NAME

Protection Key Rights for User pages - provide fast user-managed key-based access control for pages

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <machine/sysarch.h>

int
x86_pkru_get_perm(*unsigned int keyidx, int *access, int *modify*);

int
x86_pkru_set_perm(*unsigned int keyidx, int access, int modify*);

int
x86_pkru_protect_range(*void *addr, unsigned long len, unsigned int keyidx, int flag*);

int
x86_pkru_unprotect_range(*void *addr, unsigned long len*);

DESCRIPTION

The protection keys feature provides an additional mechanism, besides the normal page permissions as established by mmap(2) and mprotect(2), to control access to user-mode addresses. The mechanism gives safety measures which can be used to avoid incidental read or modification of sensitive memory, or as a debugging feature. It cannot guard against conscious accesses since permissions are user-controllable.

If supported by hardware, each mapped user linear address has an associated 4-bit protection key. A new per-thread PKRU hardware register determines, for each protection key, whether user-mode addresses with that protection key may be read or written.

Only one key may apply to a given range at a time. The default protection key index is zero, it is used even if no key was explicitly assigned to the address, or if the key was removed.

The protection prevents the system from accessing user addresses as well as the user applications. When a system call was unable to read or write user memory due to key protection, it returns the EFAULT error code. Note that some side effects may have occurred if this error is reported.

Protection keys require that the system uses 4-level paging (also called long mode), which means that it

is only available on amd64 system. Both 64-bit and 32-bit applications can use protection keys. More information about the hardware feature is provided in the IA32 Software Developer's Manual published by Intel Corp.

The key indexes written into the page table entries are managed by the **sysarch**() syscall. Per-key permissions are managed using the user-mode instructions *RDPKRU* and *WRPKRU*. The system provides convenient library helpers for both the syscall and the instructions, described below.

The **x86_pkru_protect_range**() function assigns key *keyidx* to the range starting at *addr* and having length *len*. Starting address is truncated to the page start, and the end is rounded up to the end of the page. After a successfull call, the range has the specified key assigned, even if the key is zero and it did not change the page table entries.

The *flags* argument takes the logical OR of the following values:

[*AMD64_PKRU_EXCL*]
Only assign the key if the range does not have any other keys assigned (including the zero key). You must first remove any existing key with **x86_pkru_unprotect_range**() in order for this request to succeed. If the *AMD64_PKRU_EXCL* flag is not specified, **x86_pkru_protect_range**() replaces any existing key.

[*AMD64_PKRU_PERSIST*]
The keys assigned to the range are persistent. They are re-established when the current mapping is destroyed and a new mapping is created in any sub-range of the specified range. You must use a **x86_pkru_unprotect_range**() call to forget the key.

The **x86_pkru_unprotect_range**() function removes any keys assigned to the specified range. Existing mappings are changed to use key index zero in page table entries. Keys are no longer considered installed for all mappings in the range, for the purposes of **x86_pkru_protect_range**() with the *AMD64_PKRU_EXCL* flag.

The **x86_pkru_get_perm**() function returns access rights for the key specified by the *keyidx* argument. If the value pointed to by *access* is zero after the call, no read or write permissions is granted for mappings which are assigned the key *keyidx*. If *access* is not zero, read access is permitted. The non-zero value of the variable pointed to by the *modify* argument indicates that write access is permitted.

Conversely, the **x86_pkru_set_perm**() establishes the access and modify permissions for the given key index as specified by its arguments.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

[EOPNOTSUPP] The hardware does not support protection keys.

[EINVAL] The supplied key index is invalid (greater than 15).

[EINVAL] The supplied *flags* argument for **x86_pkru_protect_range**() has reserved bits set.

[EFAULT] The supplied address range does not completely fit into the user-managed address range.

[ENOMEM] The memory shortage prevents the completion of the operation.

[EBUSY] The *AMD64_PKRU_EXCL* flag was specified for **x86_pkru_protect_range**() and the range already has defined protection keys.

SEE ALSO

mmap(2), mprotect(2), munmap(2), sysarch(2).

STANDARDS

The **Protection Key Rights for User pages** functions are non-standard and first appeared in FreeBSD 13.0.