Intel® SGX support for FreeBSD

Ruslan Bukin

University of Cambridge Computer Laboratory

BSDCam ☺ 2017

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

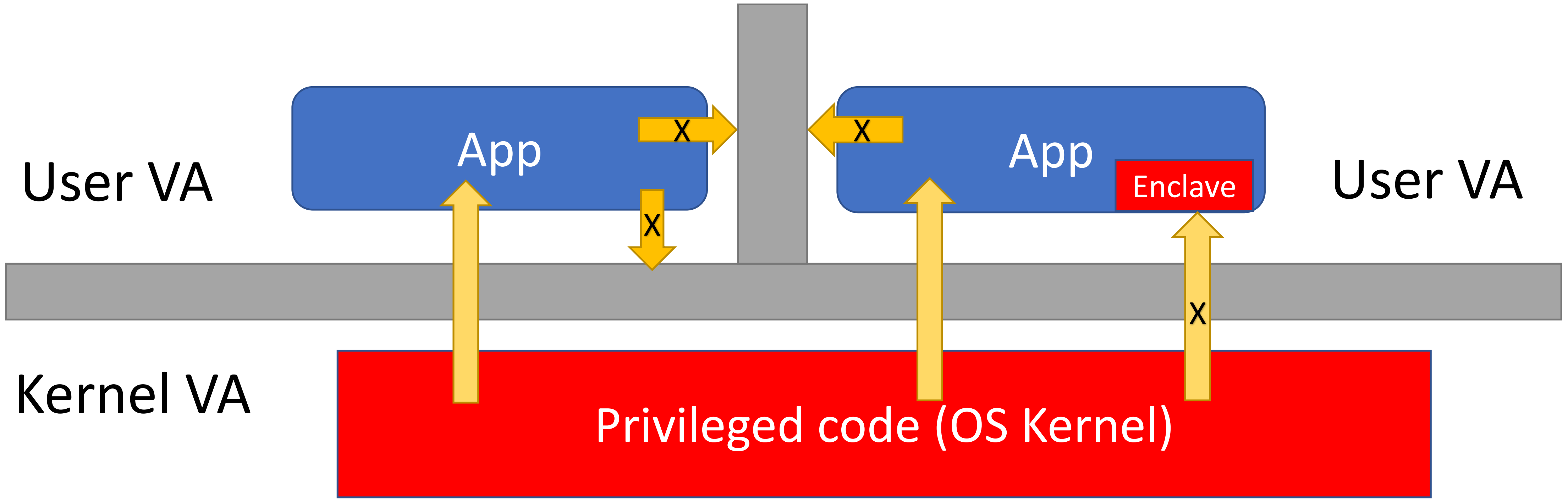
Intel® SGX

- Extension to x86
- Introduced in SkyLake
  - kvm-sgx, qemu-sgx
- Allows creation of “Enclaves”

Enclave

- Isolated compartments
- Enclave memory encrypted
- Enclaves are part of applications
    - But even less privileged
- No system calls allowed
- No direct calls between enclave and non-enclave memory
- No kernel enclaves, only userspace
- Statically linked, self-contained
    - -nostdlib -nostdinc -nodefaultlibs -nostartfiles

Traditional UNIX & Enclaves



Use cases

- When you want to securely compute data on remote machine
  - E.g. on more powerful machine
- When someone want to compute data on your machine
  - Remote guy does not trust me

