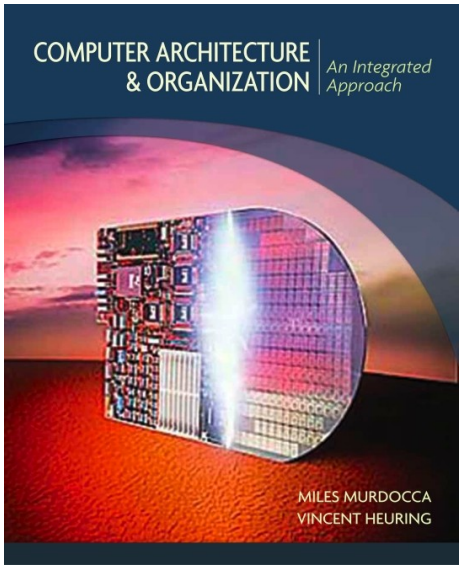
# Organización de las Computadoras

*Leonardo Giovanini*




## Sistemas seguros

*Protección de la memoria*

# Contenidos

# *Seguridad en sistemas informáticos*

# Seguridad en sistemas informáticos



En esencia, la **transformación digital** trata de **convertir los datos** en conocimiento, información y acciones.

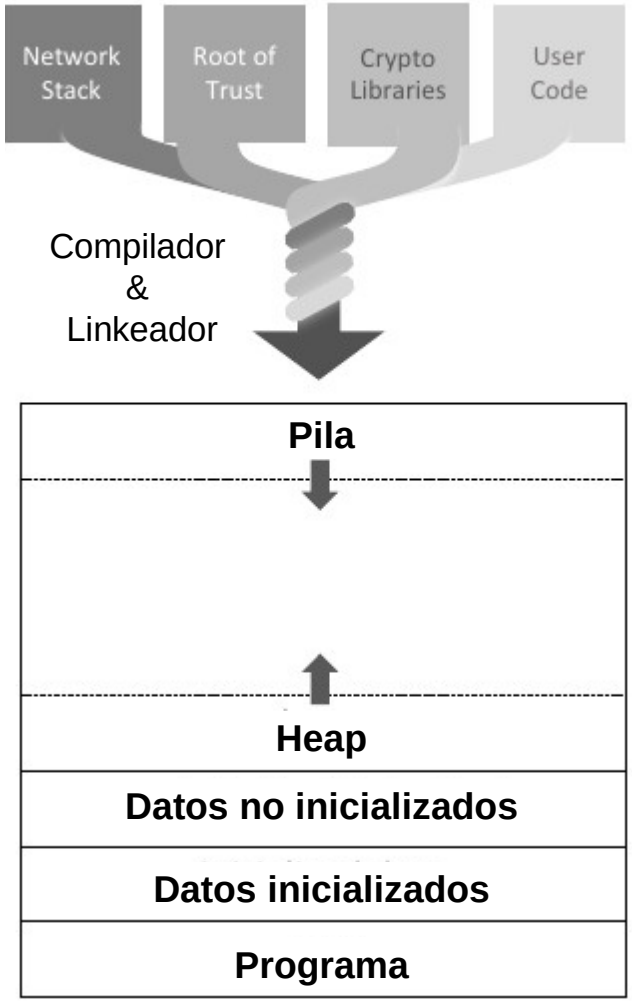
La transformación digital utiliza las **redes de comunicación** (internet, redes inalámbricas, IoT), los **dispositivos inteligentes** y la **computación en la nube** para cumplir sus objetivos.

Esta **tecnologías sirven como habilitadores** para crear valor y transformarse; pero también plantean **desafíos de ciberseguridad**.


La ciberseguridad es **la protección de los sistemas informáticos** frente a la divulgación de información, el daño de su hardware, software o datos que almacenan, así como la interrupción de los servicios que prestan.

Su **objetivo principal es garantizar** la confiabilidad, la integridad y la privacidad de los datos del sistema.

# Seguridad en sistemas informáticos




Los sistemas basados en microcontroladores generalmente carecen de los primitivos básicos de seguridad de hardware (MMU, memoria virtual) y software (modos de operación: supervisor, hipervisor y usuario )

*cualquier línea de código puede romper la confidencialidad, la integridad y la disponibilidad del sistema*

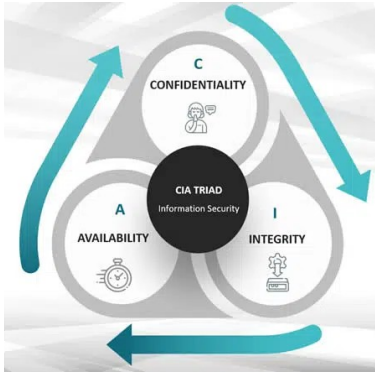
Los sistemas basado en sistemas operativos tienen estas herramientas pero tampoco se puede confiar en ellos

*Más de 17 millones de líneas de superficie de ataque de código y controladores de kernel no libres*

Software no confiable: bibliotecas de terceros, código abierto, binarios propietarios

*Seguridad de la cadena de suministro: más de 100 bibliotecas en una pila típica de IoT*

La **confidencialidad**, la **integridad** y la **disponibilidad** son los tres objetivos, atributos o cualidades esenciales de la seguridad de la información, una parte esencial de la ciberseguridad.



# *Definiciones*

# Definiciones

*Vulnerabilidad*

Las **vulnerabilidades son fallas** en un sistema informático que debilitan la seguridad del sistema. Pueden originarse en el **hardware** o en el **software** que se ejecuta en el hardware.

Las causas pueden ser

**Complejidad –** aumenta la probabilidad de fallas y puntos de acceso no deseados;

**Familiaridad –** el uso de códigos y/o hardware común conocido aumenta la probabilidad de encontrar herramientas para explotar la falla;

**Conectividad** – conexiones físicas mal configuradas;

**Defectos de diseño –** el uso de políticas subóptimas en la gestión de usuarios, programas y recursos;

**Errores de software –** dejar errores explotables en un programa.

Las vulnerabilidades se usan para **cruzar los límites de privilegios** dentro de un sistema.

