Practical Memory Deduplication Attacks in Sandboxed JavaScript

Daniel Gruss, David Bidner, and Stefan Mangard
IAIK, Graz University of Technology

September 23, 2015

Overview

- Page deduplication not only a problem in the cloud
- Can be used to eavesdrop on browser usage

Overview

- Page deduplication not only a problem in the cloud
- Can be used to eavesdrop on browser usage

The first page deduplication attack,

- in sandboxed JavaScript,
- on personal computers and smartphones,
- through malicious websites.

Overview

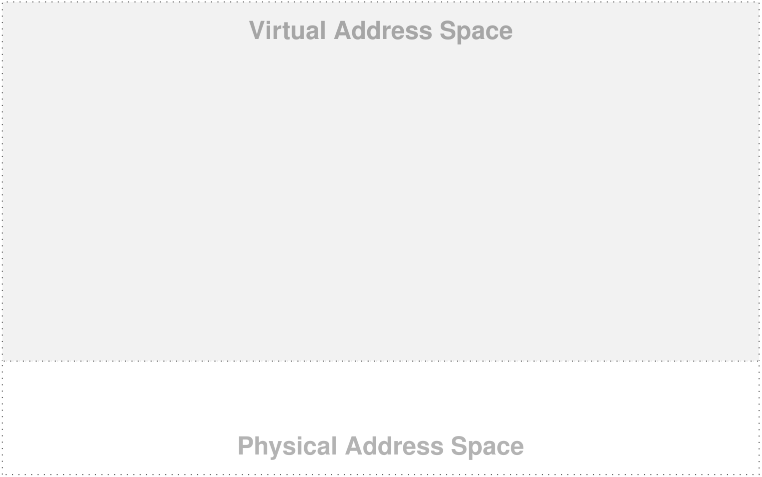
- Page deduplication not only a problem in the cloud
- Can be used to eavesdrop on browser usage

The first page deduplication attack,

- in sandboxed JavaScript,
- on personal computers and smartphones,
- through malicious websites.

