March 13th, 2024

Open Confidential Computing Conference

Extending Integrity Measurement Architecture for Attestation

Wenhui Zhang, Ken Lu, May Xie, Ruomeng Hao, Xiaocheng Dong, Ruoyu Ying, Guixiong Wei, Yuan Zhu, Zhu Song, Duan Bing, Ziye Yang, Hou Yu, Liguang Xie, Jianqing Zhang, Qirui Yang, Henry Hu, Zefan Li, Jianjun Chen, Trent Jaeger

Contents

Problem Statement:
Boot Sequence of Cloud Native Workloads

BIOS: Static Root of Trust
-> GRUB: Static Root of Trust
-> tboot (trusted boot): load tdx-module, build time measurement register (MRTD)
-> kernel: Dynamic Root of Trust
-> initrd: full disk encryption
-> init (including ACPI subsystem)
-> daemons
-> cloud native daemons
-> containers
-> sidecars and applications
-> handles workloads and requests

Problem Statement:
Attestation for TEE – TDX 1.0

Architecture Support

```
// #define TDX_GET_INFO          1 // TDCALL for Guest prepare
// #define TDX_GET_REPORT        4  // TDCALL, in Guest get integrity measure
#define TDG_VP_VMCALL                    0
#define TDG_VP_INFO                      1
#define TDG_MR_REPORT                    4
#define TDVMCALL_GET_QUOTE               0x10002
```

Table 2.9: Guest-Side (TDCALL Leaf) Interface Functions

Interface Function Name	Leaf Number	Description
TDG.MEM.PAGE.ACCEPT	6	Accept a pending private page into the TD
TDG.MR.REPORT	4	Creates a cryptographic report of the TD
TDG.MR.RTMR.EXTEND	2	Extend a TD run-time measurement register.
TDG.VP.CPUIDVE.SET	5	Control delivery of #VE on CPUID instruction execution
TDG.VP.INFO	1	Get TD execution environment information
TDG.VM.RD	7	Read a TD-scope metadata field
TDG.VM.WR	8	Write a TD-scope metadata field
TDG.VP.VEINFO.GET	3	Get Virtualization Exception Information for the recent #VE exception
TDG.VP.VMCALL	0	Call a host VM service

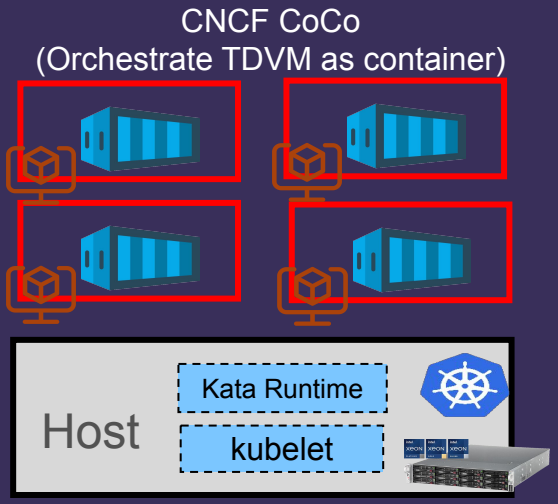
Kernel Support

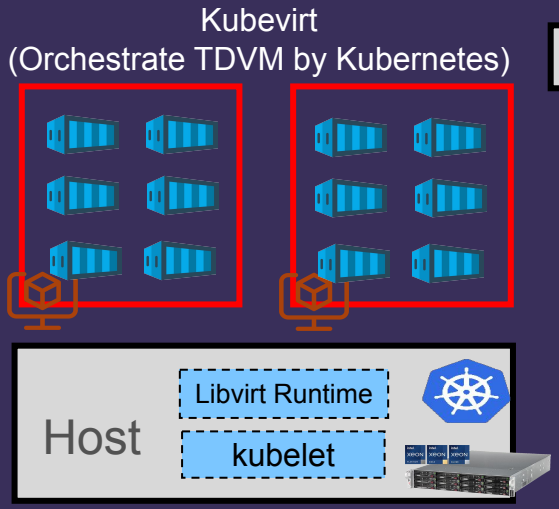
```
// u64 __tdx_module_call(u64 fn, u64 rcx, u64 rdx, u64 r8, u64 r9, struct tdx_module_output *out); // TDCALL 5.15
u64 __tdcall(u64 fn, struct tdx_module_args *args); // upstream
u64 __tdx_hypercall(struct tdx_module_args *args); // upstream

/* ./arch/x86/coco/tdx/tdx.c */
int tdx_mcall_get_report0(u8 *reportdata, u8 *tdreport); // through module call, TDCALL
int tdx_hcall_get_quote(void *tdquote, int size); // through hypercall, TDCALL
```

