蓬莱TEE：RISC-V openEuler下的安全底座

上海交通大学 ·IPADS实验室

冯二虎

RISC-V开源指令集架构

- Free and Open
  - Anyone can use
  - More competition ⇒ More innovation
  - Pick ISA, then vendor

- For Cloud & Edge
  - From large to tiny computers
- Secure/Trustworthy
  - Design own secure core
  - Open cores ⇒ no secrets

- Simple, Elegant
  - 25 years later, learn from 1st gen RISCs*
  - Far simpler than ARM and x86
  - **Can add custom instructions**
  - **Input from software/architecture experts BEFORE finalize ISA**

- Community designed
  - RISC-V Foundation owns RISC-V ISA

*: A New Golden Age for Computer Architecture: History, Challenges, and Opportunities, David Patterson, 2018

RISC-V带来的新的机会: 开源开放的芯片设计

- **硬件: 国内/国外 企业/高校**
  - 平头哥、海思、芯来、starfive等RISC-V硬件厂商
  - “香山”中科院开源RISC-V芯片
- **软件: 开源软件适配**
  - 开源OS: openEuler(官方支持的镜像)，OpenHarmony等
- **安全能力的自定义**
  - 云计算、IoT、加速器等场景需要各自的安全能力
  - 隐私数据、AI模型等需要保护

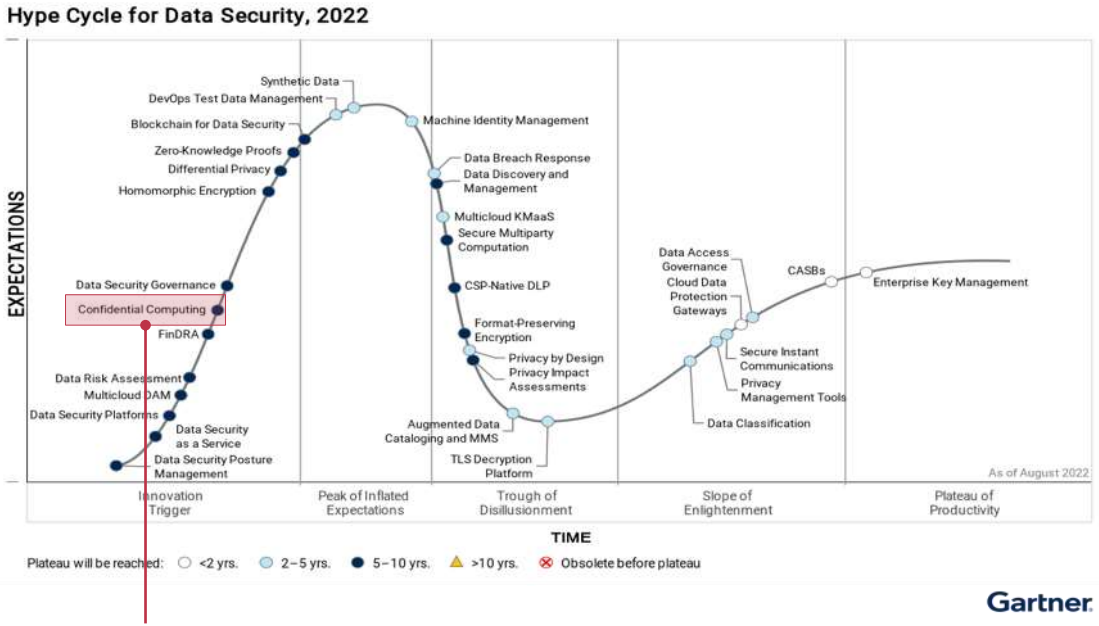
机密计算

Intel

微软

IBM

机密计算是实现 “数据可用不可见”的重要技术手段



Confidential Computing （机密计算）连续三年上榜

机密计算载体：可信执行环境（TEE）

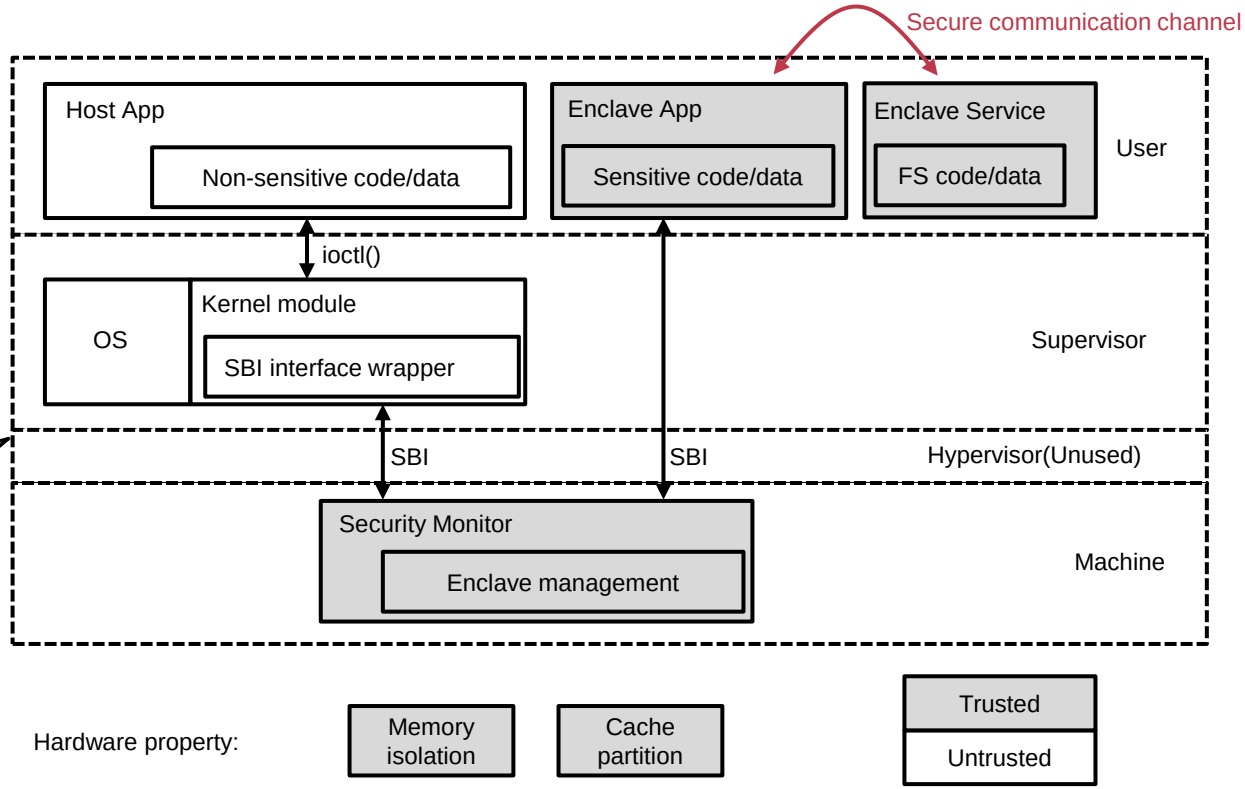
- **TEE的定义**
  - "可信执行环境" （TEE，Trusted Execution Environment），是计算机系统中一块通过底层软硬件构造的安全区域，通过保证加载到该区域的代码和数据的完整性和隐私性，实现对代码执行与数据资产的保护 —— *Wikipedia*
- **TEE的两个主要功能**
  - 远程认证：验证远程节点是否为加载了合法代码的Enclave
  - 隔离运行：TEE外无法访问TEE内部的数据
- **TEE带来的能力：限制访问数据的软件**
  - 保证数据只在提前被认证的合法节点间流动
    - 合法节点：部署了合法软件的节点