Large scale remote attacks possible

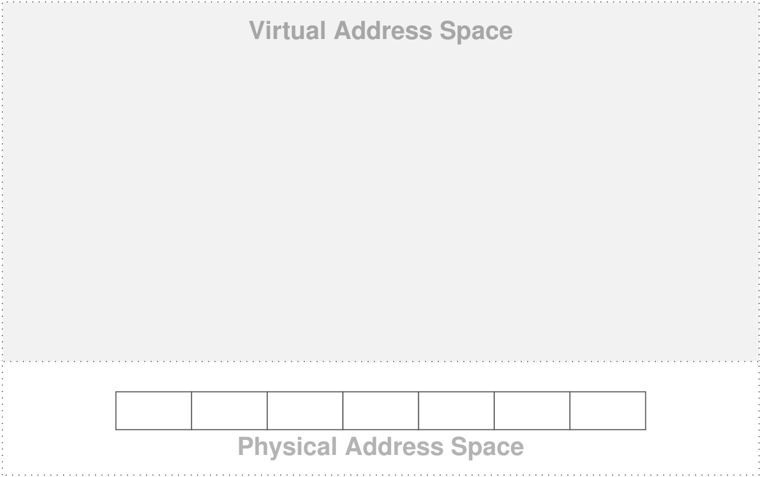
Copy-on-Write



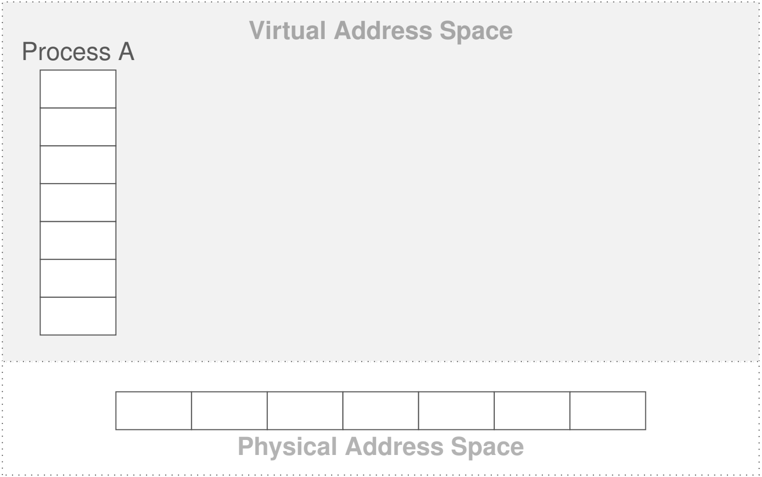
Virtual Address Space

Physical Address Space

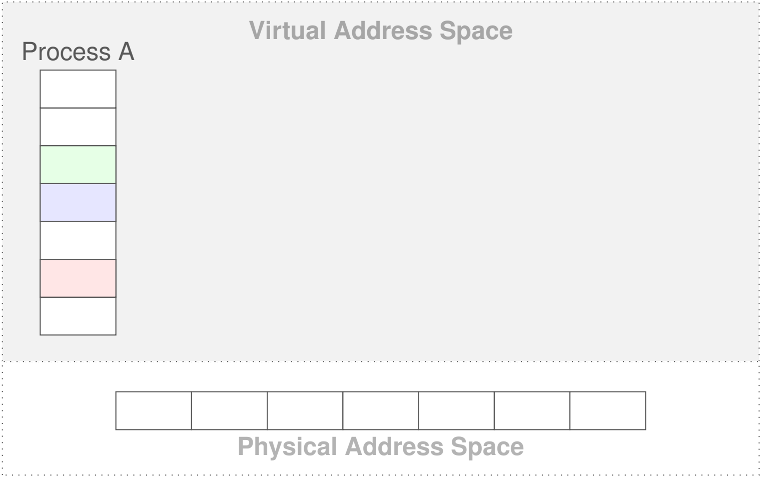
Copy-on-Write



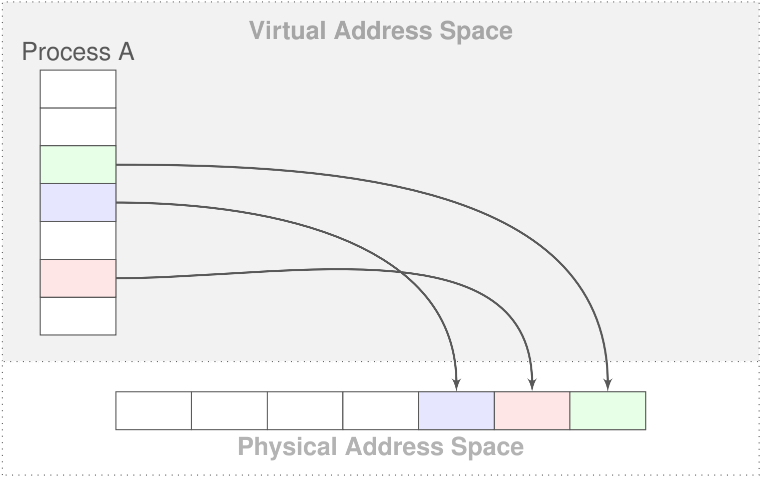
Copy-on-Write



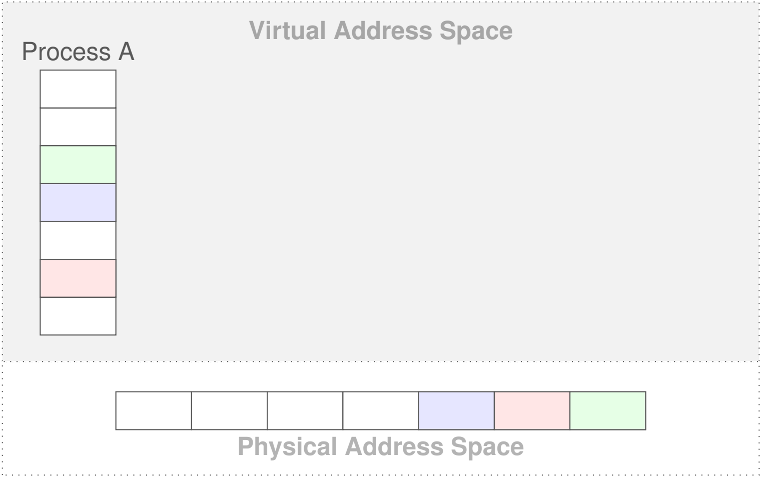
Copy-on-Write



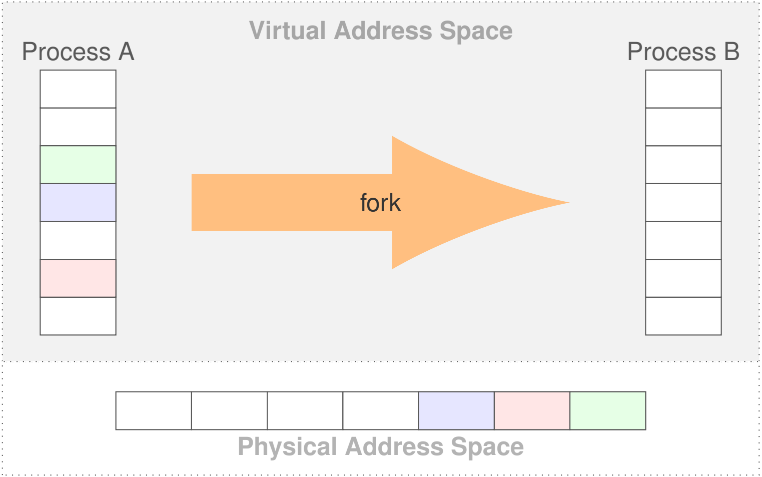
Copy-on-Write



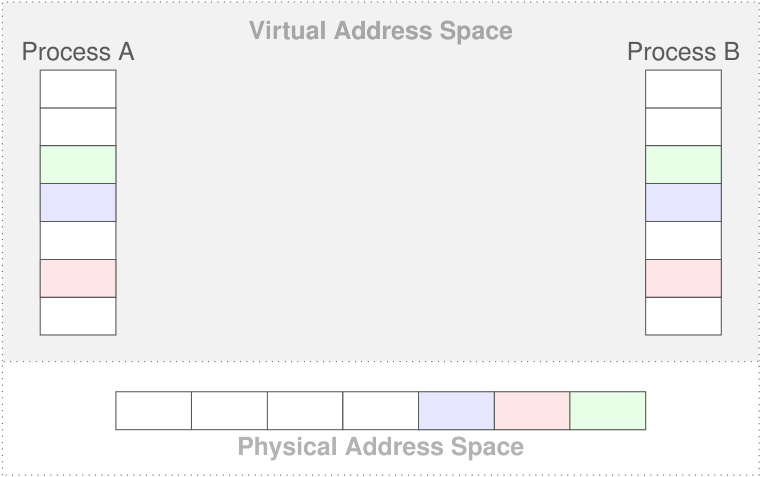
Copy-on-Write



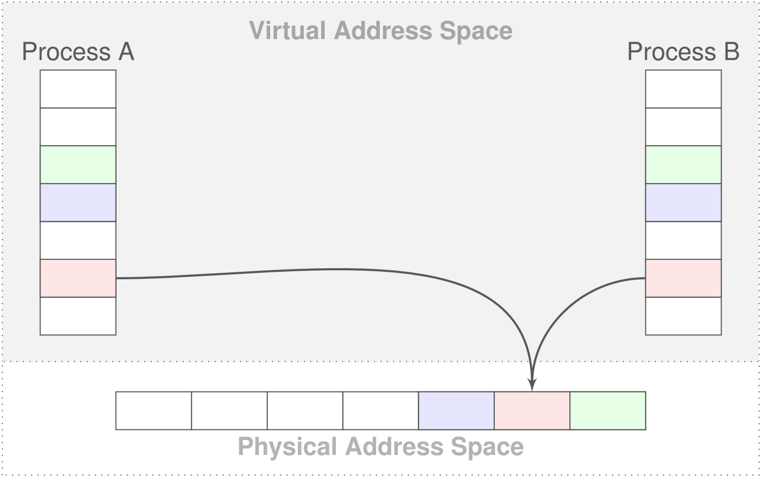
Copy-on-Write



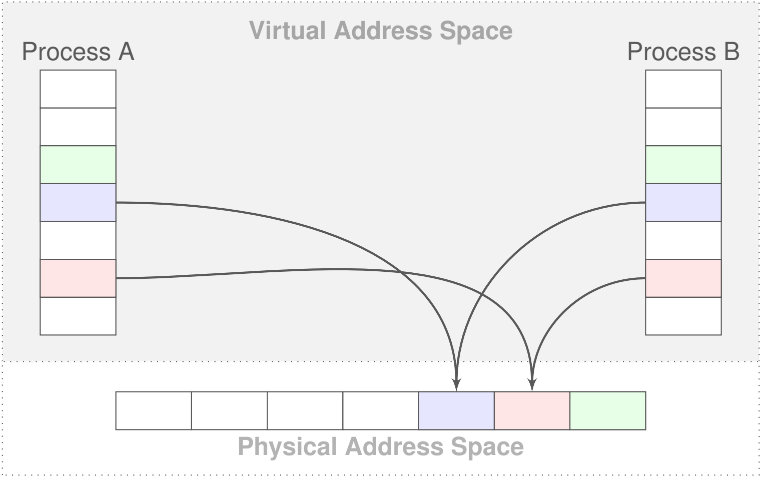
Copy-on-Write



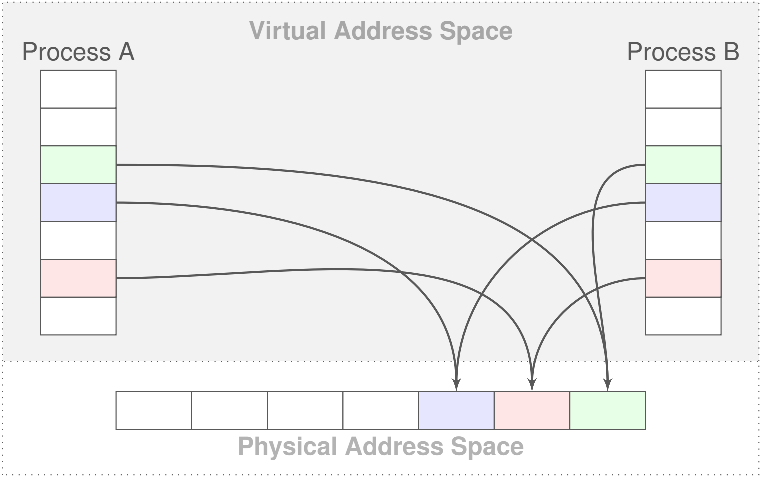
Copy-on-Write



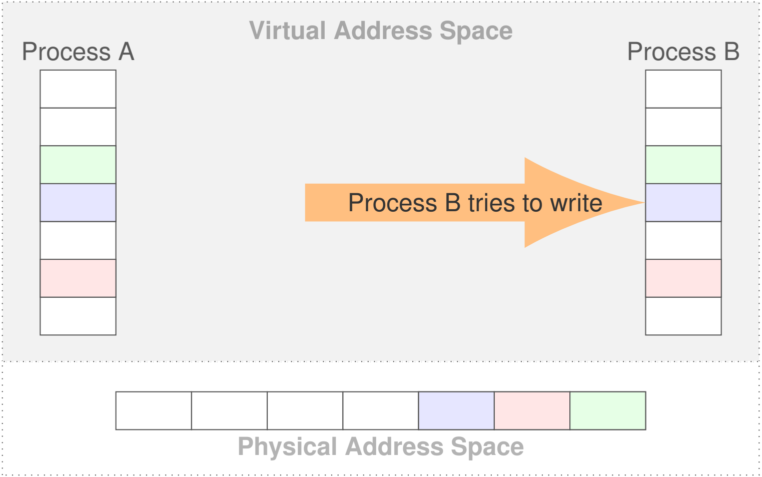
Copy-on-Write



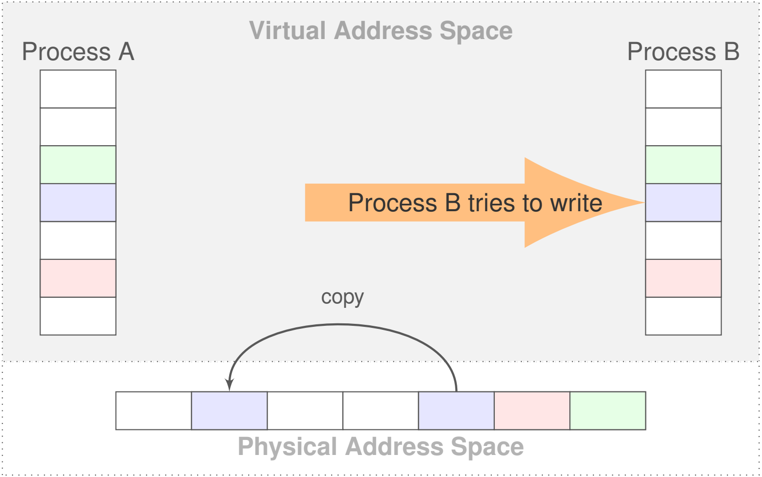
Copy-on-Write



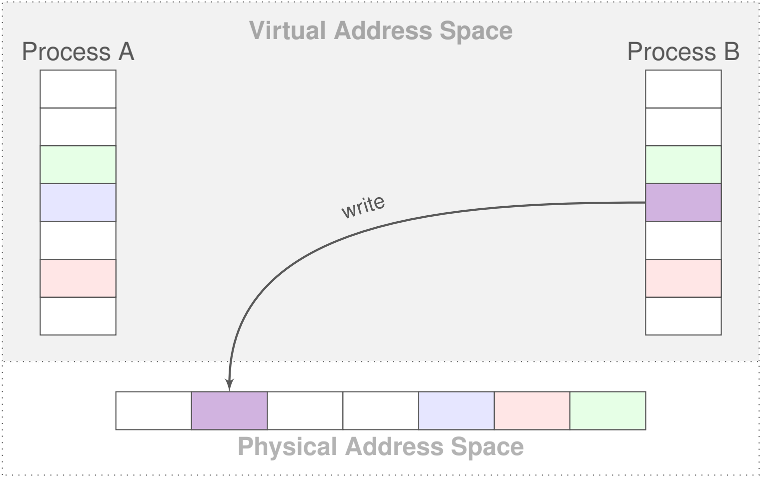
Copy-on-Write



Copy-on-Write



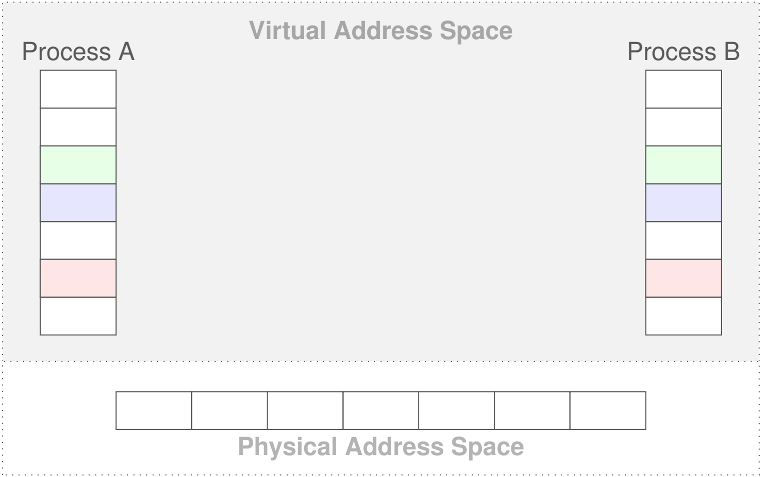
Copy-on-Write



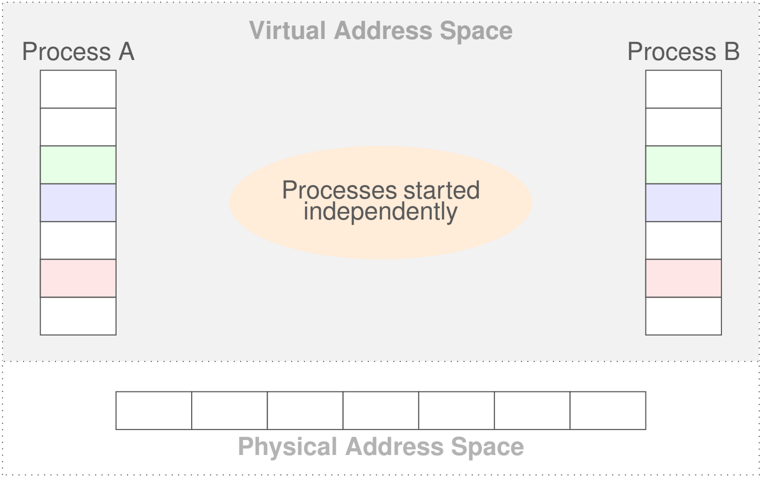
Write vs. Copy-on-Write

- Regular write access $< 0.2\mu s$
- Write access with copy-on-write pagefault $> 3.0\mu s$
- Clearly distinguishable