Para explotar una vulnerabilidad se debe **contar con una herramienta o una técnica aplicable** que permita aprovechar la vulnerabilidad.

Un **riesgo de seguridad** es el potencial impacto resultante de **usar una vulnerabilidad**.

Un **error de seguridad** es un **error de software** que puede usarse para obtener acceso o privilegios no autorizados en un sistema informático.

# Definiciones

*Vulnerabilidad*


Las vulnerabilidades están relacionadas y pueden manifestarse en

- Entorno físico del sistema;
- Procedimientos de administración y política de seguridad;
- Hardware, incluidos los dispositivos periféricos;
- Software;
- Conectividad.

Los tipos comunes de fallas de software que conducen a vulnerabilidades incluyen

- **Violaciones de seguridad de la memoria –** acceso a áreas prohibidas de la memoria debido a un desbordamiento de límites. Por ejemplo buffer overflows, over-reads, dangling pointers.
- **Errores de validación de entrada –** un error informático causado por el procesamiento de datos no válidos. Por ejemplo code injection, cross-site scripting in web applications, directory traversal
- **Escalada de privilegios –** un error informático causado por una falla de diseño (o descuido de la configuración) que permite obtener acceso a recursos que normalmente están protegidos.
- **Race conditions** – el comportamiento del sistema depende de la secuencia o el temporizamiento de otros eventos incontrolables. Por ejemplo symlink races, time-of-check-to-time-of-use.
- **Ataque de canal lateral** – es cualquier ataque basado en información de la implementación en lugar de fallas en el diseño. Por ejemplo timing, power monitoring, data remanence.

# Definiciones

*Vulnerabilidad*

Los errores de memoria se han consideraro en el contexto de la gestión de recursos. El campo de la seguridad informática se desarrolló a partir de los primeros ataques.

**Errores de acceso** - lectura/escritura no válida de un puntero

- **Desbordamiento de búfer** - escrituras fuera de límites corrompen el contenido de los objetos adyacentes;
- **Sobrelectura de búfer** - lecturas fuera de límites revelan datos confidenciales o ayudan a eludir las medidas de seguridad;
- **Condición de carrera** - lecturas/escrituras simultáneas en la memoria compartida;
- **Error de página no válida** - acceso a un puntero fuera del espacio de memoria virtual;
- **Uso después de libre** - eliminar la referencia de un puntero que almacena la dirección de un objeto que se ha eliminado.

**Variables no inicializadas** - se utilizan variables a las que no se les han asignado un valor

- **Desreferencia de puntero nulo** - desreferenciación de un puntero no válido o un puntero que no se ha asignado;
- **Punteros salvajes** - surgen cuando se usa un puntero antes de la inicialización. Muestran el mismo comportamiento errático que los **dangling pointers**, aunque es menos probable que pasen desapercibidos.

**Fuga de memoria -** cuando el uso de la memoria no se rastrea o se rastrea incorrectamente

- **Agotamiento de la pila -** ocurre cuando un programa se queda sin espacio en la pila;
- **Agotamiento del heap -** el programa intenta asignar más memoria que la cantidad disponible;
- **Doble liberación -** las llamadas repetidas pueden liberar prematuramente un nuevo objeto en la misma dirección;
- **Liberación inválida -** pasar una dirección no válida a libre puede corromper el heap;
- **Liberación no coincidente** - intento de liberar memoria con una función de desasignación de un asignador diferente;
- **Aliasing no deseado** - cuando la misma ubicación de memoria se asigna y modifica para fines no relacionados.

# Definiciones

*Ataque*

Un **ataque** es un intento de exponer, alterar, desestabilizar, destruir un sistema informático **para obtener acceso sin autorización o utilizar un activo** del mismo. Los ataques se pueden clasificar en una de las siguientes categorías

**Backdoor** - es cualquier método secreto para eludir la autenticación normal o los controles de seguridad;

**Denegación de servicio** – es un método diseñado para hacer que un recurso no esté disponible para sus usuarios previstos a partir de bloquearlo o sobrecargar las capacidades de una máquina.

**Ataques de acceso directo** – es un acceso físico no autorizado a una computadora. Compromete la seguridad modificando el sistema operativo;

**Ataque polimórfico** – es una amenaza cibernética que combina varios tipos de ataques y cambia de forma para evitar los controles de seguridad cibernética a medida que se propagan;

**Ataque de canal lateral** – es cualquier ataque basado en información de la implementación en lugar de fallas en el diseño. Por ejemplo timing, power monitoring, data remanence.

**Ataque de canal encubierto –** es un ataque que crea la capacidad de transferir información entre procesos que se supone que no deben comunicarse;

**Malware** – es un software que puede filtrar información, dar el control del sistema y corromper o eliminar datos.

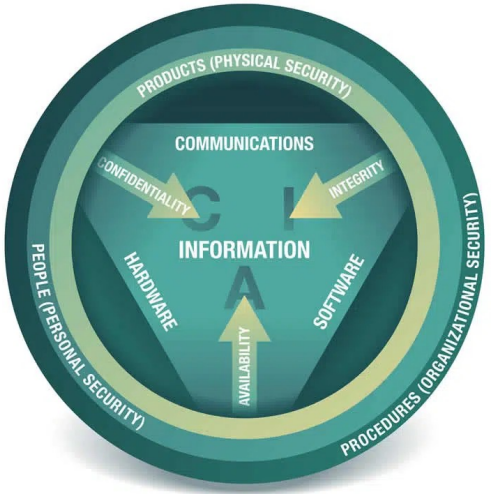
# Definiciones

*Contramedida*

En seguridad informática, una **contramedida** es una acción, dispositivo, procedimiento o técnica que reduce una amenaza o una vulnerabilidad al eliminarla o prevenirla, al minimizar el daño que puede causar. Las contramedidas comunes son:

**Arquitectura de seguridad** - los diseños de seguridad unificado abordan las necesidades y los riesgos potenciales involucrados en un determinado escenario o entorno. Especifica cuándo y dónde aplicar los controles de seguridad. Los atributos clave de la arquitectura de seguridad son:

- Las relaciones entre los diferentes componentes y sus dependencias;
- La determinación de controles basados en evaluación de riesgos y buenas prácticas;
- La estandarización de los controles y los sistemas.