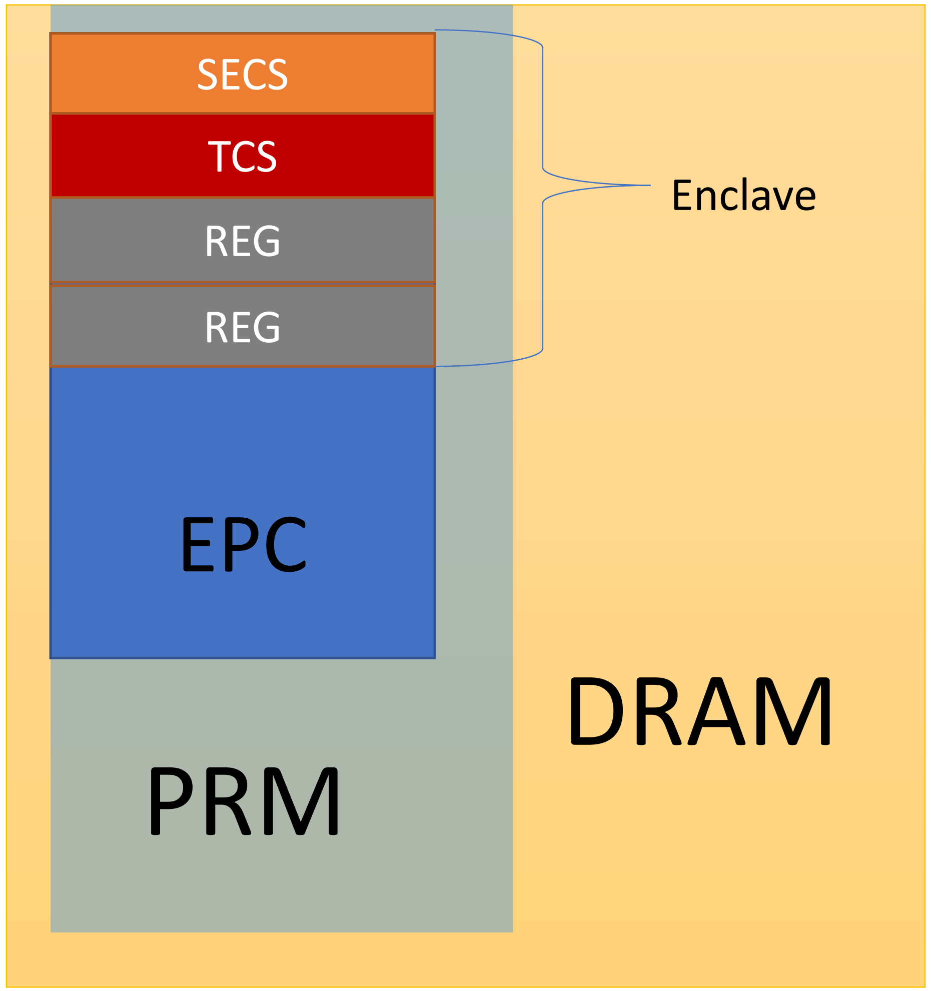
- Pros:
  - Computation without data disclosure
  - Your provider does not see what you do

Enclave Page Cache (EPC)

- Memory for enclaves (part of DRAM)
- Part of Processor Reserved Memory (RPM)
- Protected from DMA
- CPU provides 128 MB
  - 93.5mb on Lenovo X1 Carbon
  - 32mb on QEMU

Enclave and EPC page types (3 of 5)

- PT_SECS (SGX Enclave Control Structure)
  - One per enclave
  - VA, size
  - Inaccessible
- PT_TCS (Thread Control Storage)
  - One per enclave thread
  - Entry point, SSA
  - Inaccessible
- PT_REG (Regular data page)
  - SSA
  - STACK
  - DATA
  - CODE



EPC Pages types (5 of 5)

- PT_VA (Version array)
  - Used for eviction
  - 512 slots x 8 byte
- PT_TRIM (Page is removed)

Intel® SGX

- Kernel driver
  - encls(opcode, …)
- Userspace SDK
  - enclu(opcode, …)
  - Includes SGX service daemon (aesm_service)
  - Provides ABI: ecall(), ocall()
- No special compiler needed