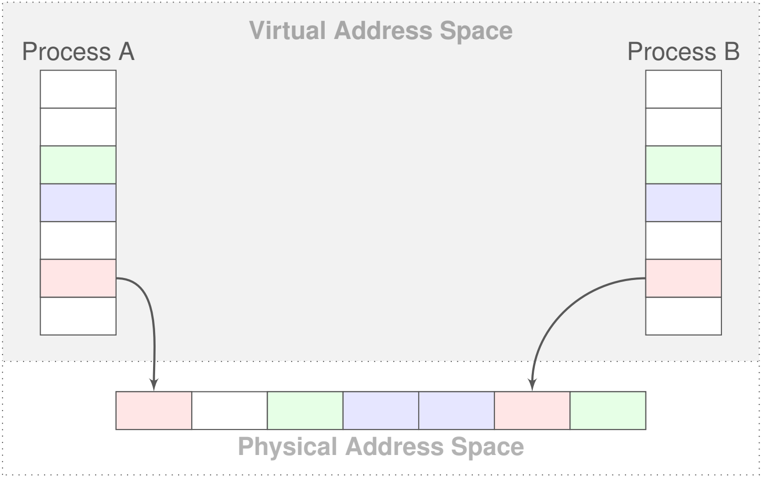
Page Deduplication



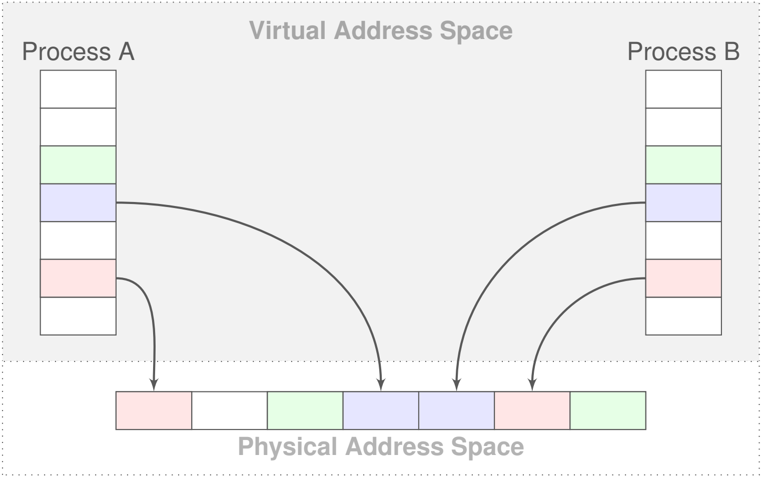
Page Deduplication



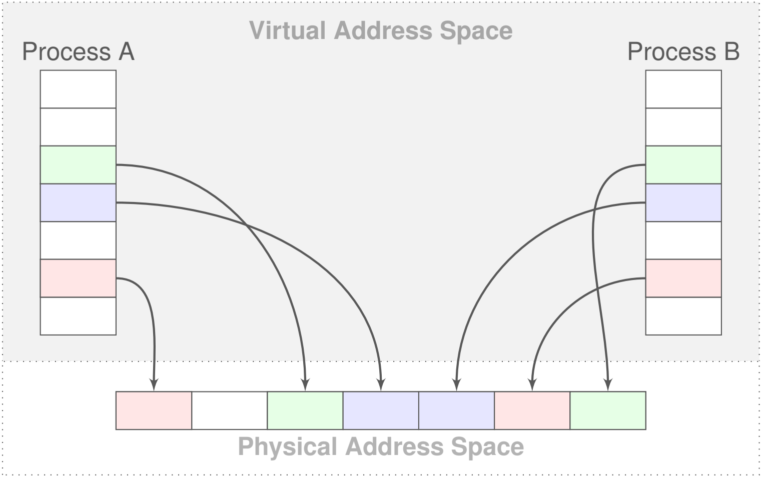
Page Deduplication



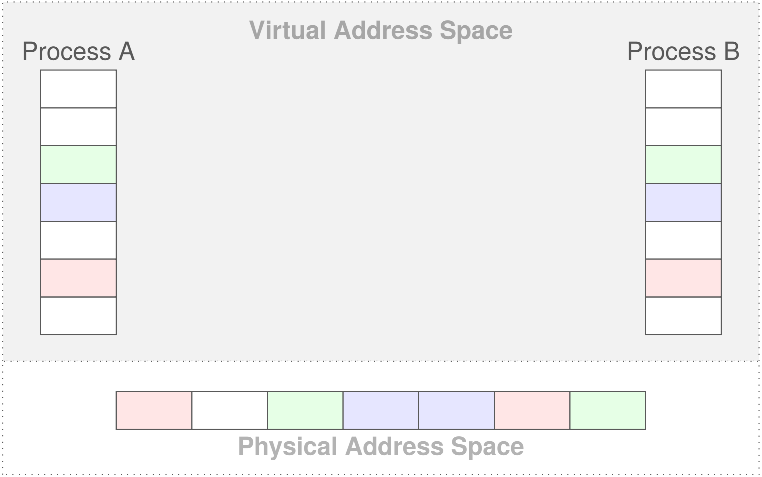
Page Deduplication



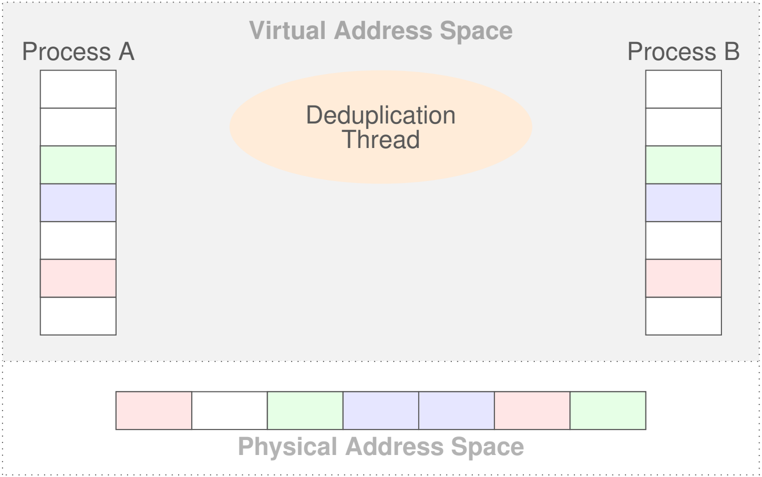
Page Deduplication



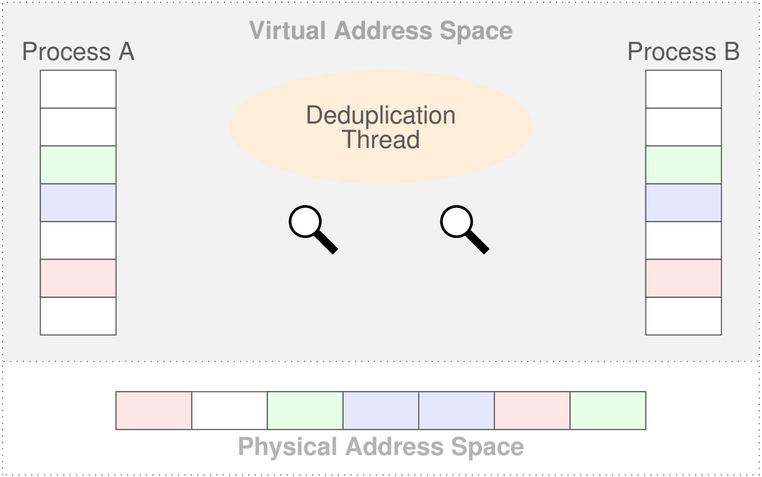
Page Deduplication



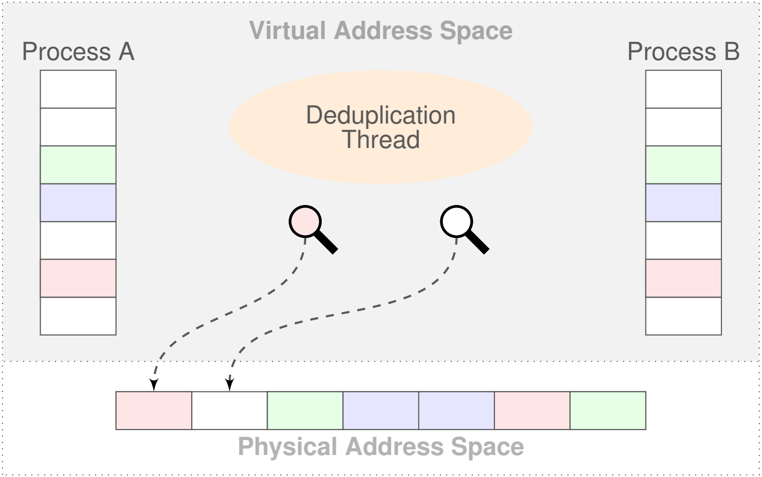
Page Deduplication



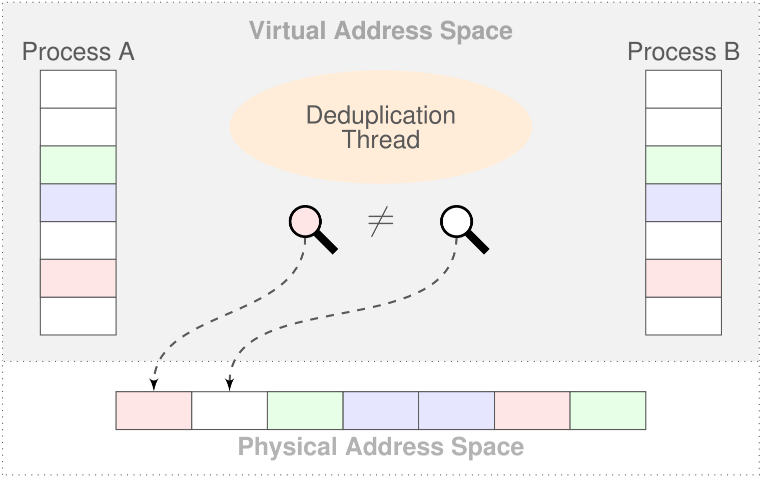
Page Deduplication



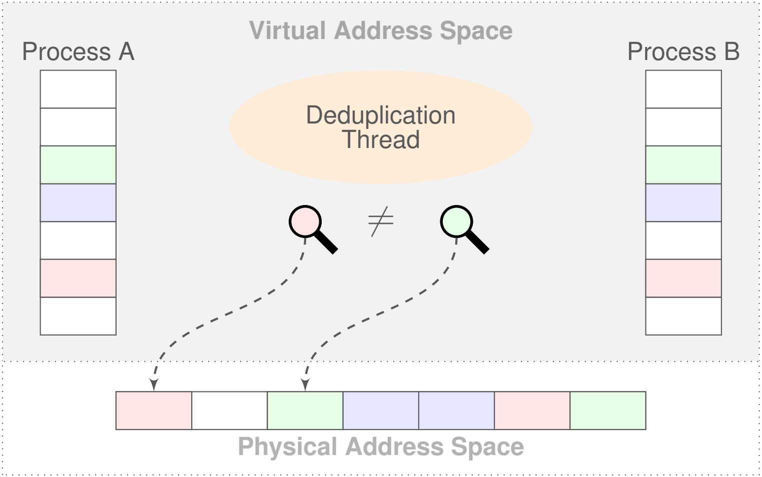
Page Deduplication



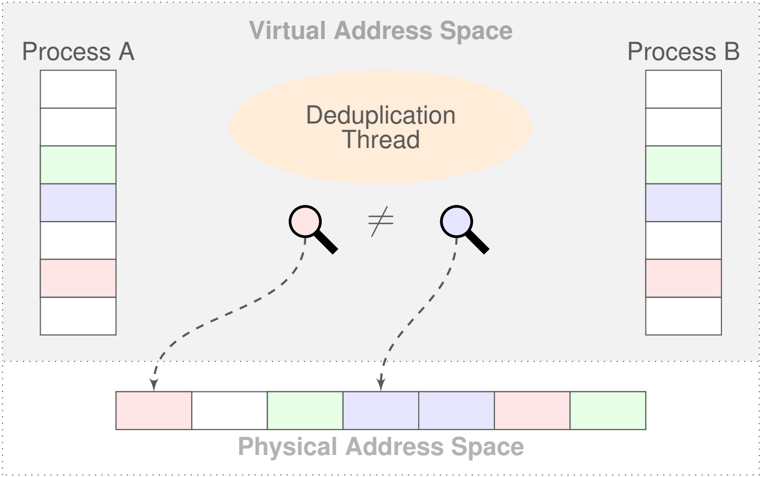
Page Deduplication



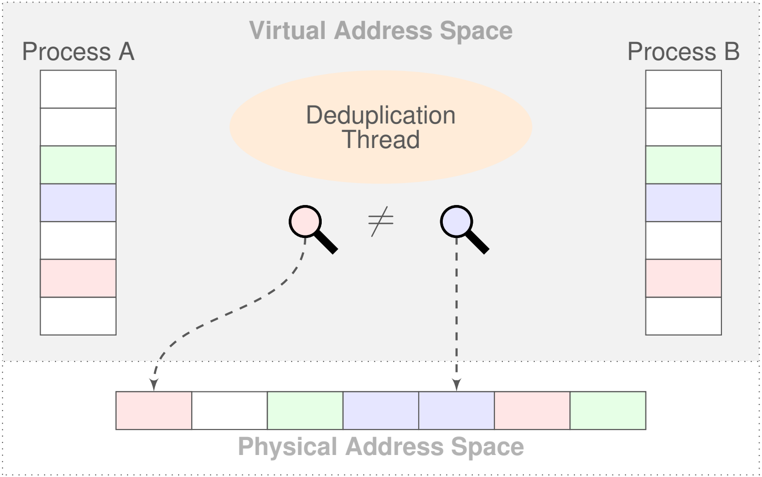
Page Deduplication



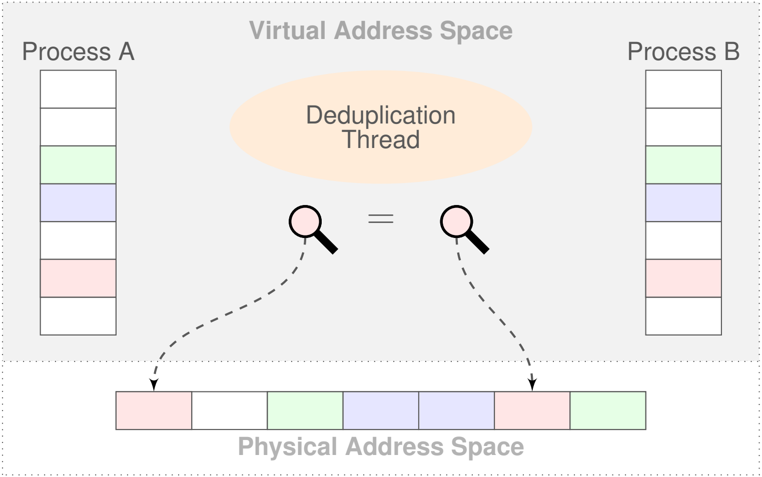
Page Deduplication



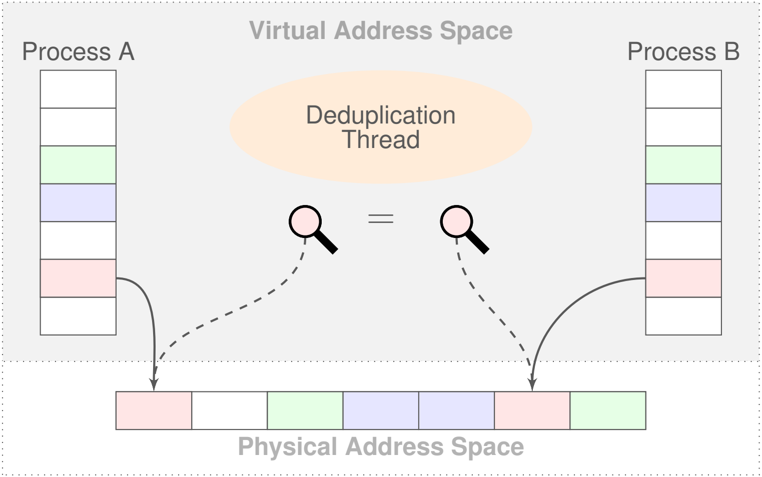
Page Deduplication



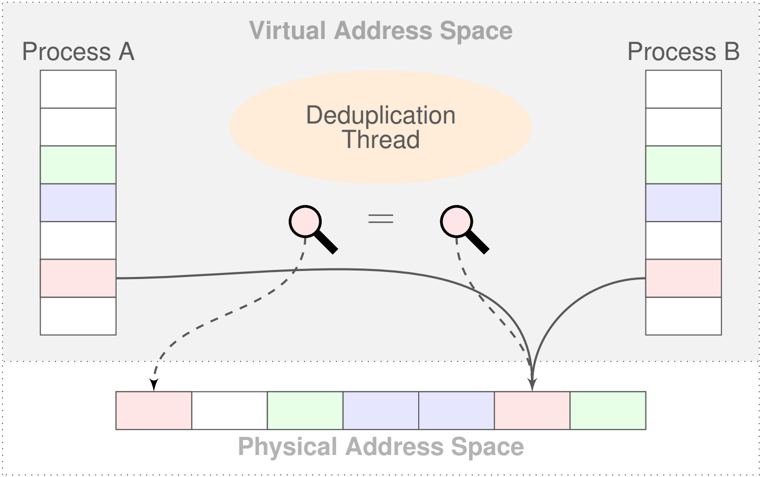
Page Deduplication



