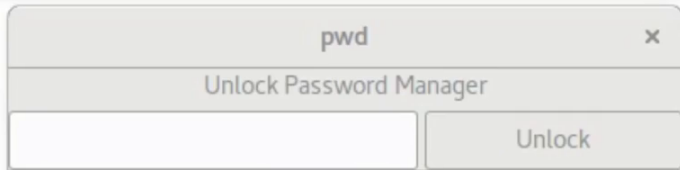
- Index of cache hit reveals data




- Flush+Reload over all pages of the array



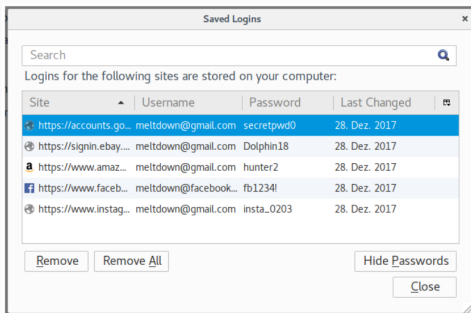
- Index of cache hit reveals data
- Permission check is in some cases not fast enough





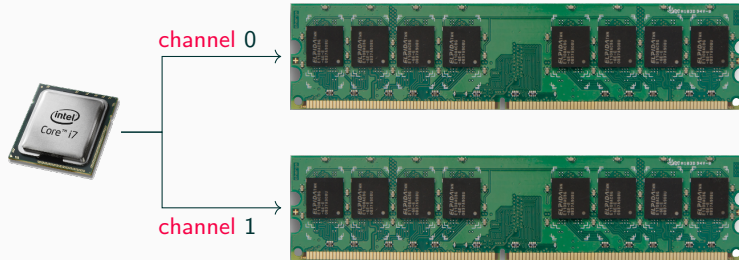
```
meltdown@meltdown ~/ppm2 % taskset 1 ./imgdump 0x375a00000 14919 > outp
ut.flif
Reading from 0xffff880375a00000
```

