A small library to modify all page-table levels of all processes from user space for x86_64 and ARMv8.

demos	first commit	last year
doc	first commit	last year
module	Fixed module for ARM64.	20 days ago
.gitignore	first commit	last year
LICENSE	first commit	last year
Makefile	first commit	last year
README.md	updated readme	last year
example.c	Fix naming convention error.	last month
ptedit.c	Fix naming convention error.	last month
ptedit.h	Fix naming convention error.	last month
ptedit_header.h	Fix naming convention error.	last month

README.md

PTEditor

A small library to modify all page-table levels of all processes from user space for x86_64 and ARMv8. It also allows to read and program memory types (i.e., PATs on x86 and MAIRs on ARM).

Installation

The library relies on the `pteditor` kernel module. Both the library and the the kernel module can be build by running

```
make
```

The library can be used by linking it to the application (see `example.c`) or as a single header (`ptedit_header.h`) which can be directly included (see the demos).

Requirements

The library requires a recent Linux kernel (tested on 4.11, but should also work on 3.x kernels). It supports both x86_64 and ARMv8.

The library does not rely on any other library. It uses only standard C functionality.

Example

The basic functionality (`ptedit_init` and `ptedit_cleanup`) is always required. After the initialization, all functions provided by the library can be used.

For examples see `example.c` or the examples in the `demo` folder. The `demo` folder contains multiple examples:

- `memmap`: Starting from the root of paging, the demo iterates through all page tables of all levels and dumps the contents of the entries.
- `map_pt`: A Rowhamer exploit simulation, which maps the page table to a user-accessible address for manipulation.
- `uncachable`: This demos manipulates the memory type of a mapping to uncachable and back to cachable.
- `nx`: After setting a function to non-executable, it uses the page tables to make the function executable again.
- `virt2phys`: Converts a virtual to a physical address.

API

Basic Functionality	Descriptions
`int ptedit_init ()`	Initializes (and acquires) PTEditor kernel module
`void ptedit_cleanup ()`	Releases PTEditor kernel module

Page tables	Descriptions
`ptedit_entry_t ptedit_resolve (void * address,pid_t pid)`	Resolves the page-table entries of all levels for a virtual address of a given process.
`void ptedit_update (void * address,pid_t pid,ptedit_entry_t * vm)`	Updates one or more page-table entries for a virtual address of a given process. The TLB for the given address is flushed after updating the entries.
`void ptedit_pte_set_bit (void * address,pid_t pid,int bit)`	Sets a bit directly in the PTE of an address.
`void ptedit_pte_clear_bit (void * address,pid_t pid,int bit)`	Clears a bit directly in the PTE of an address.
`unsigned char ptedit_pte_get_bit (void * address,pid_t pid,int bit)`	Returns the value of a bit directly from the PTE of an address.
`size_t ptedit_pte_get_pfn (void * address,pid_t pid)`	Reads the PFN directly from the PTE of an address.
`void ptedit_pte_set_pfn (void * address,pid_t pid,size_t pfn)`	Sets the PFN directly in the PTE of an address.

System Info	Descriptions
`int ptedit_get_pagesize ()`	Returns the default page size of the system

Page frame numbers (PFN)	Descriptions
`size_t ptedit_set_pfn (size_t entry,size_t pfn)`	Returns a new page-table entry where the page-frame number (PFN) is replaced by the specified one.
`size_t ptedit_get_pfn (size_t entry)`	Returns the page-frame number (PFN) of a page-table entry.

Physical pages	Descriptions
`void ptedit_read_physical_page (size_t pfn,char * buffer)`	Retrieves the content of a physical page.
`void ptedit_write_physical_page (size_t pfn,char * content)`	Replaces the content of a physical page.

Paging	Descriptions
`size_t ptedit_get_paging_root (pid_t pid)`	Returns the root of the paging structure (i.e., CR3 on x86 and TTBR0 on ARM).
`void ptedit_set_paging_root (pid_t pid,size_t root)`	Sets the root of the paging structure (i.e., CR3 on x86 and TTBR0 on ARM).

TLB/Barriers	Descriptions
`void ptedit_invalidate_tlb (void * address)`	Invalidates the TLB for a given address on all CPUs.
`void ptedit_full_serializing_barrier ()`	A full serializing barrier which stops everything.