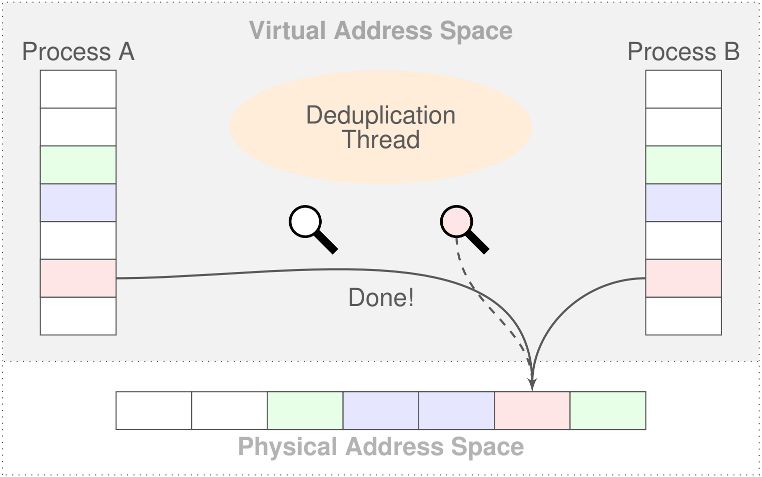
Page Deduplication



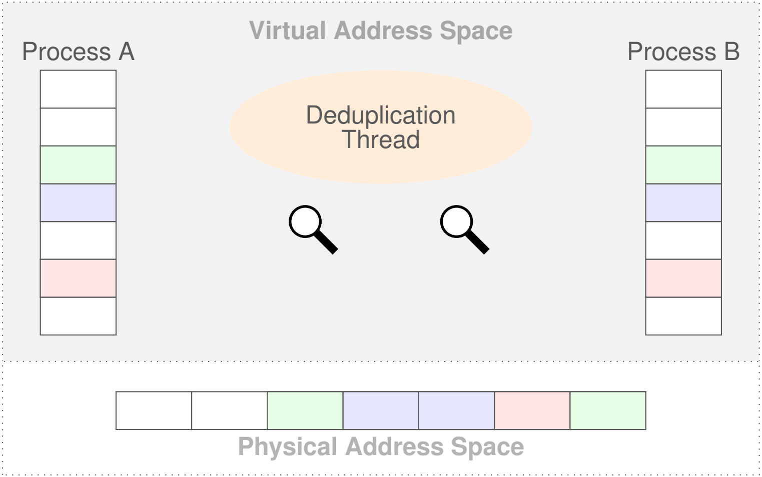
Page Deduplication



Page Deduplication



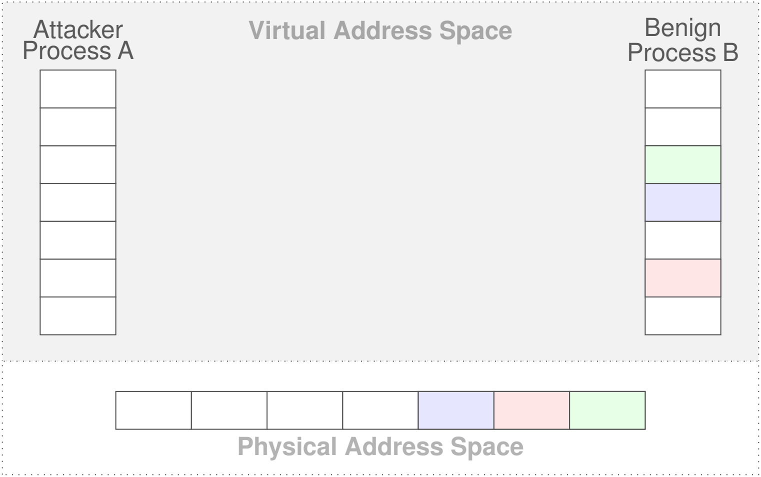
Page Deduplication



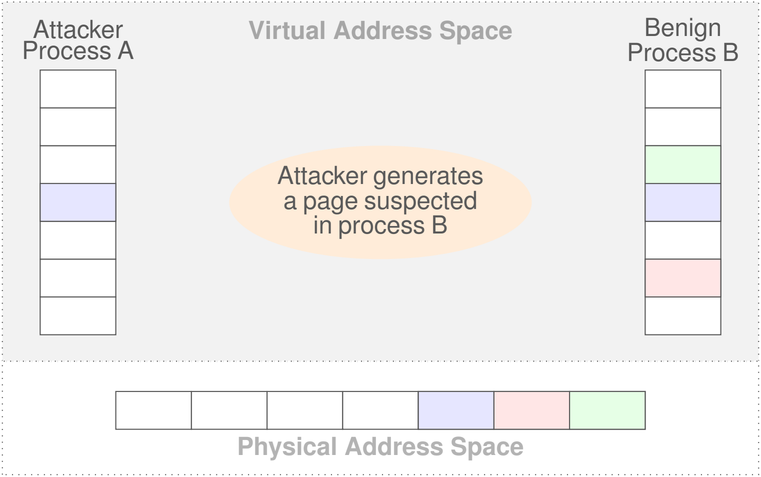
Page Deduplication

- Deduplication between processes:
    1. in same OS instance (Android, Windows)
    2. in different VMs (KVM, VMWare, ...)
- Code pages, data pages - even kernel pages
- Time until deduplication 2-45 minutes
    - depends on system configuration

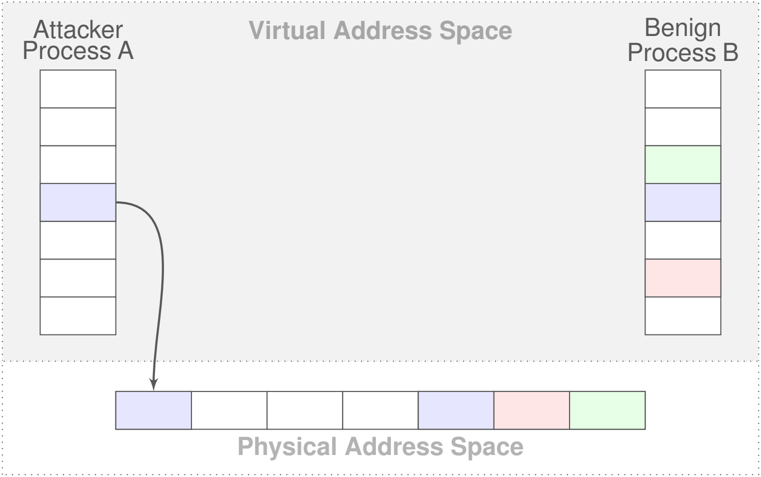
Page Deduplication Attack



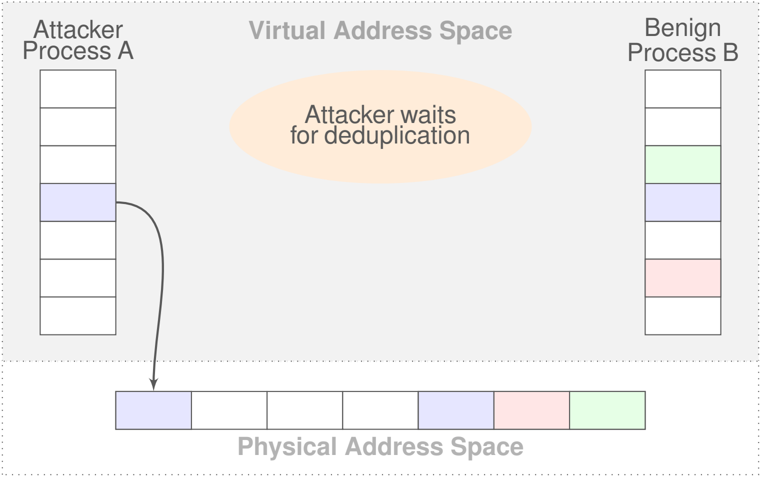
Page Deduplication Attack



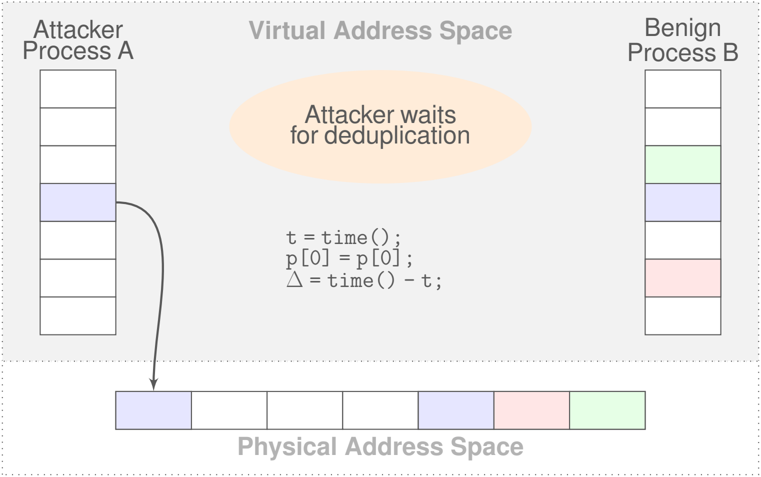
Page Deduplication Attack



Page Deduplication Attack

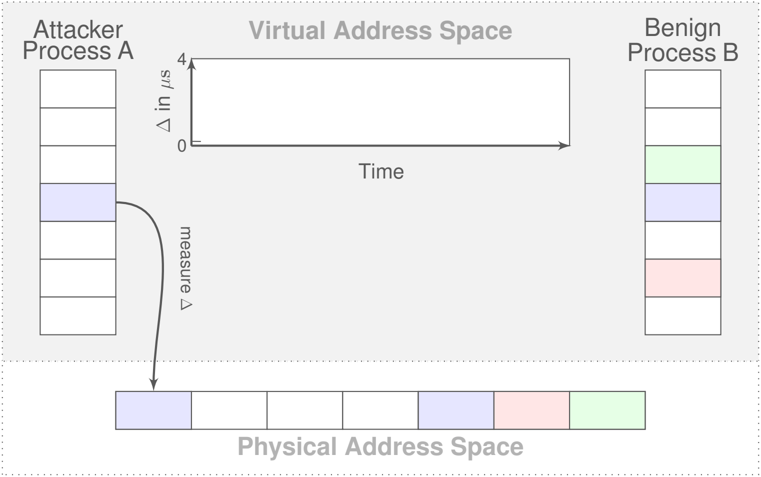


Page Deduplication Attack

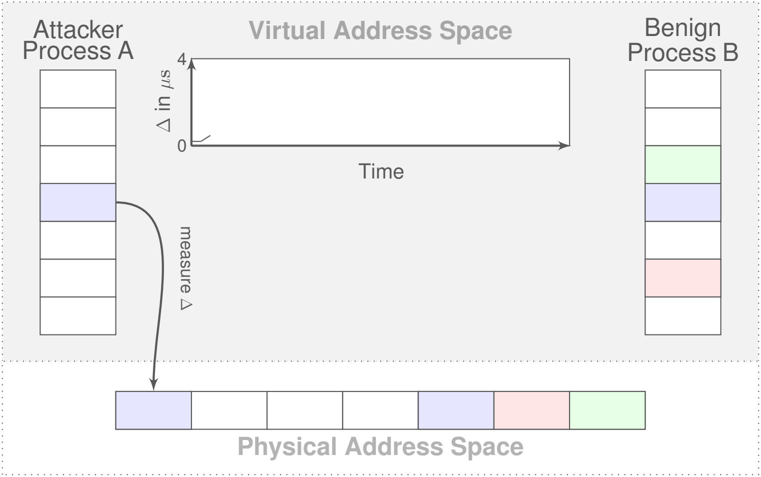


```
t = time();
p[0] = p[0];
△ = time() - t;
```