如果数据是石油，
TEE就是管道，
保证数据只通向合法的加工厂

蓬莱TEE：RISC-V架构下的开源可信执行环境

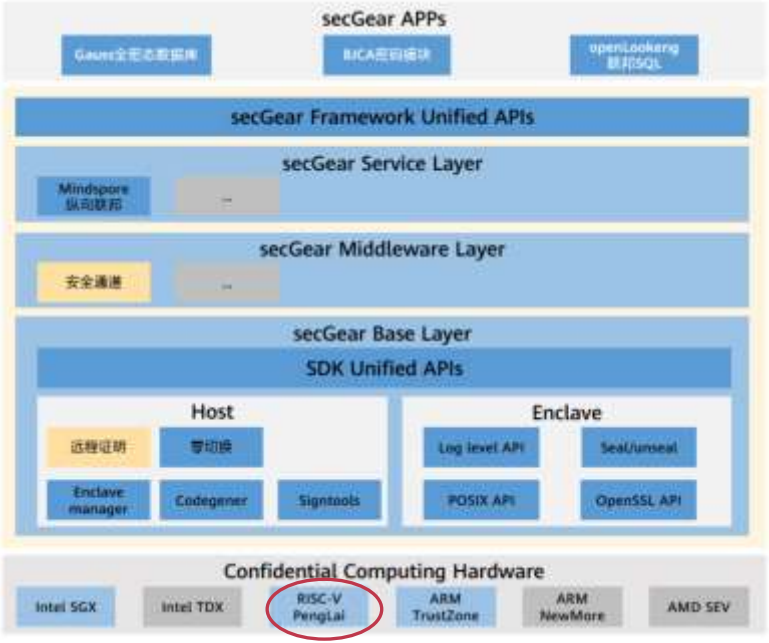
1. **底层固件（可信）**：运行在最高特权级，配合操作系统内核
2. **用户态SDK**：应用可直接调用，创建可信执行环境
3. **硬件扩展（可选）**：对性能和可扩展性取得进一步的提升

蓬莱TEE保证了，哪怕操作系统（openEuler）被攻破。也能保证应用数据与代码的安全



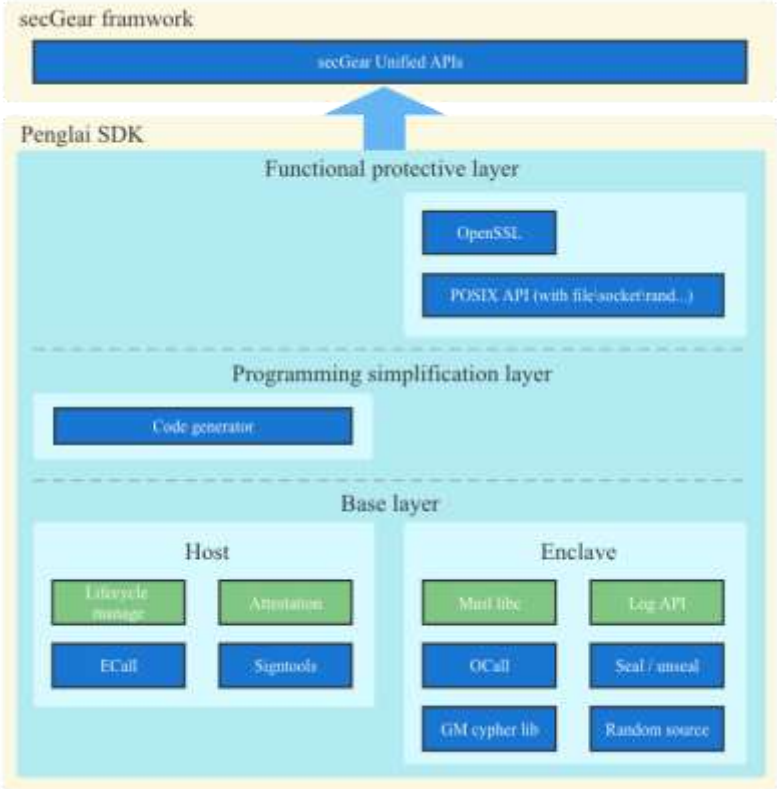
Part1: 蓬莱TEE+openEuler secGear

- **跨平台统一TEE编程抽象**
  - 一套TEE代码，多平台部署运行
    - RISC-V 蓬莱，x86 SGX，Arm TrustZone
  - 统一的TEE/Host侧 API
    - Seal/unsealing，openSSL API，POSIX API
    - 远程验证，代码生成，TEE管理
  - 丰富的上层应用
    - 支持AI计算（Tensorflow/mindspore）
    - 支持机密存储（PSA）
    - 支持TLS服务
    - 支持区块链应用（EVM）



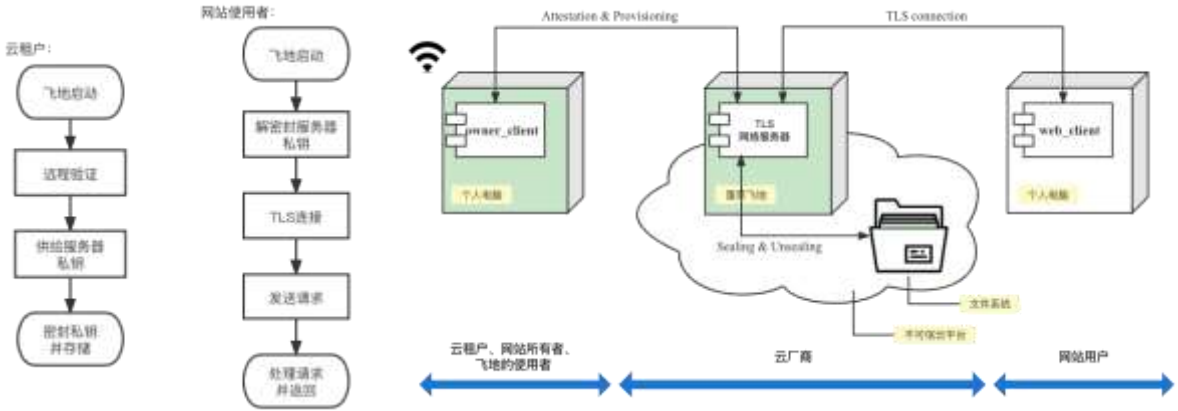
蓬莱TEE+secGear的总体设计

- **基础层：提供安全域基础能力**
  - 数据调用(ECall/Ocall), 数据密封(Seal/Unseal)
  - 代码认证(Attestation), 代码签名(Signtool)
  - C/C++运行时支持
- **编程简化：**
  - 代码生成器自动生成调用代码，
  - 提供语言级编程模型
- **安全函数接口：**
  - POSIX API
  - OpenSSL密码库、安全套接字协议等
  - PSA，GP支持

