black hat USA 2020
AUGUST 5-6, 2020
BRIEFINGS

Hiding Process Memory via Anti-Forensic Techniques

Frank Block

Hiding Process Memory via Anti-Forensic Techniques

Ralph Palutke*, Frank Block*, Patrick Reichenberger, and Dominik Stripeika

Security Research Group
Department of Computer Science
Friedrich-Alexander University Erlangen-Nürnberg (FAU)

ERNW RESEARCH pursuing knowledge.

Black Hat | USA 2020

Agenda

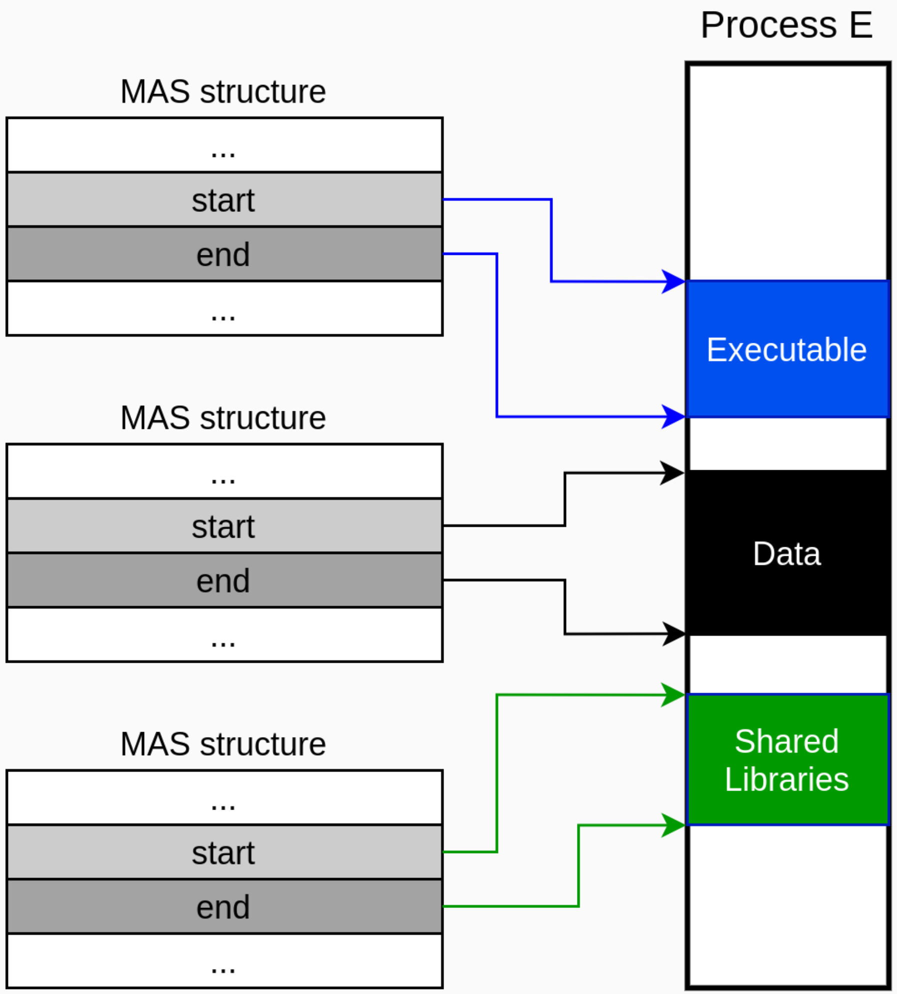
- Introduction
- Memory Subversion Techniques
- Considerations
- Memory Subversion Evaluation
- Memory Subversion Detection
- Memory Subversion Detection Evaluation
- Conclusion

Introduction

- Related Work
  - Shadow Walker by Sherri Sparks and Jamie Butler [1].
  - Gargoyle by Josh Lospinoso [2].

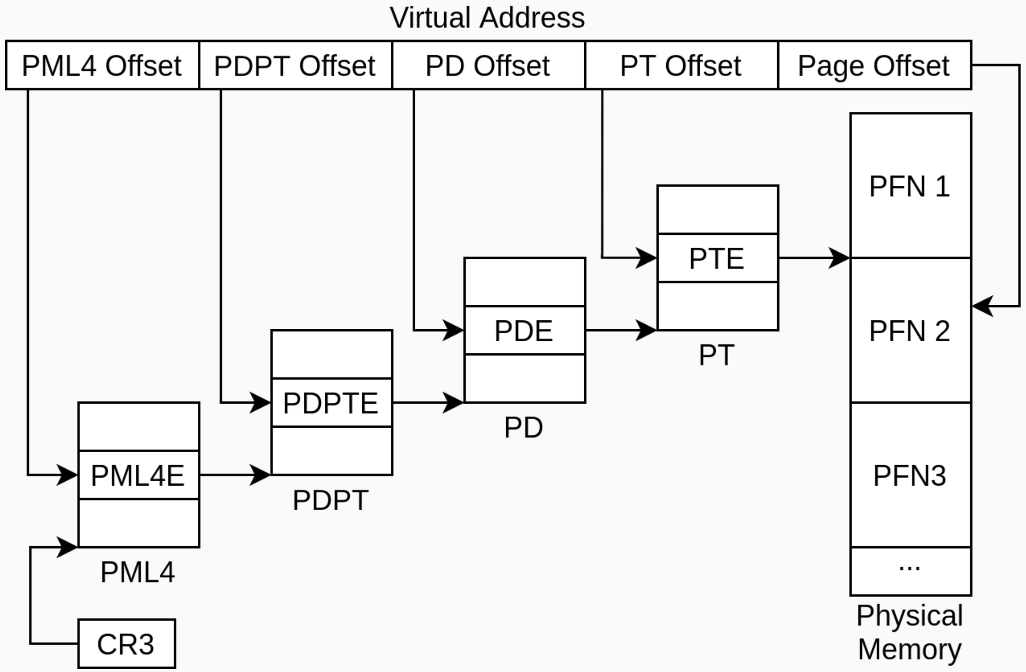
Process Address Space

- Process address space contains executables, libraries, heap, …
  - Are described by what we will call Memory Area Structures.
- Memory Area Structures (MASs) describe each memory area.
  - Linux: Virtual Memory Areas (VMAs)
  - Windows: Virtual Address Descriptors (VADs)



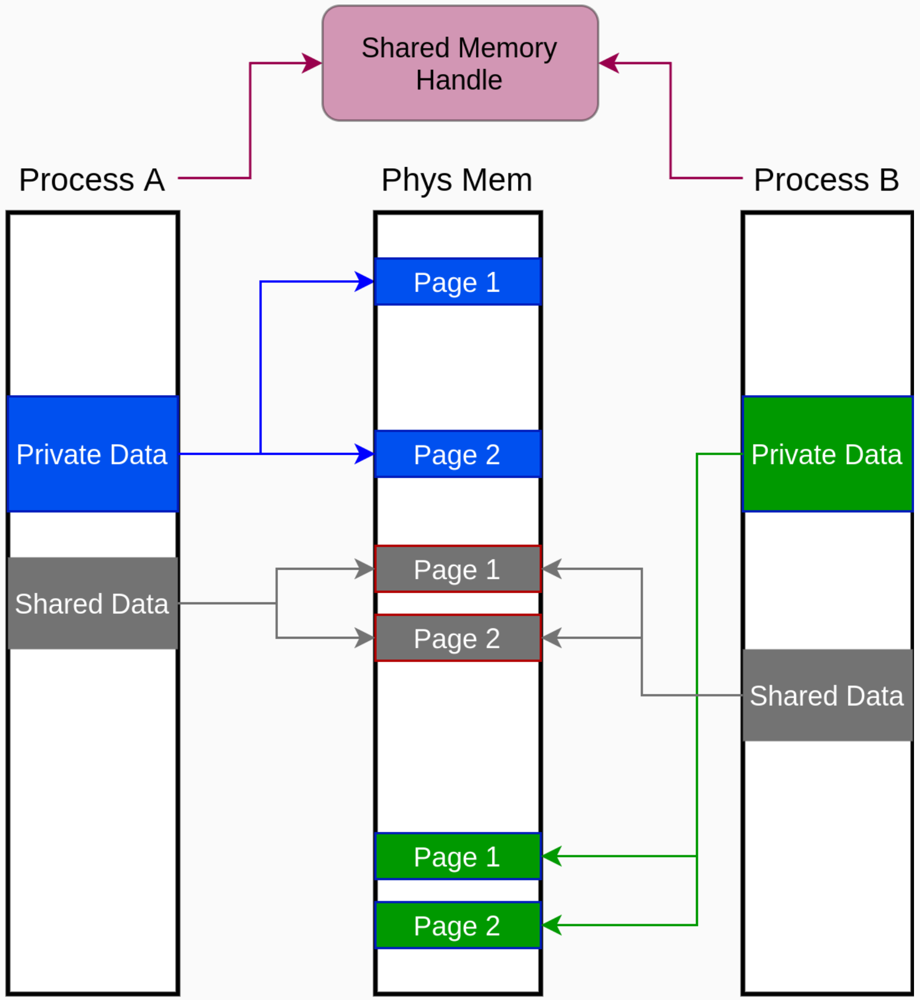
Paging

- Physical memory is structured into page frames.
  - Referenced by Page Frame Numbers (PFNs).
- Translation from virtual to physical done with paging structures.
  - Last part in translation process: PTE
- Reverse mapping: Windows PFN DB and on Linux the `page` structures
  - The physical view on memory.



Shared Memory

- Memory shared among multiple processes.
  - Allows the exchange of data.
  - We focus on anonymous/page-file-backed shared memory.
- Available until no process owns a handle anymore.
  - Private memory is lost after owning process unmaps it.
- OS-specific APIs for creating and mapping shared memory.
  - Windows:
    - Memory-mapped files (`CreateFileMapping`/`MapViewOfFile`)
  - Linux:
    - POSIX shared memory objects (`shm_open`/`mmap`)
    - Anonymous files: (`memfd_create`/`mmap`)
    - System V shared memory segments (`shmget`/`shmat`)



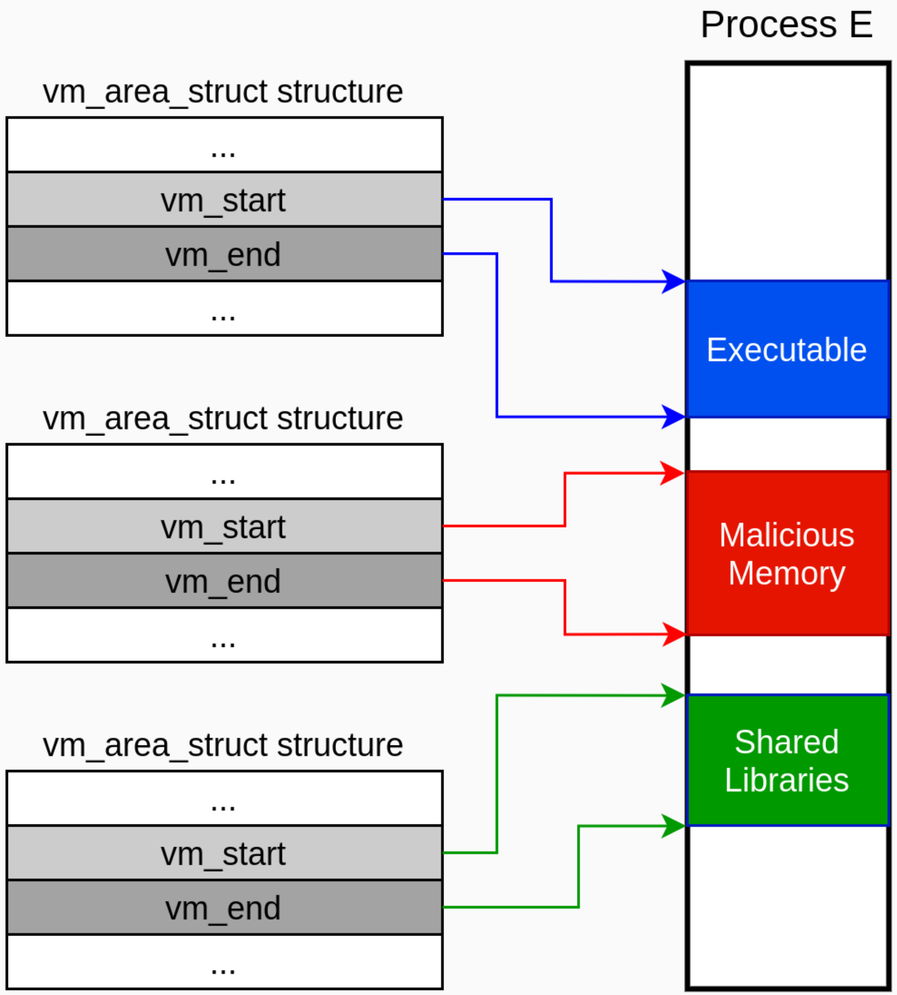
Memory Subversion Techniques

Overview

- Anti-forensic techniques that hide process memory.
  - Can be used independently or in combination.
- Attacker Scenario: Hide malicious parts of otherwise benign process.
  - Either launched or infected during run time.
  - Injected shellcode, loaded libraries, parts of the application.

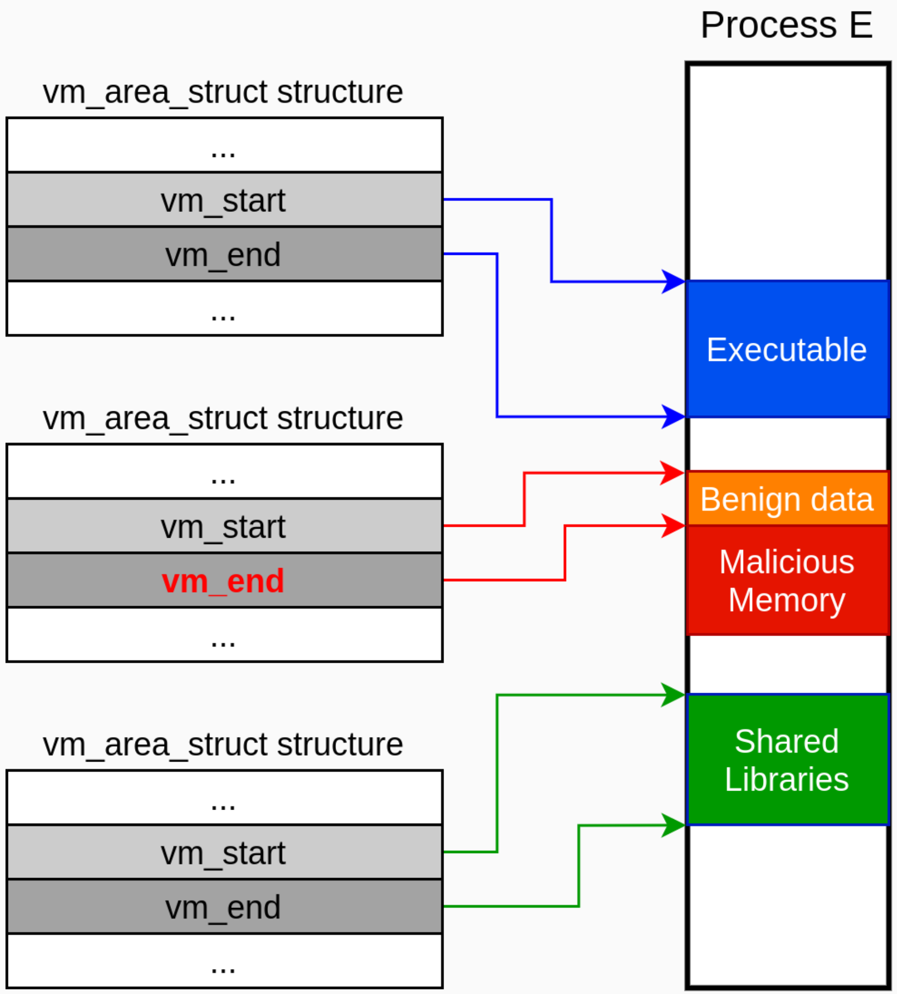
MAS Remapping

- Many forensic tools rely on the integrity of MASs.
- Idea: Remap MASs from malicious to benign areas.
- Requires kernel-level privileges.