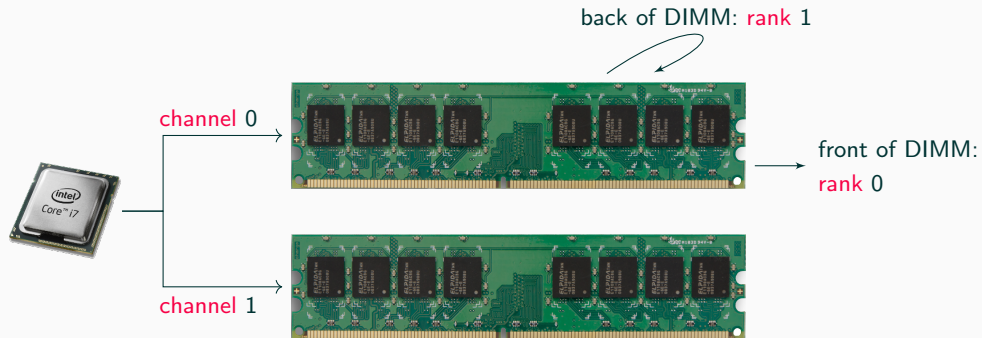


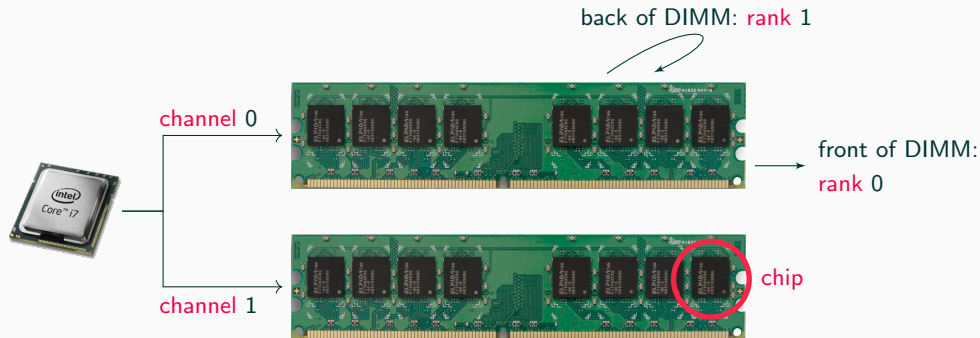


```
f94b7690: e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 | .....  
f94b76a0: e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 | .....  
f94b76b0: 70 52 b8 6b 96 7f XX XX XX XX XX XX XX XX | pR.k.....  
f94b76c0: 09 XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b76d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b76e0: XX XX XX XX XX XX XX XX XX XX XX XX XX 81 | .....  
f94b76f0: 12 XX e0 81 19 XX e0 81 44 6f 6c 70 68 69 6e 31 | .....Dolphin1  
f94b7700: 38 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 | 8.....  
f94b7710: 70 52 b8 6b 96 7f XX XX XX XX XX XX XX XX XX | pR.k.....  
f94b7720: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b7730: XX XX XX XX 4a XX XX XX XX XX XX XX XX XX XX | .....J.....  
f94b7740: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b7750: XX XX XX XX XX XX XX XX XX XX e0 81 69 6e 73 74 | .....inst  
f94b7760: 61 5f 30 32 30 33 e5 e5 e5 e5 e5 e5 e5 e5 | a_0203.....  
f94b7770: 70 52 18 7d 28 7f XX XX XX XX XX XX XX XX XX | pR.}(.  
f94b7780: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b7790: XX XX XX XX 54 XX XX XX XX XX XX XX XX XX XX | .....T.....  
f94b77a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b77b0: XX XX XX XX XX XX XX XX XX XX XX XX XX 73 65 63 72 | .....secre  
f94b77c0: 65 74 70 77 64 30 e5 e5 e5 e5 e5 e5 e5 e5 | etpwd0.....  
f94b77d0: 30 b4 18 7d 28 7f XX XX XX XX XX XX XX XX XX | 0..}(.  
f94b77e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b77f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX | .....  
f94b7800: e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 e5 | .....  
f94b7810: 68 74 74 70 73 3a 2f 2f 61 64 64 6f 6e 73 2e 63 | https://addons.c/  
f94b7820: 64 6e 2e 6d 6f 7a 69 6c 6c 61 2e 6e 65 74 2f 75 | dn.mozilla.net/u/  
f94b7830: 73 65 72 2d 6d 65 64 69 61 2f 61 64 64 6f 6e 5f | ser-media/addon_  
f94b7840: 69 63 6f 6e 73 2f 33 35 34 2f 33 35 34 33 39 39 | icons/354/354399/  
f94b7850: 2d 36 34 2e 70 6e 67 3f 6d 6f 64 69 66 69 65 64 | -64.png?modified=  
f94b7860: 3d 31 34 35 32 32 34 34 38 31 35 XX XX XX XX XX | =1452244815.....
```