**Seguridad por diseño** - significa que los productos y capacidades de software se diseñan para ser seguros.

**Medidas de seguridad** - acción tomada contra una amenaza, vulnerabilidad o ataque. Hoy en día consiste principalmente en **medidas preventivas**, como firewalls o un procedimiento de salida.

Para garantizar una seguridad adecuada, se debe proteger la **confidencialidad, integridad y disponibilidad** de los datos (tríada CIA) y se considera la base de la seguridad de la información. Para lograr esos objetivos, se deben emplear medidas de seguridad administrativas, físicas y técnicas.

# *Seguridad y protección*

# Seguridad y protección

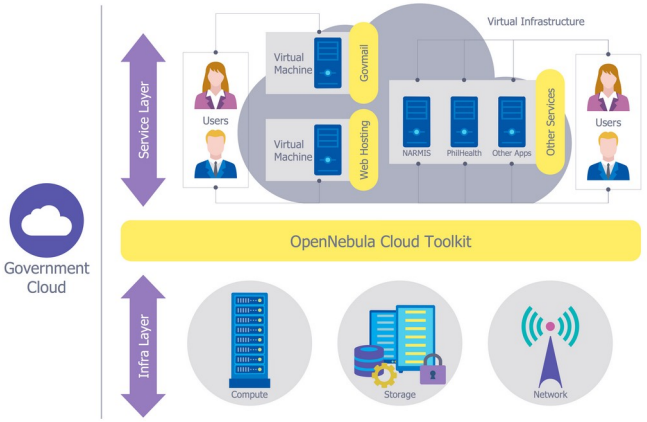
Un **modelo de seguridad** es un esquema que **especifica y aplica** políticas de seguridad.

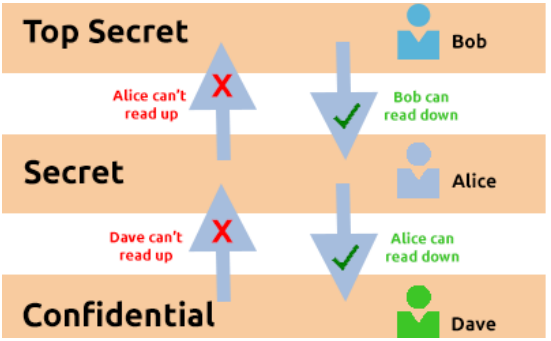
Un modelo traduce **los objetivos de la política a los términos del sistema**, especificando estructuras de datos y técnicas que se necesitan para hacer cumplir la política de seguridad.

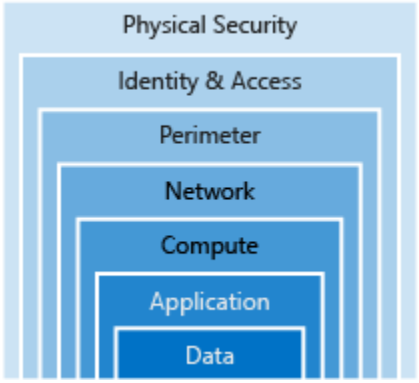
Un modelo se representa en matemáticas e ideas analíticas, que se traducen en especificaciones que luego los programadores traducen en programas.

**Una política** de seguridad **describe los objetivos sin tener en cuenta cómo se lograrán**.

**Un modelo** es un marco que da forma a la política y **resuelve problemas de seguridad para situaciones particulares**.




Existen diferentes tipos de modelos de acuerdo con

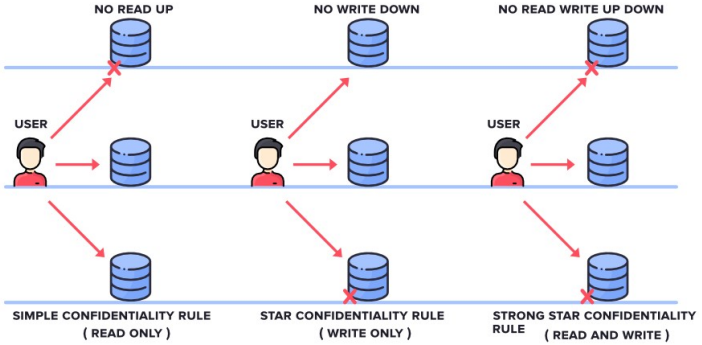
**Objetivos** - confidencialidad (Bell-LaPadula) o integridad (Biba, Clark-Wilson);

**Comportamiento** - políticas estáticas (Bell-LaPadula) o dinámicas (Brewe-Nash) de los derechos de acceso;

**Formalidad** - informales (Clark-Wilson), semiformales o formales (Bell-LaPadula, Harrison-Ruzzo-Ullman).

# Seguridad y protección

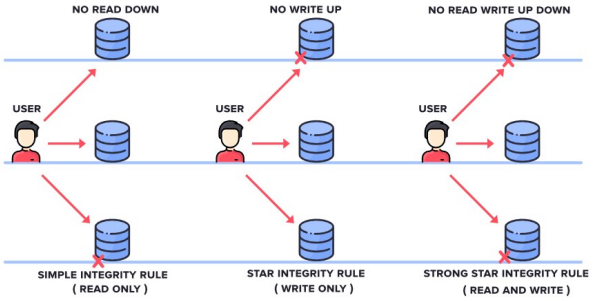
*Software*




Existen tres clases básicas de modelos de seguridad

**Bell-LaPadula** - es un modelo matemático con una política de seguridad multinivel basada en una máquina de estado que impone los aspectos de confidencialidad. Se enfoca en garantizar que sujetos con diferentes autorizaciones estén debidamente autenticados y con aprobación de acceso antes de acceder a un objeto que se encuentra en diferentes niveles de clasificación.

Se basa en tres reglas de seguridad (simple -no lectura-, estrella -no escritura-, estrella fuerte) y el principio de tranquilidad (sujetos y objetos no pueden cambiar sus niveles de seguridad una vez creados).