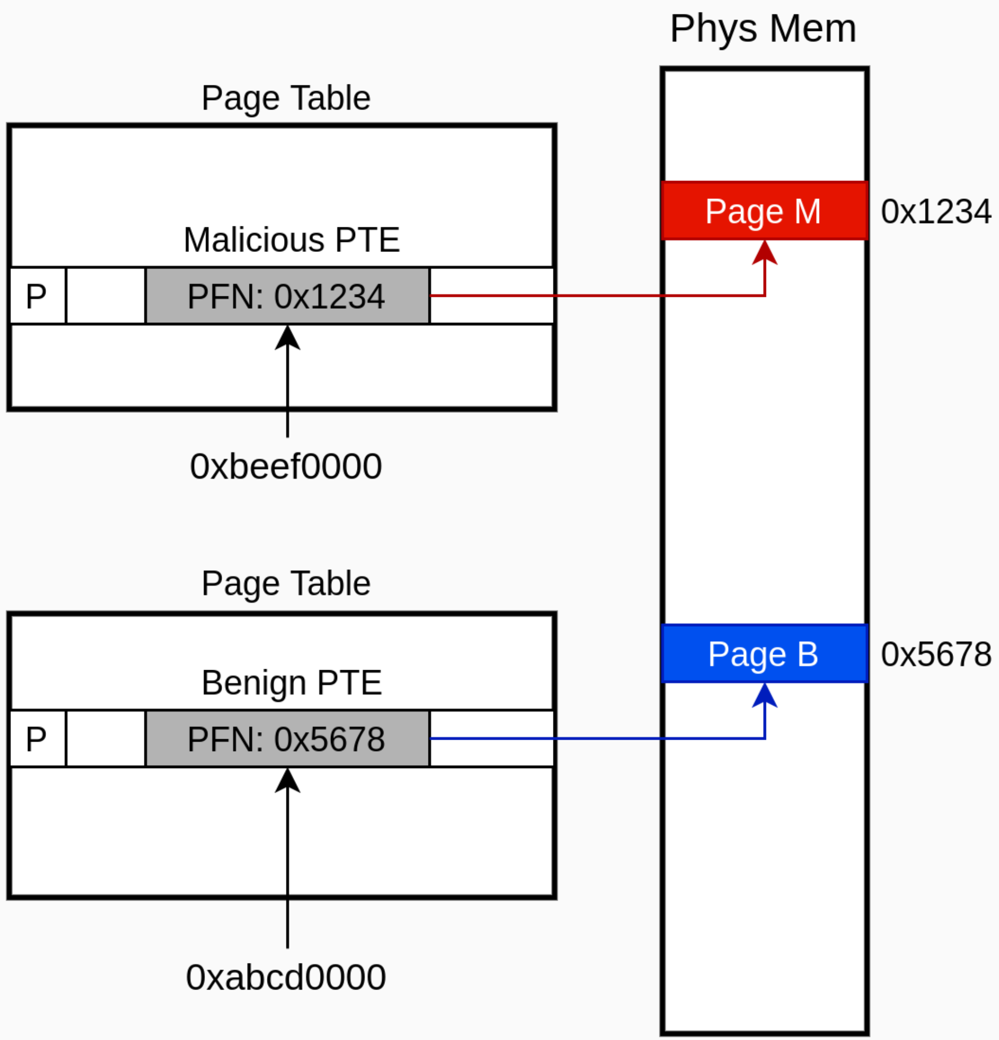
MAS Remapping

- Manipulate MASs virtual start and end addresses.
  - Windows: `StartingVPN/EndingVPN (VAD)`
  - Linux: `vm_start/vm_end` (`vm_area_struct`)
- MASs are not involved in the translation process.
  - Memory can be accessed despite being hidden.
  - Underlying PTEs still intact.
- **No necessity to revert** modifications during runtime



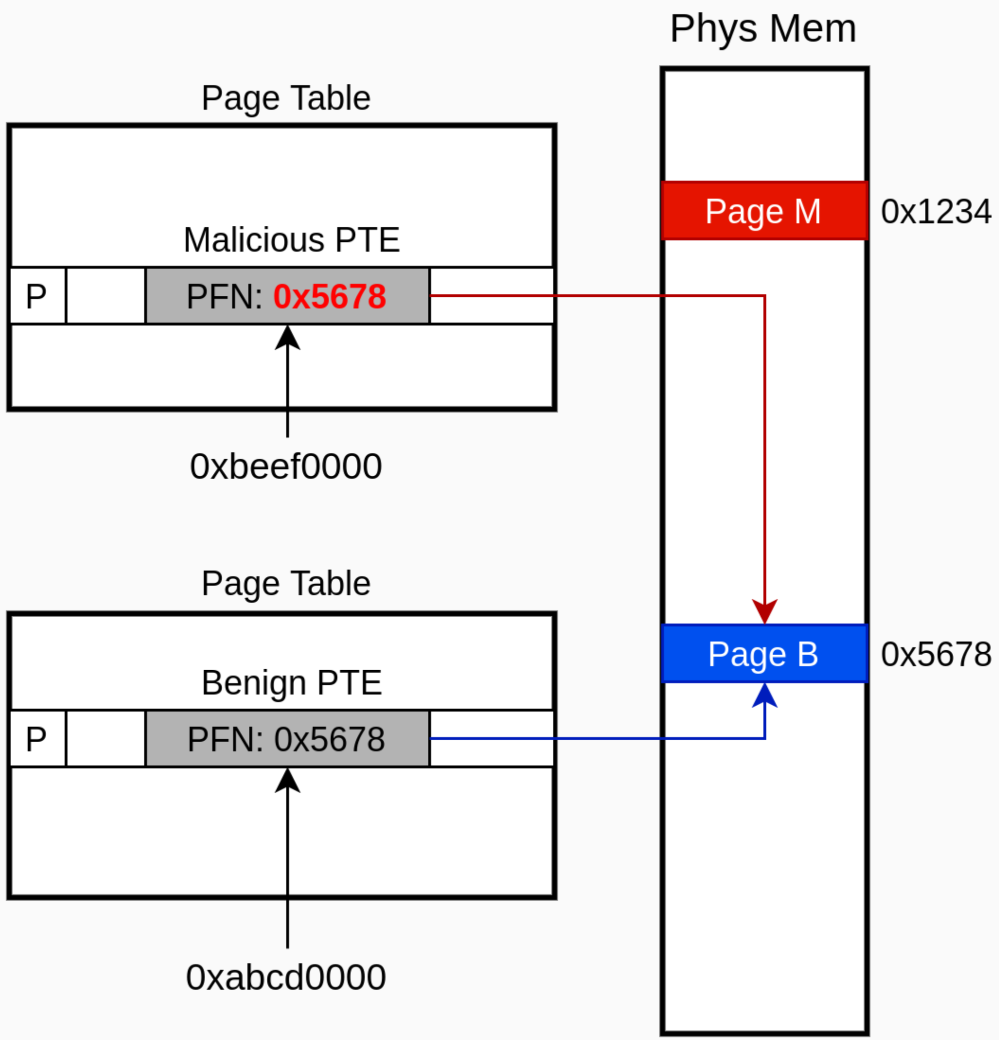
PTE Subversions

- No modification of process' virtual memory layout.
- Requires kernel-level privileges.
- Accessing malicious memory requires PTE restoration.



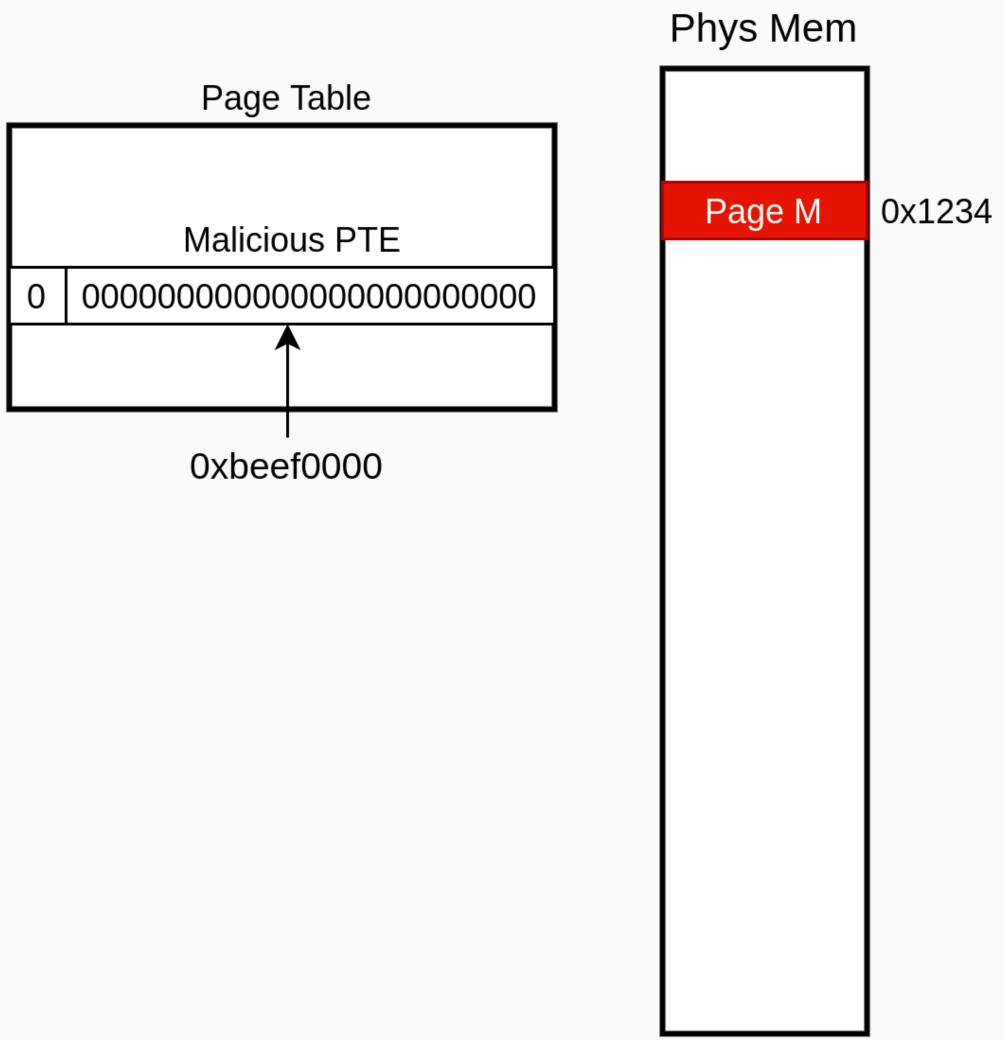
PTE Remapping

- Remap malicious page frames to equal amount of benign ones
- Virtual addresses might resolve to same PFN
- Redirection target ideally contains similar content



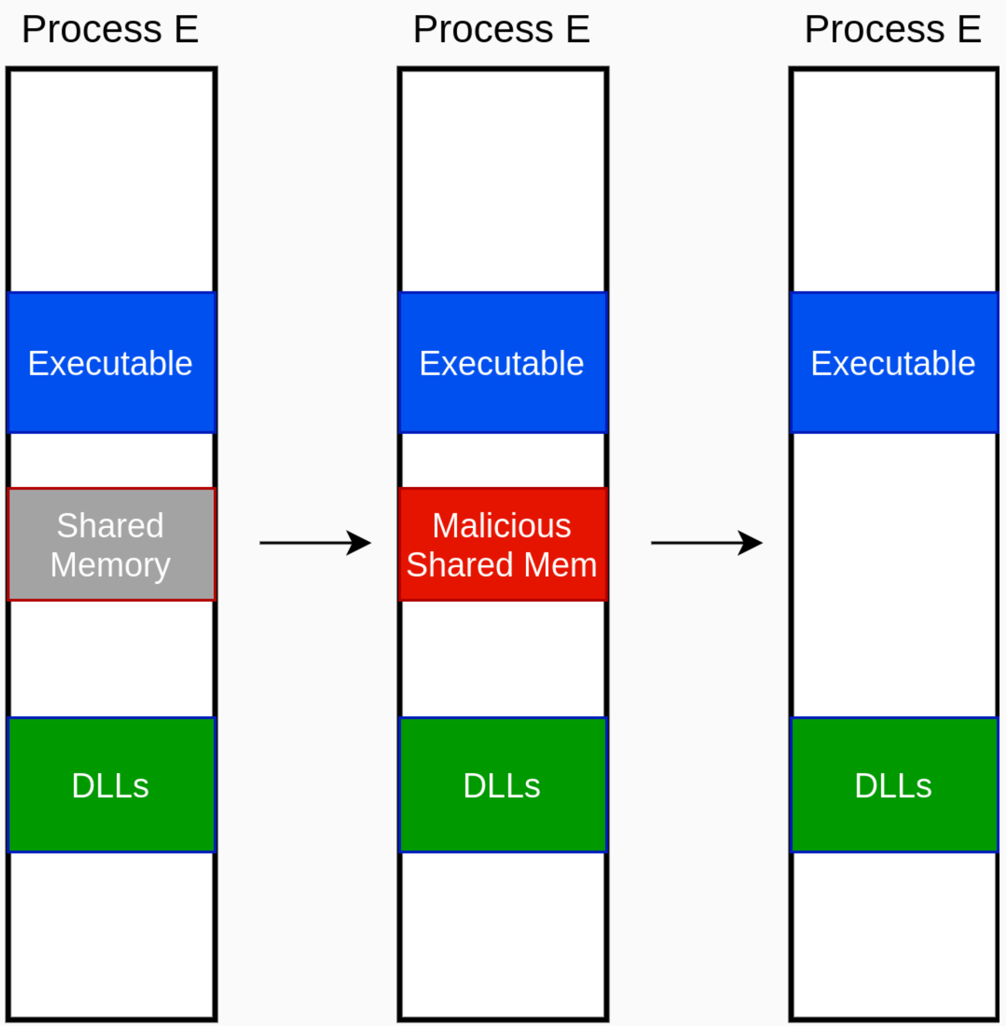
PTE Erasure

- Erase/Nullify PTEs of malicious page frames
  - PTEs seem to be not initialized, accessed, or present
  - Inherently invalidates PTE
- Solely invalidating present bit is not sufficient
  - Analysis tools could detect remaining information
- Windows/Linux does not rely on PTEs to find free memory
  - No risk of being freed or reused by the OS



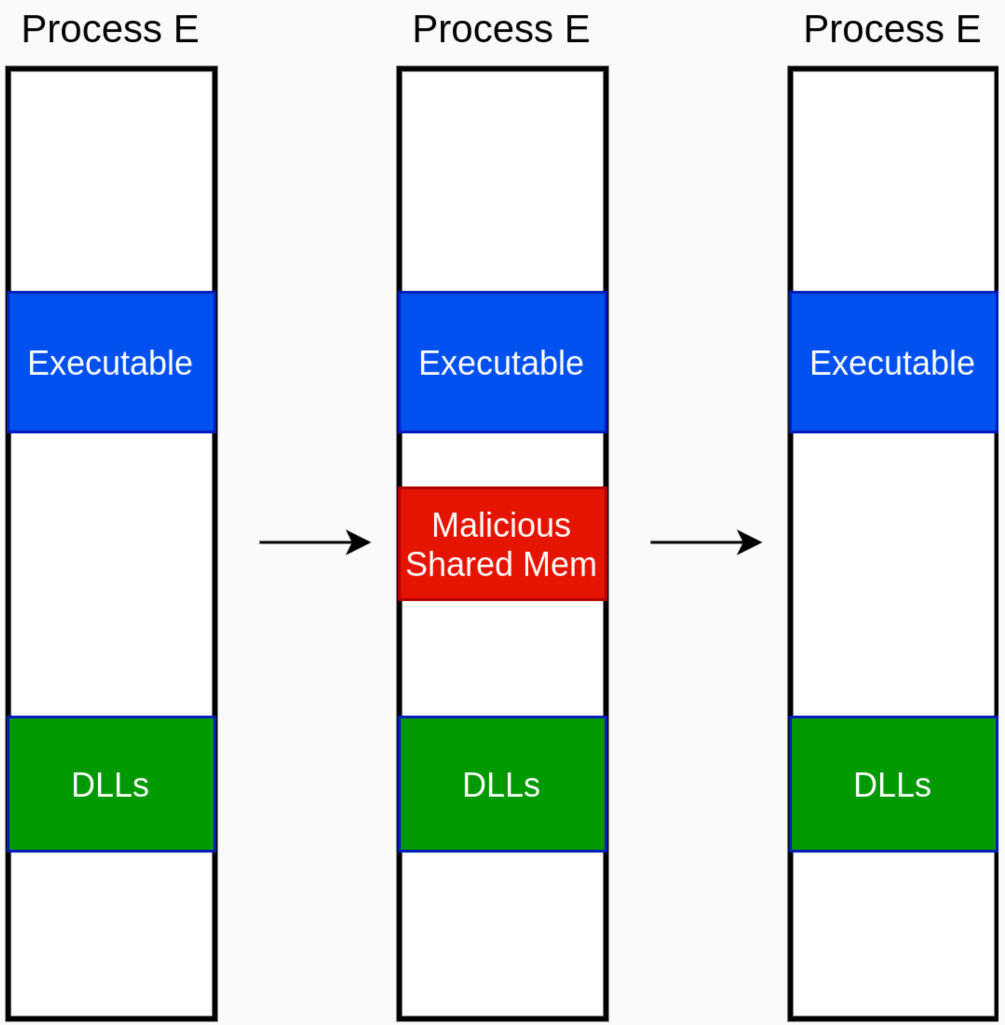
Shared Memory Subversion

- Shared memory does not have to be shared between processes
  - Can be used to store executable code
- Initial setup:
  - Create shared memory section
  - Map shared memory into target process
  - Write malicious data to shared memory
  - Unmap shared memory
- **No kernel privileges** required