Problem Statement:

Limitations in Cloud Native Scenarios





Bare Metal

Confidential VM

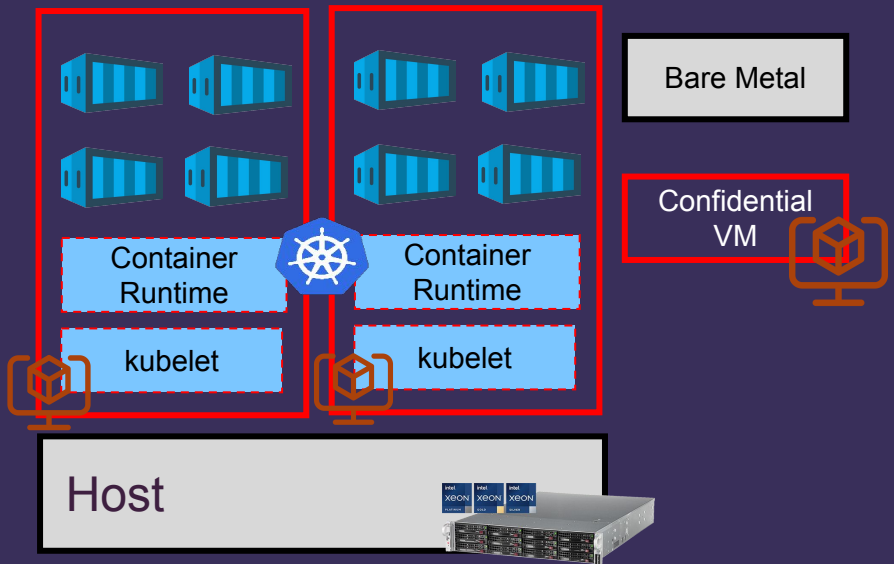
Only Supports per Secure Container level or VM level Attestation

- I/O Bottleneck
- Unable to scale up, no HPA supported (unable to snapshot)
- Unable to scale out in large scale

Goal:

Attestation for Cloud Native Deployment

Confidential Cluster (Deploy K8S cluster in TDVM)



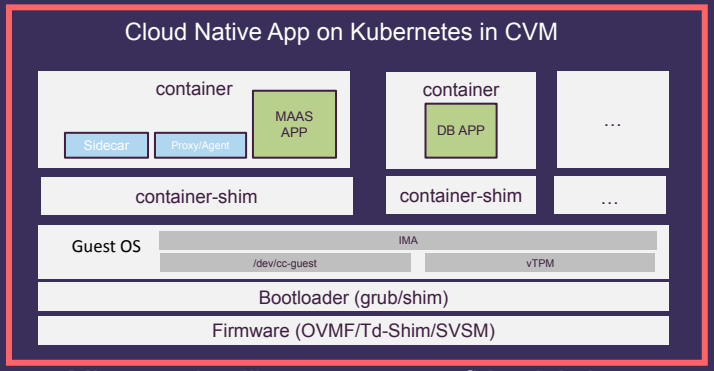
- Supports several containers in one CVM, no I/O Bottleneck between containers in one CVM
- Supports attested cluster of containers
- Supports Dynamic Resource Allocation (DRA)
  - scale up (HPA) through snapshot
  - scale out (batching) through warm pool and ps-tree fork

Goal:

Attestation for Cloud Native Deployment

Confidential **Database** Service and Confidential **Model-as-a-Service** deployments commonly encompass multiple containers distributed across several virtual machines (VMs).