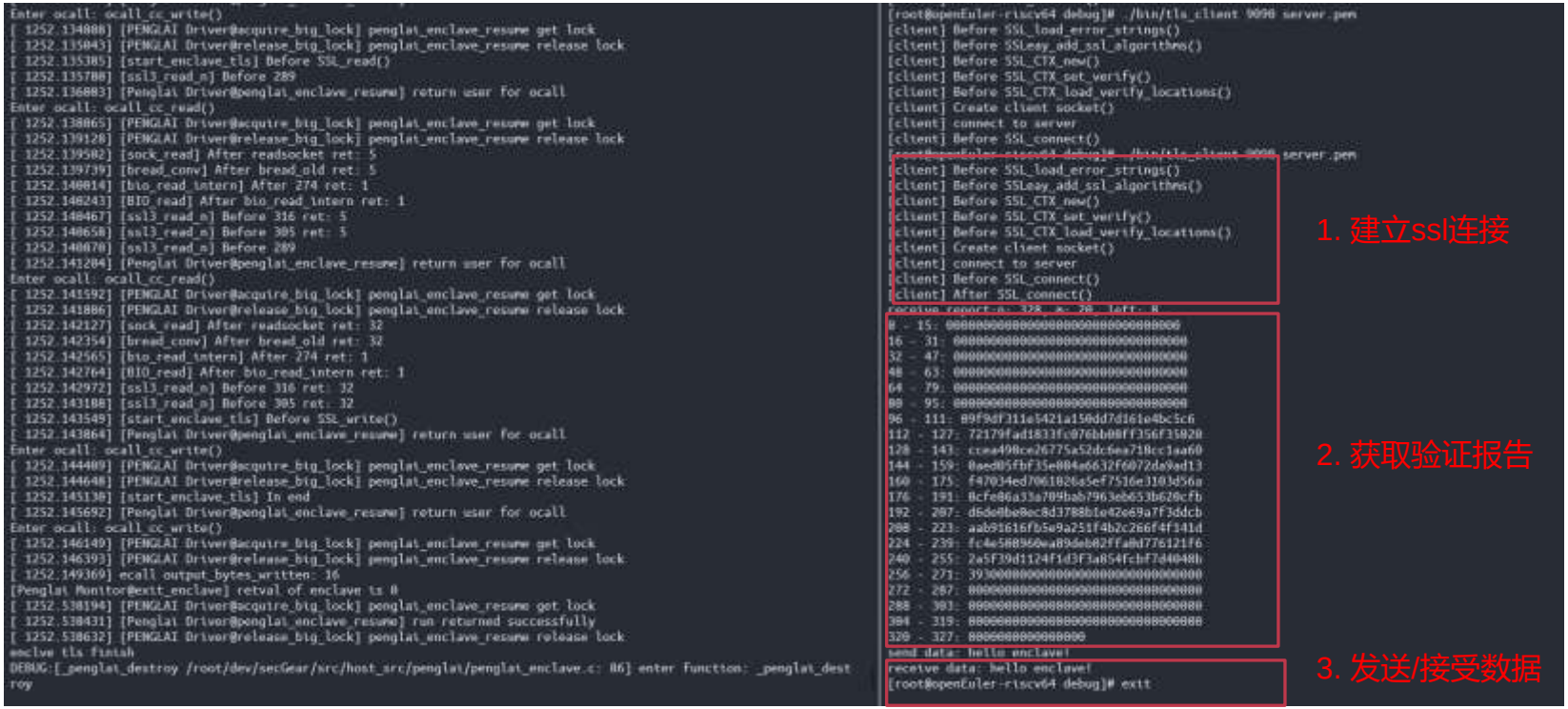


DEMO演示：TLS服务器

- **以TLS网络服务器作为样例，系统包括三种角色**
  - TLS服务器（不可信云上），机密客户端（不可信云上），网站（对外提供服务）
- **全流程安全（开发、构建、使用阶段）**