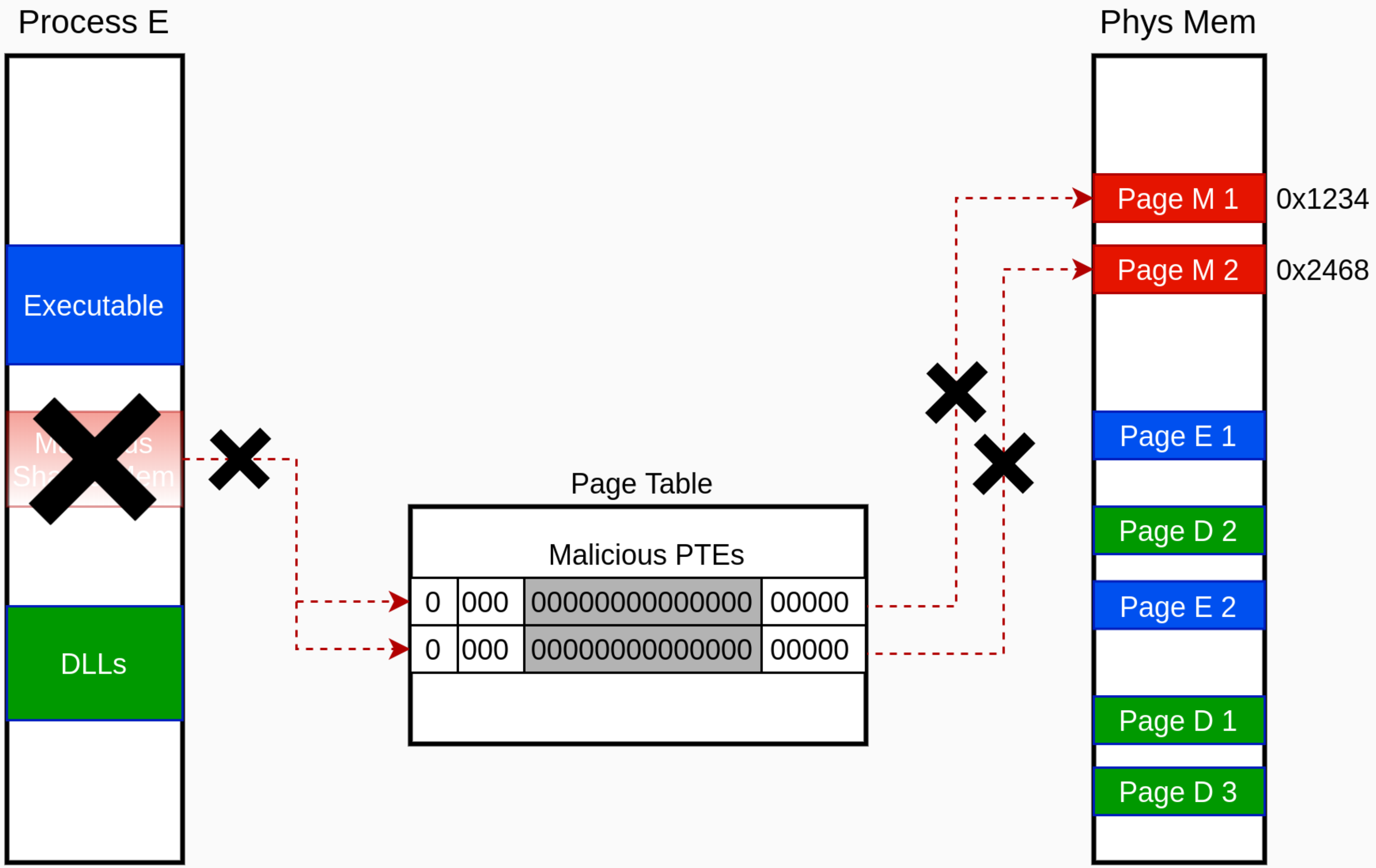


Shared Memory Subversion

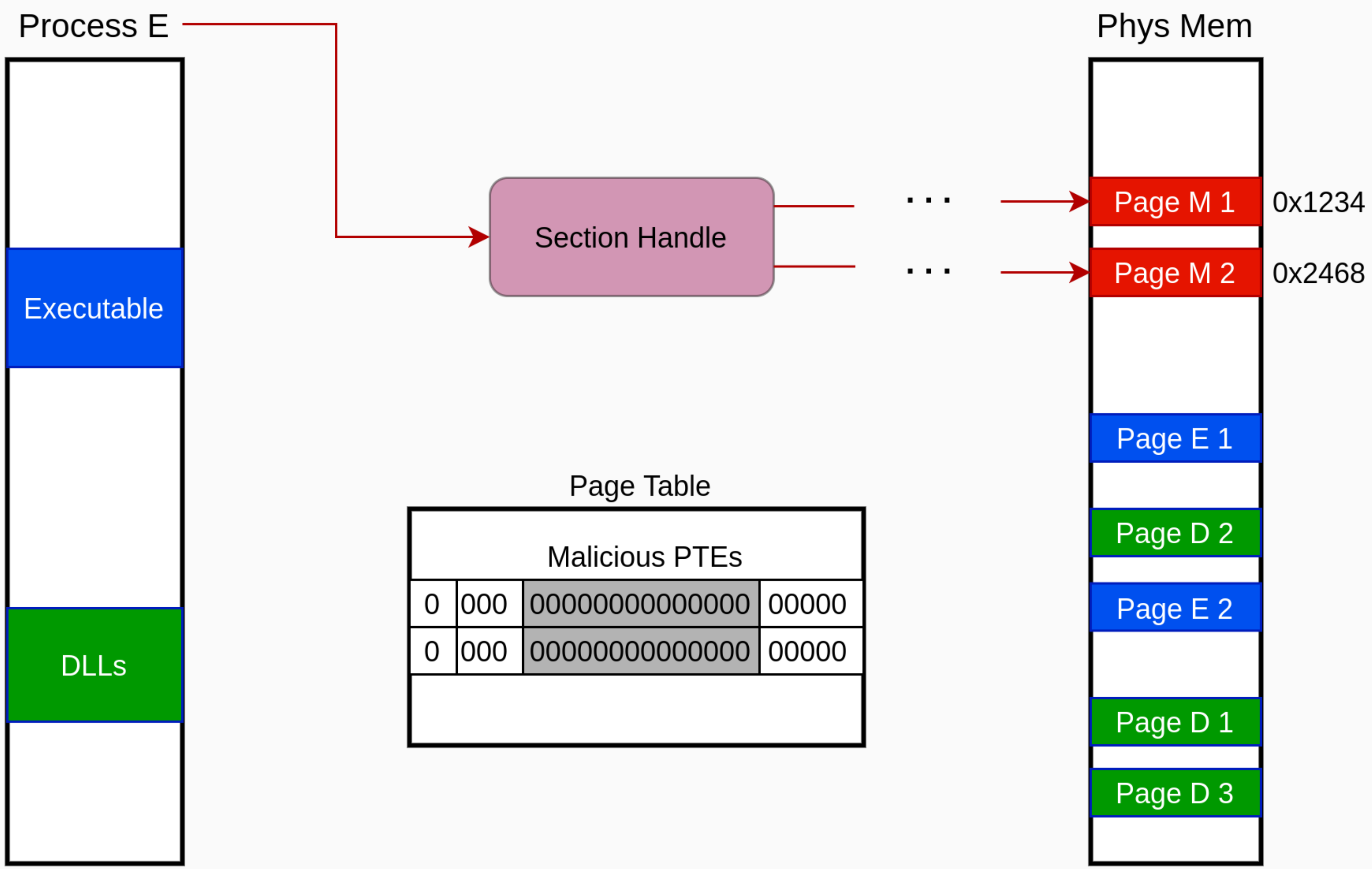
- Temporarily remap memory when being required
  - E.g. for executing included code
  - Unmap immediately afterwards
- Tools typically focus on currently mapped memory only
  - Remains undetected (if not caught while being mapped)



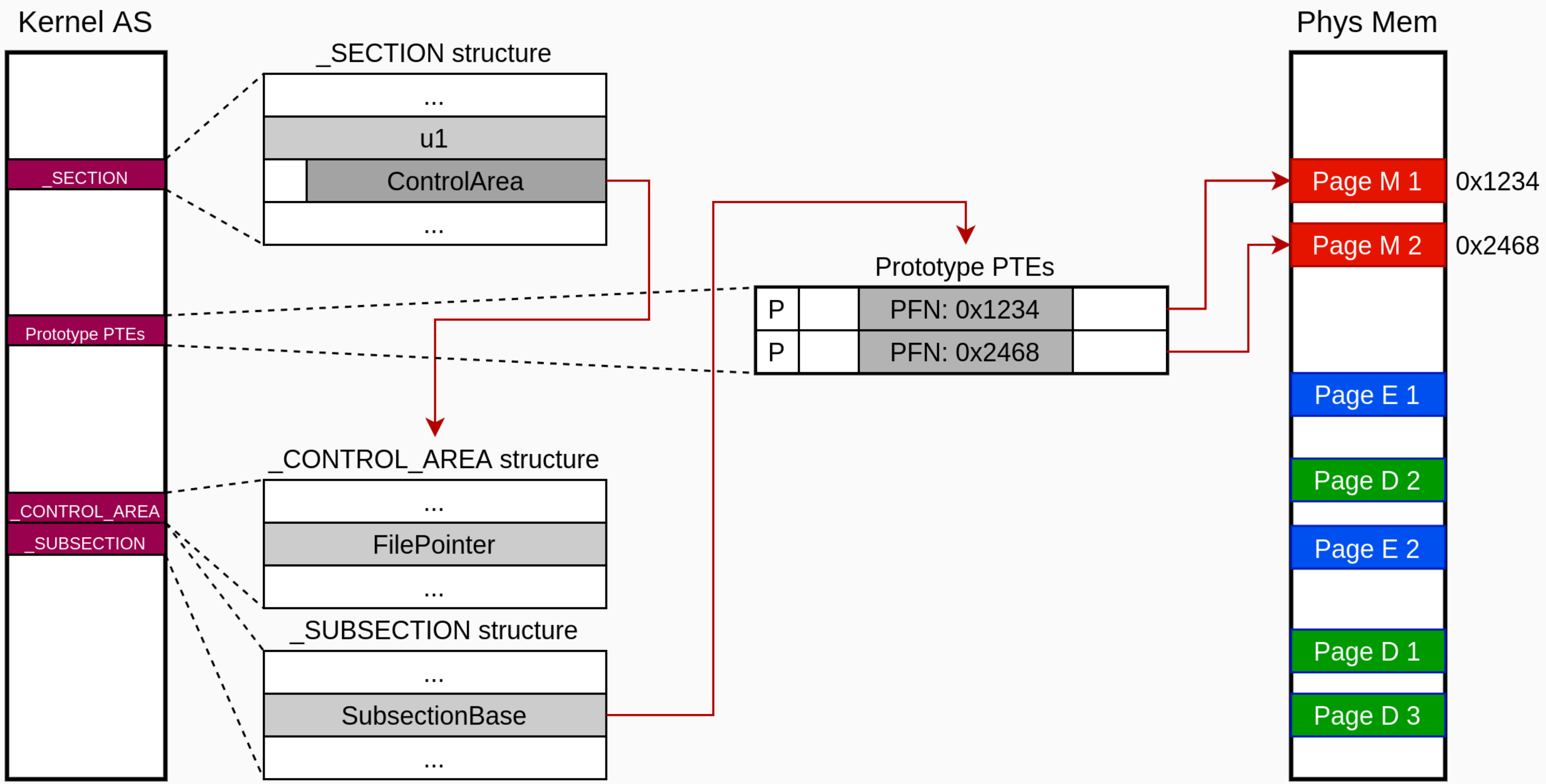
Shared Memory Subversion



Shared Memory Subversion



Shared Memory Subversion



Memory Subversion Evaluation

Evaluation

- The subversion techniques have been implemented as a Proof of Concept for Windows and Linux.

- Evaluation from a memory and live forensics perspective, on both operating systems:
  - Windows 10 Pro Version x64 (1511 Build 10586 and 1909 Build 18363)
  - Debian 9.9 4.9.0-11-amd64 (4.9.189-3+deb9u2)

Evaluation – Memory Forensics

	handles	MAS	malfind	ptenum	read	memdump	libdump	masdump	r-yarascan	v-yarascan
PTE Erasure	A [3]	A [1]	N	N	N	N	N	N	F	N
PTE Remapping	A [3]	A [1]	A [4]	A [3]	N	N	N	N	F	N
MAS Remapping	A [3]	I	A [4]	F [3]	F	F [2]	N	N	F	N
Shared Memory Windows	A	N	N	N	N	N	N	N	F	N
Shared Memory shmget	N	N	N	X	N	N	N	N	F	N
Shared Memory shmopen	A	N	N	X	N	N	N	N	F	N
Shared Memory memfd	A	N	N	X	N	N	N	N	F	N

[1] Only on Linux with a loaded Library. [2] No hit for Rekall's memdump on Windows. [3] Only on Windows. [4] Only for Shellcode scenario.

Table 1: Evaluation of Subversion Techniques with Memory Forensics

Details are included in the Research Paper [4]

Evaluation – Live Forensics

	handles	MAS	dbgdump	tdump	read	memsearch	yara
PTE Erasure	A [2]	A [1]	N	N	N	N	N
PTE Remapping	A [2]	A [1]	I [3]	I [3]	N	N	N
MAS Remapping	A [2]	I	I [4]	I [4]	F [2]	F [2]	N
Shared Memory Windows	A	N	N	N	N	N	N
Shared Memory shmget	N	N	N	N	N	N	N
Shared Memory shmopen	A	N	N	N	N	N	N
Shared Memory memfd	A	N	N	N	N	N	N

[1] Only on Linux with a loaded Library. [2] Only on Windows.
[3] Dump fails on Windows. [4] Only on Linux: Warning message with memory address.

Table 2: Evaluation of Subversion Techniques with Live Analysis

Details are included in the Research Paper [4]

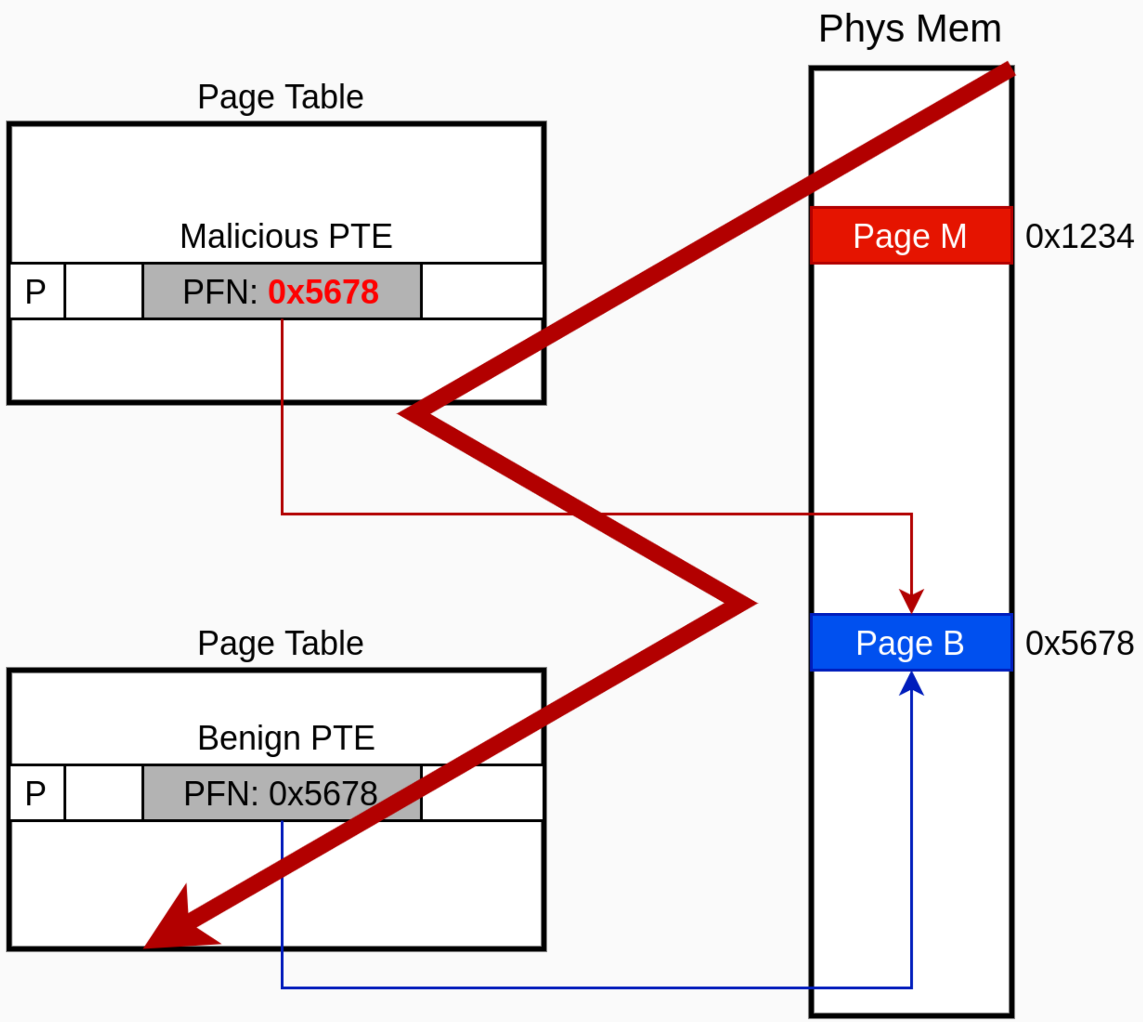
Considerations

Considerations

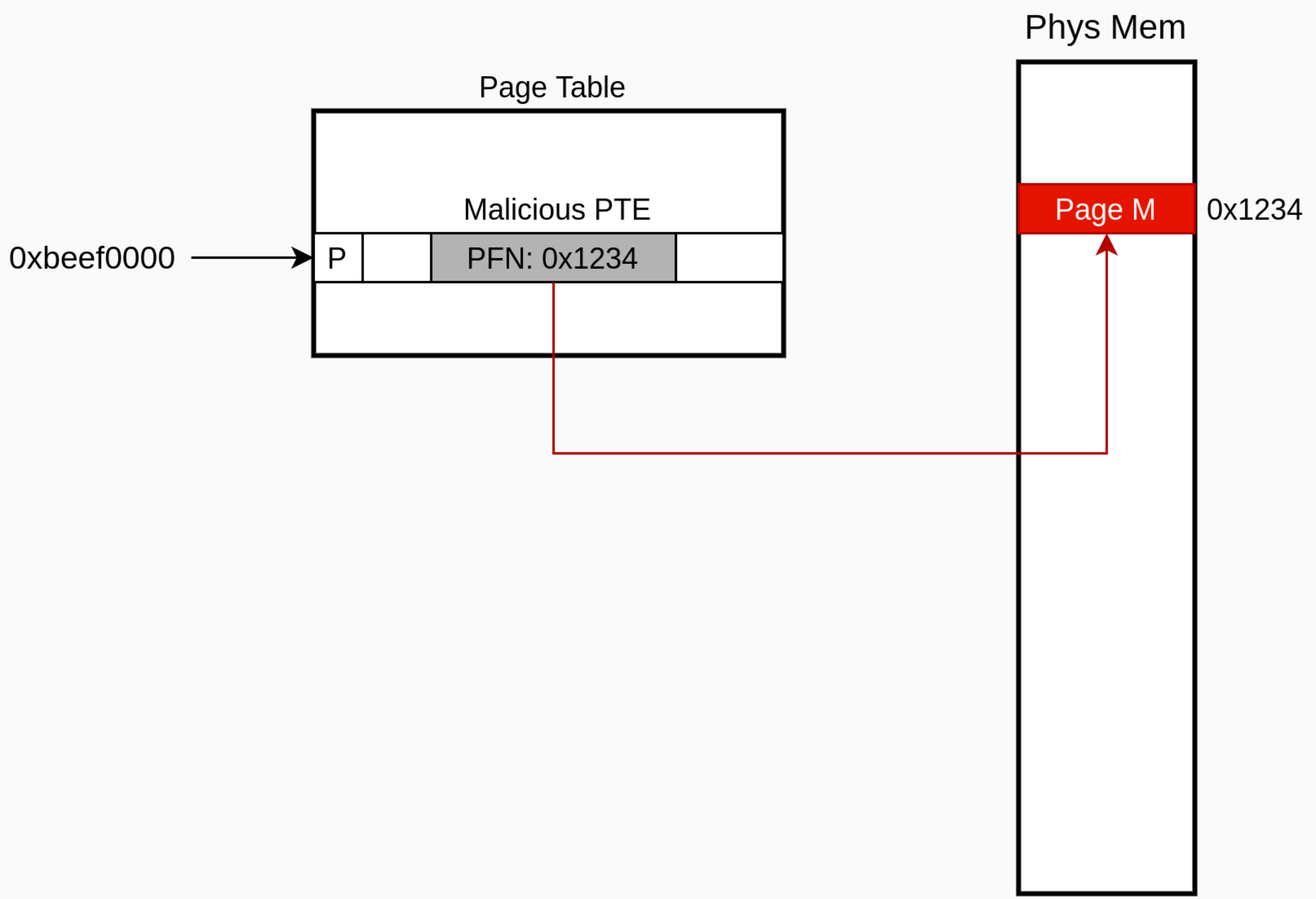
- Control code necessary to un/rehide (for PTE and shared memory subversions).
- Locking memory to prevent page swapping mechanisms to interfere with subversion techniques.
- In order to prevent side effects: Undoing modifications right before exit.
- Counters to consider on Linux:
  - The **Resident Set Size (RSS)** counters (store information about a process' occupied physical memory).
  - The mm_struct structure's **nr_ptes** counter, which specifies the number of page tables.
  - The page structure's **_refcount** field.