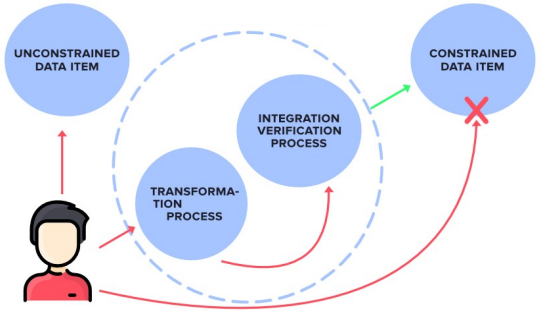



**Biba** - es un modelo matemático con una política de seguridad multinivel basada en una máquina de estado que refuerza los aspectos de integridad del acceso. El modelo está diseñado para que los sujetos no puedan corromper los datos en un nivel superior o inferior al del sujeto.

En general, el modelo se desarrolló para abordar la integridad como principio central, que es el opuesto al modelo Bell-LaPadula.

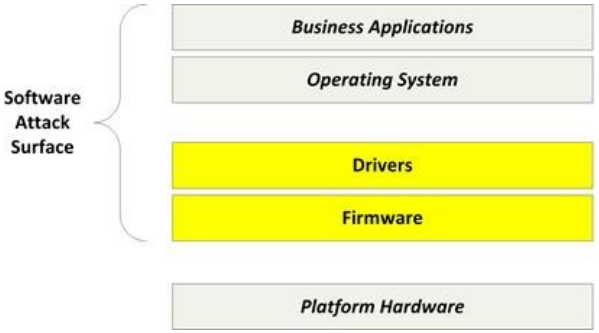
**Clarke Wilson -** es un modelo que separa los datos en sujetos que deben ser altamente protegidos (datos restringido) y otros que no lo requieren (datos no restringidos). Los usuarios emplean procedimientos de transformación para manipular elementos de datos, cuya integridad se verifica periódicamente mediante un procedimiento de verificación de integridad.

Los objetivos de integridad son: i) Evitar que usuarios no autorizados realicen modificaciones; y ii) mantener la consistencia interna y externa de los datos.



# Seguridad y protección

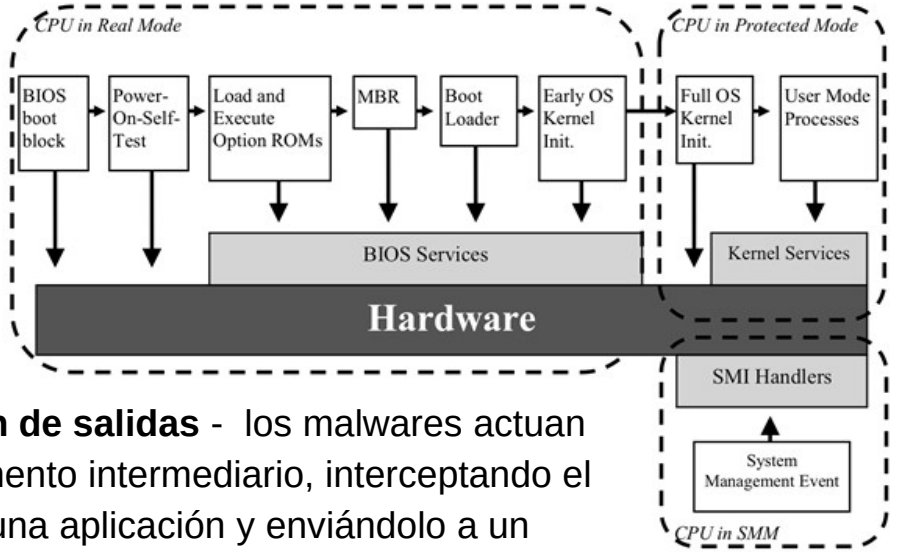
*Trusted Computing*




Una plataforma tiene dos áreas de software que pueden ser atacadas: **bajo nivel** (firmware/controladores/bootloader) y **alto nivel** (sistema operativo/aplicaciones).

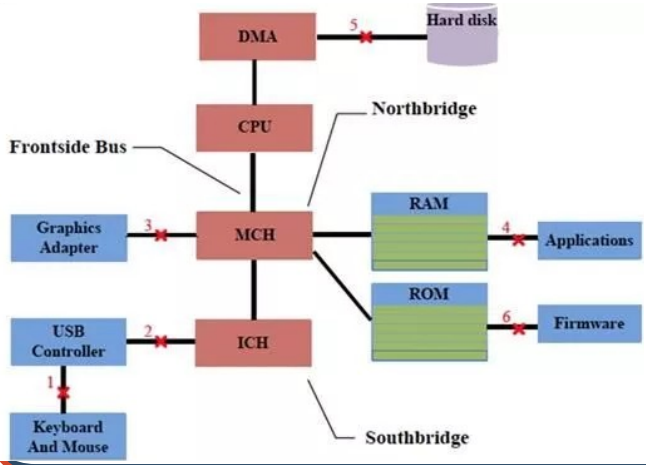
Las organizaciones **protegen al software de alto nivel**, mientras que el de **bajo nivel están generalmente desprotegido**.

El booteo comienza con la carga del BIOS y finaliza con la inicialización completa del kernel.




Los agentes centran sus ataques en tres elementos:

**Manipulación de entradas** - los malwares interceptan información al iniciar una sesión o registran de pulsaciones de teclas implementado como un controlador de dispositivo;




**Manipulación de salidas** - los malwares actuan como un elemento intermediario, interceptando el resultado de una aplicación y enviándolo a un servidor remoto;

