SGX Upstreaming Story

Linux Plumbers Conference 2019

Jarkko Sakkinen <jarkko.sakkinen@linux.intel.com>

First, a little bit of history

- Skylake 2015
- First attempt 2016/04/25:
  - Only Intel blessed enclaves :(
  - `https://lwn.net/Articles/686808/`
- At LPC 2016 first plans for flexible launch control.
- In September 2017 new series was started.
- In December Geminilake launchedx.
- `https://lwn.net/Articles/786487/`
- Latest version is v22.

Enclaves

- Reserved address space.
- Memory is committed from a reserved memory area called Enclave Page Cache (EPC).
- Predefined entry points (ring-3).
- CPU asserted access.
- Memory encryption (outside LLC).
- Local and remote attestation.

The kernel assets

- Sources
  - arch/x86/kernel/cpu/sgx
  - tools/testing/selftests/x86/sgx
- Devices
  - /dev/sgx/enclave
    - `SGX_IOC_ENCLAVE_CREATE`
    - `SGX_IOC_ENCLAVE_ADD_PAGE`
    - `SGX_IOC_ENCLAVE_INIT`
    - `SGX_IOC_ENCLAVE_SET_ATTRIBUTE`
  - /dev/sgx/provision
- Community
  - linux-sgx@vger.kernel.org
  - https://github.com/jsakkine-intel/linux-sgx.git

The kernel assets: arch/x86/kernel/cpu/sgx

```
$ wc -l arch/x86/kernel/cpu/sgx/*
  423 arch/x86/kernel/cpu/sgx/arch.h
  275 arch/x86/kernel/cpu/sgx/driver.c
   34 arch/x86/kernel/cpu/sgx/driver.h
  718 arch/x86/kernel/cpu/sgx/encl.c
  133 arch/x86/kernel/cpu/sgx/encl.h
   56 arch/x86/kernel/cpu/sgx/encls.c
  263 arch/x86/kernel/cpu/sgx/encls.h
  721 arch/x86/kernel/cpu/sgx/ioctl.c
  311 arch/x86/kernel/cpu/sgx/main.c
    5 arch/x86/kernel/cpu/sgx/Makefile
  472 arch/x86/kernel/cpu/sgx/reclaim.c
   89 arch/x86/kernel/cpu/sgx/sgx.h
 3500 total
```

The kernel assets: tools/testing/selftests/x86/sgx

```
$ wc -l tools/testing/selftests/x86/sgx/*
   39 tools/testing/selftests/x86/sgx/defines.h
   94 tools/testing/selftests/x86/sgx/encl_bootstrap.S
   20 tools/testing/selftests/x86/sgx/encl.c
   34 tools/testing/selftests/x86/sgx/encl.lds
  371 tools/testing/selftests/x86/sgx/main.c
   47 tools/testing/selftests/x86/sgx/Makefile
   49 tools/testing/selftests/x86/sgx/sgx_call.S
  493 tools/testing/selftests/x86/sgx/sgxsign.c
   39 tools/testing/selftests/x86/sgx/signing_key.pem
 1186 total
```

A short breakdown

- Constructing enclaves (/dev/sgx/enclave)
- Executing enclaves
- Overcommitment
- Access control (e.g. DAC, SELinux, AppArmor)
- Provisioning (/dev/sgx/provision)

Constructing enclaves

- /dev/sgx/enclave
- `mmap()` with `PROT_NONE`.
- `SGX_IOC_ENCLAVE_CREATE` (secs)
  - SGX Enclave Control Structure (SECS)
- `SGX_IOC_ENCLAVE_ADD_PAGE` (addr, page, secinfo, mrmask)
- `SGX_IOC_ENCLAVE_INIT` (sigstruct)
- `mprotect()` (capped by EADD)
  - `vma->may_protect()`

Constructing enclaves: ENCLS[EINIT]

- IA32_SGXLEPUBKEYHASH{0, 1, 2, 3} MSRs
- FEATURE_CONTROL_SGX_LE_WR
- Locked MSRs: requires a Launch Enclave.
  - Tokens generated by the LE and passed to EINIT.
- Linux runs enclaves only with unlocked MSRs.

Executing enclaves

- ENCLU[EENTER] (rbx=TCS, rcx=AEP/rip successor)
  - Thread Control Structure (TCS)
  - Asynchronous Exit Point (AEP)
- Exit to Asychronous Exit Point (AEP).
  - ENCLU[ERESUME] (rbx=TCS, rcx=AEP)
- ENCLU[EEXIT] (rbx=outside address, rcx=AEP)

Executing enclaves: TCS

```
.section ".tcs", "a"
.balign         4096

.fill           1, 8, 0             # STATE (set by CPU)
.fill           1, 8, 0             # FLAGS
.quad           encl_ssa            # OSSA
.fill           1, 4, 0             # CSSA (set by CPU)
.fill           1, 4, 1             # NSSA
.quad           encl_entry          # OENTRY
.fill           1, 8, 0             # AEP (set by EENTER/RESUME)
.fill           1, 8, 0             # OFSBASE
.fill           1, 8, 0             # OGSBASE
.fill           1, 4, 0xFFFFFFFF    # FSLIMIT (32-bit)
.fill           1, 4, 0xFFFFFFFF    # GSLIMIT (32-bit)
.fill           4024, 1, 0          # Reserved
```

Executing enclaves: `__vdso_sgx_enter_enclave`

- Enclaves generate exceptions as part of their normal operation.
- Permisson conflict: #PF with PF_SGX
- Illegal instructions: #UD
  - `https://software.intel.com/en-us/node/703005`
- `__vdso_sgx_enter_enclave`
  - Exception: di=exception (e.g. #PF), si=error (e.g. PF_SGX), rdx=addr

Access control: DAC

- /dev/sgx/enclave permissions control who can **build** enclaves.
  - The build process also caps mmap() and mprotect().
- /dev/sgx/provision permissions control who can grant access to provision an enclave.
- Enclaves always need an outside delegate for syscalls. They can read and write process memory but cannot affect outside system.
- The end game is that there needs to be a process that is able to change writable pages executable pages unconditionally.

Access control: LSM hooks

- `security_enclave_load(vma, prot)`: Allow LSM intervene when a a page is loaded into enclave.
  - Prevent loading a non-executable file.
  - Deny WX from unprivileged process (as defined by the LSM).
- `security_enclave_map(vma, prot)`: Allow LSM intervene mmap() or mprotect() of an enclave.
  - Deny WX.

That's all folks, thank you.