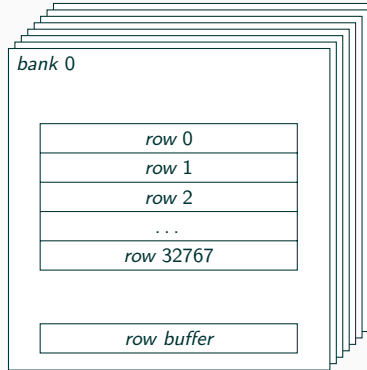








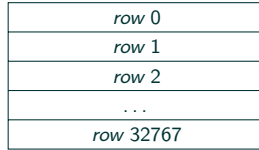
chip



chip



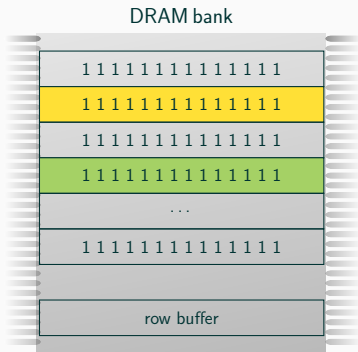
bank 0



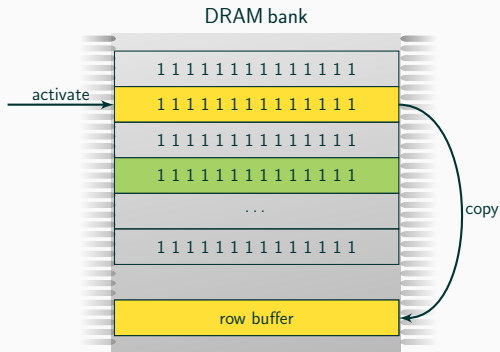
row buffer



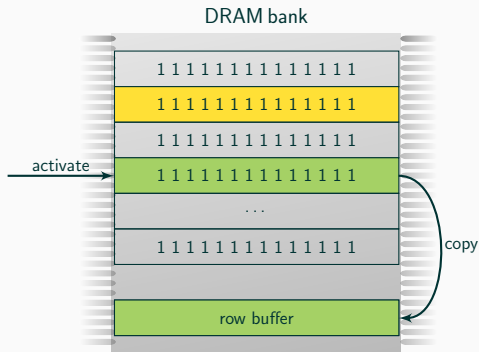
64k cells
1 capacitor,
1 transistor each



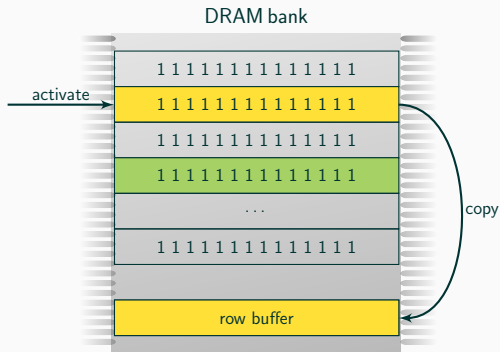
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer



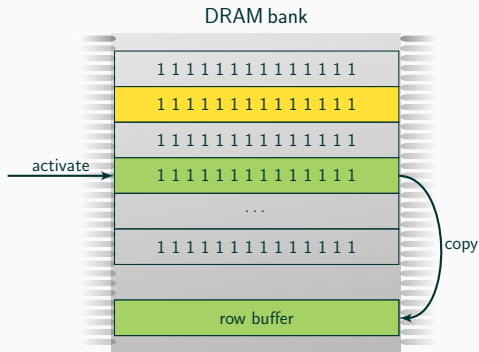
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer



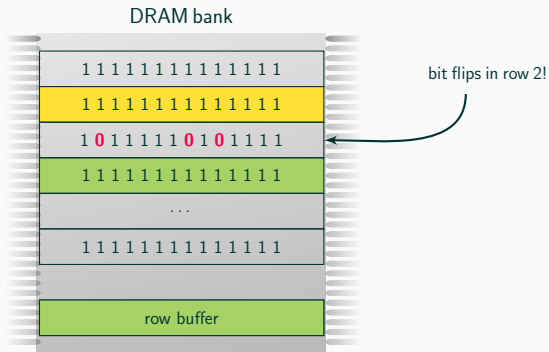
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer



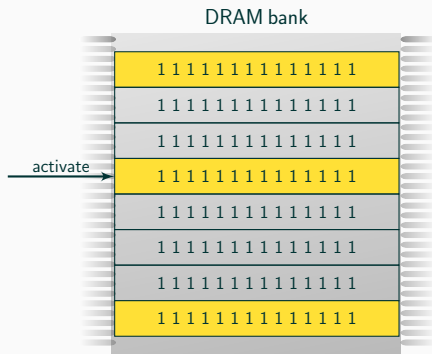
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer

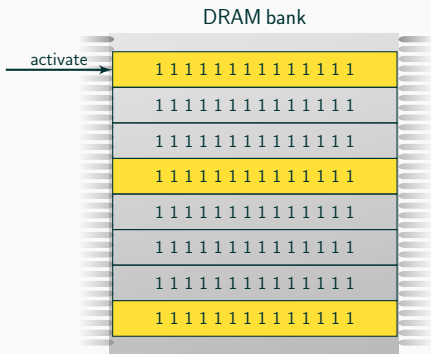


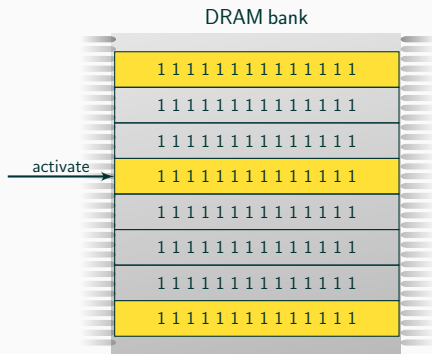
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer