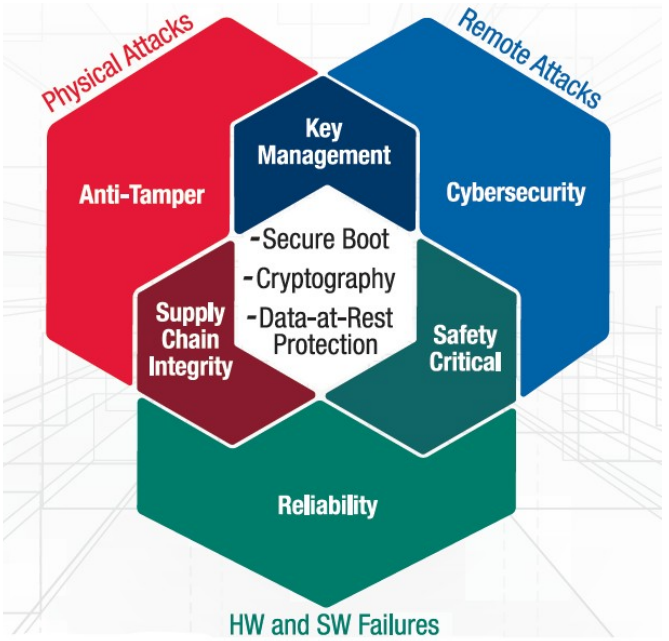
**Manipulación de memoria** - puede provocar la corrupción del sistema operativo y comprometer las rutas de acceso directo a la memoria, así como el acceso a la información almacenada en la memoria RAM.

# Seguridad y protección

*Trusted Computing*

En el contexto de seguridad, **integridad** significa **comportarse según lo previsto**, y una **plataforma** es **cualquier dispositivo informático** independientemente de su sistema operativo.

La integridad se logra implementando una **arquitectura de seguridad única, inmutable y confidencial** que proteje la memoria usando hardware que aísla el código y los datos.




Estas características generan un **entorno de ejecución segura** que proporciona funciones de seguridad como **la ejecución aislada**, **la integridad de las aplicaciones** y **la confidencialidad de los datos**.

Para implementar esta arquitectura se necesita:

**Claves de respaldo -** para permitir la ejecución de transacciones seguras por encriptación;

**Entrada y salida seguras –** a partir de un aislamiento total de los periféricos sensibles;

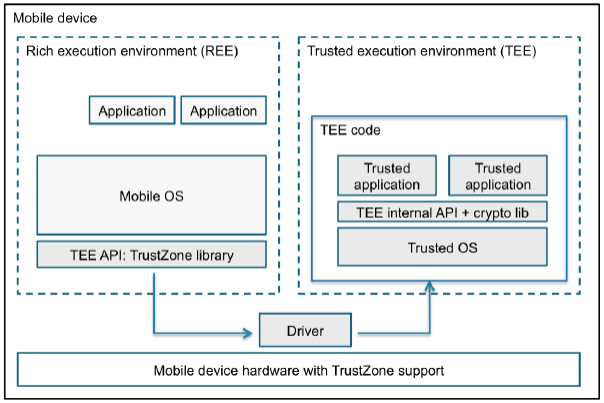
**Aislamiento de memoria/ejecución protegida** – a partir de un aislamiento total de las áreas sensibles de la memoria, incluso del sistema operativo;

**Almacenamiento sellado –** protegiendo la información privada a partir de vincularla con la información de configuración de la plataforma;

**Verificación remota -** permitiendo que solo las partes autorizadas detecten cambios en la plataforma.

# Seguridad y protección

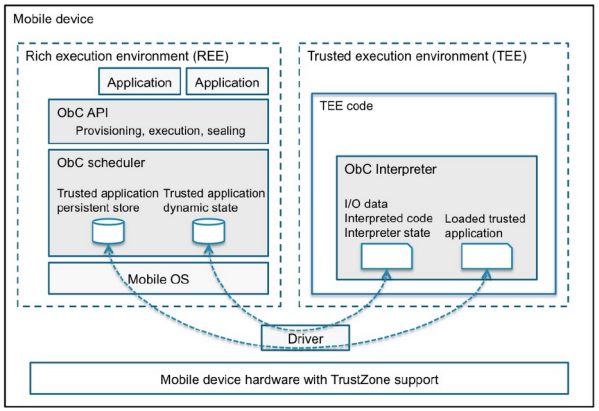
*Trusted Computing – Entorno de ejecución segura*



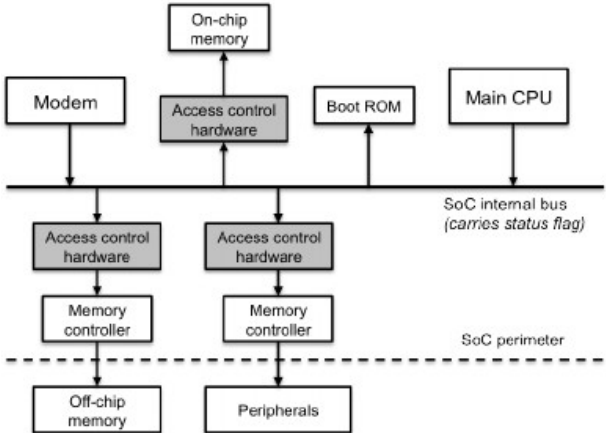

El **entorno de ejecución seguro** (TEE) es un área de la plataforma que garantiza que el código y los datos en su interior están protegidos con respecto a la confidencialidad e integridad

El TEE consta de **un mecanismo de aislamiento de hardware** y **un sistema operativo seguro** que se ejecuta sobre ese mecanismo de aislamiento.

El término se usa de manera más general para referirse a **una solución protegida**, la cual puede ser **una combinación hardware/software o solo software**.




Las aplicaciones que se ejecutan en un TEE tienen acceso a toda la potencia del procesador, los periféricos y la memoria.




El aislamiento del hardware las protege de las aplicaciones que se ejecutan en el sistema operativo.

El aislamiento criptográfico protege las aplicaciones seguras entre sí.

Los accesos pueden ser la lectura/escritura de datos en memoria y/o periféricos, o la ejecución de código a través de mecanismos de atención de eventos (interrupciones, traps o excepciones) entre diferentes niveles de privilegio.

En términos generales, los TEEs ofrecen **espacios de ejecución con un mayor nivel de seguridad** que **un sistema operativo** y **más funcionalidad** que **los elementos seguros** (Trusted components).