Considerations

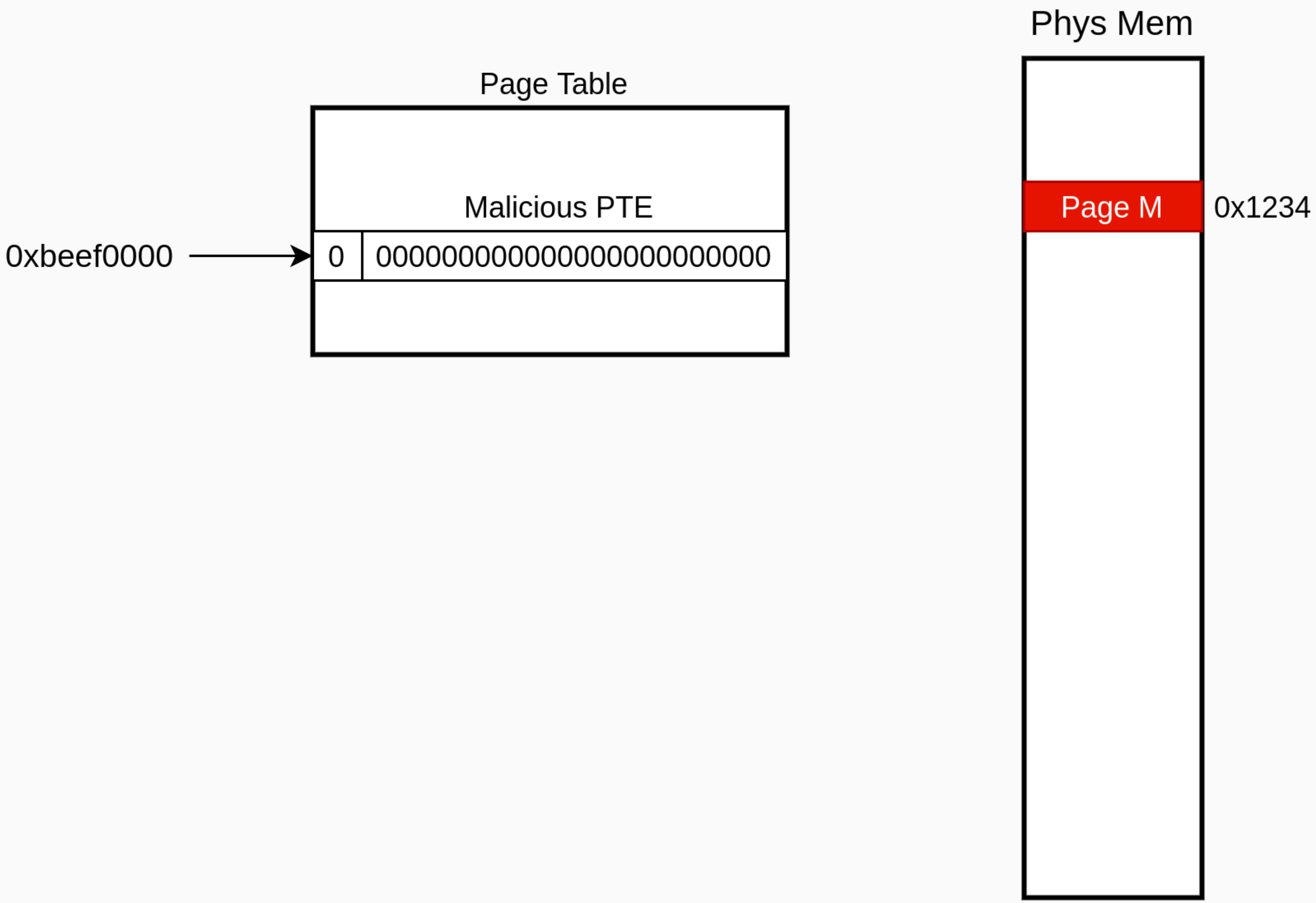
- “Standard” PFN remapping on Windows reliably leads to crashes e.g. when yara tries to scan the process’ memory.



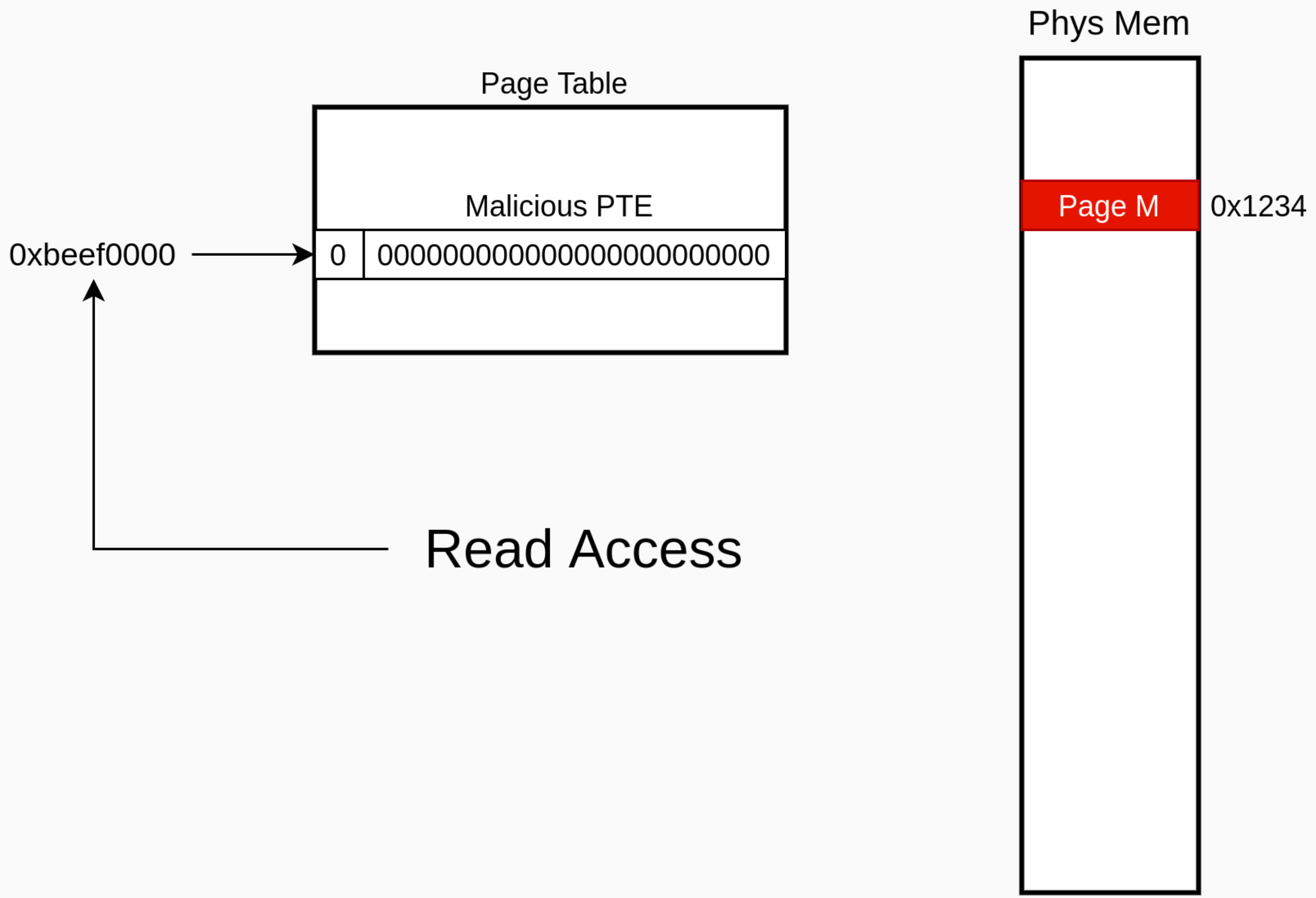
Modified PFN Remapping on Windows



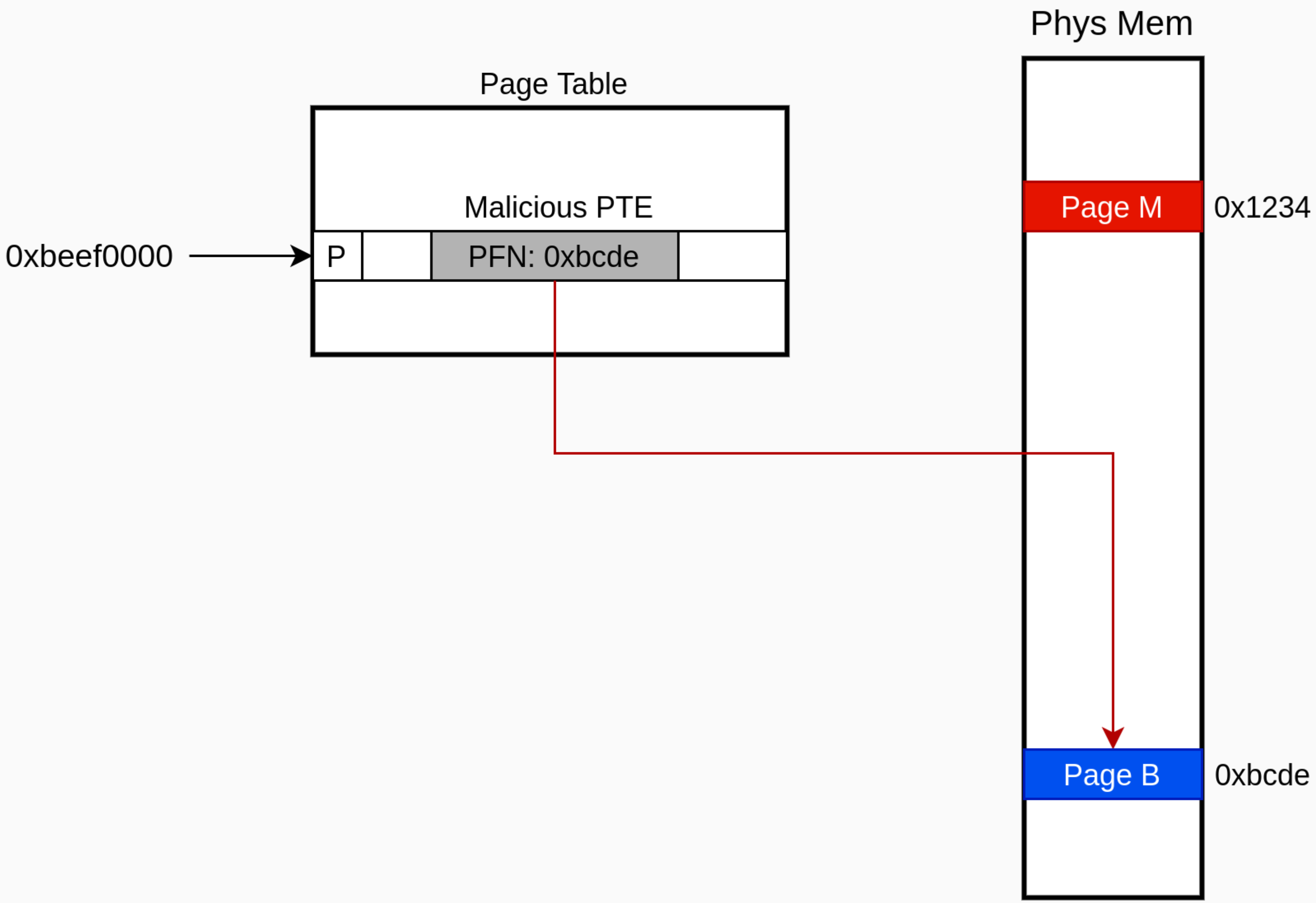
Modified PFN Remapping on Windows



Modified PFN Remapping on Windows



Modified PFN Remapping on Windows

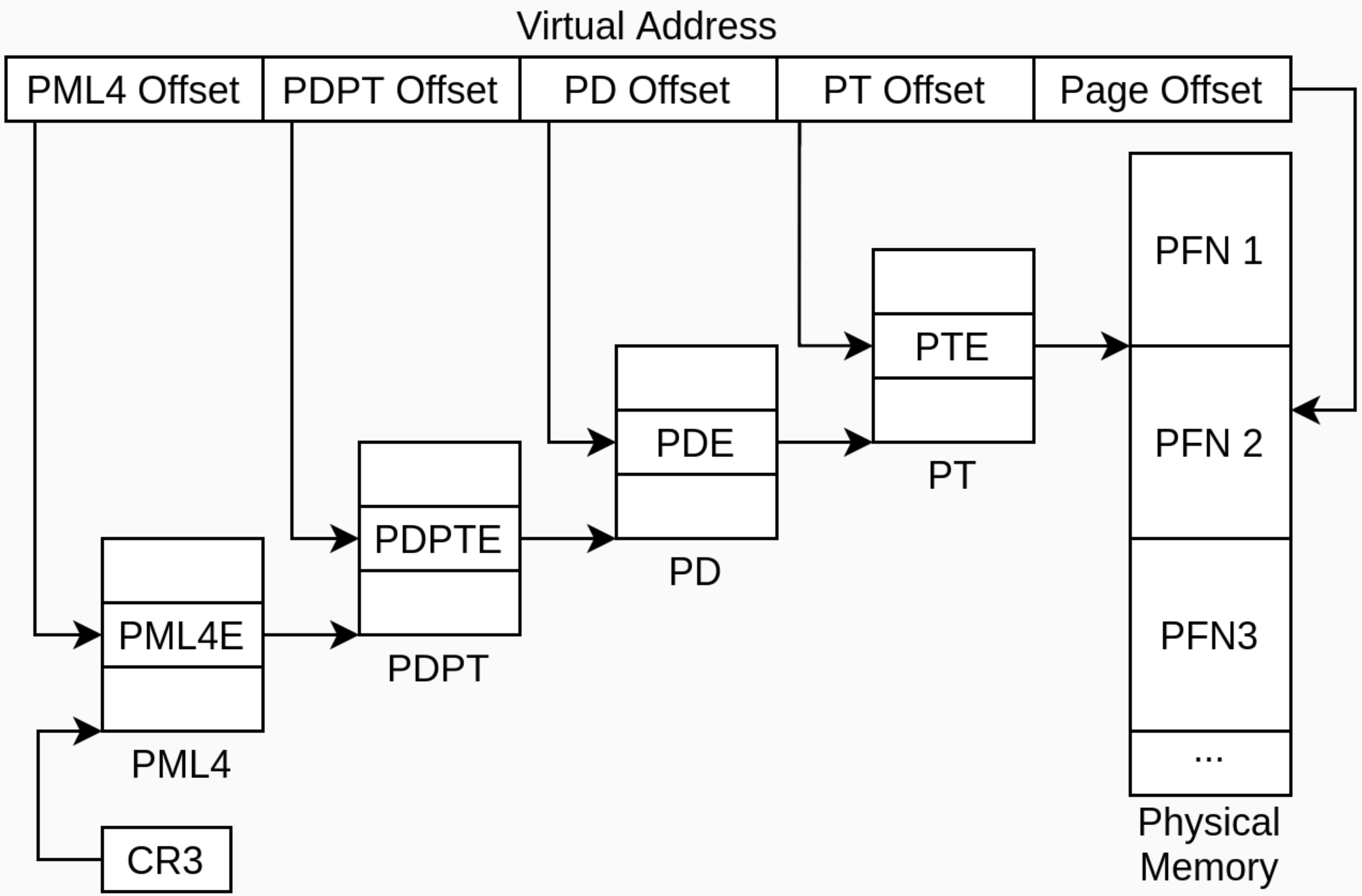


Considerations

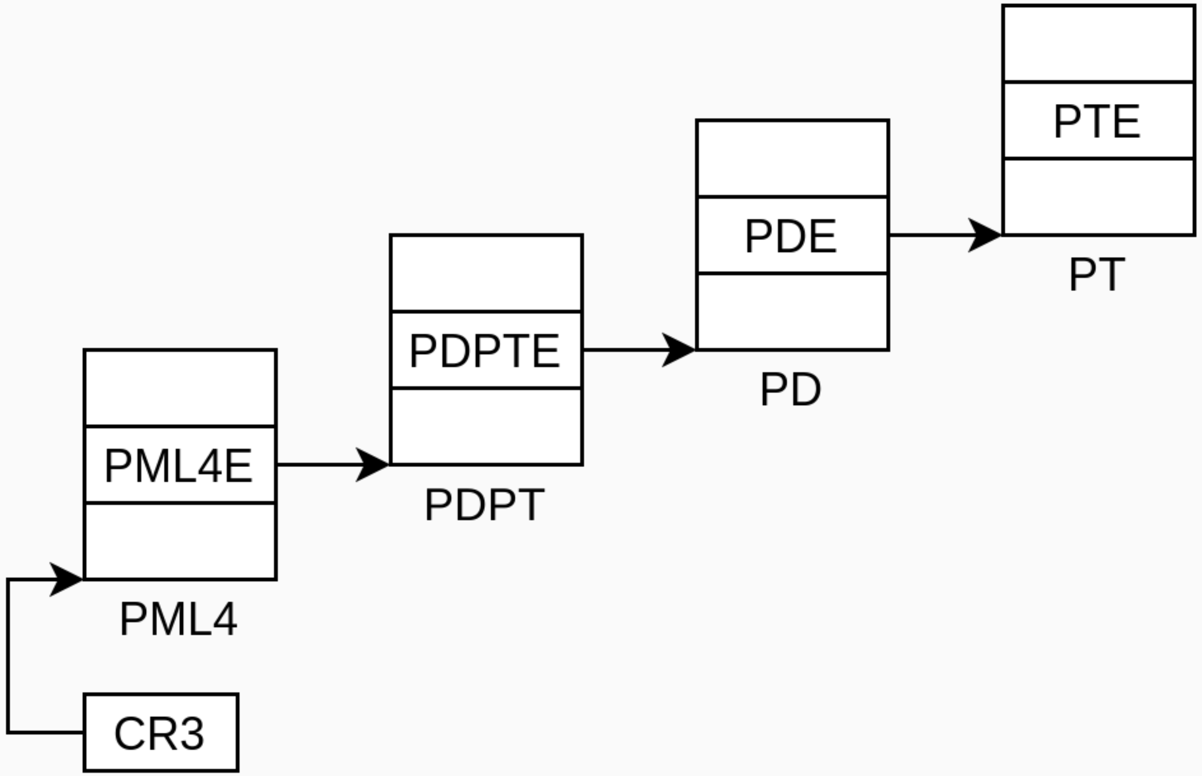
- But still, blue screen on process exit.
- Adjusting Working Set Size, does not fix this issue.
  - -> Analysis TBD