Memory types (PATs/MAIRs)	Descriptions
`size_t ptedit_get_mts ()`	Reads the value of all memory types (x86 PATs / ARM MAIRs). This is equivalent to reading the MSR 0x277 (x86) / MAIR_EL1 (ARM).
`void ptedit_set_mts (size_t mts)`	Programs the value of all memory types (x86 PATs / ARM MAIRs). This is equivalent to writing to the MSR 0x277 (x86) / MAIR_EL1 (ARM) on all CPUs.
`char ptedit_get_mt (unsigned char mt)`	Reads the value of a specific memory type attribute (PAT/MAIR).
`void ptedit_set_mt (unsigned char mt,unsigned char value)`	Programs the value of a specific memory type attribute (PAT/MAIR).
`unsigned char ptedit_find_mt (unsigned char type)`	Generates a bitmask of all memory type attributes (PAT/MAIR) which are programmed to the given value.
`int ptedit_find_first_mt (unsigned char type)`	Returns the first memory type attribute (PAT/MAIR) which is programmed to the given memory type.
`size_t ptedit_apply_mt (size_t entry,unsigned char mt)`	Returns a new page-table entry which uses the given memory type (PAT/MAIR).
`unsigned char ptedit_extract_mt (size_t entry)`	Returns the memory type (i.e., PAT/MAIR ID) which is used by a page-table entry.
`const char * ptedit_mt_to_string (unsigned char mt)`	Returns a human-readable representation of a memory type (PAT/MAIR value).

Pretty print	Descriptions
`void ptedit_print_entry (size_t entry)`	Pretty prints a page-table entry.
`void ptedit_print_entry_line (size_t entry,int line)`	Prints a single line of the pretty-print representation of a page-table entry.

Basic Functionality

`int ptedit_init ()`

Initializes (and acquires) PTEditor kernel module

Returns 0 Initialization was successful

Returns -1 Initialization failed

`void ptedit_cleanup ()`

Releases PTEditor kernel module

Page tables

`ptedit_entry_t ptedit_resolve (void * address,pid_t pid)`

Resolves the page-table entries of all levels for a virtual address of a given process.

Parameters

- `address` The virtual address to resolve
- `pid` The pid of the process (0 for own process)

Returns A structure containing the page-table entries of all levels.

`void ptedit_update (void * address,pid_t pid,ptedit_entry_t * vm)`

Updates one or more page-table entries for a virtual address of a given process. The TLB for the given address is flushed after updating the entries.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)
- `vm` A structure containing the values for the page-table entries and a bitmask indicating which entries to update

`void ptedit_pte_set_bit (void * address,pid_t pid,int bit)`

Sets a bit directly in the PTE of an address.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)
- `bit` The bit to set (one of PTEDIT_PAGE_BIT_*)

`void ptedit_pte_clear_bit (void * address,pid_t pid,int bit)`

Clears a bit directly in the PTE of an address.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)
- `bit` The bit to clear (one of PTEDIT_PAGE_BIT_*)

`unsigned char ptedit_pte_get_bit (void * address,pid_t pid,int bit)`

Returns the value of a bit directly from the PTE of an address.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)
- `bit` The bit to get (one of PTEDIT_PAGE_BIT_*)

Returns The value of the bit (0 or 1)

`size_t ptedit_pte_get_pfn (void * address,pid_t pid)`

Reads the PFN directly from the PTE of an address.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)

Returns The page-frame number (PFN)

`void ptedit_pte_set_pfn (void * address,pid_t pid,size_t pfn)`

Sets the PFN directly in the PTE of an address.

Parameters

- `address` The virtual address
- `pid` The pid of the process (0 for own process)
- `pfn` The new page-frame number (PFN)

System info

`int ptedit_get_pagesize ()`

Returns the default page size of the system

Returns Page size of the system in bytes

Page frame numbers (PFN)

`size_t ptedit_set_pfn (size_t entry,size_t pfn)`

Returns a new page-table entry where the page-frame number (PFN) is replaced by the specified one.

Parameters

- `entry` The page-table entry to modify
- `pfn` The new page-frame number (PFN)

Returns A new page-table entry with the given page-frame number

`size_t ptedit_get_pfn (size_t entry)`

Returns the page-frame number (PFN) of a page-table entry.

Parameters

- `entry` The page-table entry to extract the PFN from

Returns The page-frame number

Physical pages

`void ptedit_read_physical_page (size_t pfn,char * buffer)`

Retrieves the content of a physical page.