M (workload) = M (vm/secure container) ||
M (trusted domain) ||
M (trusted container launcher) ||
M (application)
\



Microservice-like app on a confidential cluster

This deployment model necessitate attestation for userspace code:

- **container launch time** integrity measurement and attestation
- **application initialization time** integrity measurement and attestation
- attestation for **Trusted Computing Base(TCB)** only

Key Idea:

Extending IMA for Workload Attestation

→ CC Trusted API & CCNP

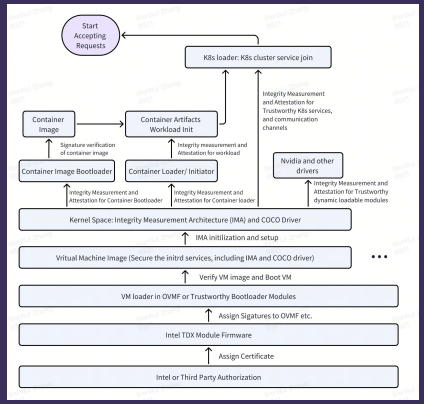
- RTMR Backend
- Digest list
- Trusted Domain
- Trusted Container Launcher

Extending Integrity Measurement Architecture for Workload Attestation

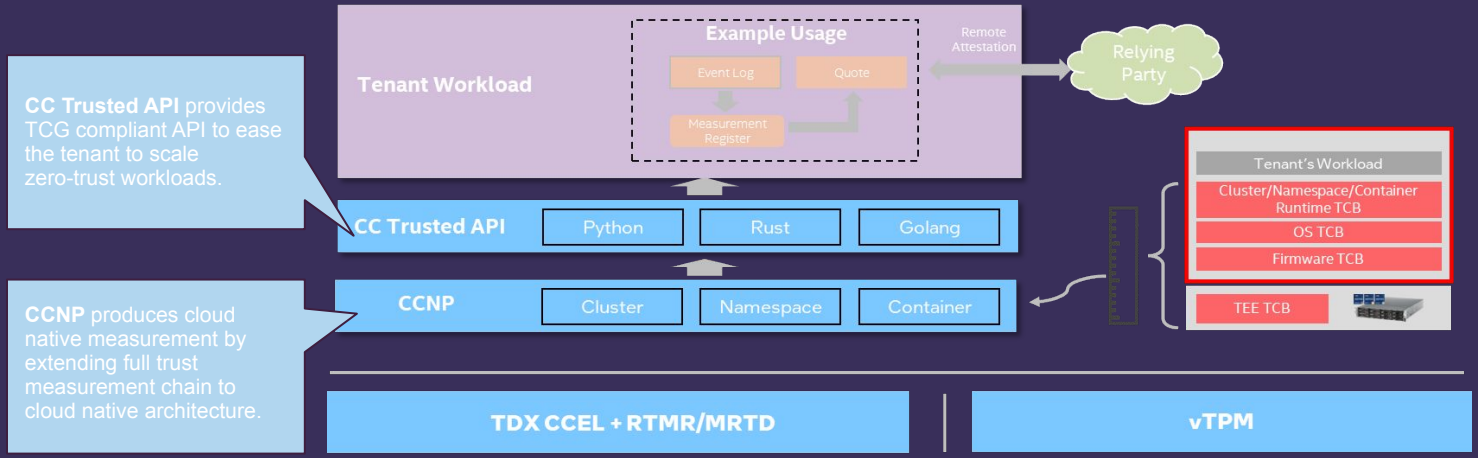
M (workload) = M (vm/secure container) ||**M (trusted domain)** || **M (trusted container launcher)** || M (application)

This work focuses on enabling measurement for:

- trusted domain (e.g., nvidia.ko, k8s daemons, container runtimes, ccnp toolkit, trusted launcher etc..),



Overview:
CC Trusted API & CCNP



CC Truste API: https://github.com/cc-api/cc-trusted-api
CCNP: https://github.com/cc-api/confidential-cloud-native-primitives

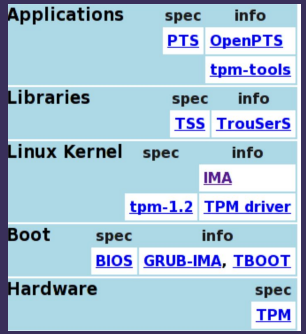
Background:
Integrity Measurement Architecture

- Measure all files before they are used according to the measurement policy
- Store measurements in kernel list, a.k.a., IMA measurement list
- Extend measurements into **TPM/vTPM** PCR. configurable, 10 for application
- TPM signed attestation report of measurement list, through TPM_QUOTE