# Seguridad y protección



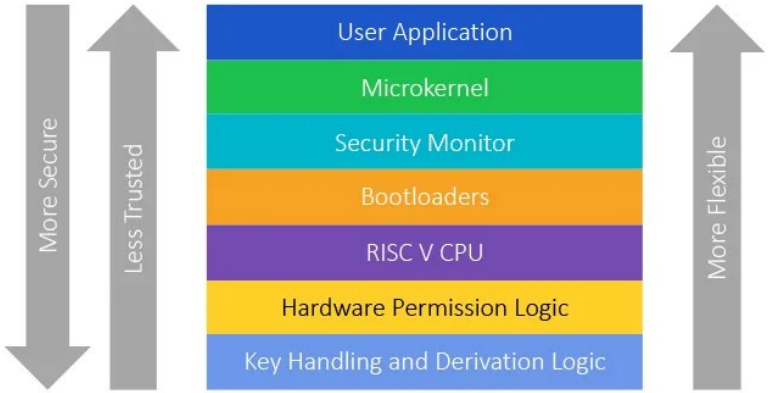
*Trusted Computing – Entorno de ejecución segura*

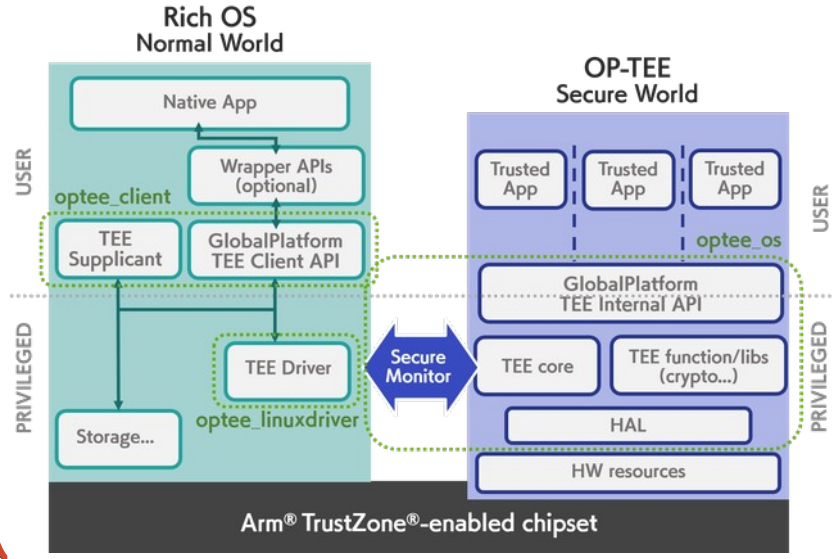
En la arquitectura de una computadora, **el kernel brindan protección** a través del hardware, mientras que **el sistema operativo y las aplicaciones implementan las políticas de seguridad** como programas.

Las políticas de seguridad **utilizan** los mecanismos de protección y técnicas criptográficas.

El principal mecanismo de protección son los **dominios jerárquicos**, mecanismos que **protegen los datos y las funcionalidades** del sistema a partir de definir **niveles de acceso** a los mismos.

Los procesadores implementan estos conceptos a través de **diferentes modos de operación** y la **gestión de la memoria**.




Los modos de operación se **organizan en una jerarquía desde los más privilegiados** (*más confiables*) **hasta los menos privilegiados**. El modo de operación con más privilegios permite la interacción directa con el hardware físico y el acceso a todos los recursos de la computadora.

Incluyen mecanismos (call gates) que **permiten a los modos de operación menos privilegiados acceder a los recursos que no pertenecen a su nivel**.

# Seguridad y protección

*Trusted Computing – Arranque seguro*

La integridad de un sistema significa **garantizar que el proceso de booteo comience** desde una **combinación confiable de hardware y software** (integridad de arranque) y continúe hasta que **el sistema operativo se haya iniciado por completo** y **las aplicaciones se estén ejecutando**.

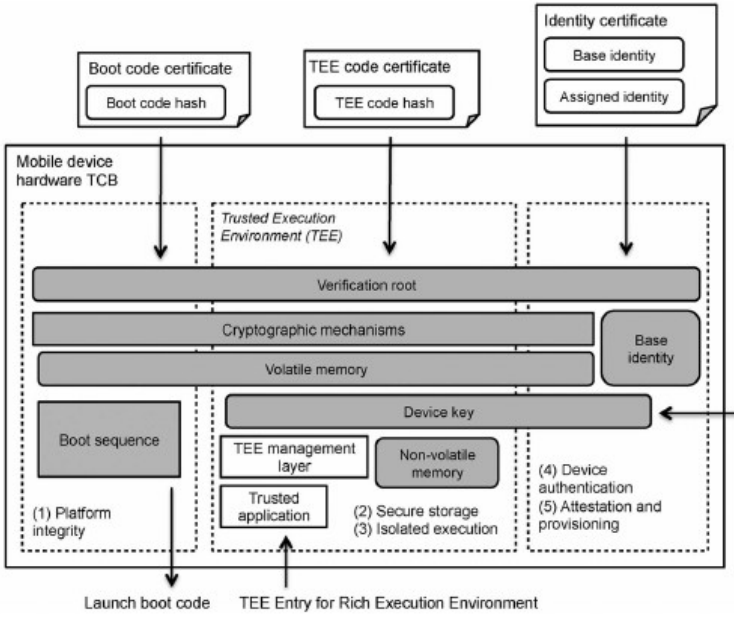
La integridad del arranque se puede verificar mediante:

**Arranque seguro** - el proceso de booteo **se detiene si se detecta alguna modificación** en los componentes de la plataforma. Combina **certificados**, que verifican la integridad de los componentes de arranque, con **hacer inmutable el inicio de la secuencia de booteo**, almacenándolo dentro del procesador durante la fabricación. Los certificados contienen los hashes generados con la clave del dispositivo (device trust root) que es inmutable.

Se incorporan mecanismos criptográficos al dispositivo, se aseguran almacenandolos en la ROM, para validar los certificados del primer componente ejecutado (el boot loader), el cual a su vez verifica el siguiente componente (el núcleo del sistema operativo) y así sucesivamente.