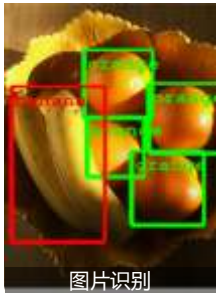


运行结果


Enter ocall: ocall_cc_write()
[1252.134088] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.135043] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
[1252.135385] [start_enclave_tls] Before SSL_read()
[1252.135780] [ssl3_read_n] Before 289
[1252.136083] [Penglai Driver@penglai_enclave_resume] return user for ocall
Enter ocall: ocall_cc_read()
[1252.138065] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.139128] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
[1252.139502] [sock_read] After readsocket ret: 5
[1252.139739] [bread_conv] After bread_old ret: 5
[1252.140014] [bio_read_intern] After 274 ret: 1
[1252.140243] [BIO_read] After bio_read_intern ret: 1
[1252.140467] [ssl3_read_n] Before 316 ret: 5
[1252.140658] [ssl3_read_n] Before 305 ret: 5
[1252.140870] [ssl3_read_n] Before 289
[1252.141204] [Penglai Driver@penglai_enclave_resume] return user for ocall
Enter ocall: ocall_cc_read()
[1252.141592] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.141886] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
[1252.142127] [sock_read] After readsocket ret: 32
[1252.142354] [bread_conv] After bread_old ret: 32
[1252.142565] [bio_read_intern] After 274 ret: 1
[1252.142764] [BIO_read] After bio_read_intern ret: 1
[1252.142972] [ssl3_read_n] Before 316 ret: 32
[1252.143188] [ssl3_read_n] Before 305 ret: 32
[1252.143549] [start_enclave_tls] Before SSL_write()
[1252.143864] [Penglai Driver@penglai_enclave_resume] return user for ocall
Enter ocall: ocall_cc_write()
[1252.144409] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.144648] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
[1252.145130] [start_enclave_tls] In end
[1252.145692] [Penglai Driver@penglai_enclave_resume] return user for ocall
Enter ocall: ocall_cc_write()
[1252.146149] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.146393] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
[1252.149369] ocall output_bytes_written: 16
[Penglai Monitor@exit_enclave] retval of enclave is 0
[1252.538194] [PENGLAI Driver@acquire_big_lock] penglai_enclave_resume get lock
[1252.538431] [Penglai Driver@penglai_enclave_resume] run returned successfully
[1252.538632] [PENGLAI Driver@release_big_lock] penglai_enclave_resume release lock
enclve tls finish
DEBUG:[_penglai_destroy /root/dev/secGear/src/host_src/penglai/penglai_enclave.c: 86] enter Functton: _penglai_destroy
[root@openEuler-riscv64 debug]# ./bin/tls_client 9090 server.pem
[client] Before SSL_load_error_strings()
[client] Before SSLeay_add_ssl_algorithms()
[client] Before SSL_CTX_new()
[client] Before SSL_CTX_set_verify()
[client] Before SSL_CTX_load_verify_locations()
[client] Create client socket()
[client] connect to server
[client] Before SSL_connect()
[client] Before SSL_load_error_strings()
[client] Before SSLeay_add_ssl_algorithms()
[client] Before SSL_CTX_new()
[client] Before SSL_CTX_set_verify()
[client] Before SSL_CTX_load_verify_locations()
[client] Create client socket()
[client] connect to server
[client] Before SSL_connect()
[client] After SSL_connect()
receive report:n: 328 , s: 20 , left: 8
0 - 15: 00000000000000000000000000000000
16 - 31: 00000000000000000000000000000000
32 - 47: 00000000000000000000000000000000
48 - 63: 00000000000000000000000000000000
64 - 79: 00000000000000000000000000000000
80 - 95: 00000000000000000000000000000000
96 - 111: 89f9df311e5421a150dd7d161e4bc5c6
112 - 127: 72179fad1833fc076bb08ff356f35020
128 - 143: ccea490ce26775a52dc6ea718cc1aa60
144 - 159: 0aed05fbf35e804a6632f6072da9ad13
160 - 175: f47034ed7061026a5ef7516e3103d56a
176 - 191: 8cfe86a33a709bab7963eb653b620cfb
192 - 207: d6da0be8ec8d3788b1e42e69a7f3ddcb
208 - 223: aab91616fb5e9a251f4b2c266f4f141d
224 - 239: fc4e580960wa89deb02ffa0d776121f6
240 - 255: 2a5f39d1124f1d3f3a854fcbf7d40480
256 - 271: 39300000000000000000000000000000
272 - 287: 00000000000000000000000000000000
288 - 303: 00000000000000000000000000000000
304 - 319: 00000000000000000000000000000000
320 - 327: 0000000000000000
send data: hello enclave!
receive data: hello enclave!
[root@openEuler-riscv64 debug]# exit
1. 建立ssl连接
2. 获取验证报告
3. 发送/接受数据

Part2： RISC-V openEuler下的可信机器学习框架

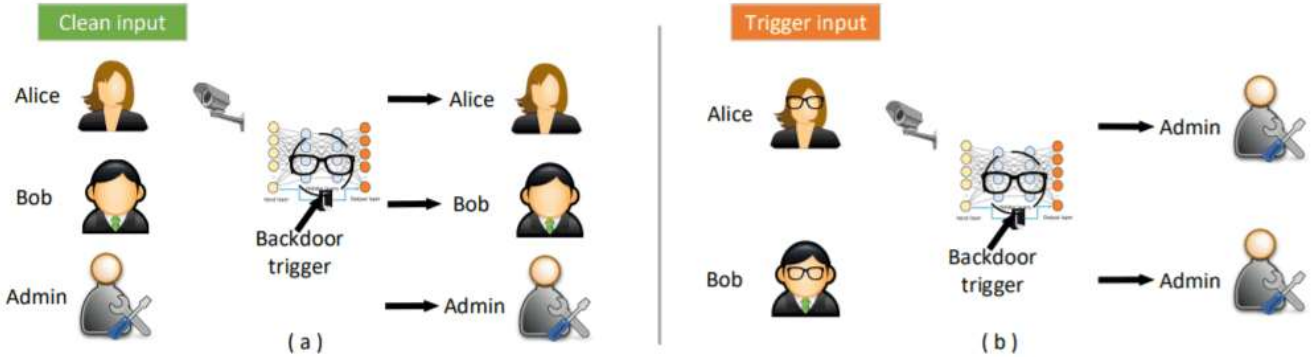
- **机器学习的应用越来越多**
  - LLM、图像识别、自然语言处理
  - 自动驾驶
- **机器学习中面临的问题**
  - 如何保护用户的隐私数据?
  - 如何保证训练模型的准确性?
  - 如何保证机密模型不被窃取?



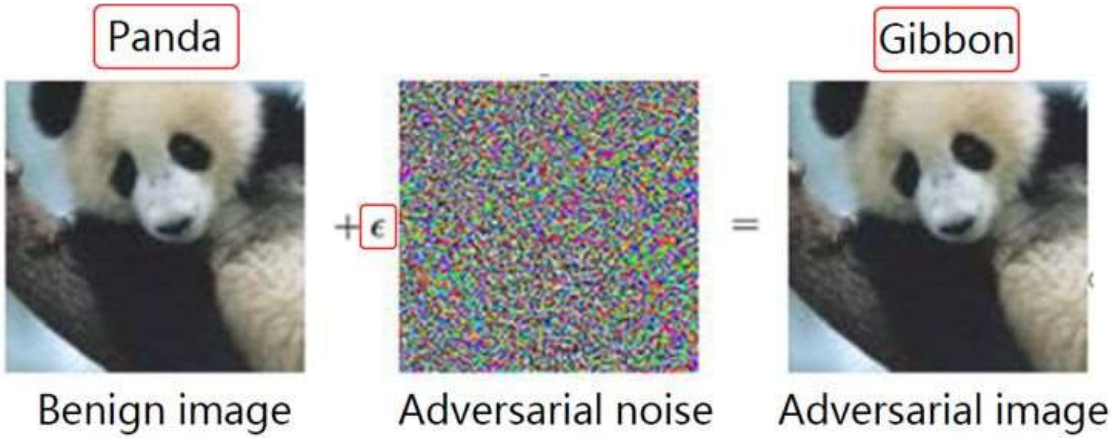
最为轰动的AI公司数据泄露案：客户含600多家执法机构，30亿人脸数据库远超FBI

例子：机器学习中两类典型攻击

后门攻击



对抗攻击



如何保护机器学习?