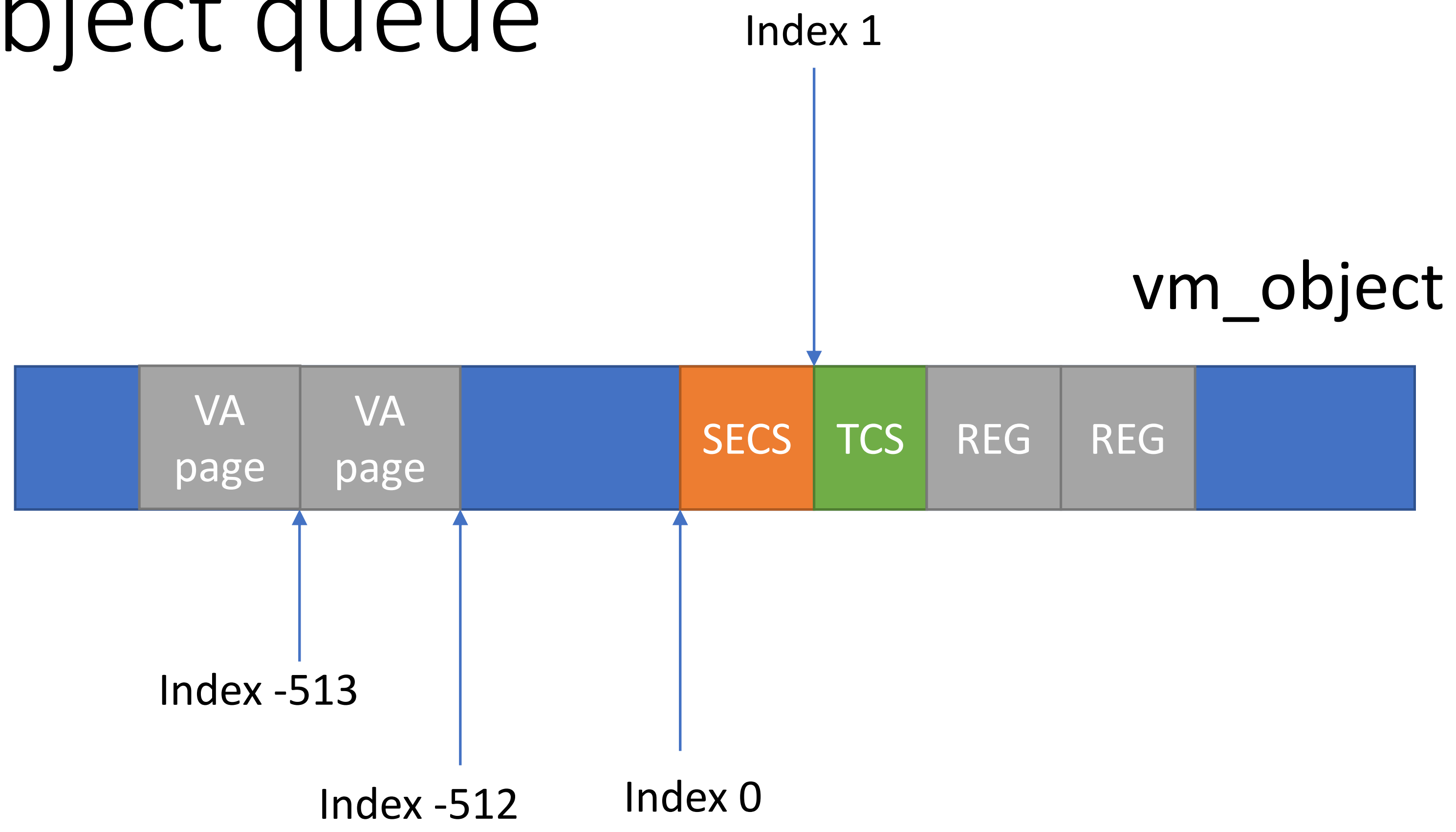
Intel® SGX userspace SDK

- A set of libraries
  - tlibc (openbsd)
  - tlibstdcxx -> tlibcxx (LLVM)
  - tlibthread
  - tseal
  - cpprt
  - sign_tool
- ABI
  - sgx_ecall(), sgx_ocall()
- SGX Service daemon
  - aesmd_service
- Sample Appications
- BSD licensed