Memory Subversion Detection

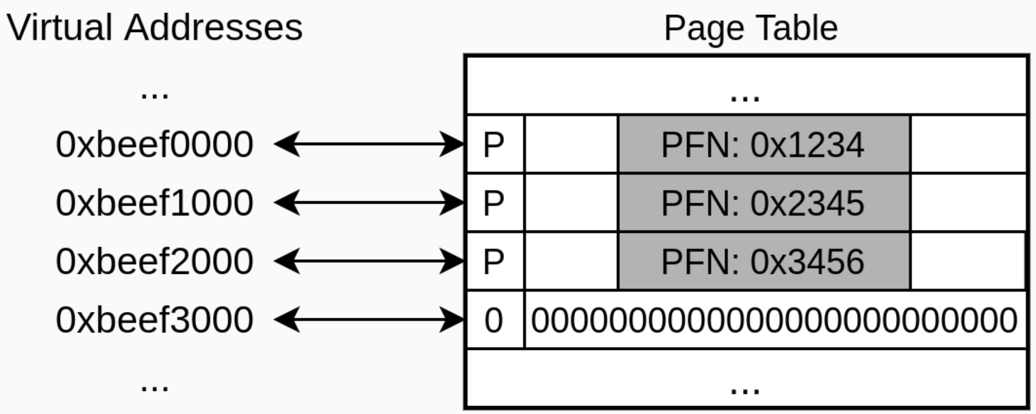
MAS Remapping Detection



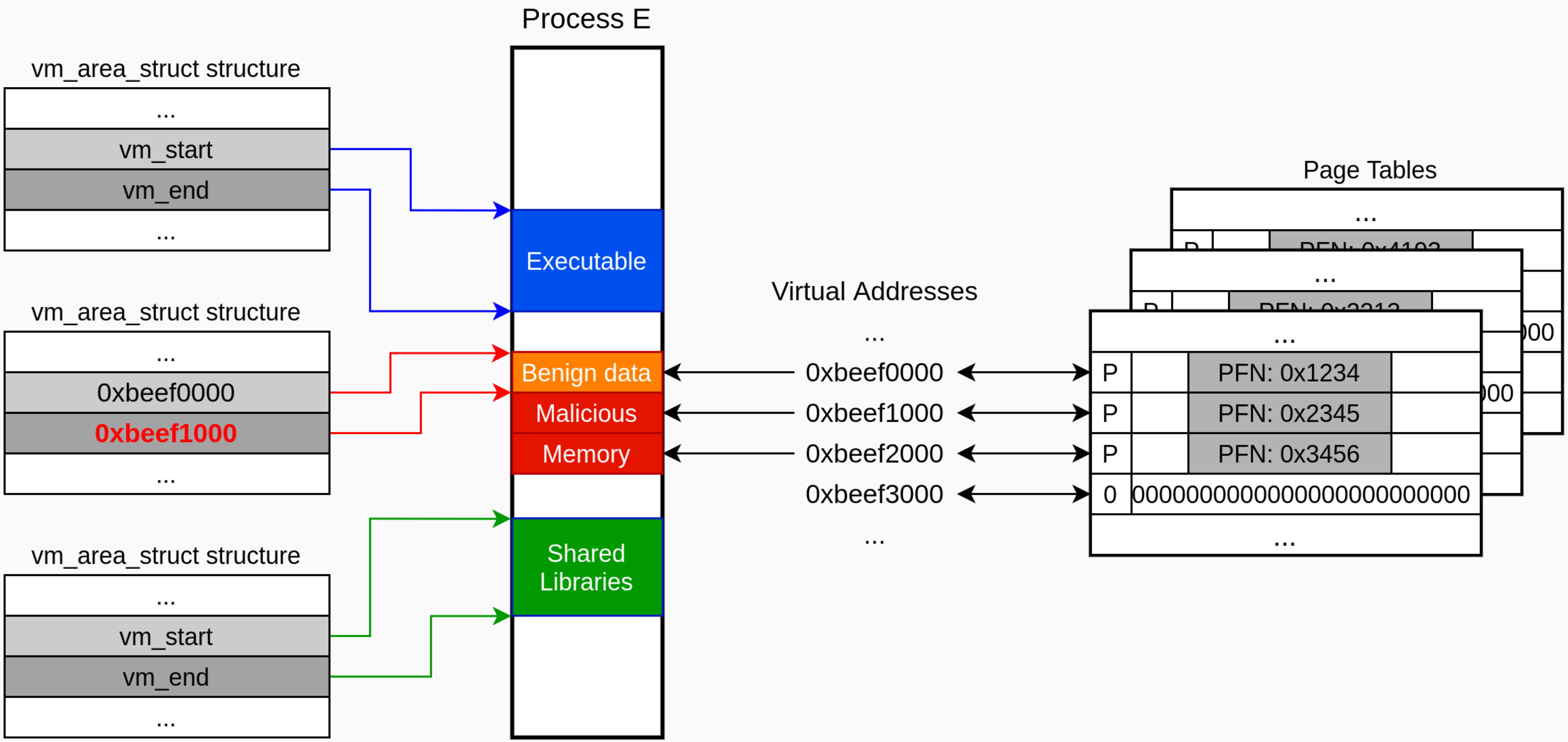
MAS Remapping Detection



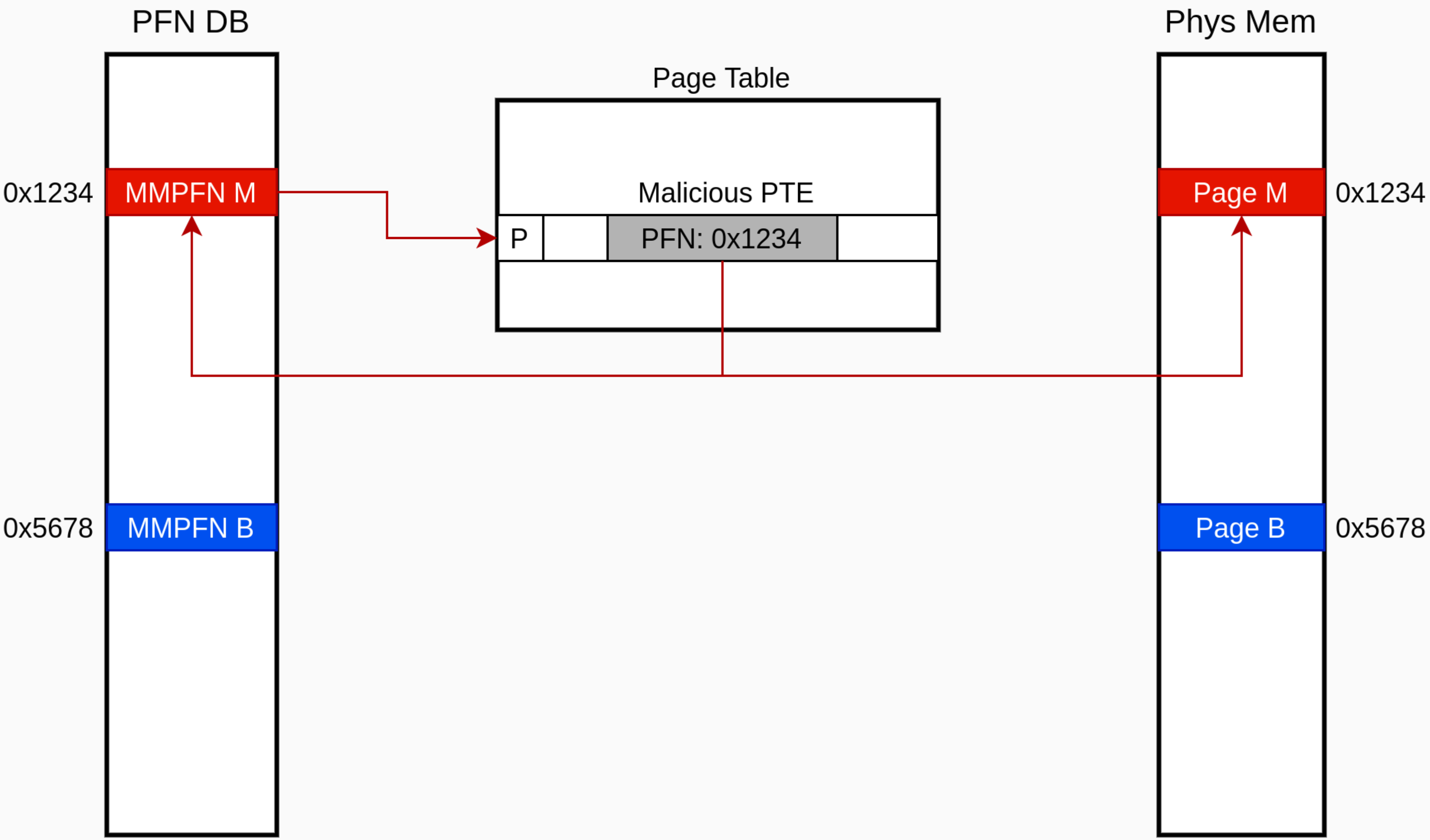
MAS Remapping Detection



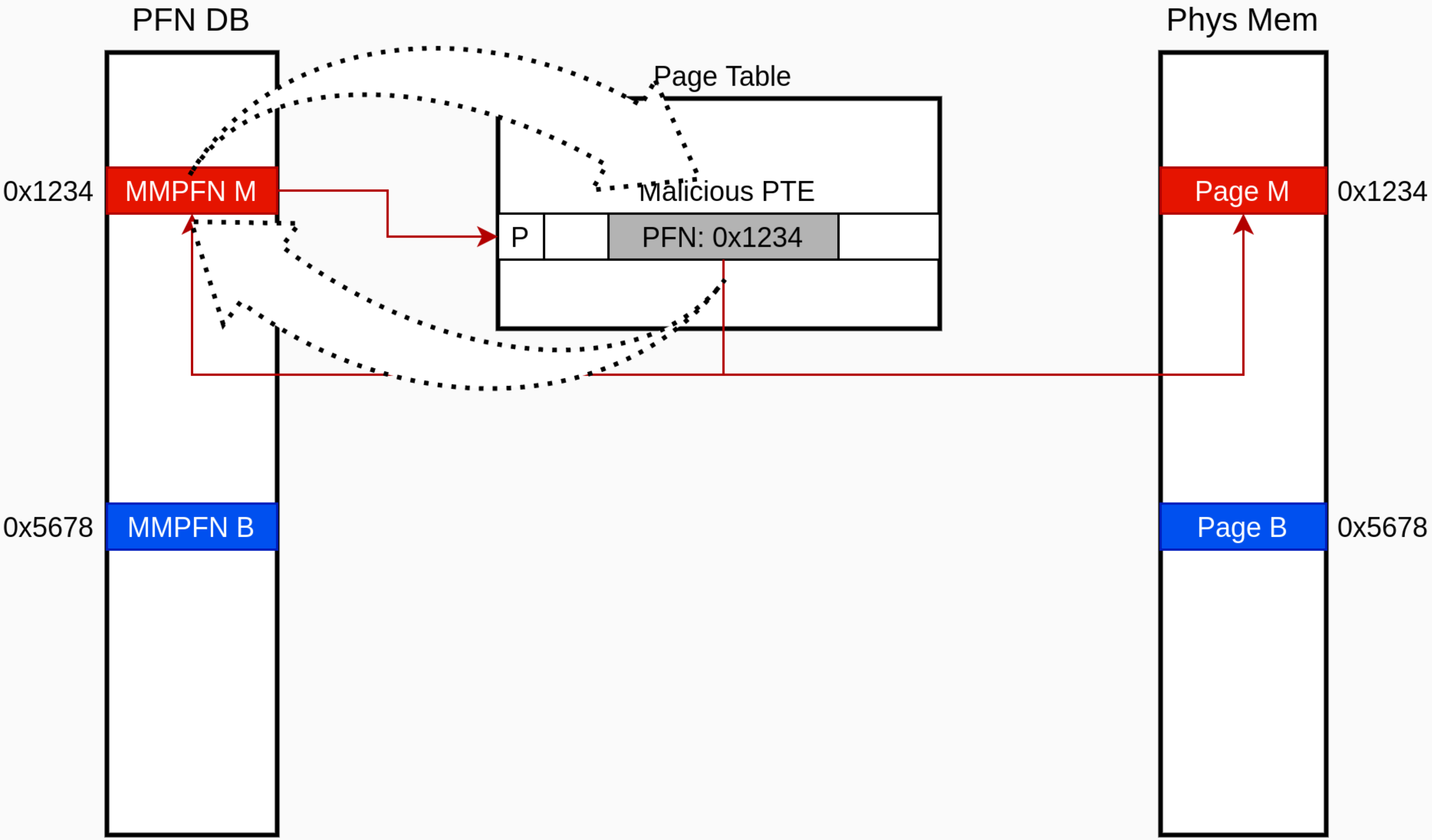
MAS Remapping Detection



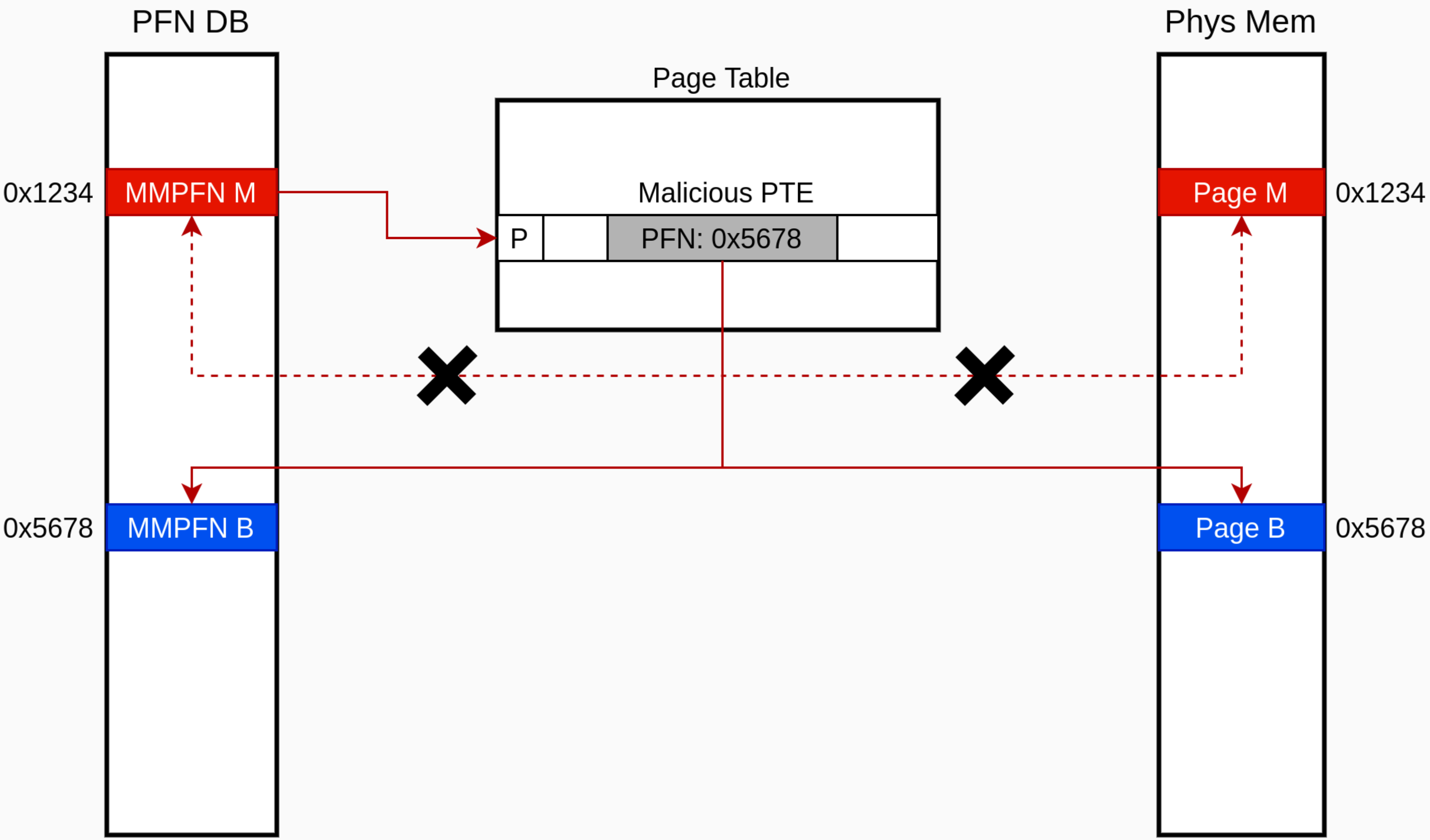
PTE Subversion Detection - Windows



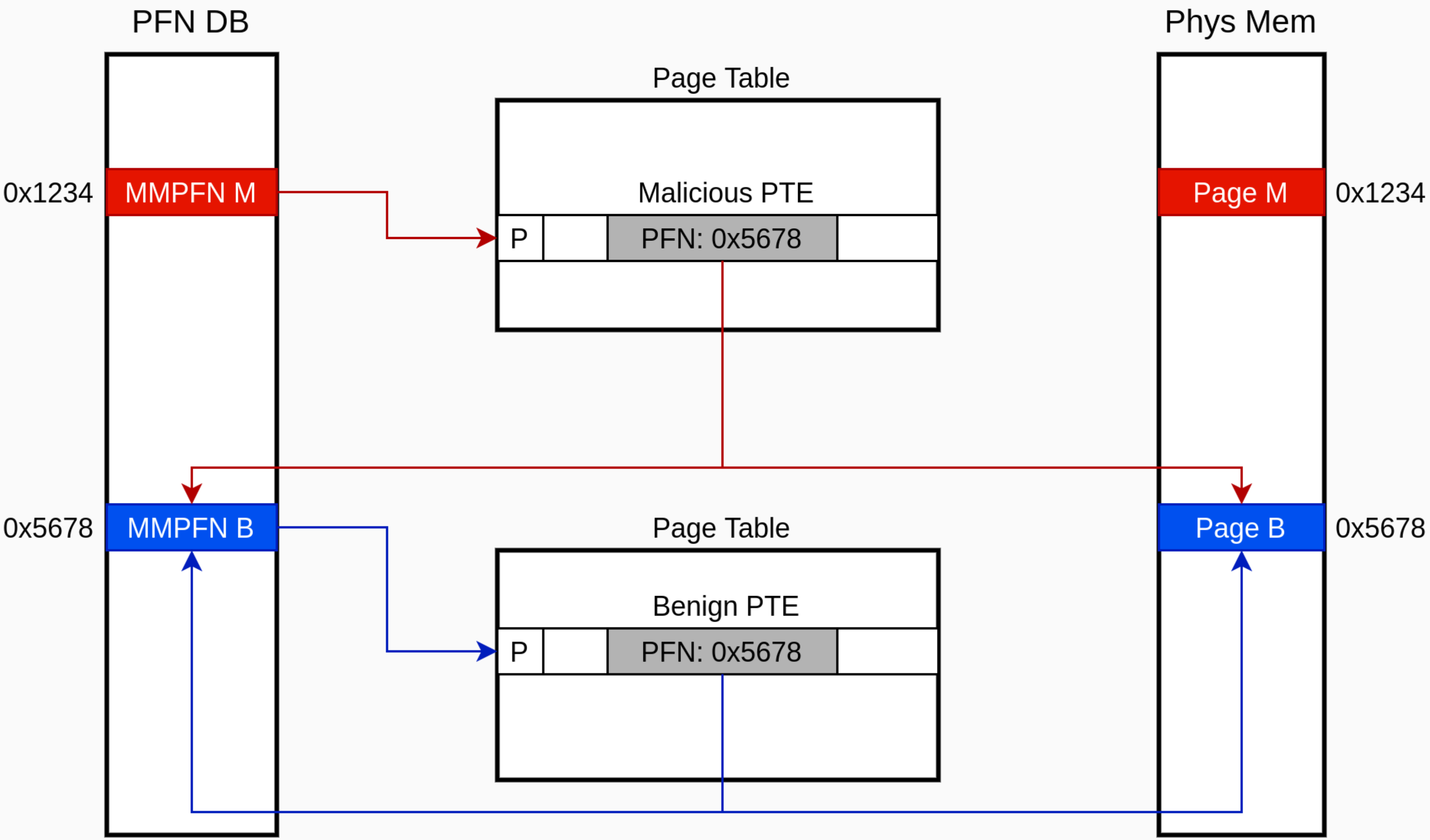
PTE Subversion Detection - Windows



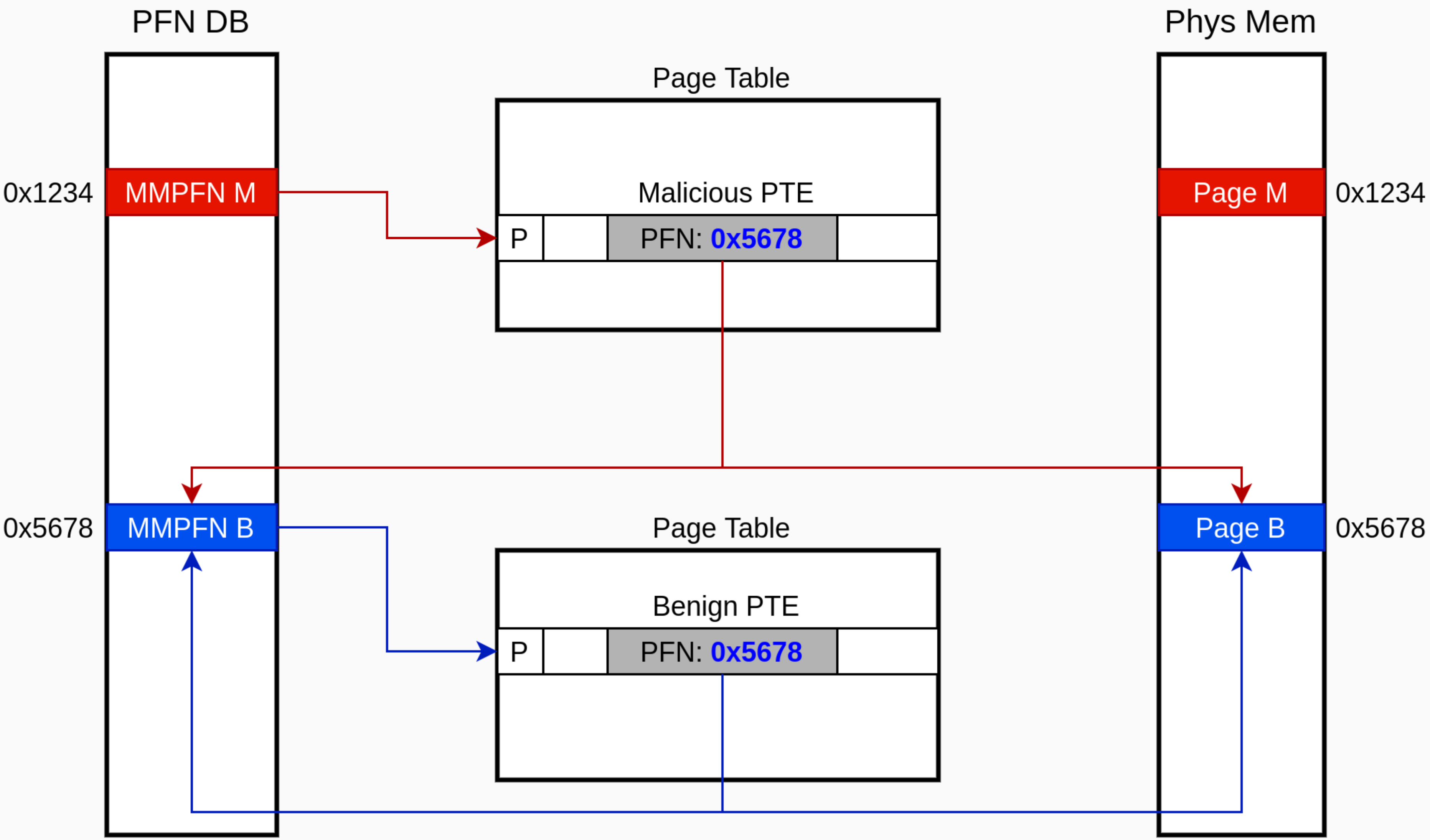
PTE Subversion Detection - Windows



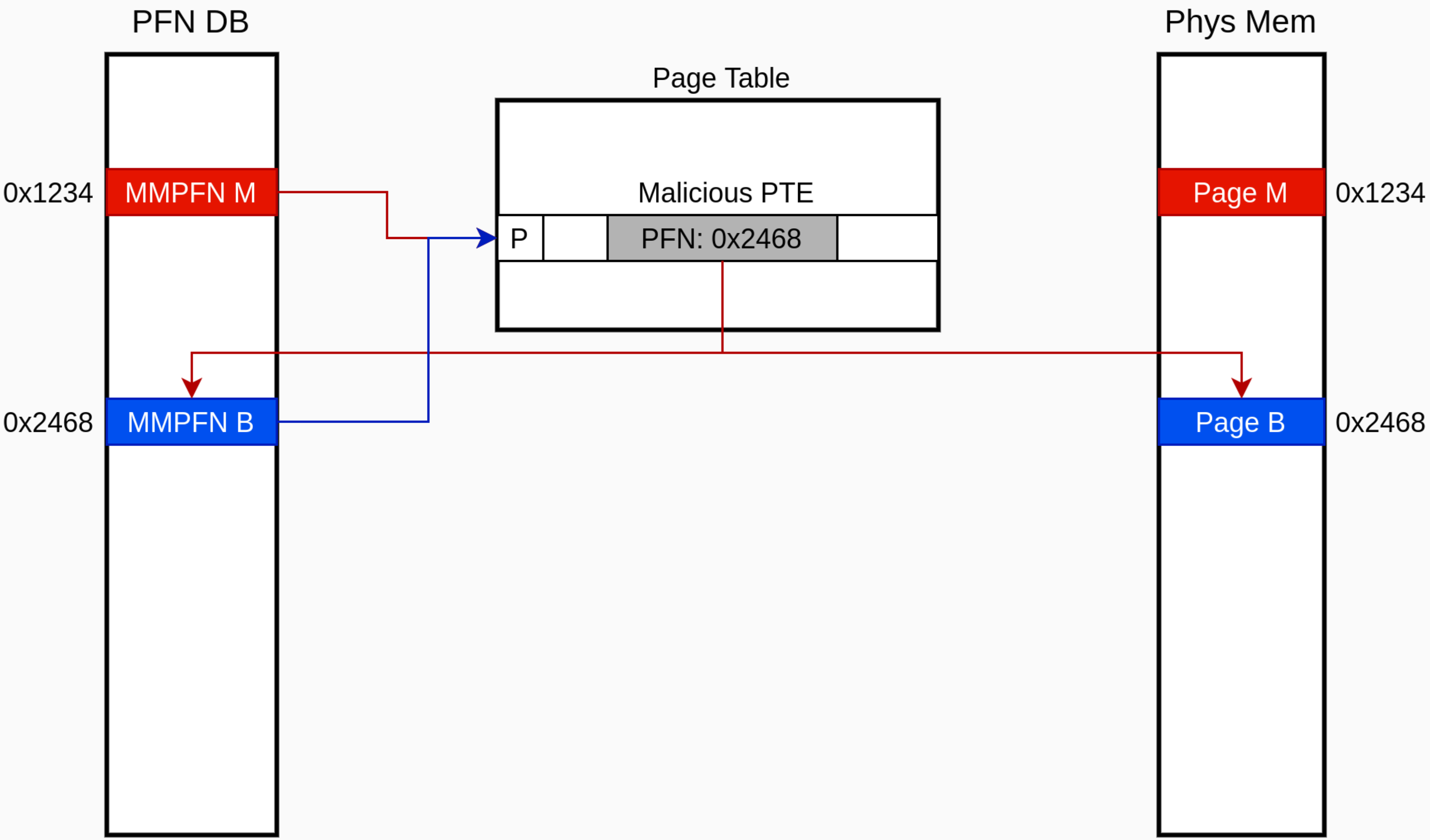
PTE Subversion Detection - Windows



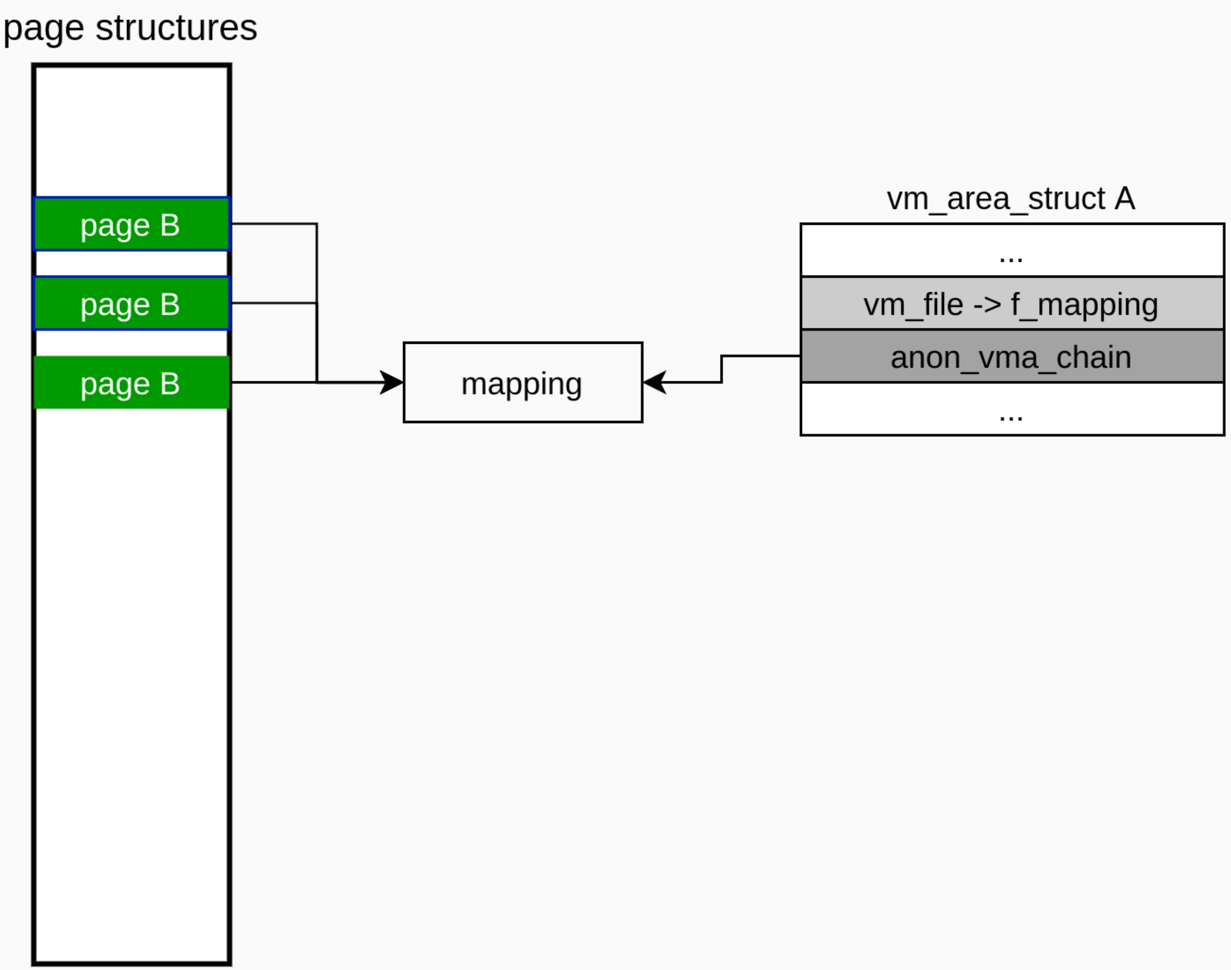
PTE Subversion Detection - Windows



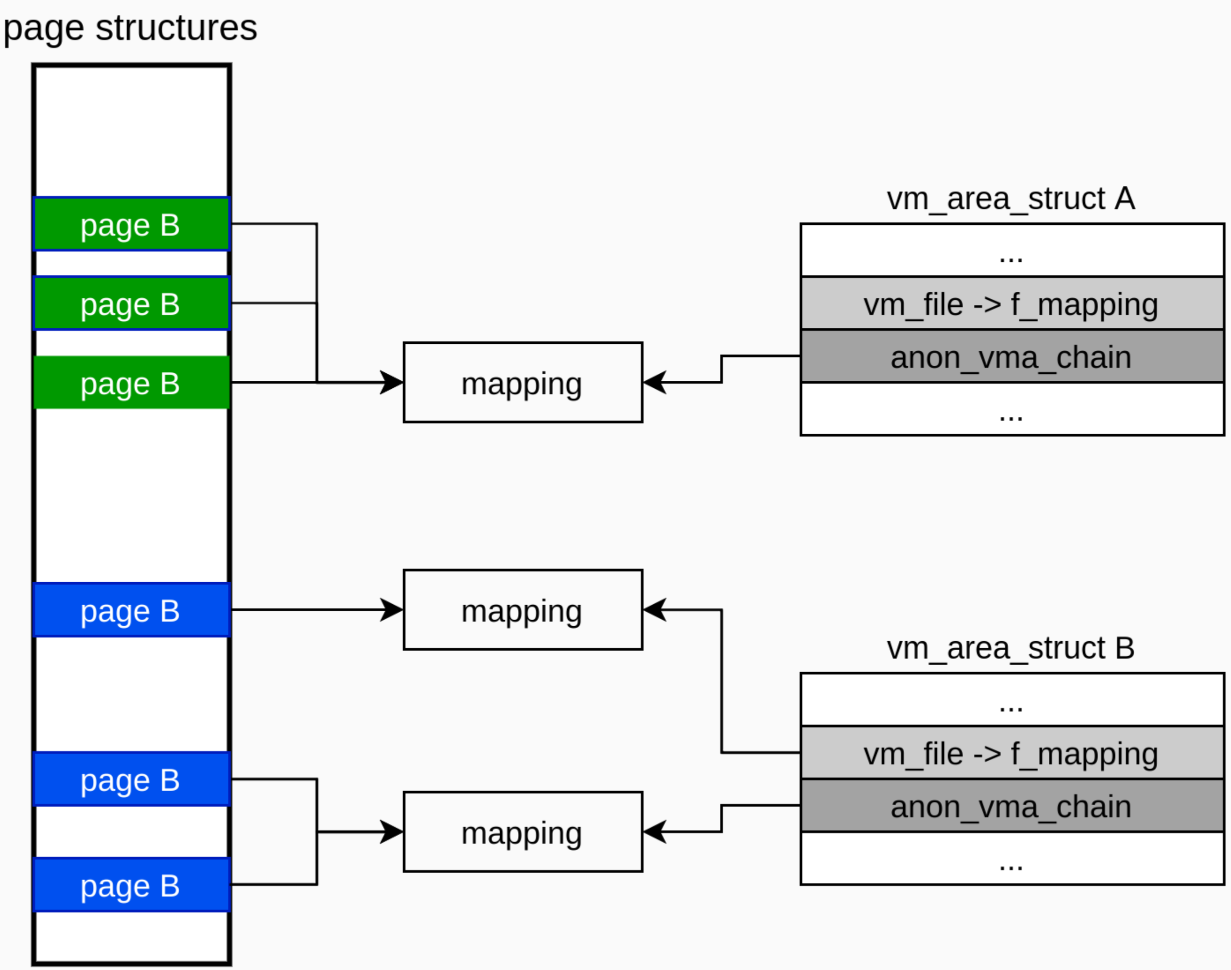
PTE Subversion Detection - Windows



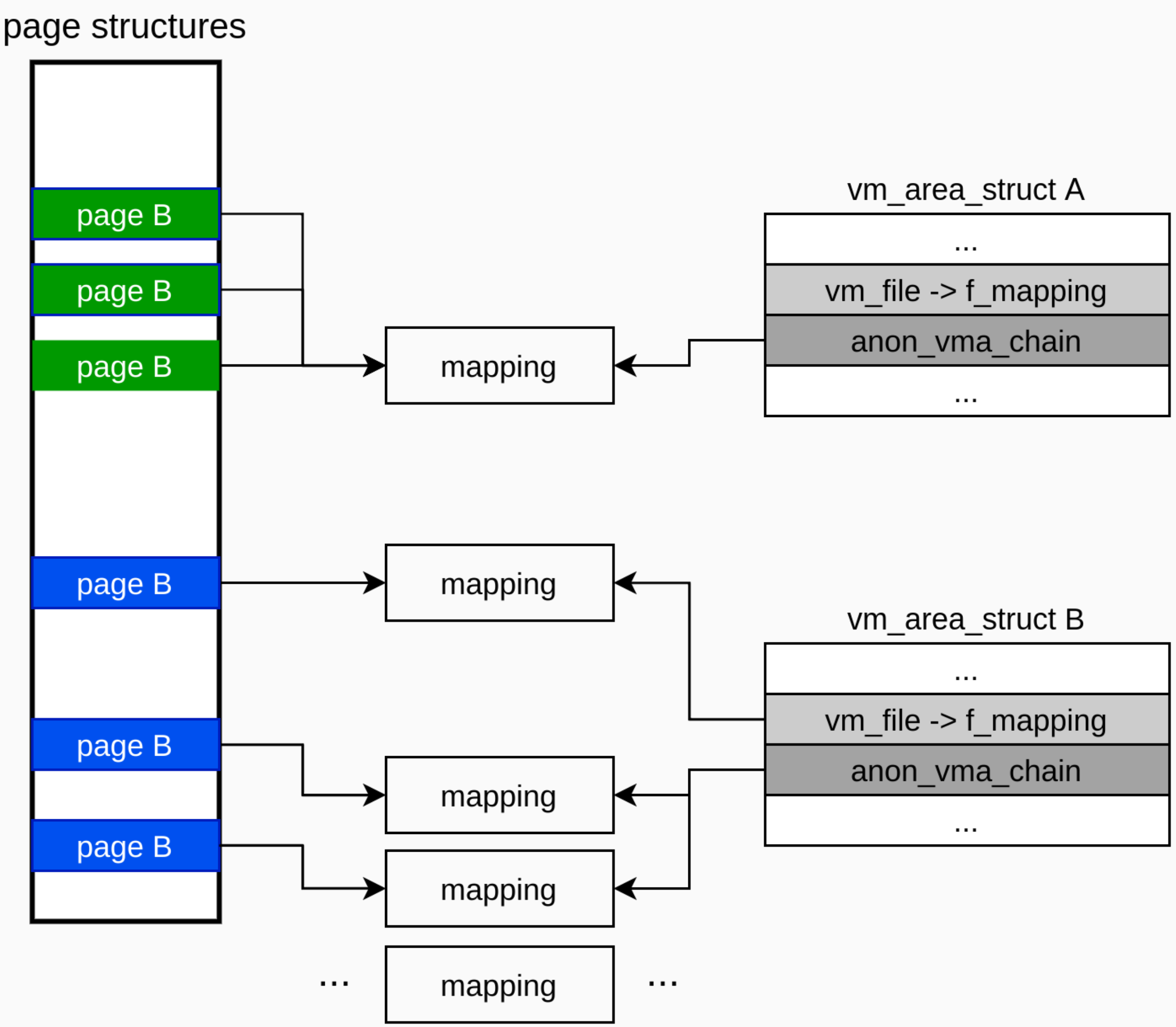
PTE Subversion Detection - Linux



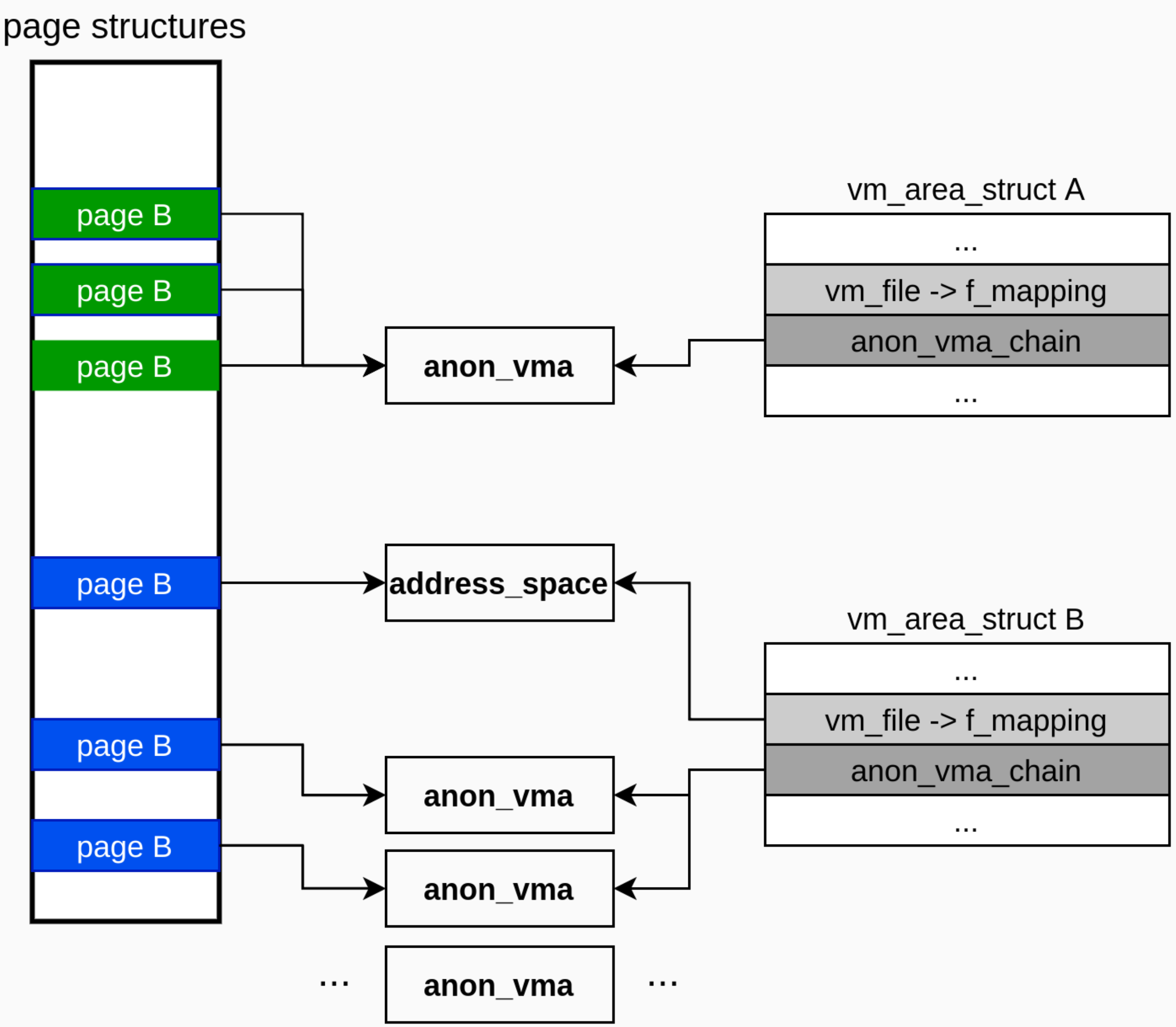
PTE Subversion Detection - Linux



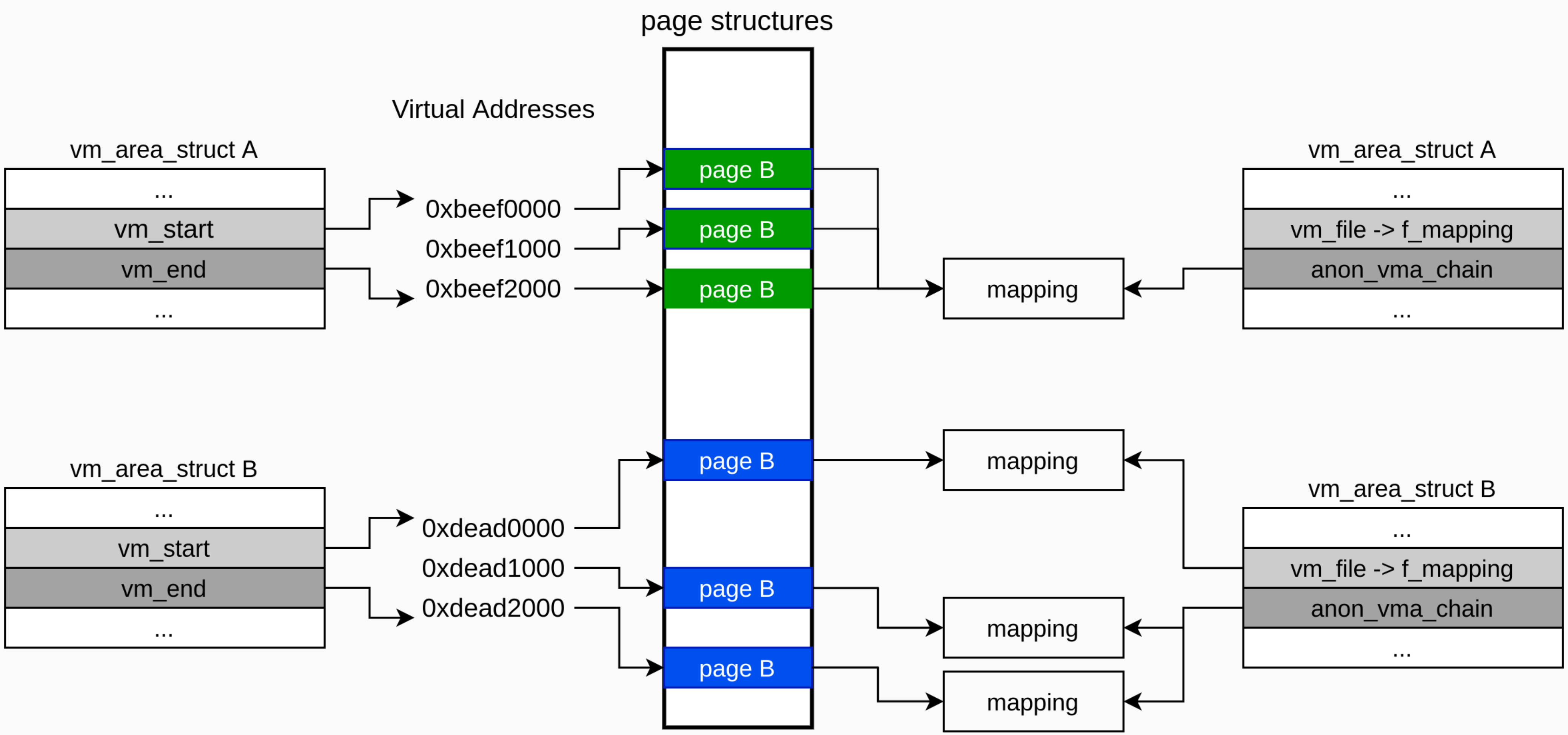
PTE Subversion Detection - Linux



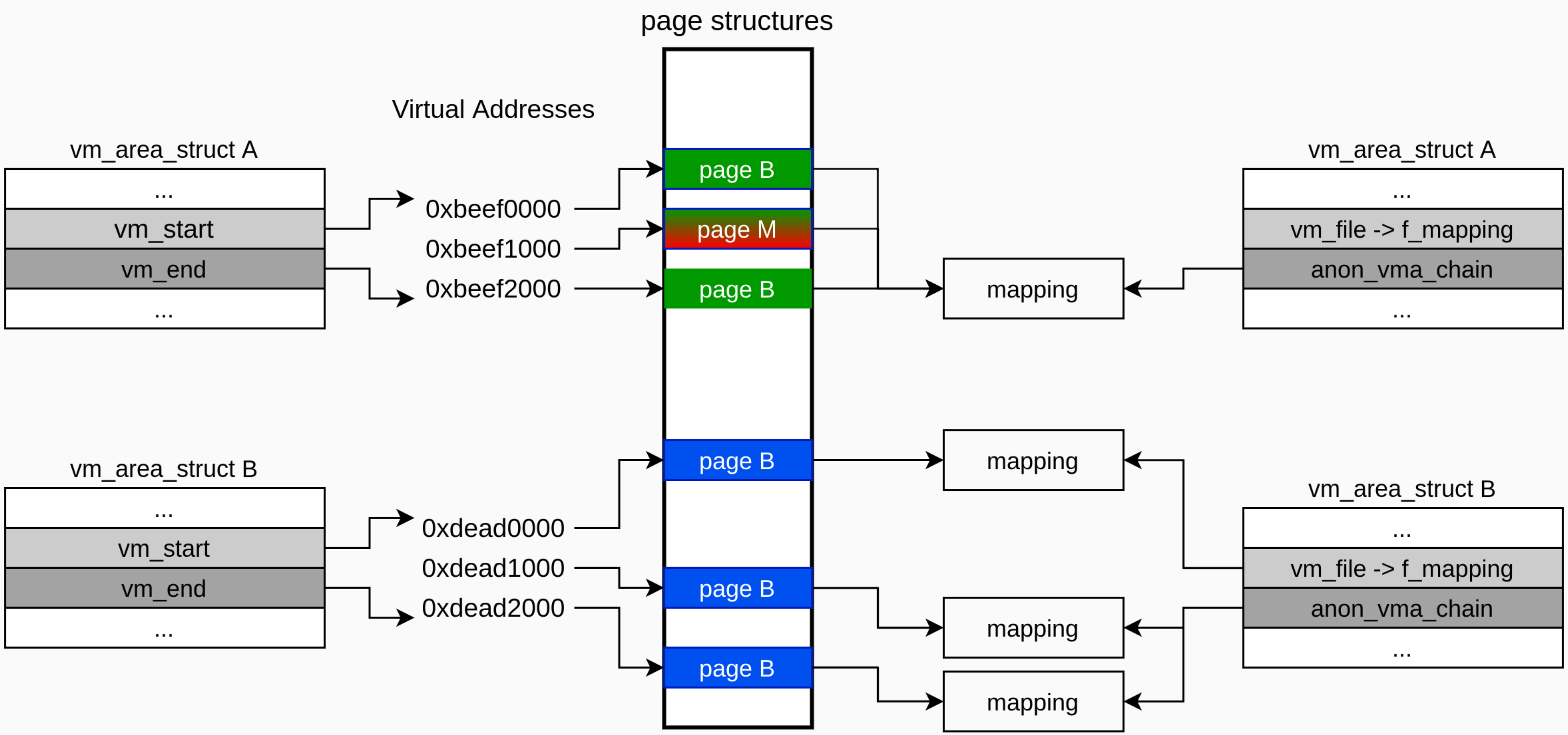
PTE Subversion Detection - Linux



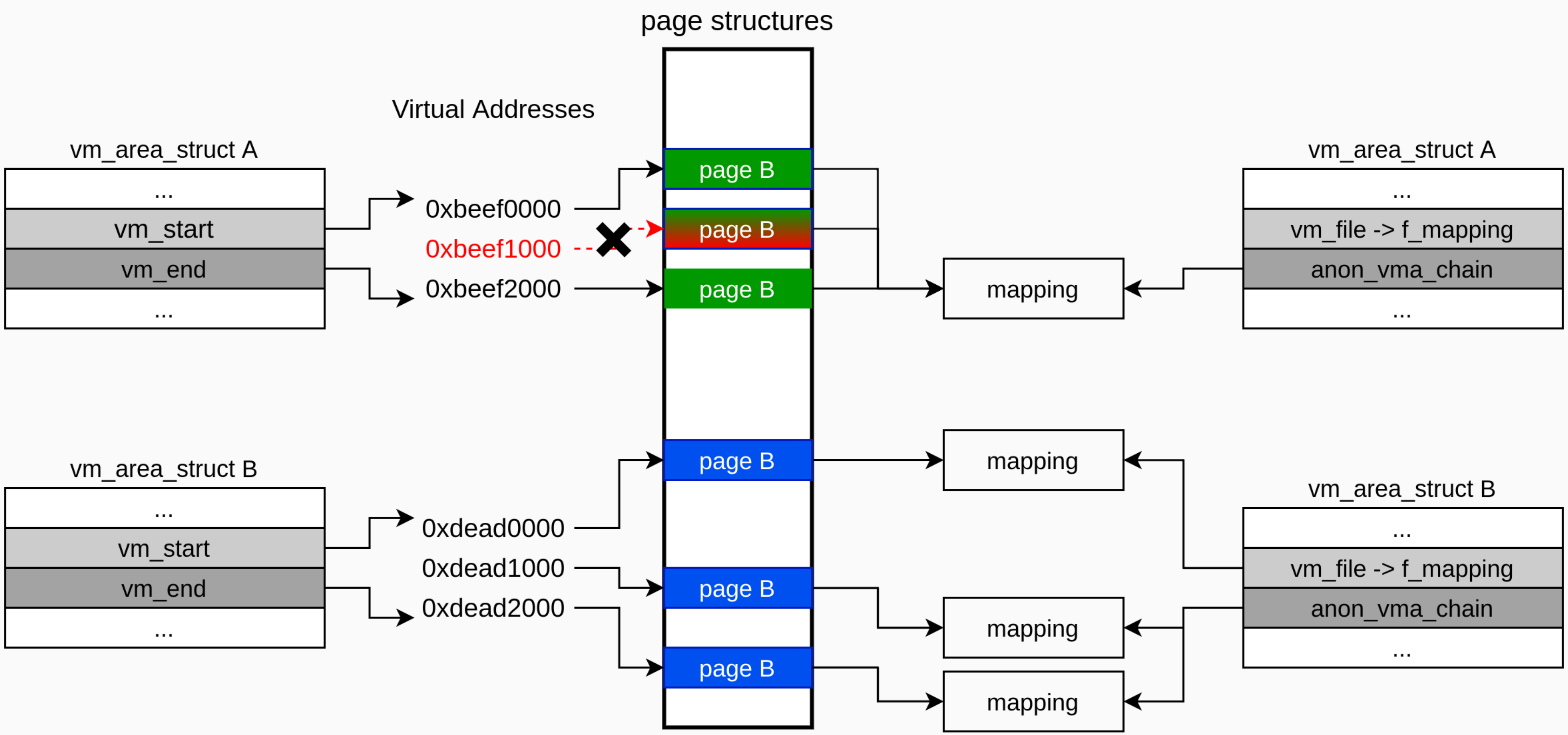
PTE Subversion Detection - Linux



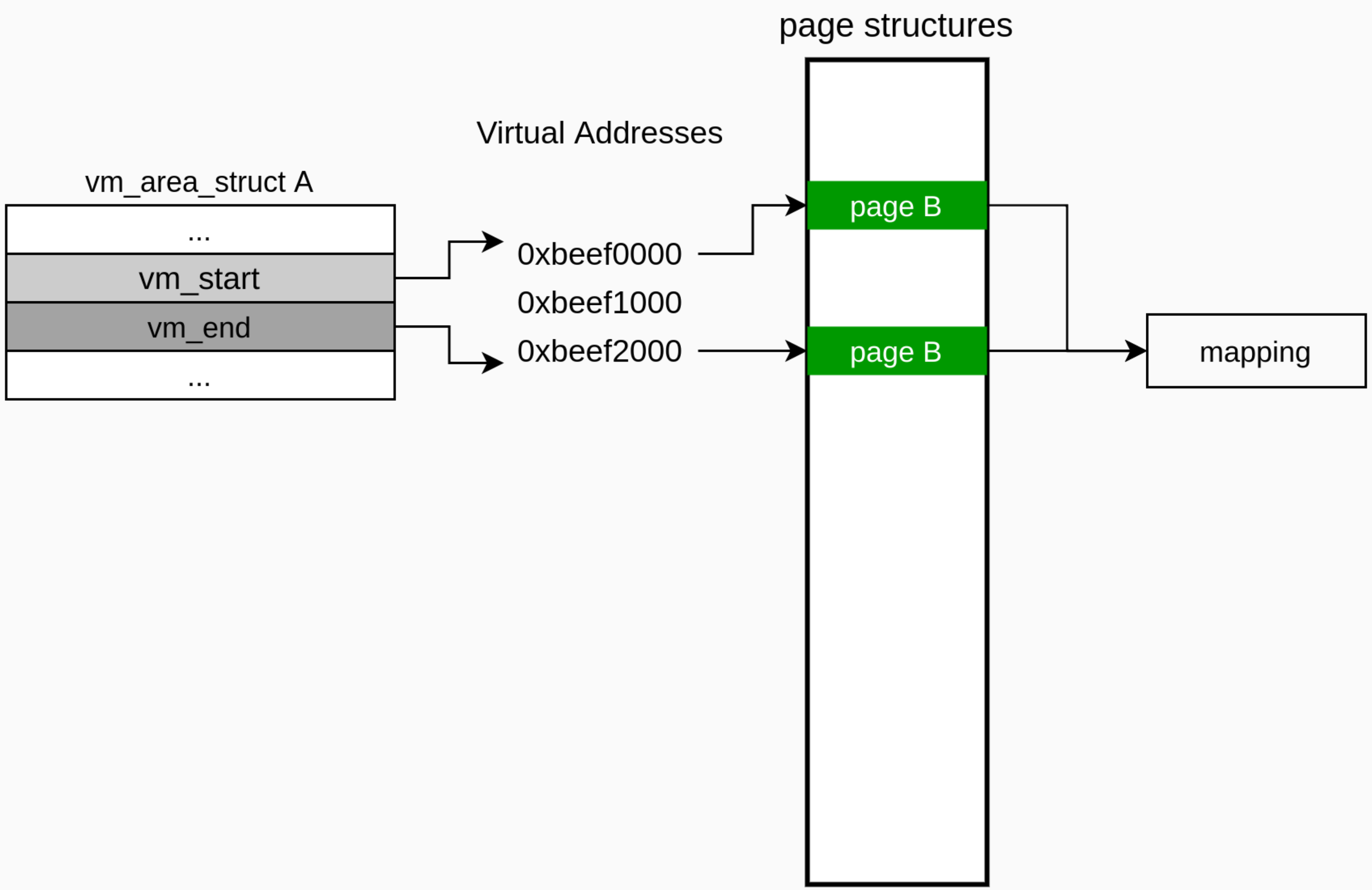
PTE Subversion Detection - Linux



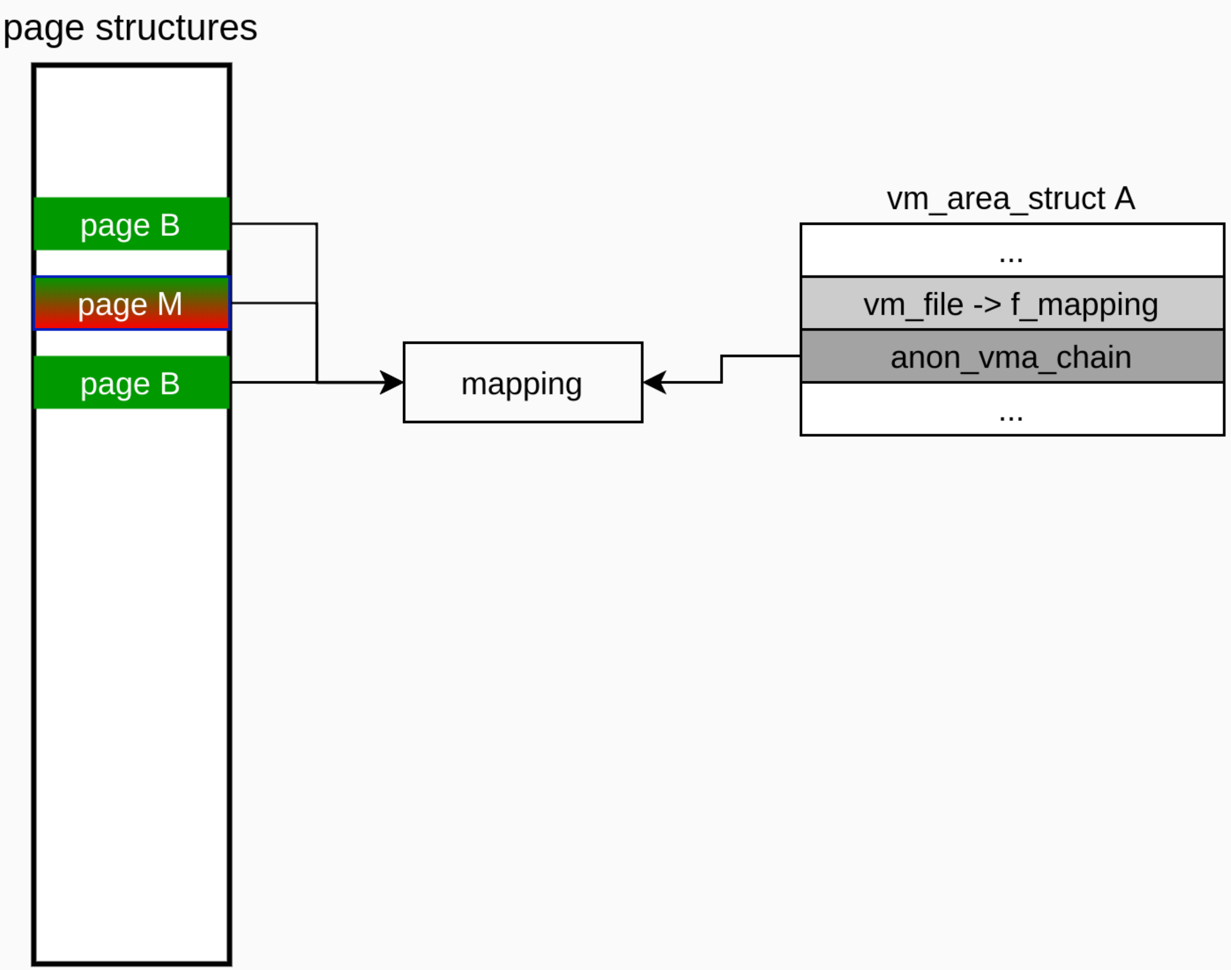
PTE Subversion Detection - Linux



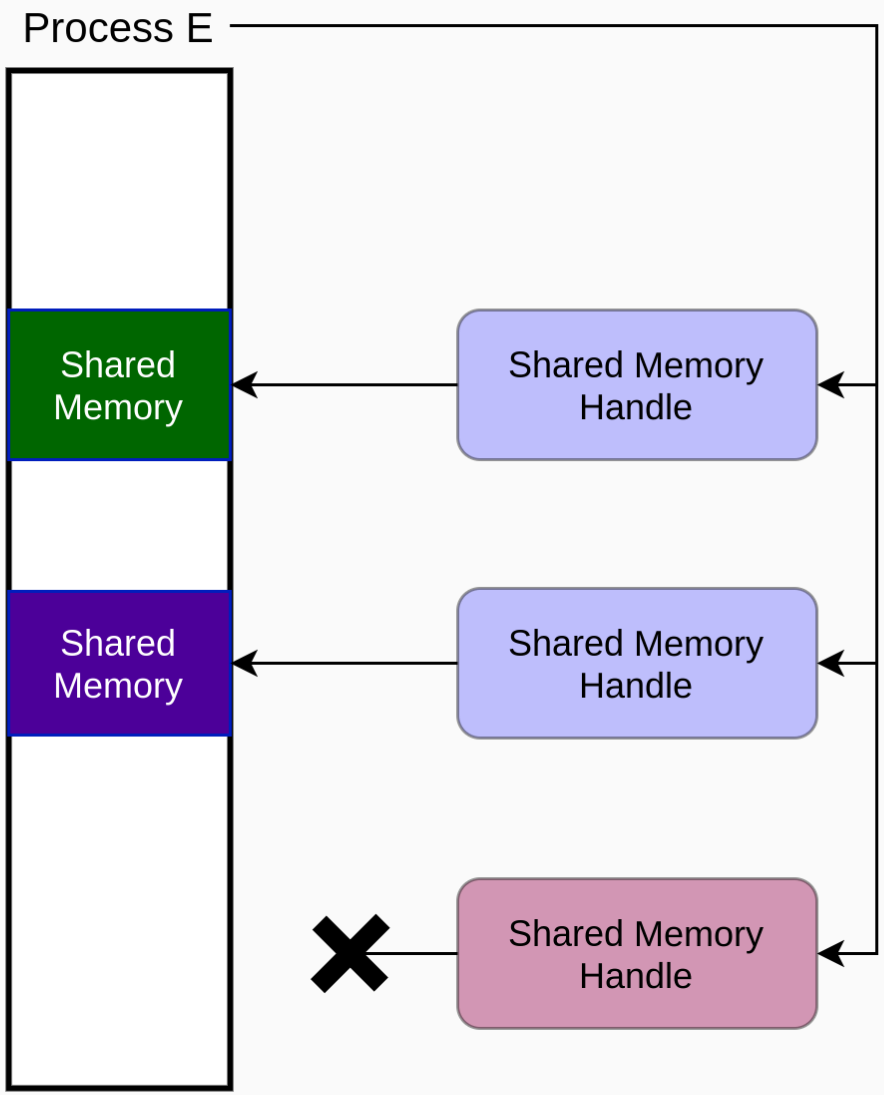
PTE Subversion Detection - Linux



PTE Subversion Detection - Linux



Shared Memory Subversion Detection



Memory Subversion Detection Evaluation

Test environment

- Same VMs as for the memory subversion evaluation.

- Several running applications:
  - Browsers (Firefox, Microsoft Edge, Chromium)
  - Office applications (Microsoft Word, LibreOffice)
  - PDF documents opened in reader application

Detection Evaluation - Windows

- "Standard" PFN remapping and PTE erasure

task	vaddr	pfn	pte_value	orphaned_page	dup_pfn	dup_pte_ptr	pte_ptr_diff	zero_pte
908	0x1a208310000	0x205f8	0xa8b00000205f8867	False	True	False	False	False
3292	0x183f0140000	0x205f8	0x9db00000205f8867	False	True	False	True	False