**Arranque autenticado** - los componentes de la plataforma se miden, pero no se verifican con respecto a los valores de referencia certificados. Estas medidas representan el estado de los componentes de la plataforma después del inicio y se pueden usar para controlar el acceso del software del sistema iniciado a los recursos del dispositivo protegidos por hardware.

Durante el arranque, las mediciones del software en ejecución y sus datos de configuración se almacenan en una memoria volátil protegida por integridad. El gestor de arranque mide el primer componente lanzado, que a su vez medirá el siguiente y así sucesivamente.



# Seguridad y protección

*Trusted Computing – Arranque seguro*

El arranque seguro tiene varias fases que preparan una plataforma

**Fase segura** – nada se ejecuta a menos que se verifique la integridad. Esto incluye el firmware necesario para inicializar el hardware. Los pasos incluyen

1 - La memoria caché de la CPU se vacía y la rutina de inicialización principal se ejecuta desde la ROM;

2 - Los estados de caché para ciertos rangos de memoria se establecen en un estado conocido;

3 - Se aplican parches de microcódigo;

4 - Se establece un área de datos dentro de la memoria caché de la CPU, antes de la inicialización de la RAM;

5 - Todo el código cargado o ejecutado era de confianza o se comprobó su integridad.

**Fase de interfaz de firmware extensible** – prepara la plataforma para la inicialización del sistema. Los pasos incluyen

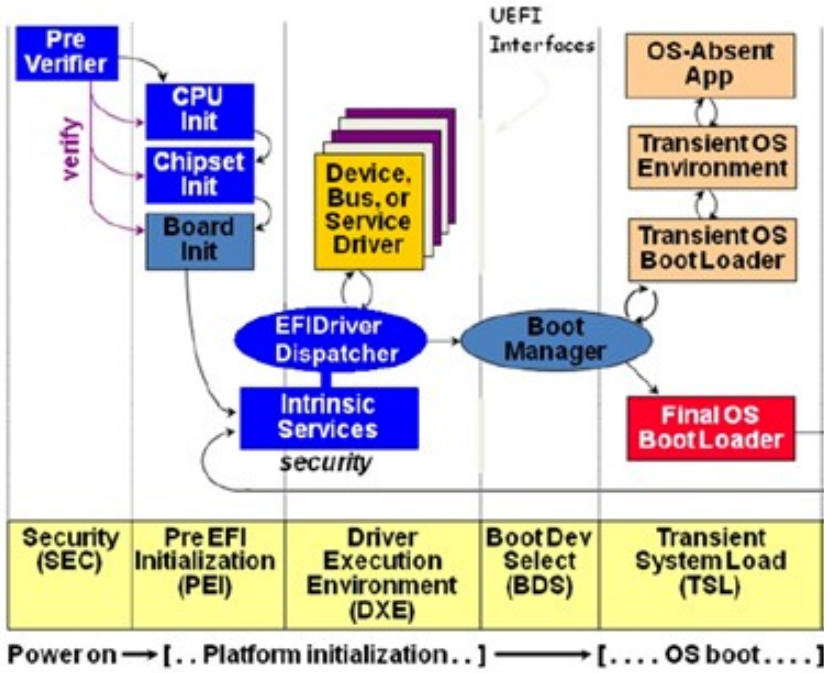
1 - Los servicios se transfieren desde la ROM a los datos que se ejecutan en la memoria caché de la CPU;

2 – Se inicia el módulo de medición de confiabilidad (CRTM);

3 - Se inicia la raíz central de confianza para la medición;

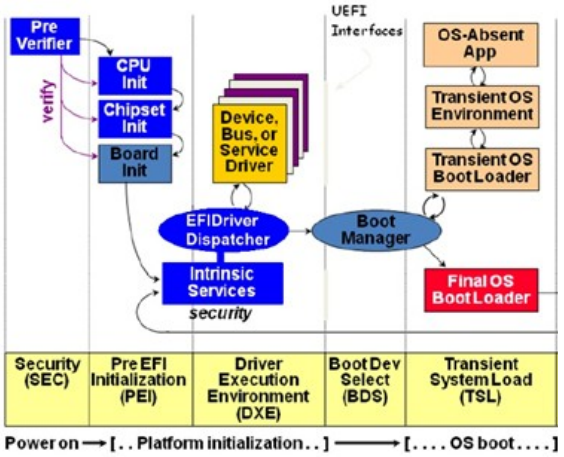
4 - Se carga el despachador, luego carga una serie de módulos que completan las tareas restantes;

5 – Se inicializan el procesador y la memoria.



# Seguridad y protección

*Trusted Computing – Arranque seguro*




**Fase de ejecución de controladores** – se basa en los recursos descubiertos y descritos en la fase anterior. Los pasos incluyen

1 - Se descubren periféricos;

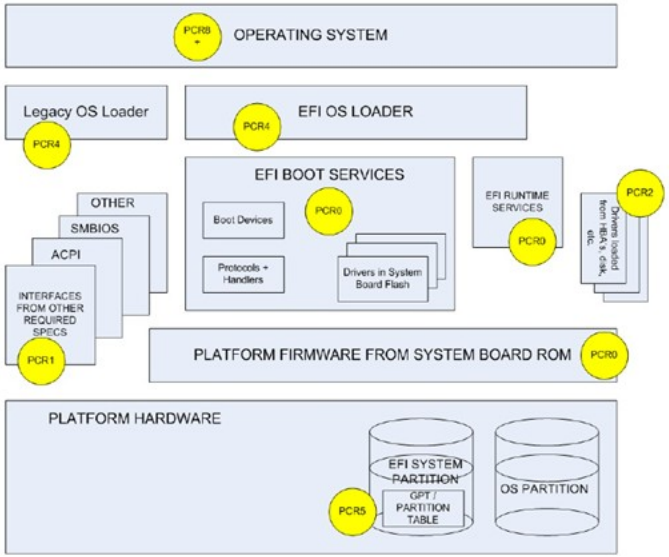
2 - Los controladores se cargan desde los periféricos.