Intel® SGX kernel driver

- EPC pages management
- Character device
- MMAP
  - VM object constructor (sgx_pg_ctor)
  - VM object destructor (sgx_pg_dtor)
  - VM object fault handler (not in use)
- IOCTLS (not SGX standard)
  - IOCTL_ENCLAVE_CREATE
  - IOCTL_ENCLAVE_ADD_PAGE
  - IOCTL_ENCLAVE_INIT
- ENCLS locking, #pg-s protection
- Linux compatibility layer

vm_object queue



Enclave lifecycle

Kernel	ECREATE	Adding SECS page
	EADD	Adding TCS, REG pages
	EEXTEND	Measuring page
	EPA	Adding Version Array (VA) slot
	EINIT	Finalize enclave creation
Userspace	EENTER	Go to entry point
	EEXIT	Return to main application
	ERESUME	Resume operation, e.g. after Interrupt

Operation flow 1: mmap

- fd = open("/dev/sgx", …);
- secs_ptr = mmap(NULL, fd, …);

Userspace

- sgx_mmap_single():

Kernel driver creates VM object

Kernel

Operation flow 2: enclave creation (SECS)

- struct sgx_secs secs {
  base = secs_ptr;
  size = 8192;
  attributes = …
  …}
- error = ioctl(fd,
  ENCLAVE_CREATE, &secs);

Userspace

- Copyin(secs)
- Validate SECS
- Lookup for VM object in process VM space
- Allocating struct enclave
- Enclave->object = object
- Get EPC from pool
- encls(EPA,…)
- vm_page_insert
- encls(ECREATE, ..)
- vm_page_insert

Kernel

Operation flow 3: adding TCS page

```
• Struct sgx_tcs tcs {
          flags = …
          oentry = offset;
          ossa = offset;
          …
}
• Struct add_page {
          src_pge = &tcs;
          secs = secs_ptr;
}
• ioctl(fd, ADD_PAGE, &add_page);
```

Userspace

- Copyin(tcs)
- Validate TCS
- Lookup for enclave
- Get EPC page from pool
- encls(EADD, ..)
- encls(EEXTEND,..)
- encls(EPA,…)
- vm_insert(page, enclave->obj)
- vm_insert(vaslot, enclave->obj)

Kernel

Operation flow 4: adding REG pages

- fd = open(“enclave1.bin”)
- ptr = mmap(NULL, fd, …)

- For each page:

```
struct sgx_addp {
            src_pge = ptr[..];
            secs = secs_ptr;
}
ioctl(fd, ADD_PAGE, &addp);
```

- munmap(fd)

Userspace

- copyin(secinfo)
- Lookup for enclave
- Allocate EPC page
- encls(EADD, ..)
- encls(EEXTEND,..)
- encls(EPA,…)
- vm_insert(epc_page, enclave->object)

Kernel

Operation flow 5: init enclave

```
• struct sgx_initp {
      sigstruct = …
      einittoken = …
      addr = secs_ptr;
}


ioctl(fd, EINIT, einitp);
```

Userspace

- Lookup for enclave
- copyin(sigstruct)
- copyin(einittoken)
- encls(EINIT, …)

Kernel

Operation flow 6: enter enclave

- enclu(EENTER,..)
- sgx_ecall()
- sgx_ocall()

Userspace

Operation flow 7: removing enclave

- enclu(EEXIT, …)
- munmap(secs_ptr, size);

Userspace

- pager_dtor():

1) Enclave lookup
2) for each page in object:
    - vm_object_remove(page)
    - encls(EREMOVE,..)
    - EPC returned to pool

Kernel

Enclave and exceptions

- Exceptions generate AEX: Asynchronous Enclave Exit

  AEP pointer is called

```
enclu(EENTER, aep, ..)

void
aep() {
    …exception handled here…
    enclu(ERESUME, ..)
}
```

Attestation

- Don’t know how exactly works
- Enclaves can prove to us that they are enclaves they clamed to be
- Each CPU has its own unique key as a root in key hierarchy, created on manufacture time
    - Enclave requests it using EGETKEY ?

Review

- D11113
- http://wiki.freebsd.org/Intel_SGX

Questions?