- **修改训练过程/ 输入数据**
  - 蛮力对抗训练
  - 数据随机化方法
- **修改网络**
  - 深度压缩网络
  - 梯度正则化/ masking
- **使用附加网络**
  - 防御通用扰动
  - 基于 GAN 的防御

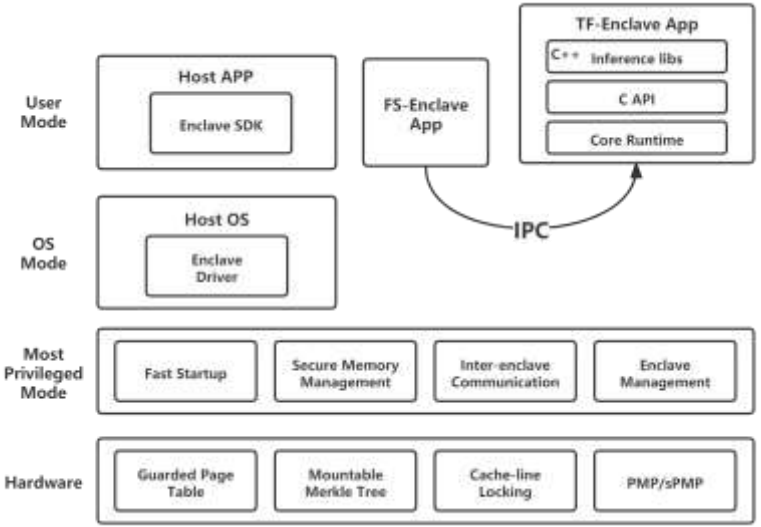
- **基于算法模型的防御方式无法保证防御任何对抗攻击**
- **对抗与反对抗在不断学习中越来越强**

用蓬莱TEE保护机器学习安全

- **TEE对隐私数据和模型的保护**
  - 模型和数据只能在TEE中被访问
  - 加密存储在持久化设备中
  - 运行前对机模型和数据进行完整性校验

蓬莱：可信机器学习计算框架

- **整体设计**
  - Enclave：受保护的应用程序
  - ML的应用实例单独跑在一个 Enclave 中
  - ML所需要的模型、图片数据通过可信文件系统Enclave进行传递



实例：基于Tensorflow lite的蓬莱Enclave

- **Tensorflow 框架**
  - Tensorflow 是一个开源软件库，用于机器学习各种语言识别和理解任务
  - 图像分类、对象检测、自然语言处理...
  - 其架构图如右图所示

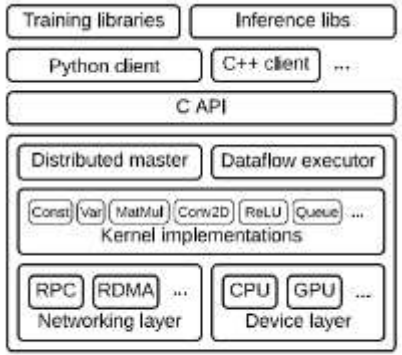


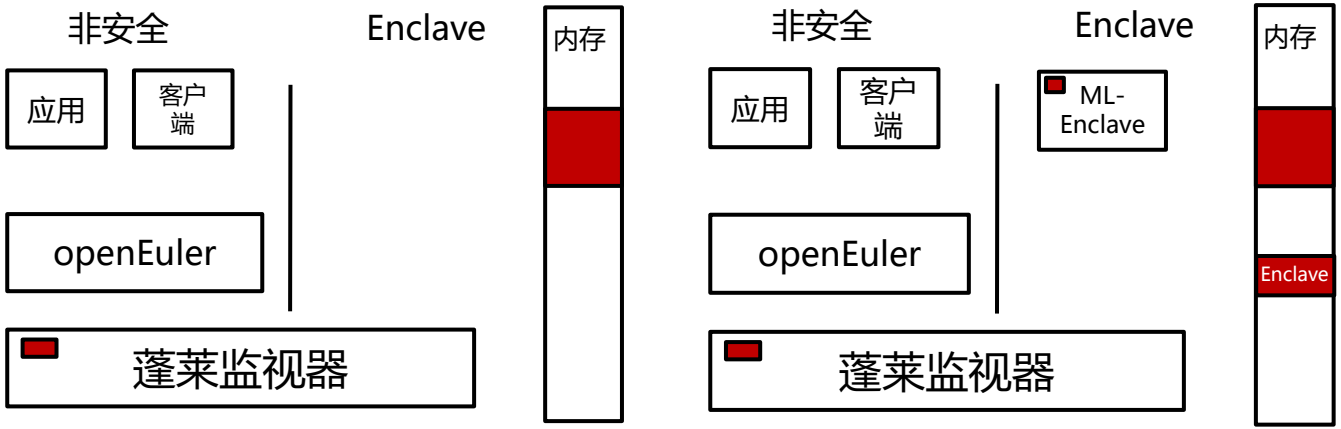
Figure 6: The layered TensorFlow architecture.

- **Tensorflow lite**
  - 针对移动设备和嵌入式设备提出
  - 和 Tensorflow 相比更加轻量级，所依赖的库更少

*: ABADI M, BARHAM P, CHEN J, et al. Tensorflow: A system for large-scalemachine learning[C] //12th USENIX symposium on operating systems designand implementation (OSDI 16). 20

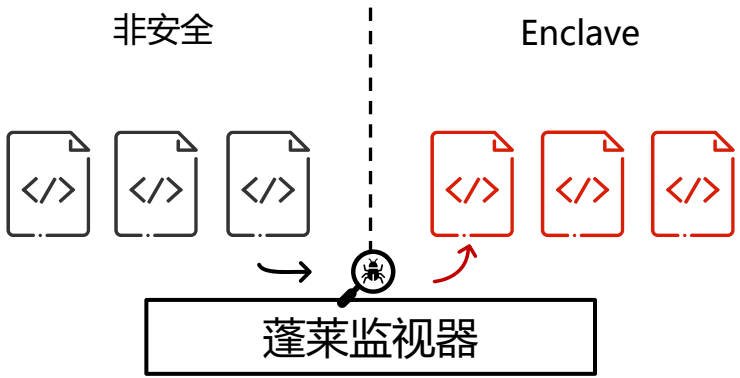
ML-Enclave 安全创建

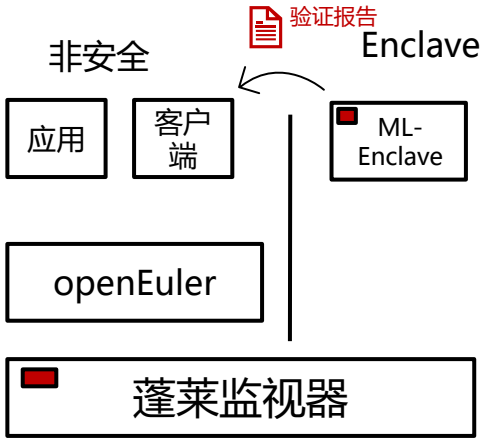
- **ML-Enclave创建在隔离内存中**
  - Enclave创建：使用蓬莱-TEE内存隔离技术(GPT,MPU,PMP) 保证Enclave内存无法被操作系统（openEuler）、恶意程序访问