3292	0x183f0260000	0xfff9	0x0	Not applicable	Not applicable	False	Not applicable	True

- MAS Remapping

task	vaddr	pfn	pte_value	orphaned_page	dup_pfn	dup_pte_ptr	pte_ptr_diff	zero_pte
3604	0x1b4b1701000	0x2be3f	0x6000002be3f867	True	False	False	False	False
3604	0x1b4b1702000	0x2abc0	0x7000002abc0825	True	False	False	False	False

Detection Evaluation - Linux

- PTE remapping and PTE erasure

```
  task        vma         flags      vm_start          vm_end           mapping      file_path
 ------ ---------------- ------ ---------------- ---------------- ---------------- ---------
------------------------------------------------------
Mapping 0x95a2c9ea5ec0 with a page diff of count of 111
------------------------------------------------------
  173    0x95a2cec36320   r, w   0x7ff7170cd000   0x7ff71714d000   0x95a2c9ea5ec0   /run/log/journal/f2fdea958f704b75bcc3001b55fa2dce/system.journal
  173    0x95a2cec36e10   r, w   0x7ff7171d9000   0x7ff7171da000   0x95a2c9ea5ec0   /run/log/journal/f2fdea958f704b75bcc3001b55fa2dce/system.journal
----------------------------------------------------
Mapping 0x95a2cc54c6c0 with a page diff of count of 2
----------------------------------------------------
  1676   0x95a2cefd0e10   r, w   0x7fa58d945000   0x7fa58d948000   0x95a2cc54c6c0
```

- Shared memory

```
    task        file_object                                 devname
------------ ----------------- --------------------------------------
683          0x91e84c675000    /dev/shm/SHM_TEST
```

False Positives - Windows

- MAS remapping
  - There have been false positives with chromium browser in some cases.
  - Also, potentially a page frame in each process on Windows XP SP3 and Windows 7 SP1 containing _KUSER_SHARED_DATA [3], but in our test environments it is part of a VAD.
- PTE subversions
  - None known at the moment.
  - A lot of PFN DB entries for the System process would show up as suspicious, but we are not investigating kernel memory at the moment, so we are lucky there ;)