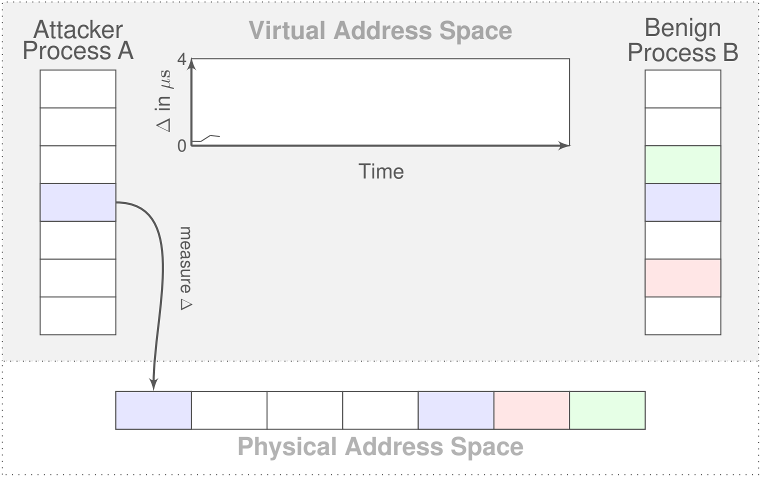
Page Deduplication Attack



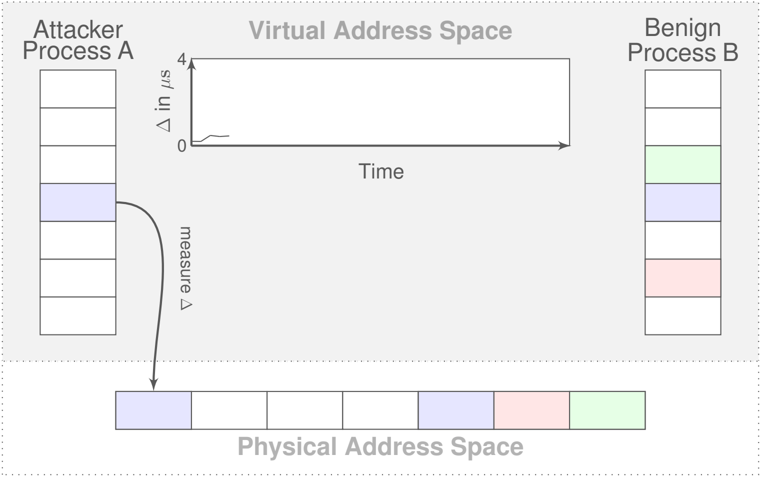
Page Deduplication Attack



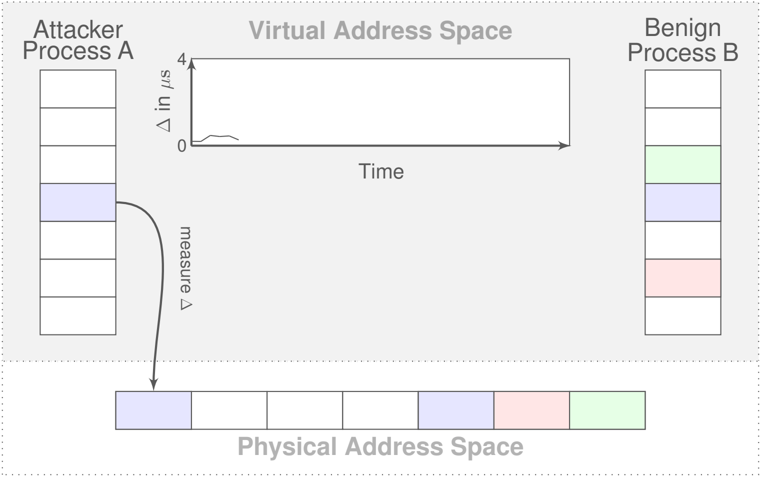
Page Deduplication Attack



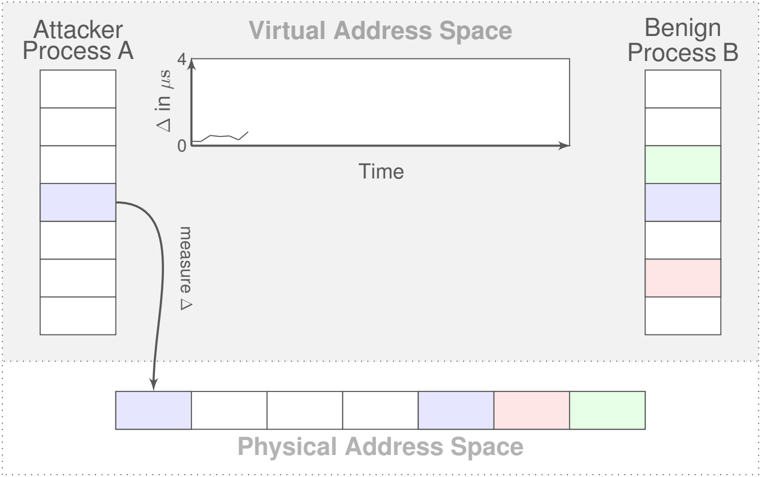
Page Deduplication Attack



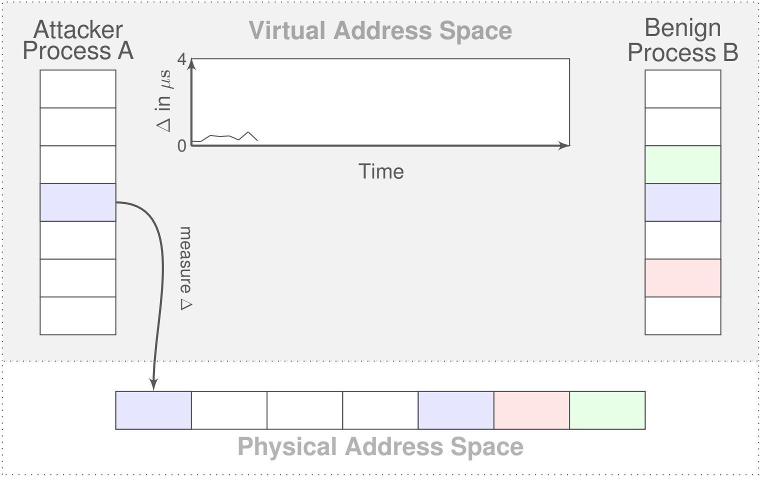
Page Deduplication Attack



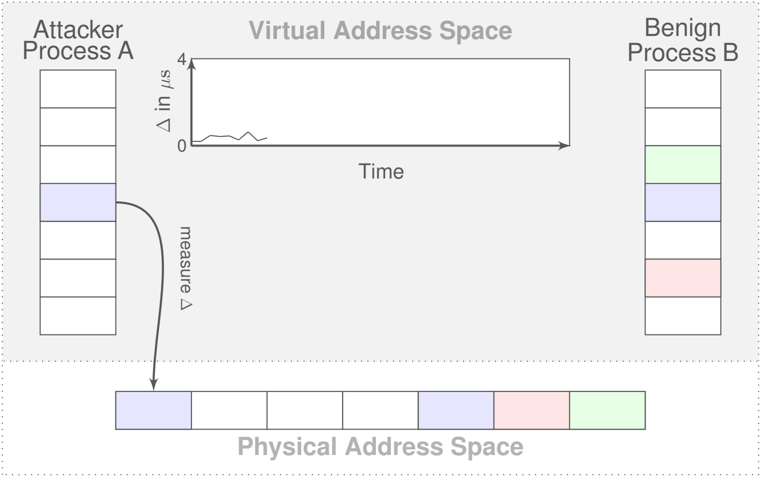
Page Deduplication Attack



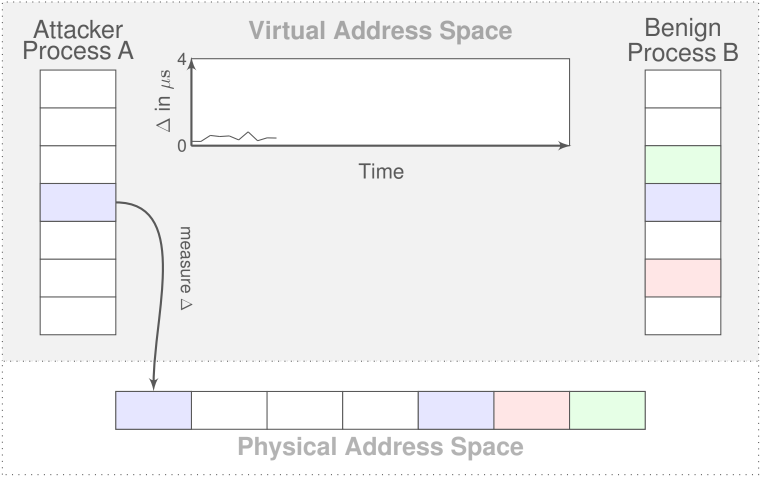
Page Deduplication Attack



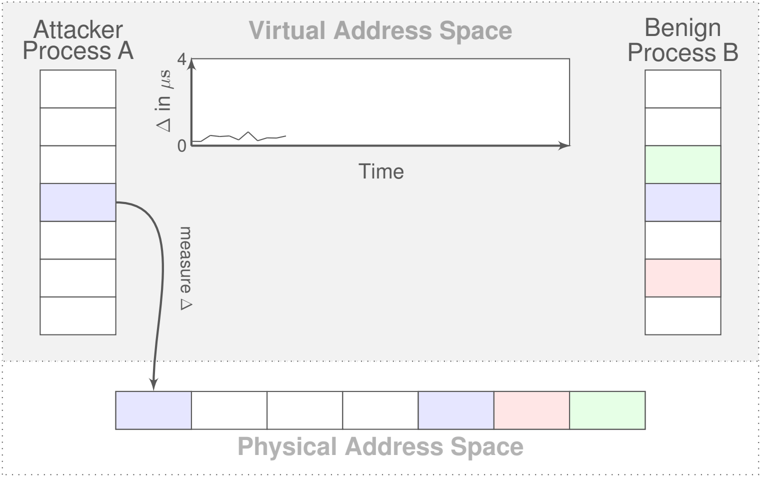
Page Deduplication Attack



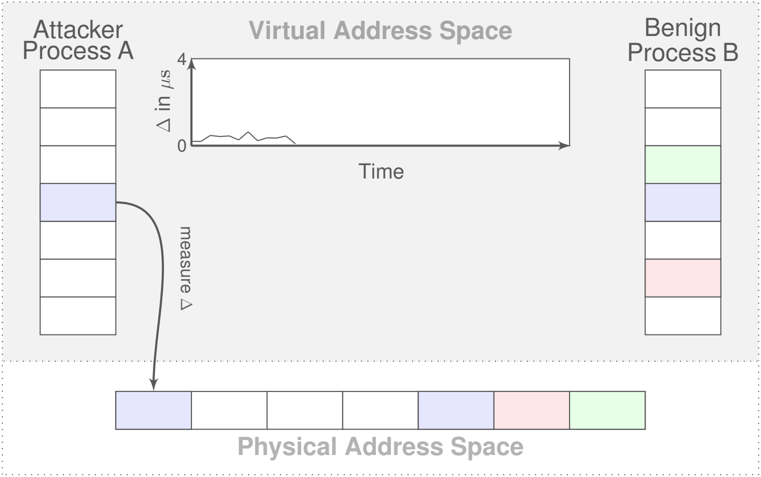
Page Deduplication Attack



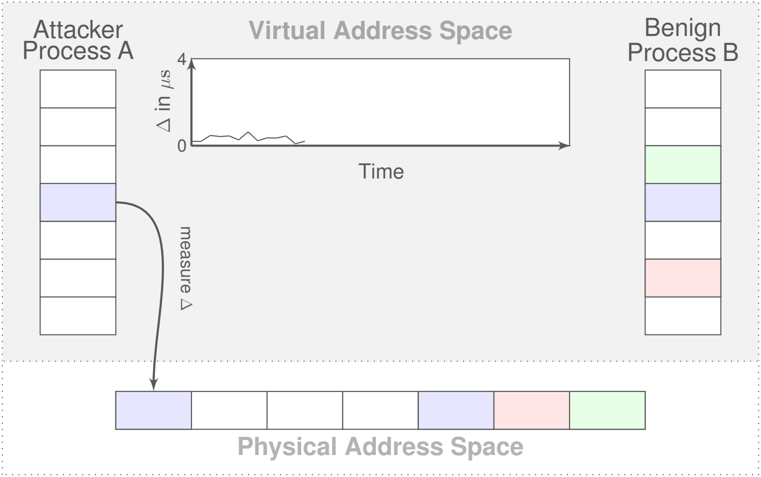
Page Deduplication Attack



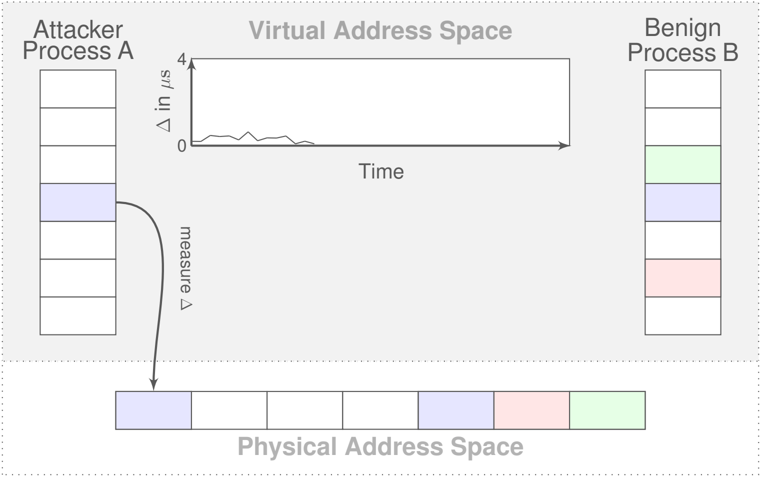
Page Deduplication Attack



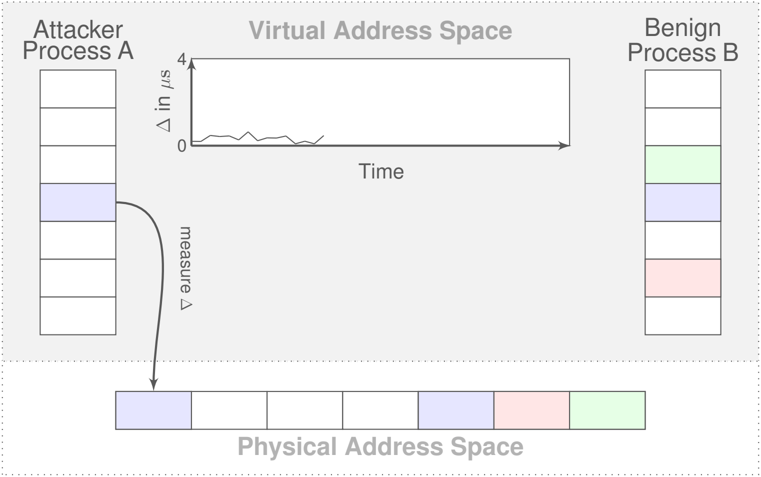
Page Deduplication Attack



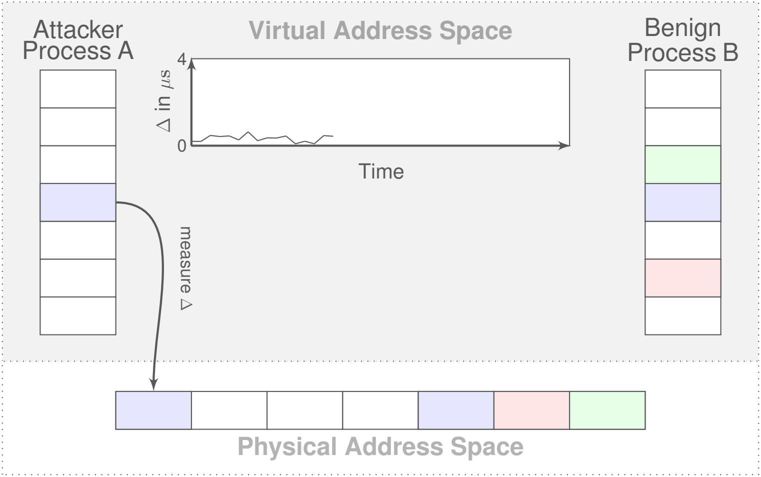
Page Deduplication Attack



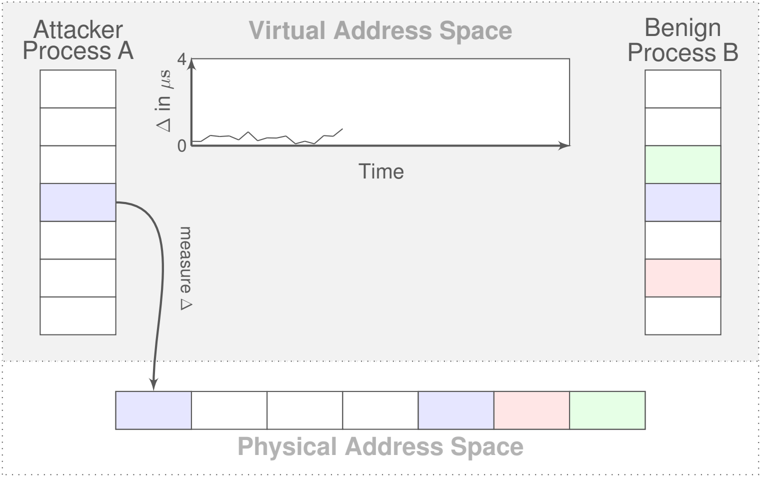