IMA Raw output

```
10 1609097048173d7c1659ff33713741b98020d0d8 ima a36e075c9dc23bd71886d55e0daee556143f8e7e boot_aggregate
10 3d82394c528268a62687c8b59b34c39137b4fb3c ima ff65625e34131617ef3ac5fead5cb285b6aa73b9 /init
10 231309f206f12296cff6c59c6e3bceb45dc285bb ima 602acdac8864b113d4546bf8f2d7b96c81607792 /init
10 9d20d222ae9a3a9c80b2a9f0b3c08a5786847ed6 ima ff4c31959c04f865c859d8df331a6bf6b6967cef ld-2.10.1.so
10 88a2bee776f4bd1305ce6ded08611fdecb5bf0db ima 44e727b6c99370d373d0acef95631e0950ef8c00 ld.so.cache
10 0c86c858b30abf3f2c3b0269a56204f642f3ef71 ima a0ed012c34fee03e9fecee8d4a7bcab1ff98f091 libnash.so.6.0.87
10 676a27e4112c9fb5677b7ef38cc488d87fca0846 ima bf094171a04b06d642c931d7c2ce0a707a659827 libbdevid.so.6.0.87
10 04c918d9ccb56ca92a118b72f8b2f60da0791887 ima 9355215ed19d72287c446d0f3b7b41dcaddbd254 libdevmapper.so.1.02
10 6585544486aee180aa950b1bb6132434cf0f71d6 ima caca8bbf8ff4957584d114c753b7c4f15936af7c libparted-1.8.so.8.0.0
```

After typical boot (Fedora 11), 1600 measurements.
Overhead at boot time, 10% (3-5 seconds).
Slight performance improvement at runtime (a lot prefetched).



Pictures credited to David Safford, Mimi Zohar, Reiner Sailer, https://blog.linuxplumbersconf.org/2009/slides/David-Stafford-IMA_LPC.pdf

Background:
Integrity Measurement Architecture

```
#define __ima_hooks(hook)                                   \
        hook(NONE, none)                                    \
        hook(FILE_CHECK, file)                              \
        hook(MMAP_CHECK, mmap)                              \
        hook(MMAP_CHECK_REQPROT, mmap_reqprot)              \
        hook(BPRM_CHECK, bprm)                              \
        hook(CREDS_CHECK, creds)                            \
        hook(POST_SETATTR, post_setattr)                    \
        hook(MODULE_CHECK, module)                          \
        hook(FIRMWARE_CHECK, firmware)                      \
        hook(KEXEC_KERNEL_CHECK, kexec_kernel)              \
        hook(KEXEC_INITRAMFS_CHECK, kexec_initramfs)        \
        hook(POLICY_CHECK, policy)                          \
        hook(KEXEC_CMDLINE, kexec_cmdline)                  \
        hook(KEY_CHECK, key)                                \
        hook(CRITICAL_DATA, critical_data)                  \
        hook(SETXATTR_CHECK, setxattr_check)                \
        hook(MAX_CHECK, none)
```

```
static const char *const
mask_tokens[] = {
        "^MAY_EXEC",
        "^MAY_WRITE",
        "^MAY_READ",
        "^MAY_APPEND"
  };
```

Source: https://elixir.bootlin.com/linux/v6.8-rc7/source/security/integrity/ima/ima.h

Contribution 1:
RTMR Backend

Architecture Support

u64 __tdx_module_call(u64 fn, u64 rcx, u64 rdx, u64 r8, u64 r9, struct tdx_module_output *out); // TDCALL

```
#define TDX_GET_INFO          1 // TDCALL for Guest prepare
#define TDX_EXTEND_RTMR       2  // TDCALL , in Guest invoke integrity measure
#define TDX_GET_REPORT        4  // TDCALL, in Guest get integrity measure
#define TDX_VERIFYREPORT      22 // TDCALL 1.5 only, in Guest-driver attestation
```

Table 2.9: Guest-Side (TDCALL Leaf) Interface Functions

Interface Function Name	Leaf Number	Description
TDG.MEM.PAGE.ACCEPT	6	Accept a pending private page into the TD
TDG.MR.REPORT	4	Creates a cryptographic report of the TD
TDG.MR.RTMR.EXTEND	2	Extend a TD run-time measurement register
TDG.VP.CPUIDVE.SET	5	Control delivery of #VE on CPUID instruction execution
TDG.VP.INFO	1	Get TD execution environment information
TDG.VM.RD	7	Read a TD-scope metadata field
TDG.VM.WR	8	Write a TD-scope metadata field
TDG.VP.VEINFO.GET	3	Get Virtualization Exception Information for the recent #VE exception
TDG.VP.VMCALL	0	Call a host VM service