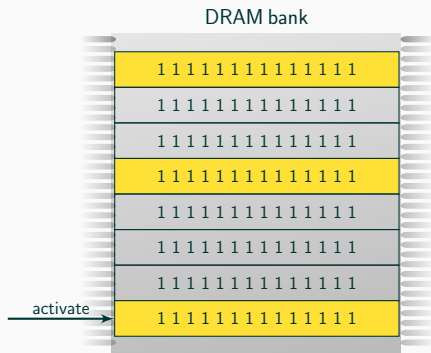


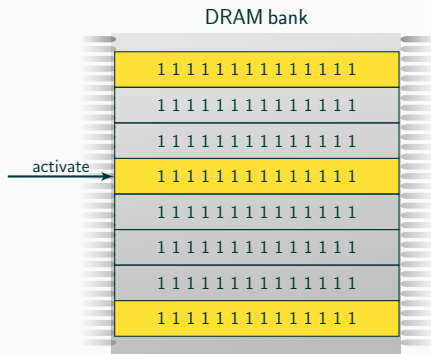
- Cells leak → repetitive **refresh** necessary
- Maximum interval between refreshes to guarantee **data integrity**
- Cells leak faster upon proximate accesses → Rowhammer

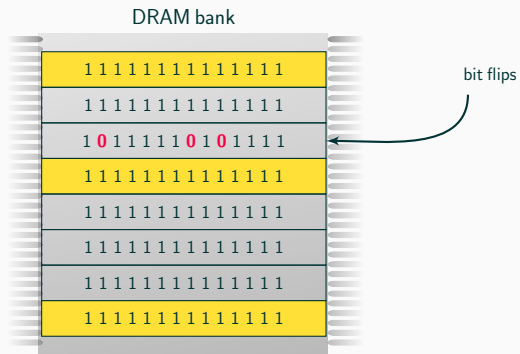


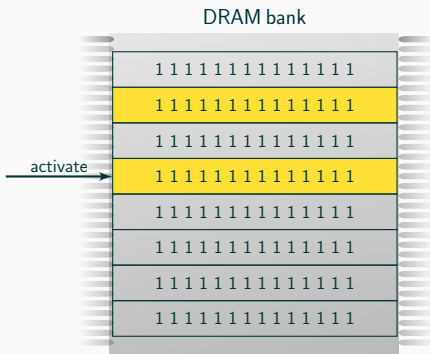


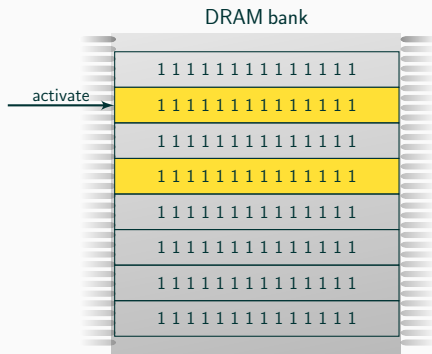


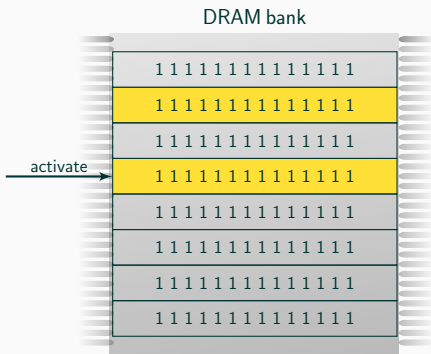


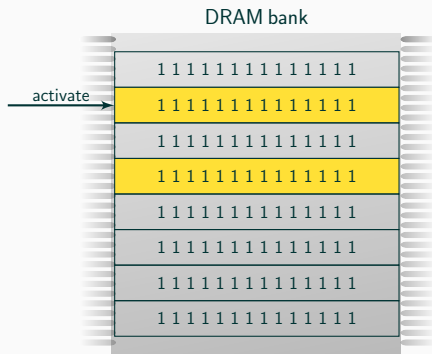


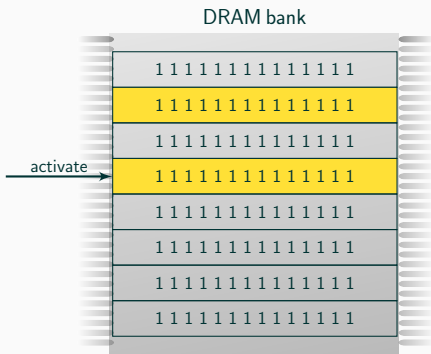


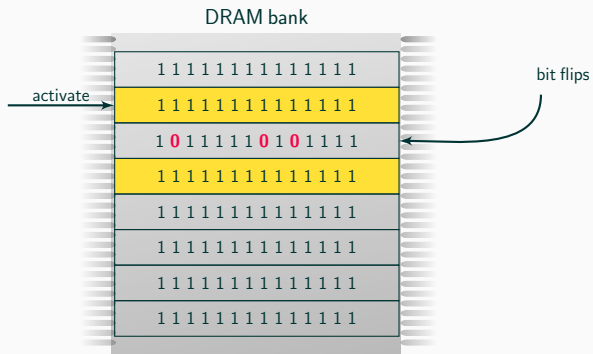


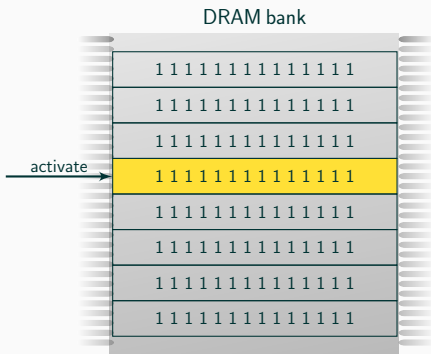


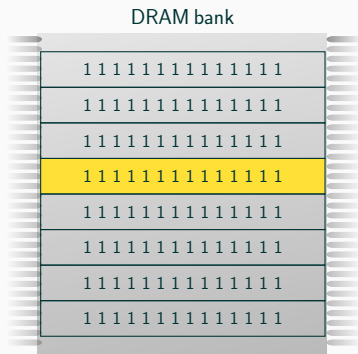


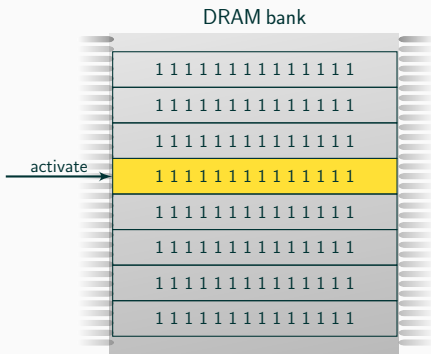


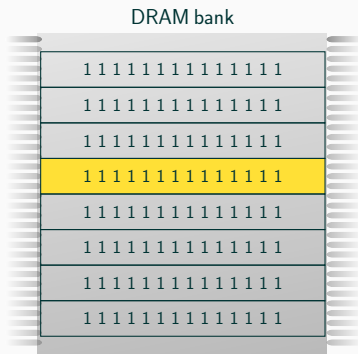


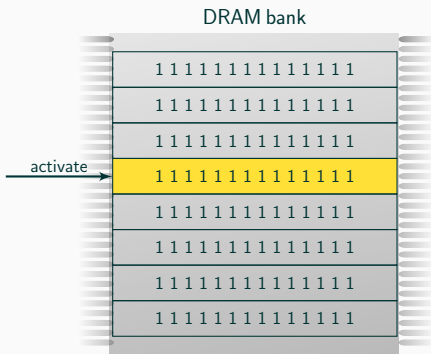


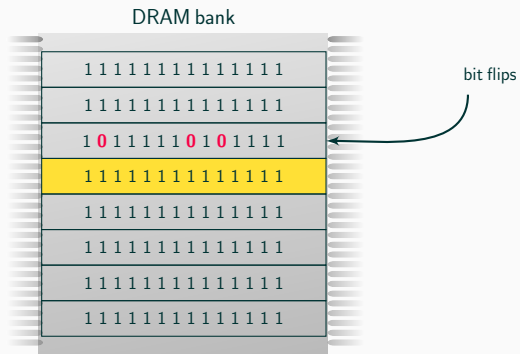












- They are not random → highly reproducible flip pattern!
 1. Choose a (kernel) data structure that you can place at arbitrary memory locations
 2. Scan for “good” flips
 3. Place (kernel) data structure there
 4. Trigger bit flip again

- Many applications perform actions as root

- Many applications perform actions as root
- They can be used by unprivileged users as well

- Many applications perform actions as root
- They can be used by unprivileged users as well