Page Deduplication Attack



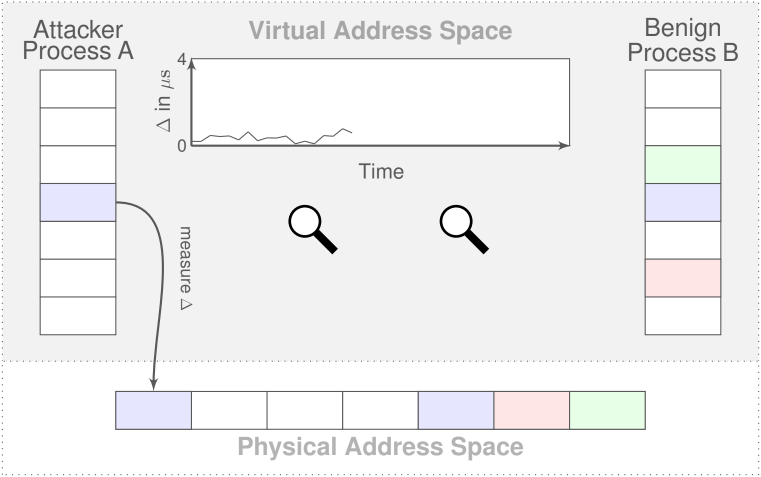
Page Deduplication Attack



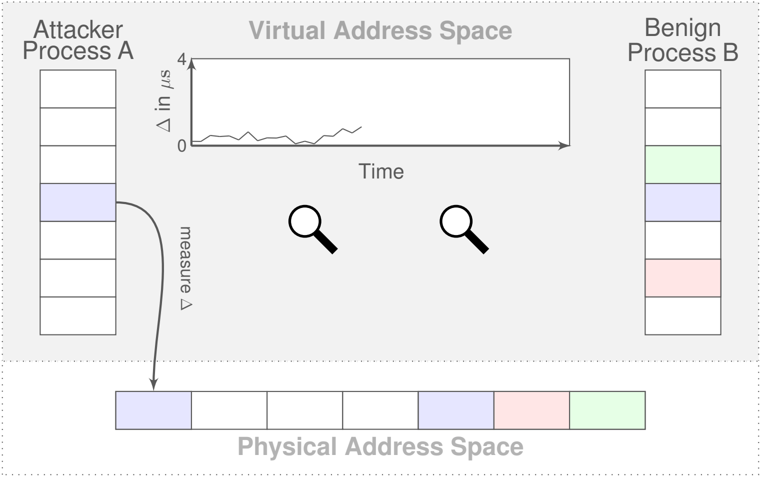
Page Deduplication Attack



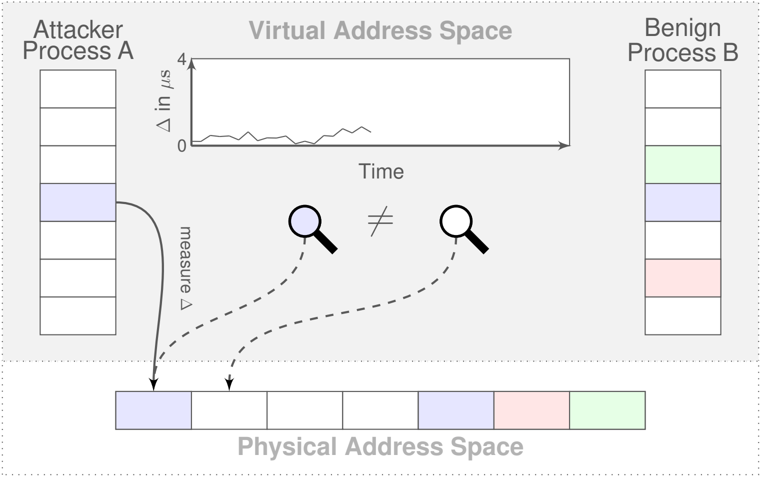
Page Deduplication Attack



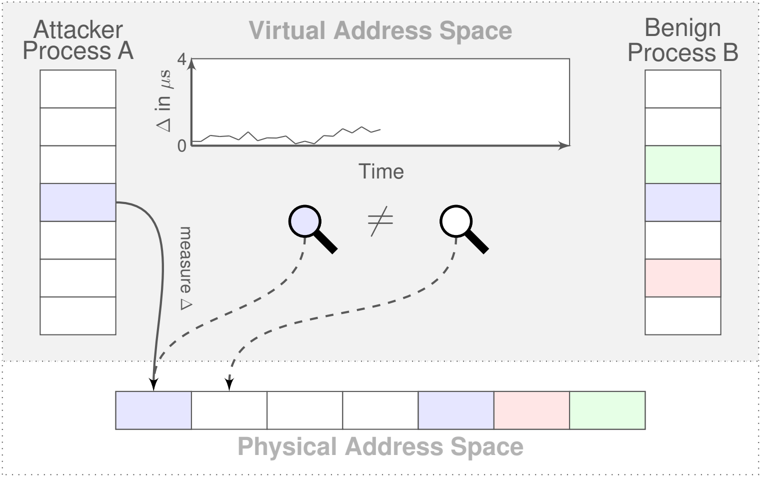
Page Deduplication Attack



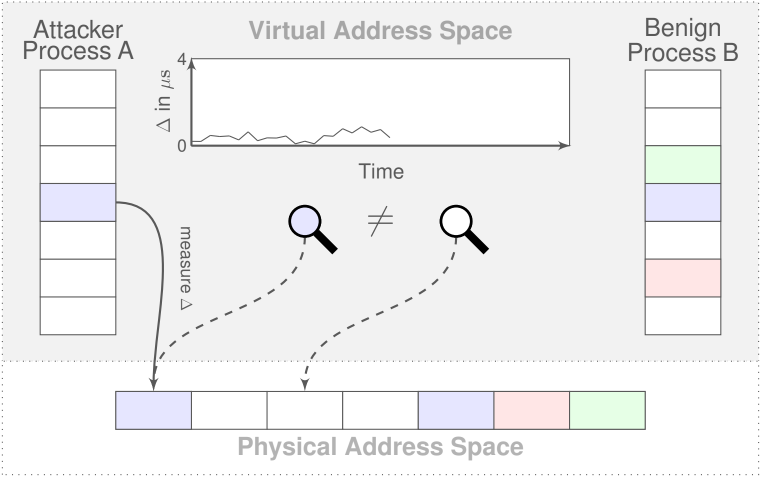
Page Deduplication Attack



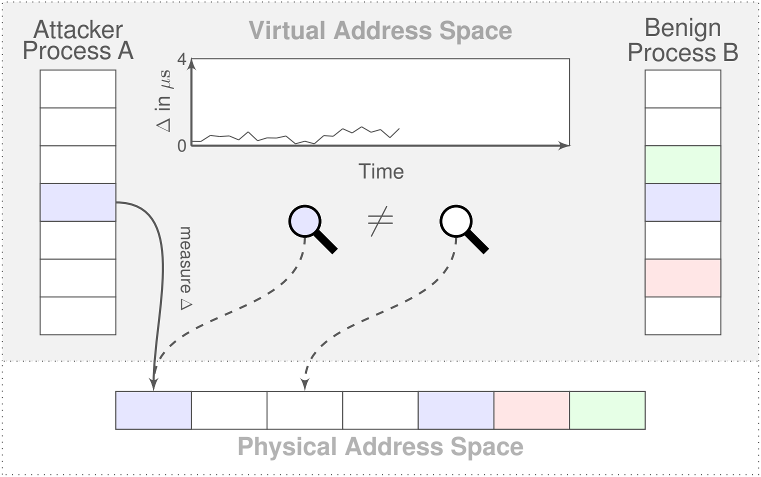
Page Deduplication Attack



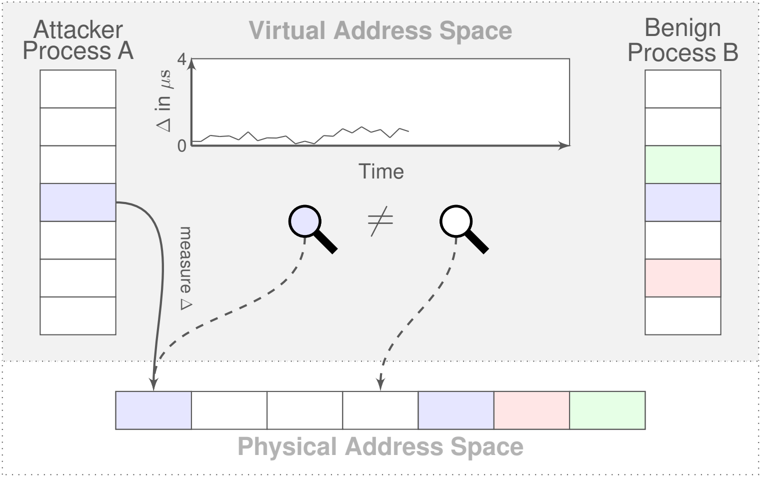
Page Deduplication Attack



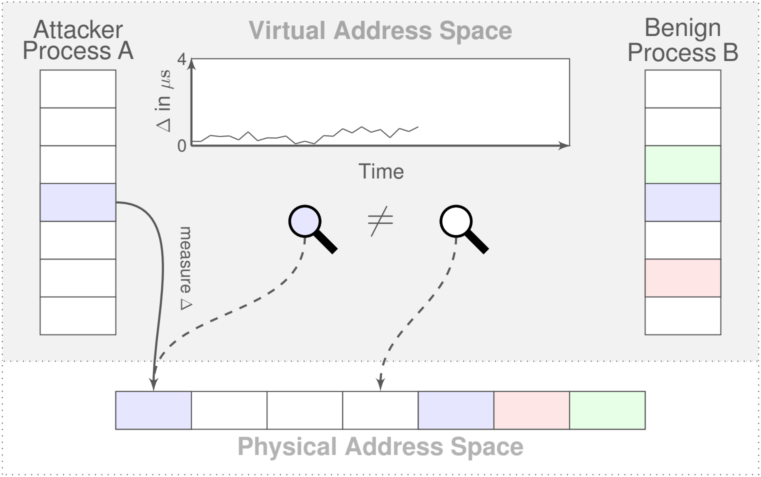
Page Deduplication Attack



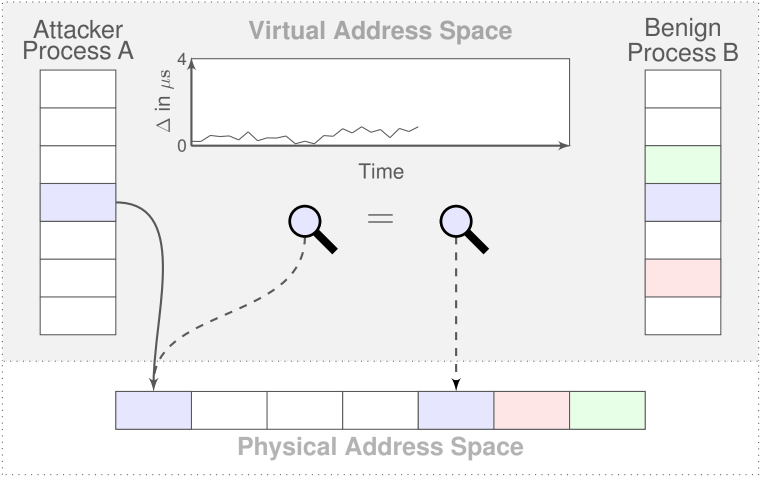
Page Deduplication Attack



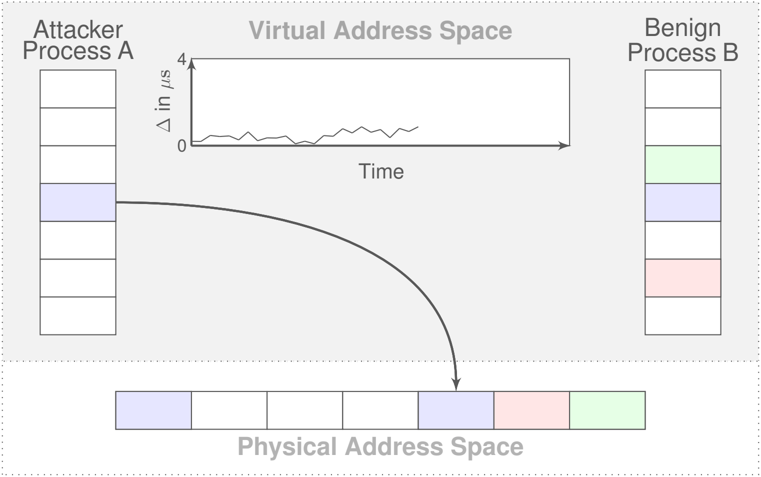
Page Deduplication Attack



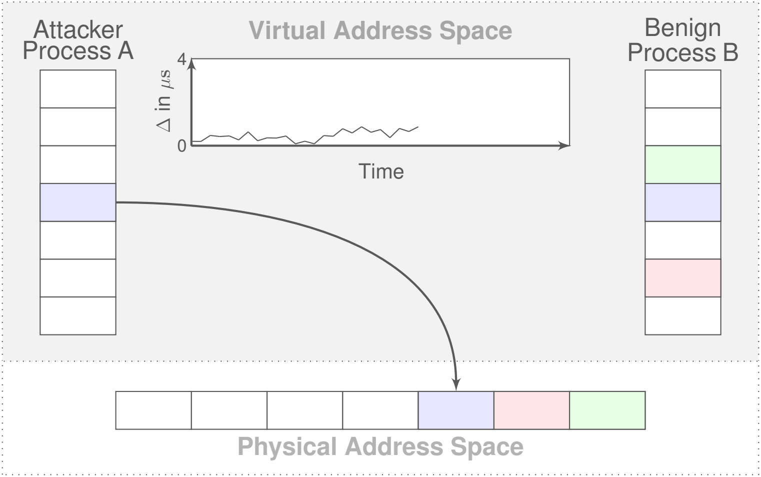
Page Deduplication Attack



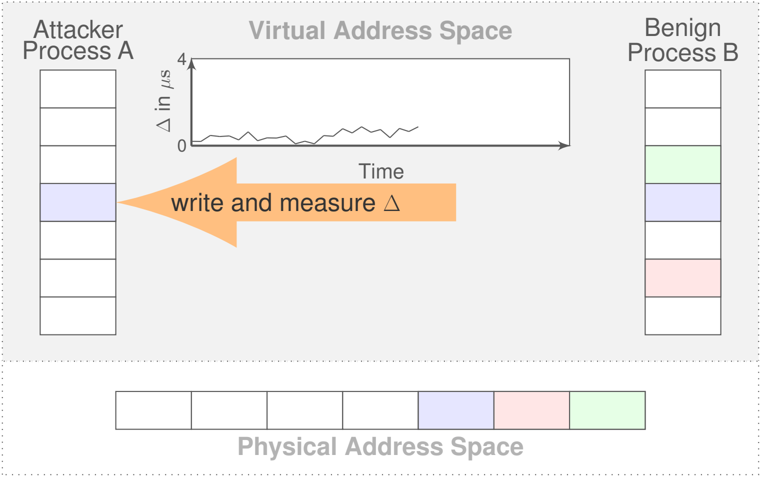
Page Deduplication Attack



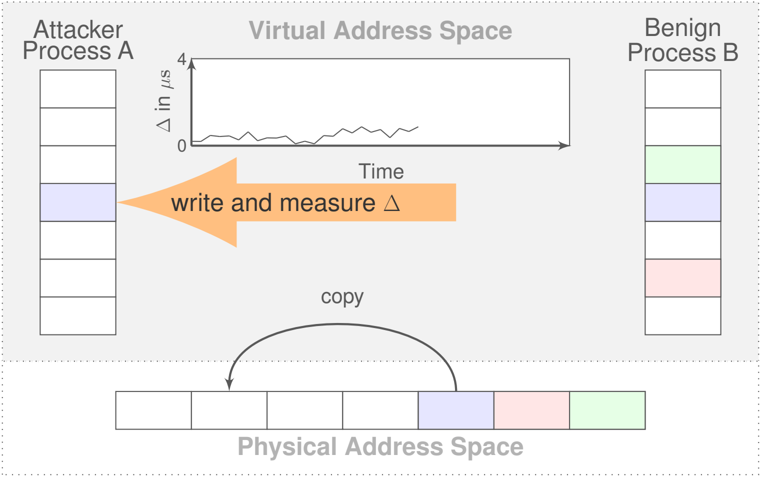
Page Deduplication Attack