Kernel and TDX Module Support

/* ./arch/x86/coco/tdx/tdx.c */

```
int tdx_mcall_extend_rtmr(u8 *data, u8 index);
int tdx_mcall_get_report0(u8 *reportdata, u8 *tdreport);
u64 tdx_mcall_verify_report(u8 *reportmac);
int tdx_hcall_get_quote(void *tdquote, int size);
```

Contribution 1:
RTMR Backend

Kernel Support to vTPM Driver

```
/* ./arch/x86/include/asm/tdx-rtmr.h */
struct tpm_chip *tdx_rtmr_device(void); // a fake TPM device for IMA usage,TPM_ALG_SHA384
int ima_extend_rtmr(struct tpm_chip *chip, u32 rtmr_idx, struct tpm_digest *digests);
int tdx_get_boot_measurements(struct tdx_boot_digests *boot_digests);
/* ./arch/x86/include/asm/tdx.h */
void ccel_record_eventlog(void *data, u8 index);
```

Contribution 1:
RTMR Backend

Kernel Support to vTPM Driver

```
/* ./arch/x86/include/asm/tdx-rtmr.h */
struct tpm_chip *tdx_rtmr_device(void); // a fake TPM device for IMA usage,TPM_ALG_SHA384
int ima_extend_rtmr(struct tpm_chip *chip, u32 rtmr_idx, struct tpm_digest *digests);
int tdx_get_boot_measurements(struct tdx_boot_digests *boot_digests);
/* ./arch/x86/include/asm/tdx.h */
void ccel_record_eventlog(void *data, u8 index);
```

Tool and SDK Support

```
ACPI_TABLE_FILE = "/sys/firmware/acpi/tables/CCEL"
ACPI_TABLE_DATA_FILE = "/sys/firmware/acpi/tables/data/CCEL"
IMA_DATA_FILE = "/sys/kernel/security/integrity/ima/ascii_runtime_measurements"
# The name of the device in different kernel version
DEVICE_NODE_NAME_DEPRECATED = "/dev/tdx-attest"    # deprecated
DEVICE_NODE_NAME_1_0 = "/dev/tdx-guest"
DEVICE_NODE_NAME_1_5 = "/dev/tdx_guest"
```

To be replaced by *Trusted Security Module (TSM)* Interface : https://elixir.bootlin.com/linux/v6.8-rc2/source/drivers/virt/coco/tsm.c

Contribution 1:
RTMR Backend

- td_infor : rtmr
- tdreport: quoted event log

```
/**
 * struct td_info - TDX guest measurements and configuration.
 * @attr: TDX Guest attributes (like debug, spet_disable, etc).
 * @xfam: Extended features allowed mask.
 * @mrtd: Build time measurement register.
 * @mrconfigid: Software-defined ID for non-owner-defined configuration
 *              of the guest - e.g., run-time or OS configuration.
 * @mrowner: Software-defined ID for the guest owner.
 * @mrownerconfig: Software-defined ID for owner-defined configuration of
 *                 the guest - e.g., specific to the workload.
 * @rtmr: Run time measurement registers.
 * @reserved: Added for future extension.
 *
 * It contains the measurements and initial configuration of the TDX guest
 * that was locked at initialization and a set of measurement registers
 * that are run-time extendable. More details can be found in TDX v1.0
 * Module specification, sec titled "TDINFO_STRUCT".
 */
struct td_info {
    __u8 attr[8];
    __u64 xfam;
    __u64 mrtd[6];
    __u64 mrco  id[6];
    __u64 mrown  [6];
    __u64 mrownerconfig[6];
    __u64 rtmr[24];
    __u64 servtd_hash[6];
    __u64 reserved[8];
};
```

```
/*
 * struct tdreport - Output of TDCALL[TDG.MR.REPORT].
 * @reportmac: Mac protected header of size 256 bytes.
 * @tee_tcb_info: Additional attestable elements in the TCB are not
 *                reflected in the reportmac.
 * @reserved: Added for future extension.
 * @tdinfo: Measurements and configuration data of size 512 bytes.
 *
 * More details can be found in TDX v1.0 Module specification, sec
 * titled "TDREPORT_STRUCT".
 */
struct tdreport {
    struct reportmac reportmac;
    __u8 tee_tcb_info[239];
    __u8 reserved[17];
    struct td_info tdinfo;
};
```

```
/**
 * struct reportmac - TDX guest report data, MAC and TEE hashes.
 * @type: TDREPORT type header.
 * @reserv  eserved for future extension.
 * @cpu_svn: CPU security version.
 * @tee_tcb_info_hash: SHA384 hash of TEE TCB INFO.
 * @tee_td_info_hash: SHA384 hash of TDINFO_STRUCT.
 * @reportdata: User defined unique data passed in TDG.MR.REPORT request.
 * @reserved2: Reserved for future extension.
 * @mac: CPU MAC ID.
 *
 * It is MAC-protected and contains hashes of the remainder of the
 * report structure along with user provided report data. More details can
 * be found in TDX v1.0 Module specification, sec titled "REPORTMACSTRUCT"
 */
struct reportmac {
    struct tdreport_type type;
    __u8 reserved1[12];
    __u8 cpu_svn[16];
    __u8 tee_tcb_info_hash[48];
    __u8 tee_td_info_hash[48];
    __u8 reportdata[64];
    __u8 reserved2[32];
    __u8 mac[32];
};
```

```
struct tpm_digest {
    u16 alg_id;
    u8 digest[TPM_MAX_DIGEST_SIZE];
} __packed;

struct tpm_bank_info {
    u16 alg_id;
    u16 digest_size;
    u16 crypto_id;
};
```