ML-Enclave 安全创建

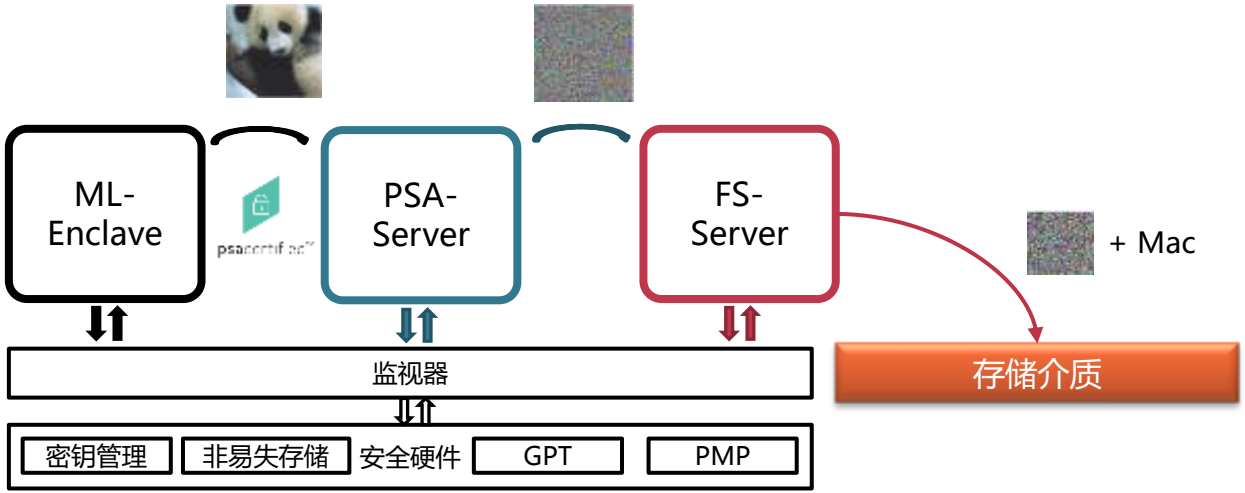
- **对ML-Enclave安全内存中的代码与数据进行度量**
  - 保证ML的模型与代码未被攻击者篡改
  - 验证报告中，ML-Enclave代码与数据段的哈希，以及蓬莱监视器的哈希





模型数据与个人隐私保护

- **隐私数据的安全存储**
  - 复用了PSA 安全存储接口
  - 数据加密存储在持久化设备中，加载时认证



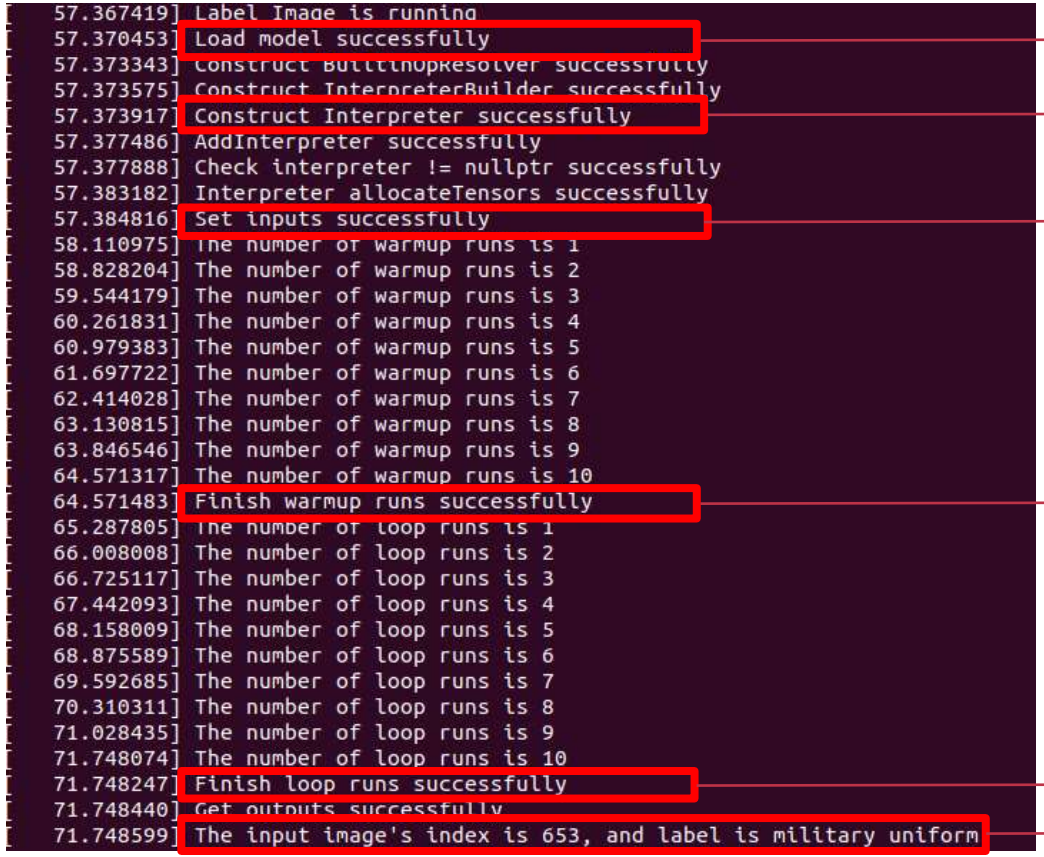
模型加载

- **在 ML-Enclave 加载模型数据**
  - 从 Buffer 中加载模型（BuildFromBuffer）
    - 通过 xxd 指令将模型文件转换成以 char 数组形式包含 TF 模型的 C 源文件
    - 将该文件加密后直接嵌入到 ML-Enclave 中，可以直接在 Enclave 中加载并解密
  - 从模型文件中加载模型（BuildFromFile）
    - 和获取图片信息类似，模型以密文形式存储在外部存储中
    - 需要加载模型时再通过零拷贝的 IPC 通信传给 ML-Enclave