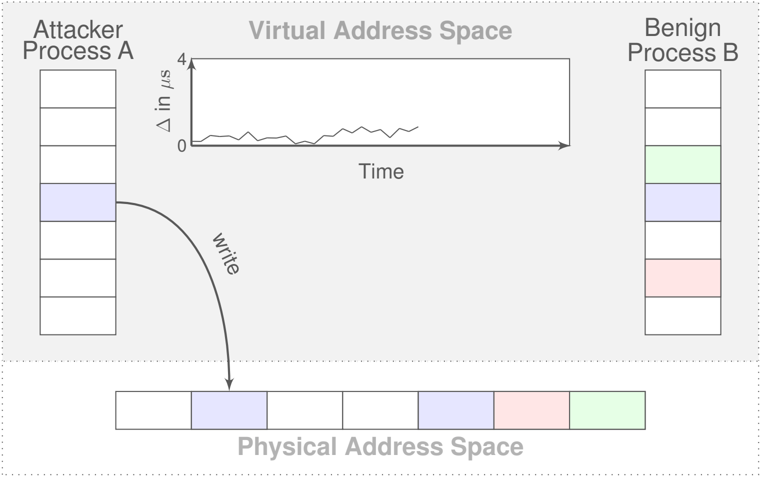
Page Deduplication Attack



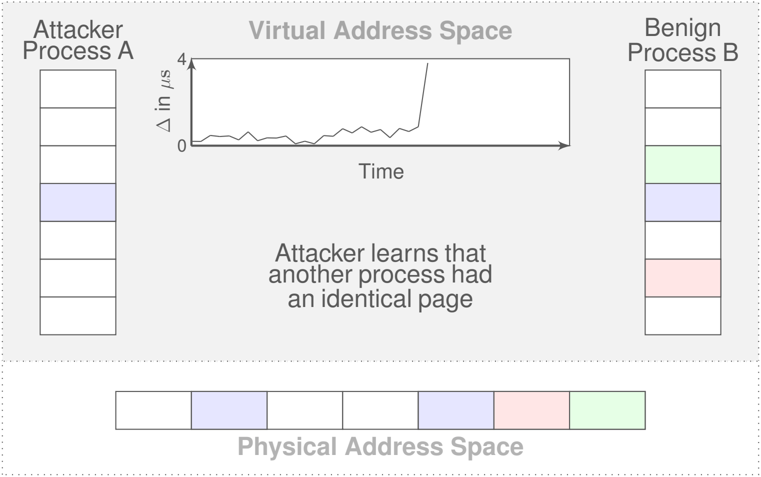
Page Deduplication Attack



Page Deduplication Attack

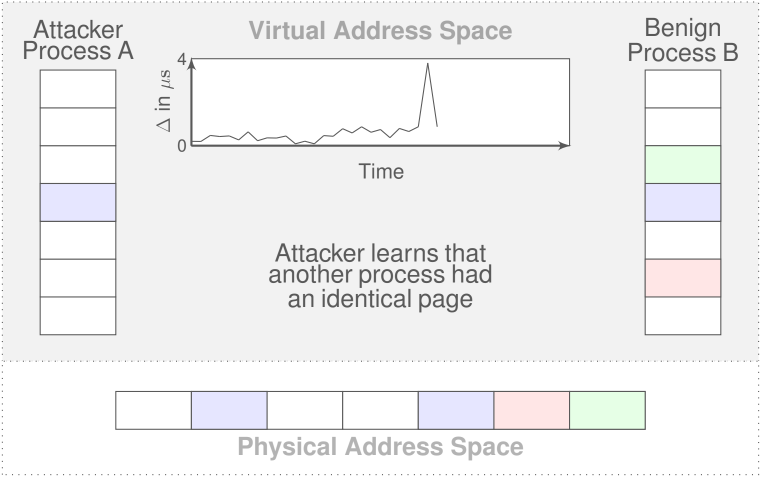


Page Deduplication Attack



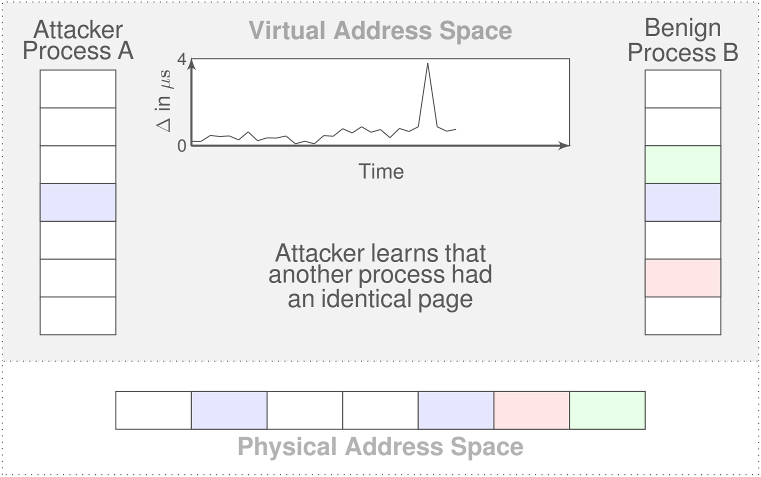
Attacker learns that another process had an identical page

Page Deduplication Attack



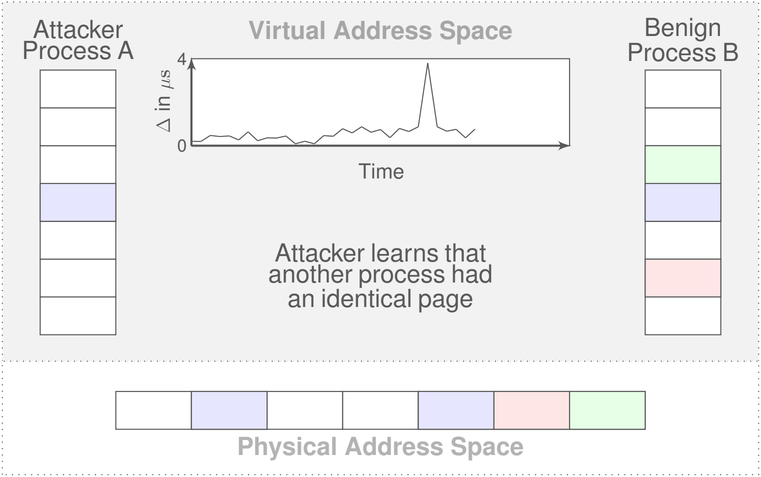
Attacker learns that another process had an identical page

Page Deduplication Attack



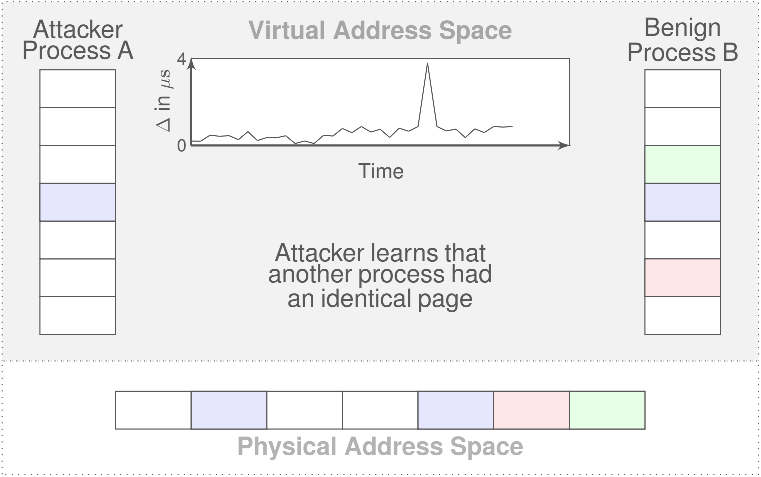
Attacker learns that another process had an identical page

Page Deduplication Attack



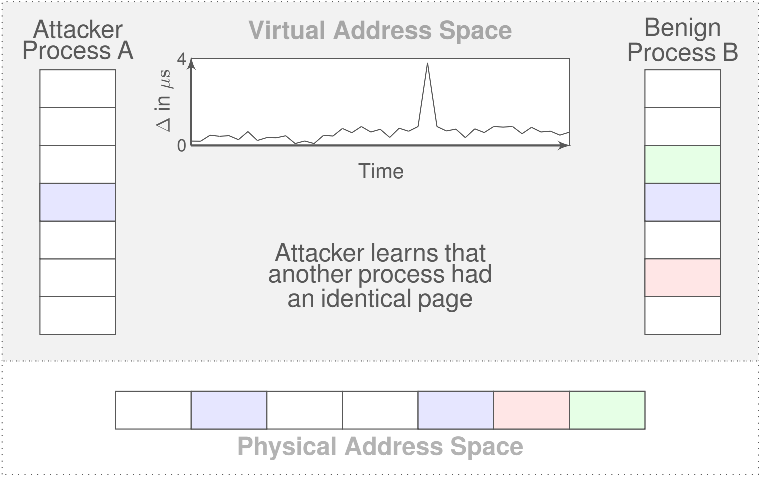
Attacker learns that another process had an identical page

Page Deduplication Attack



Attacker learns that another process had an identical page

Page Deduplication Attack



Attacker learns that another process had an identical page

What can be attacked?