- `sudo`



















We have ignored microarchitectural attacks for many many years:



We have ignored microarchitectural attacks for many many years:

- attacks on crypto



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
- Rowhammer attacks



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
- Rowhammer attacks → “only affects cheap sub-standard modules”



We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
 - attacks on ASLR → “ASLR is broken anyway”
 - attacks on SGX and TrustZone → “not part of the threat model”
 - Rowhammer attacks → “only affects cheap sub-standard modules”
- for years we solely optimized for performance



After learning about a side channel you realize:



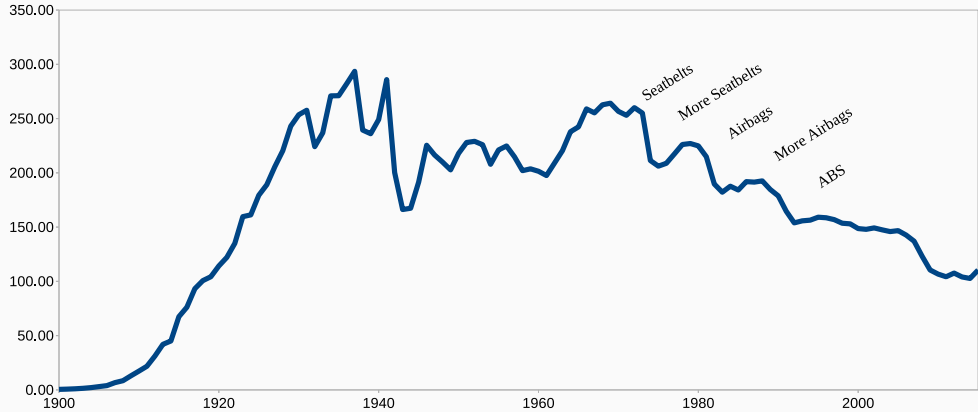
After learning about a side channel you realize:

- the side channels were documented in the Intel manual



After learning about a side channel you realize:

- the side channels were documented in the Intel manual
- only now we understand the implications



Motor Vehicle Deaths in U.S. by Year





- moral obligation to invest more time on defenses than on attacks



- moral obligation to invest more time on defenses than on attacks
- **dangerous**: we overlooked Meltdown and Spectre for decades



- moral obligation to invest more time on defenses than on attacks
- **dangerous**: we overlooked Meltdown and Spectre for decades
- we don't know all problems. do we know at least the most important subset?



- moral obligation to invest more time on defenses than on attacks
- **dangerous**: we overlooked Meltdown and Spectre for decades
- we don't know all problems. do we know at least the most important subset?
- are we hammering on a small subset of problems and forgot about the bigger picture?





A unique chance to

- rethink processor design



A unique chance to

- rethink processor design
- grow up, like other fields (car industry, construction industry)



A unique chance to

- rethink processor design
- grow up, like other fields (car industry, construction industry)
- find good trade-offs between security and performance



A unique chance to

- rethink processor design
- grow up, like other fields (car industry, construction industry)
- find good trade-offs between security and performance
- dedicate more time into identifying problems and not solely in mitigating known problems

Microarchitectural Attacks: From the Basics to Arbitrary Read and Write Primitives without any Software Bugs

Daniel Gruss

April 11, 2018

Graz University of Technology