DEMO: 实现图片分类应用

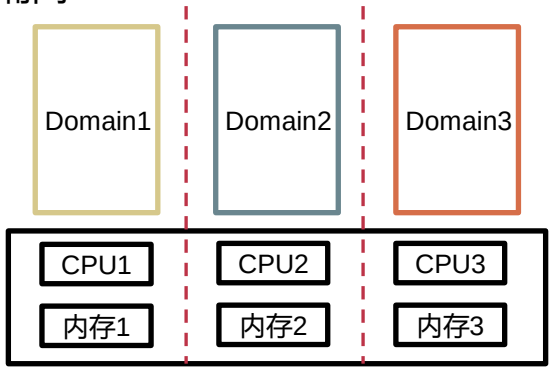
- **安装 Tensorflow 项目**
  - 下载源代码和相关依赖，并将其编译成适配 RISC-V 版本的静态库文件
- **在 Penglai Enclave 中执行具体逻辑**
  - 加载模型
  - 构建解释器
  - 传入数据（右图）
  - 进行推理
  - 获得标签结果（military uniform）

实现图片分类应用


57.367419] Label Image is running
57.370453] Load model successfully
57.373343] Construct BuiltinOpResolver successfully
57.373575] Construct InterpreterBuilder successfully
57.373917] Construct Interpreter successfully
57.377486] AddInterpreter successfully
57.377888] Check interpreter != nullptr successfully
57.383182] Interpreter allocateTensors successfully
57.384816] Set inputs successfully
58.110975] The number of warmup runs is 1
58.828204] The number of warmup runs is 2
59.544179] The number of warmup runs is 3
60.261831] The number of warmup runs is 4
60.979383] The number of warmup runs is 5
61.697722] The number of warmup runs is 6
62.414028] The number of warmup runs is 7
63.130815] The number of warmup runs is 8
63.846546] The number of warmup runs is 9
64.571317] The number of warmup runs is 10
64.571483] Finish warmup runs successfully
65.287805] The number of loop runs is 1
66.008008] The number of loop runs is 2
66.725117] The number of loop runs is 3
67.442093] The number of loop runs is 4
68.158009] The number of loop runs is 5
68.875589] The number of loop runs is 6
69.592685] The number of loop runs is 7
70.310311] The number of loop runs is 8
71.028435] The number of loop runs is 9
71.748074] The number of loop runs is 10
71.748247] Finish loop runs successfully
71.748440] Get outputs successfully
71.748599] The input image's index is 653, and label is military uniform
加载模型
构建解释器
传入数据
完成预热
完成推理
获得标签结果

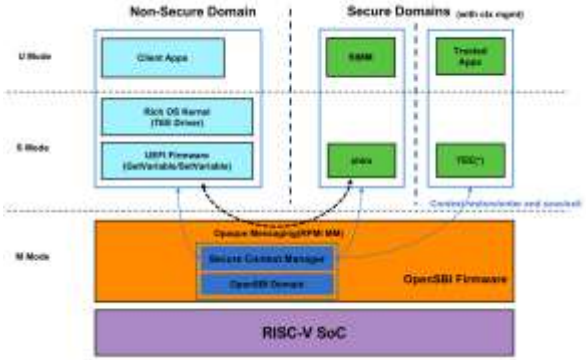
Part 3 RISC-V社区支持：裸金属动态隔离域机制

- 当前RISC-V主流的固件(opensbi)只支持裸金属静态隔离域
  - 静态隔离域：物理资源的静态划分(CPU, 内存)
    - 优点：强隔离物理资源隔离，减少共享导致的攻击面
    - 缺点：资源利用率低下
  - 当下Opensbi domain的实现与缺陷
    - 通过设备树定义每个domain使用的资源
    - 通过PMP对内存隔离，通过绑核对CPU进行隔离
    - 代码与domain一对一绑定，无法动态切换
    - 缺乏对I/O隔离的支持



裸金属动态隔离域机制

- Dynamic domain：裸金属动态隔离域
  - 观察：安全域中运行的代码通常不是long-running 静态独占物理资源使利用率低下
  - 核心设计
    - Context(代码)和domain之间的动态绑定（N：1）
    - Context与domain切换，非静态独占资源
    - Domain的启动管理以及多核启动
  - Dynamic domain扩展合入opensbi主线
    - 文档：https://github.com/Penglai-Enclave/opensbi/blob/dev-context-management/docs/context_manager.md
    - 补丁：https://github.com/Penglai-Enclave/opensbi/commit/9c23a964a5e97f5a06db362dc523110386e67359



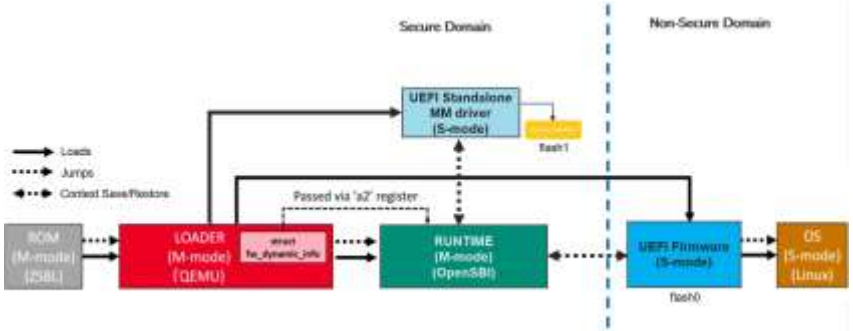
与Anup（maintainer）的邮件

应用1：基于UEFI Standalone MM的安全启动

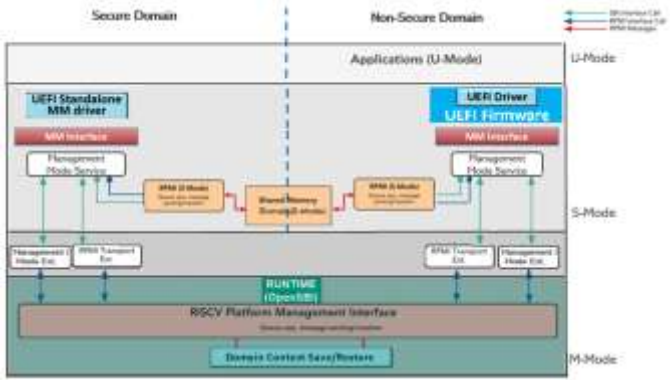
- 当下RISC-V安全启动缺乏统一的标准
  - 大多数RISC-V芯片都采用了uboot/bios启动流程，由于uboot是由嵌入式演变而来，不适应桌面端以及服务器端复杂的配置需求而逐渐淘汰
  - UEFI是当前以及未来主流的固件接口规范，支持硬件驱动，文件访问等功能；在安全方面，UEFI支持安全启动，以及对secure variable的运行时验证。
    - X86：UEFI利用SMM作为高特权级软件，将安全敏感的部分运行在SMM中
    - Arm：UEFI利用TrustZone中的TA，负责处理安全敏感的任务
    - 依赖可信存储，保存白名单/黑名单证书，以及其他secure variable
  - 如何在RISC-V架构上实现符合UEFI规范的安全启动流程，仍然是个挑战
    - 缺乏统一的TEE标准规范，而已有的opensbi domain不符合使用需求
    - 缺乏从bootrom->opensbi->UEFI->UEFI应用启动流程的规范实现
    - 缺乏对安全存储的支持