Parameters

- `pfn` The page-frame number (PFN) of the page to read
- `buffer` A buffer which is large enough to hold the content of the page

`void ptedit_write_physical_page (size_t pfn,char * content)`

Replaces the content of a physical page.

Parameters

- `pfn` The page-frame number (PFN) of the page to update
- `content` A buffer containing the new content of the page (must be the size of a physical page)

Paging

`size_t ptedit_get_paging_root (pid_t pid)`

Returns the root of the paging structure (i.e., CR3 on x86 and TTBR0 on ARM).

Parameters

- `pid` The proccess id (0 for own process)

Returns The phyiscal address (not PFN!) of the first page table (i.e., the PGD)

`void ptedit_set_paging_root (pid_t pid,size_t root)`

Sets the root of the paging structure (i.e., CR3 on x86 and TTBR0 on ARM).

Parameters

- `pid` The proccess id (0 for own process)
- `root` The physical address (not PFN!) of the first page table (i.e., the PGD)

TLB/Barriers

`void ptedit_invalidate_tlb (void * address)`

Invalidates the TLB for a given address on all CPUs.

Parameters

- `address` The address to invalidate

`void ptedit_full_serializing_barrier ()`

A full serializing barrier which stops everything.

Memory types (PATs/MAIRs)

`size_t ptedit_get_mts ()`

Reads the value of all memory types (x86 PATs / ARM MAIRs). This is equivalent to reading the MSR 0x277 (x86) / MAIR_EL1 (ARM).

Returns The memory types in the same format as in the IA32_PAT MSR / MAIR_EL1

`void ptedit_set_mts (size_t mts)`

Programs the value of all memory types (x86 PATs / ARM MAIRs). This is equivalent to writing to the MSR 0x277 (x86) / MAIR_EL1 (ARM) on all CPUs.

Parameters

- `mts` The memory types in the same format as in the IA32_PAT MSR / MAIR_EL1

`char ptedit_get_mt (unsigned char mt)`

Reads the value of a specific memory type attribute (PAT/MAIR).

Parameters

- `mt` The PAT/MAIR ID (from 0 to 7)

Returns The PAT/MAIR value (can be one of PTEDIT_MT_*)

`void ptedit_set_mt (unsigned char mt,unsigned char value)`

Programs the value of a specific memory type attribute (PAT/MAIR).

Parameters

- `mt` The PAT/MAIR ID (from 0 to 7)
- `value` The PAT/MAIR value (can be one of PTEDIT_MT_*)

`unsigned char ptedit_find_mt (unsigned char type)`

Generates a bitmask of all memory type attributes (PAT/MAIR) which are programmed to the given value.

Parameters

- `type` A memory type, i.e., PAT/MAIR value (one of PTEDIT_MT_*)

Returns A bitmask where a set bit indicates that the corresponding PAT/MAIR has the given type

`int ptedit_find_first_mt (unsigned char type)`

Returns the first memory type attribute (PAT/MAIR) which is programmed to the given memory type.

Parameters

- `type` A memory type, i.e., PAT/MAIR value (one of PTEDIT_MT_*)

Returns A PAT/MAIR ID, or -1 if no PAT/MAIR of this type was found

`size_t ptedit_apply_mt (size_t entry,unsigned char mt)`

Returns a new page-table entry which uses the given memory type (PAT/MAIR).

Parameters

- `entry` A page-table entry
- `mt` A PAT/MAIR ID (between 0 and 7)

Returns A new page-table entry with the given memory type (PAT/MAIR)

`unsigned char ptedit_extract_mt (size_t entry)`

Returns the memory type (i.e., PAT/MAIR ID) which is used by a page-table entry.

Parameters

- `entry` A page-table entry

Returns A PAT/MAIR ID (between 0 and 7)

`const char * ptedit_mt_to_string (unsigned char mt)`

Returns a human-readable representation of a memory type (PAT/MAIR value).

Parameters

- `mt` A memory type (PAT/MAIR value, e.g., one of PTEDIT_MT_*)

Returns A human-readable representation of the memory type

Pretty print

`void ptedit_print_entry (size_t entry)`

Pretty prints a page-table entry.

Parameters

- `entry` A page-table entry

`void ptedit_print_entry_line (size_t entry,int line)`

Prints a single line of the pretty-print representation of a page-table entry.

Parameters

- `entry` A page-table entry
- `line` The line to print (0 to 3)