Existing Attacks:

- Detect binary versions in co-located VMs
- Detect downloaded image in Firefox under certain conditions

→ Attacks on hypervisors

- Native code only

Suzaki et. al. 2011, Owens et. al. 2011, Xiao et. al. 2012, 2013

What can be attacked?

Our Contribution:

- Detect CSS files and images of opened websites
  - Chrome, Firefox and Internet Explorer
- Perform the attack in JavaScript

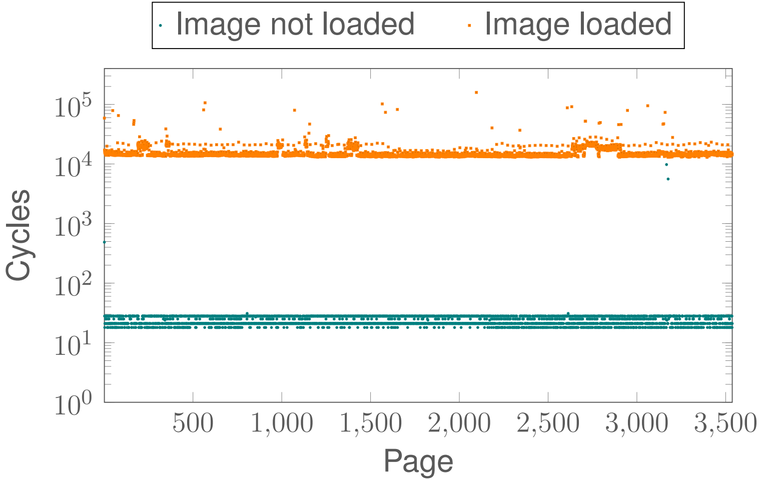
→ Attacks on KVM, Windows 8.1 and Android

Attacking Browsers

- Images and CSS files are page-aligned in memory
- Load them into memory for all websites of interest
- Detect deduplication

→ Malicious ad networks: alternative to tracking pixels?

Detect Image (Native, Cross-VM, KVM)



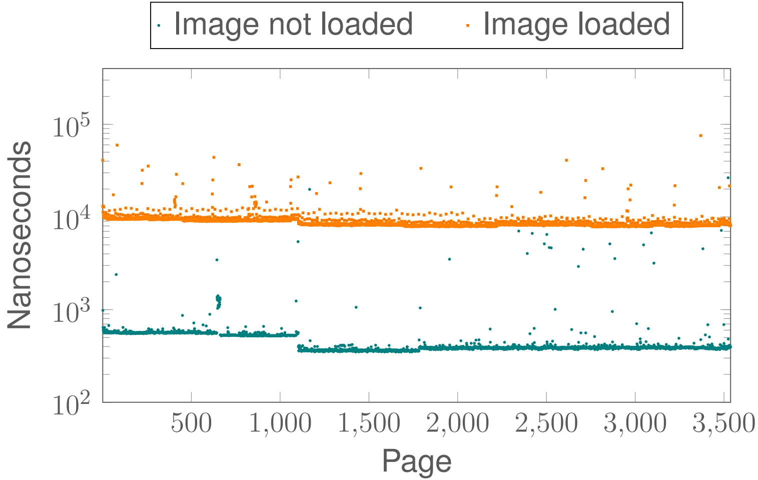
Challenges of JavaScript-based attacks

- No cycle counting (`rdtsc`)
- No access to virtual addresses

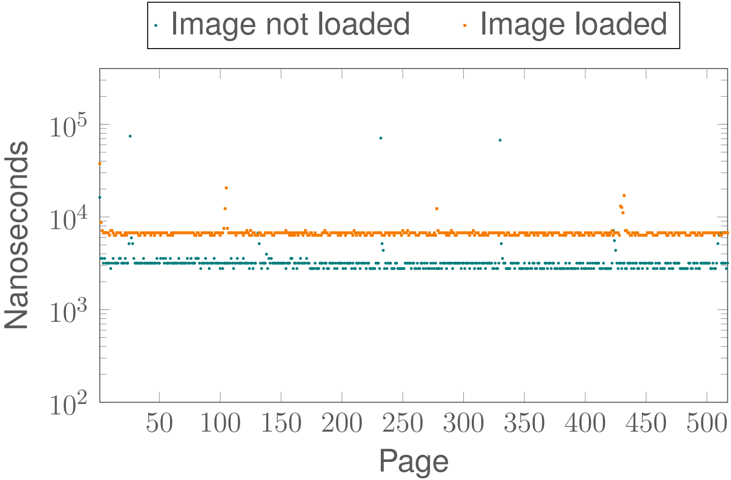
Page Deduplication Attacks in JavaScript

- Only require microsecond accuracy
  - `performance.now()` is accurate enough
  - Can even work with millisecond accuracy
    - Accumulate time difference
    - Only possible with enough image/CSS data
- Large typed arrays are allocated page-aligned

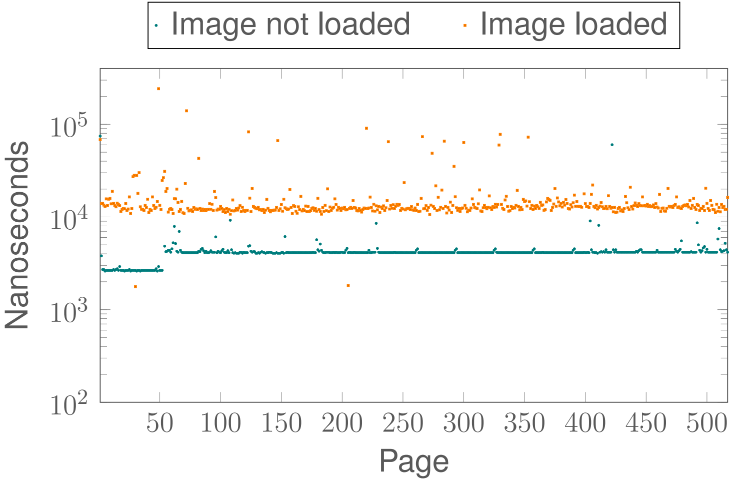
Detect Image (JavaScript, Cross-VM, KVM)



Detect Image (JavaScript, Windows 8.1)



Detect Image (JavaScript, Android 4.4.4)



Detection of Open Websites

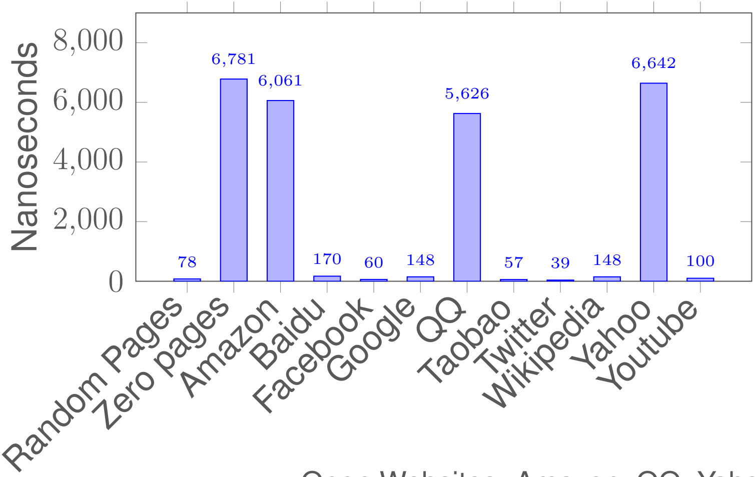
- Attacker chosen set of websites
- Load website images and CSS files into arrays
    - Served image and CSS files depend on: Browser, OS, resolution, etc.

  → Reuse HTTP headers of system under attack
- Reduce noise by measuring write accesses to several pages

Detection of Open Websites

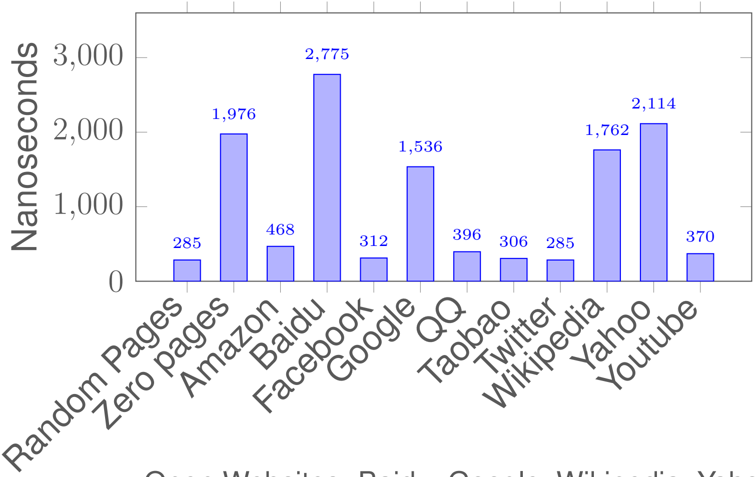
- Compare with:
  - zero pages (always deduplicated)
  - random pages (never deduplicated)
- Top 10 websites:
  Amazon, Baidu, Facebook, Google, QQ, Taobao, Twitter, Wikipedia, Yahoo, Youtube
- Examples for different platforms

Example: JavaScript, Cross-VM, KVM



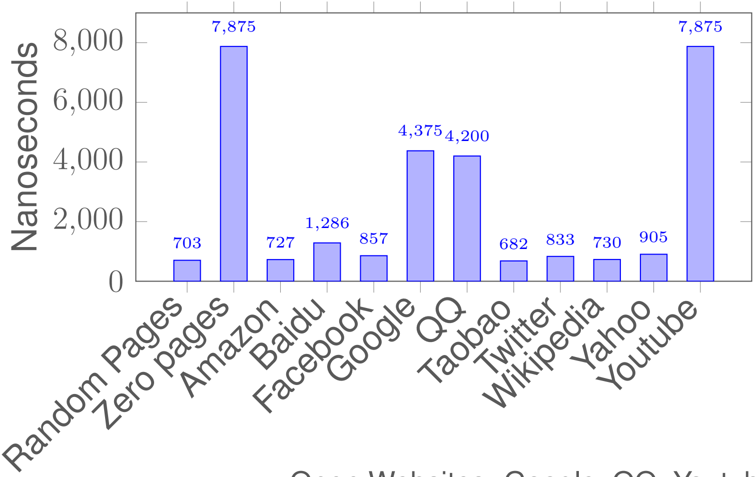
Open Websites: Amazon, QQ, Yahoo

Example: JavaScript, Windows 8.1



Open Websites: Baidu, Google, Wikipedia, Yahoo

Example: JavaScript, Android 4.4.4



Open Websites: Google, QQ, Youtube

Countermeasures

JavaScript:

- Reduce timer accuracy?
- Prevent page-aligned arrays?
- Website diversification?
- Prevent control over full pages
    - Every n-th byte not part of JavaScript array

Countermeasures

JavaScript:

- Reduce timer accuracy?
- Prevent page-aligned arrays?
- Website diversification?
- Prevent control over full pages
    - Every n-th byte not part of JavaScript array

Generic:

- Disable page deduplication (for writable pages)

Conclusion

- Page deduplication not only a problem on IaaS clouds

→ Privacy issue on personal computers and smartphones

- Remote attacks through malicious websites
- Same code for all platforms → large scale attacks
- Disable page deduplication if possible

Practical Memory Deduplication Attacks in Sandboxed JavaScript

Daniel Gruss, David Bidner, and Stefan Mangard
IAIK, Graz University of Technology

September 23, 2015