- Shared Memory Subversion
  - There can be false positives if shared memory is currently just not mapped, but at least in our test environment we did encounter none.
  - There are, however, hundreds if considering also non-executable memory.

False Positives - Linux

- MAS Remapping
  - Not yet implemented
- PTE subversions
  - 127 page frames for to 12 mappings for Firefox-ESR (related to shared memory), and systemd-journal process almost always pops up with its system.journal file (with 1 to 400 page frames).
- Shared Memory Subversion
  - In our test case 42: One for the Cron process, 40 for Firefox-ESR and 1 for Chromium
  - We are only considering anonymous shared memory. If not -> 1000 more false positives.

Conclusion

Comparison – Attacker's Point of View

Technique	Requires Kernel Access	Requires Control Code	Advantages	Disadvantages
MAS Remapping	Yes	No	• Easy to setup.	• Detectable with already existing approaches.
PTE Subversions	Yes	Yes	• Detection can be problematic.	• Not as easy and totally safe to set up.
Shared Memory Subversion	No	Yes	• Easy and safe to set up. • Most stealthy approach.	• Easy to detect with our approach.

Conclusion

- Three novel techniques, which successfully hide memory from live and memory forensics, on Linux and Windows.
- Proof-of-concept implementations for both Windows and Linux [5].
- Rekall and Volatility plugins for detection [5] (with limitations).

Limitations

- Blue Screen with PTE subversions on Windows when process exits.
- Detection approaches for PTE subversions on Windows can be circumvented by using shared memory.
- False positives with detections on Linux.
- Additional Problem on Linux: page instances without mapping.
- Our PTE subversion detection for Linux takes a sh**load of time (there is an easy fix).
  - For the same reason, MAS Remapping detection not yet available. TBD

Future Work

- Mapping just a sub view of the shared memory.
- Evaluation with kernel space.
- MAS remapping detection for Linux.