Contribution 2:
Digest List

IMA hooks: ima_bprm_check, ima_file_mmap, and ima_file_check

Policy Location: /sys/kernel/security/ima/digestlist_policy
Measurement Location: /sys/kernel/security/ima/digestlist_measurements

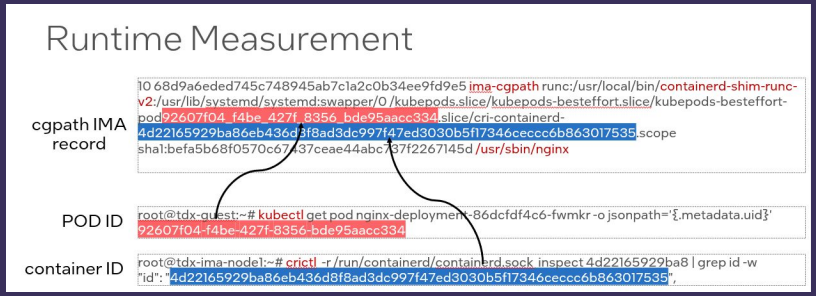
digestlist_policy format:
bprm /XXX/XXX/nvidia.ko
exe_file /XXX/XXX/file.file
config_file /XXX/XXX/file.file

digestlist_measurement format:
start with **RTMR[0]||RTMR[1],** extend to **PCR[8-15] map to RTMR[2]**
follow ima measurement list format
convert to canonical eventlog (in progress)
add IMA log to CCNP eventlog (in progress)

Contribution 2:
Digest List

two attestation modes supports:

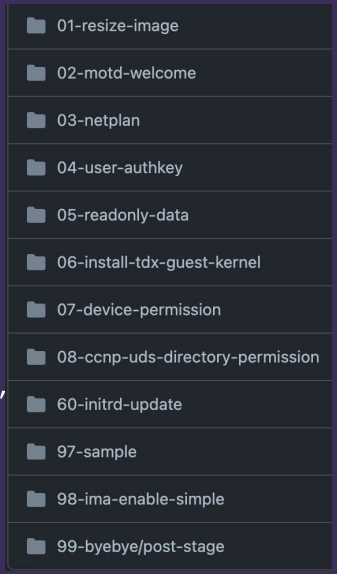
- **Local Attestation:**
  - Enabling IMA update and enabling i_version mount, Registering files hashes in i_version field as golden value for attestation for trusted domain components, distinguished by PODID-containerID
  - Code: Preregister golden values for trusted domain through cvm-rewriter https://github.com/cc-api/cc-trusted-vmsdk/tree/main/src/cvm-image-rewriter/plugins/98-ima-enable-simple
- **Remote Attestation: (in progress)**
  - Instead of Pre-load the golden values in the VM image, allow register golden value of trusted domain to remote attestor, and get attested there.