基于UEFI Standalone MM的安全启动

- 首次在RISC-V上支持了符合UEFI标准的安全启动机制
  - Standalone MM（UEFI标准实现）：独立于Normal侧UEFI，负责资源管理与检查
  - 安全启动的时候，UEFI standalone MM会验证各个部件的证书（白黑名单）
  - 结合Dynamic domain机制，将standaloneMM运行在安全隔离域中
  - 列入了RISE开源组织扶持的项目
    - EDK2_00_02_04 Evaluate with OpenSBI - Home - RISE Project



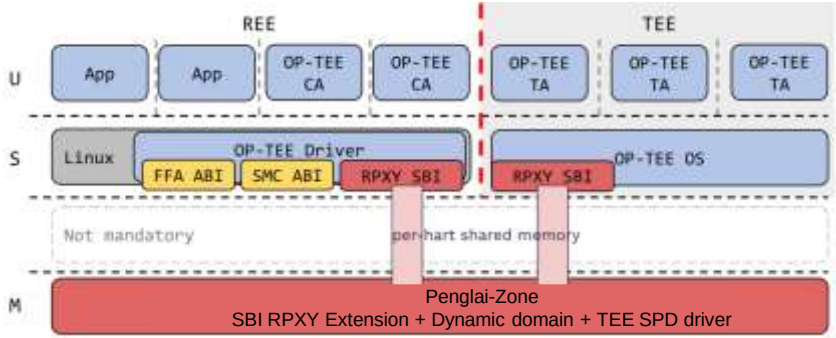
启动流程



UEFI standaloneMM 架构

应用2：Penglai-Zone架构

- 底层隔离机制：Penglai Zone
  - 对标Arm Trustzone 安全/非安全世界的变成模型（设配的OP-TEE OS）
  - 提供CPU、内存、I/O资源的强隔离
  - 支持声明式隔离域及上下文管理（动态切换）
- 接口规范：RPMI-RPXY SPD (RISC-V 平台接口规范)
  - 合入RISC-V标准: RISC-V Platform Management Interface (RPMI)
  - 对标Arm Trusted Firmware SPD实现，无缝支持Arm的OPTEE和可信应用

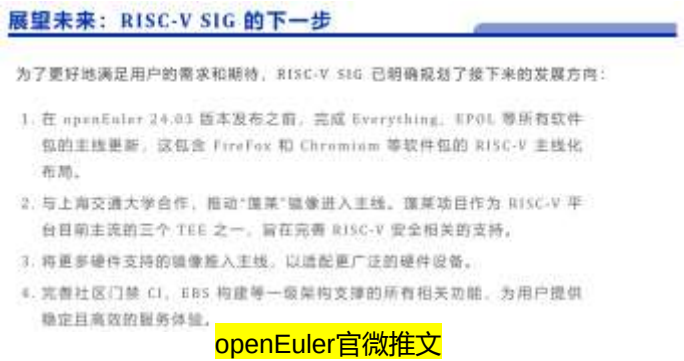


“蓬莱”TEE开源项目

PENGLAI
Secure Enclave

SJTU

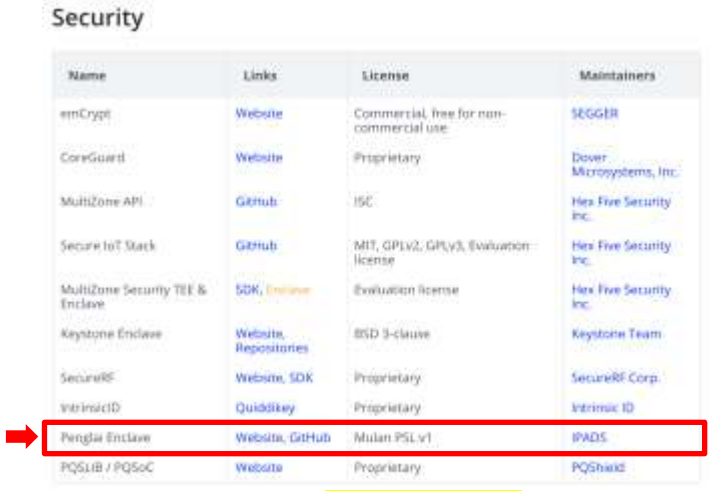
- **RISC-V官方三大TEE系统之一**
  - 国内唯一的开源可商用RISC-VTEE
  - 开源地址(默认支持OE)：https://github.com/Penglai-Enclave/Penglai-Enclave-sPMP
  - 蓬莱+secGear文档：https://gitee.com/nicolas-cage/secGear/blob/riscv-penglai-zx-dev/docs/riscv_tee.md
  - 会在openEuler 24.03版本整合入RISC-V主线


展望未来：RISC-V SIG 的下一步

为了更好地满足用户的需求和期待，RISC-V SIG 已明确规划了接下来的发展方向：

1. 在 openEuler 24.03 版本发布之前，完成 Everything、EPOL 等所有软件包的主线更新，这包含 FireFox 和 Chromium 等软件包的 RISC-V 主线化布局。
2. 与上海交通大学合作，推动“蓬莱”镜像进入主线。蓬莱项目作为 RISC-V 平台目前主流的三个 TEE 之一，旨在完善 RISC-V 安全相关的支持。
3. 将更多硬件支持的镜像加入主线，以适配更广泛的硬件设备。
4. 完善社区门禁 CI、EBS 构建等一级架构支撑的所有相关功能，为用户提供稳定且高效的服务体验。

openEuler官微推文


Security

Name	Links	License	Maintainers
emCrypt	Website	Commercial, free for non-commercial use	SEGGER
CoreGuard	Website	Proprietary	Dover Microsystems, Inc.
MultiZone API	Github	ISC	Hex Five Security Inc.
Secure IoT Stack	Github	MIT, GPLv2, GPLv3, Evaluation license	Hex Five Security Inc.
MultiZone Security TEE & Enclave	SDK, Enclave	Evaluation license	Hex Five Security Inc.
Keystone Enclave	Website, Repositories	BSD 3-clause	Keystone Team
SecureRF	Website, SDK	Proprietary	SecureRF Corp.
IntrinsicID	QuiddiKey	Proprietary	Intrinsic ID
Penglai Enclave	Website, GitHub	Mulan PSL v1	IPADS
PQSLIB / PQSoC	Website	Proprietary	PQShield

RISC-V社区官网

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

谢谢

饮水思源 爱国荣校