- Resolve Windows crash with PTE subversions.
- Testing against the Windows 10 Memory combining feature.
- Decreasing the memory footprint of the control code.
- Manipulating everything that we’ve used for detection.

I wanted to thank

- The people behind Rekall, Volatility and "The Art Of Memory Forensics" for their amazing work.
- All mentioned (and not explicitly mentioned) researchers, who inspired or helped in doing this research with their work.
- Enrico Martignetti for his great book on Windows memory management: "What Makes It Page? The Windows 7 (X64) Virtual Memory Manager"
- My girlfriend for her patience during the last months, while I was primarily sitting on the couch in my underpants, doing nerd stuff. Promise: I will take care of cooking now (at least for the next weeks ;)

Thank you!

Resources

- The Research paper, memory subversion PoC code and any material to reproduce our research results: https://github.com/DFRWS-memory-subversion/DFRWS-USA-2020

- All Rekall/Volatility plugins and a Shared Memory implementation with C&C: https://github.com/f-block/BlackHat-USA-2020

References

[1] Sparks, S., Butler, J., 2005. Shadow walker: Raising the bar for rootkit detection. Black Hat Japan 11, 504–533.

[2] Lospinoso, J., 2017. Gargoyle - a memory scanning evasion technique. URL: https://github.com/JLospinoso/gargoyle

[3] White, A., Schatz, B., Foo, E., 2012. Surveying the user space through user allocations. Digital Investigation 9, S3–S12. URL: https://www.dfrws.org/sites/default/files/session-files/paper-surveying_the_user_space_through_user_allocations.pdf

[4] Palutke, R., Block, F., Reichenberger, P., Stripeika, D., 2020. Hiding Process Memory via Anti-Forensic Techniques. Digital Investigation. URL: https://dfrws.org/presentation/hiding-process-memory-via-anti-forensic-techniques/

[5] Reichenberger, P., Stripeika, D., Block, F., Palutke, R., 2020. The public repository containing the code and binaries used in this work. URL: https://github.com/DFRWS-memory-subversion/DFRWS-USA-2020