Contribution 3:
Trusted Domain

(defined in https://github.com/cc-api/cc-trusted-vmsdk)

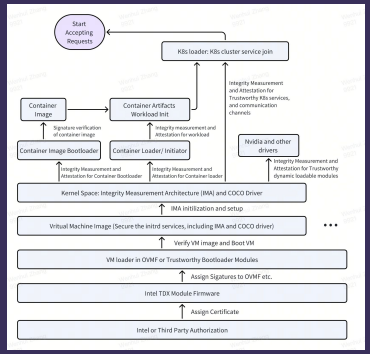
1. **Dynamic loaded modules**
   - Binary: `nvidia.ko`
2. **Container Runtimes:** containerd, containerd-shim and runc
   - Binary: `containerd`, `containerd-shim-runc-v2`, `runc`
   - Configuration File: `/etc/containerd/config.toml`
3. **K8S Management Plane:** kubelet
   - Binary: kubelet
   - Configuration File: /var/lib/kubelet/config.yaml and /etc/kubernetes/kubelet.conf
4. **CCNP services**
   - Binary: `CCNP userspace attestation tool, Remote Attestation Agent, KBS`
   - Code: https://github.com/cc-api/cc-trusted-api, https://github.com/cc-api/cc-trusted-vmsdk https://github.com/cc-api/trustauthority-kbs
5. **Trusted Container Launcher (in progress)**
   - Code: https://github.com/cc-api/cc-trusted-launcher
6. **Others:** `net`, `motd`, `passwd etc.`



Contribution 4:

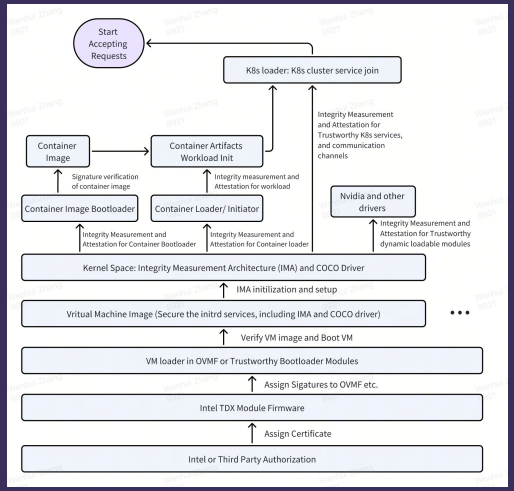
Trusted Container Launcher (In Progress)

```
Work as a daemonset: measure daemon and quote daemon
Start with RTMR[0]||RTMR[1]||RTMR[2]
Measurement:
 Container image loader (kubelet)
 Container launcher (containerd)
Attestation:
 Golden value of container images (artifacts) follows SPIFEE/SVID
 Containerd cmd with container name (PODID-containerID mapped to SVID)
Labels:
 Using containerID for container
 Using PODID for PODs
 Using groupLabels field (.group) for cluster
```

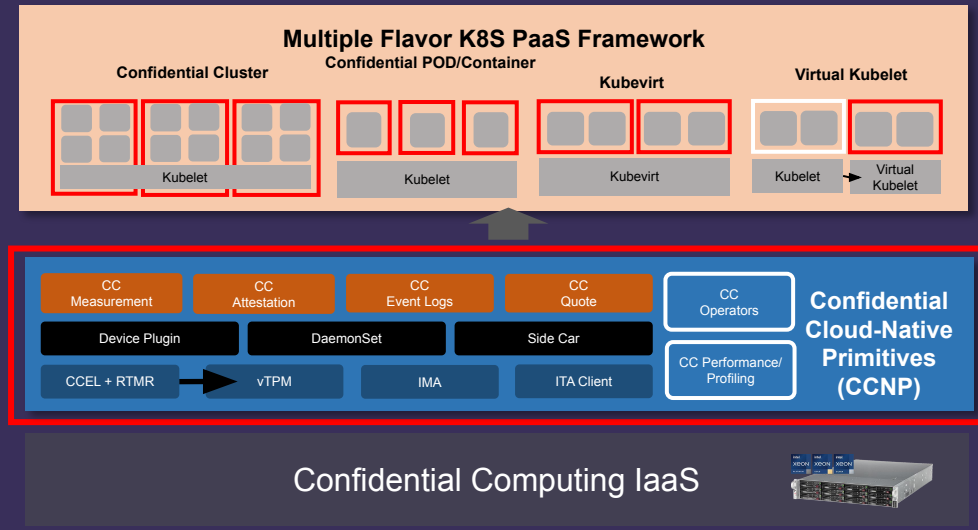


Summary

Code: https://github.com/cc-api
To be production ready for LLM and DB workloads by 06/24, upstream push by 12/24



Trust Chain



Thanks!

wenhuizhang.psu@gmail.com