**Fase medición** - es un proceso de determinar los valores de **los registros de configuración de la plataforma** (PCR), que son registros de solo escritura y se utilizan para validar los componentes al comparar los valores almacenados con los esperados. Los pasos incluyen

1 – Todos las meadiciones de PCRs son reseteados durante el reseteo o encendido de la plataforma;

2 – Se calcula la medidada del firmware con el algoritmo SHA-1 y se la almacena en PCR0;

3 – Para cada controlador del sistema




- Se calcula la medidada del controlador con el SHA-1;
- Se concatena la medidada calculada con la almacenada en PCR0

$$PCR0 = H(PCR0 \,||\, data),$$

donde H() es la la función SHA-1, || es la función de concatenavión y data es el nuevo bloque de datos. Si el bloque de datos es menos de 20 bytes, no se codifica

4 – Repita el paso 3 con todos los controladores y servicios del sistema.

El resultado final es una medida específica del estado del sistema.

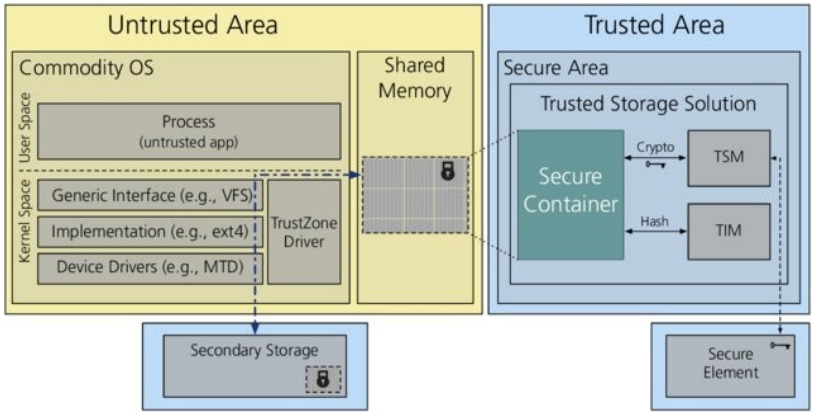
# Seguridad y protección

## *Trusted Computing – Almacenamiento seguro*

La implementación del almacenamiento seguro requiere una **clave confidencial específica del dispositivo** a la que solo se puede acceder mediante un código autorizado. Dicha clave puede inicializarse durante la fabricación y almacenarse en un área de memoria protegida en el chip del procesador.

Además se usan implementaciones confiables de **mecanismos criptográficos** y **una memoria no volátil** (por ejemplo, un contador monotónico) que almacene su estado de las versiones (protección de reversiones).

Los mecanismos de almacenamiento seguro combinados con algoritmos criptográficos **se utilizan para el REE**, con la garantía de que **las claves criptográficas nunca salen del hardware de la plataforma**.



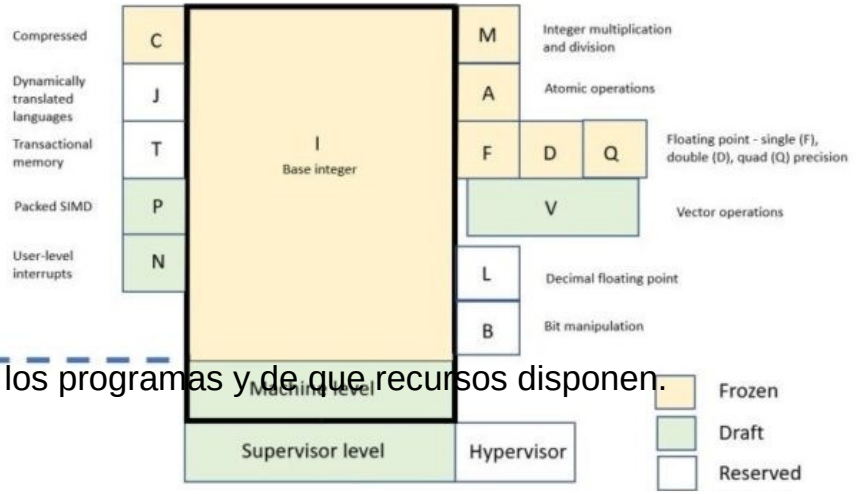

Si bien las operaciones criptográficas comunes son suficientes para muchos servicios, sin embargo ciertas aplicaciones del REE pueden requir la ejecución de algoritmos específicos de forma aislada del OS y de las aplicaciones (ejecución aislada).

Para ampliar el mecanismo de almacenamiento seguro a la ejecución aislada de código arbitrario, la configuración del hardware del dispositivo debe proporcionar una interfaz (API TEE) a través de la cual se pueda cargar el código ejecutable (código TEE) para su ejecución utilizando la memoria volátil protegida.

Un certificado de código TEE que contiene un hash del código TEE y está firmado con respecto a la raíz de confianza del dispositivo, puede autorizar la ejecución del código dentro del TEE y autorizar el código TEE para acceder a la clave del dispositivo. Además, el acceso que tiene cualquier código TEE a la clave del dispositivo puede controlarse en función del estado de la plataforma que se midió y guardó en una memoria volátil protegida por integridad durante un proceso de arranque autenticado.

# Seguridad por separación

*RISC-V – Modos de operación*




El RISC-V tiene cuatro niveles de privilegios (en orden creciente de capacidades) denominados modos operación: usuario, hipervisor, supervisor y máquina.

El procesador solo puede operar en un modo de operación.

Los modos de operación definen que es lo que pueden hacer los programas y de que recursos disponen.

Los modos tienen las siguientes carcaterísticas

**Modo usuario** (U-mode) – es el modo de operación con nivel más bajo. Se utiliza solo para ejecutar los procesos de los usuarios, no gestiona la memoria física y registros de estado y configuracion (CSRs);

**Modo hipervisor** (HS-mode) – virtualiza la arquitectura de supervisor para ejecutar sistemas operativos sobre un hipervisor. Este modo agrega otra etapa de traducción de direcciones para virtualizar la memoria y los subsistemas de E/S asignados un sistema operativo invitado. El modo HS actúa igual que el modo S pero con instrucciones y CSR adicionales que controlan la nueva etapa de traducción de direcciones. Un sistema operativo o hipervisor que se ejecuta en modo HS utiliza los CSR del supervisor para interactuar con los subsistemas de excepciones, interrupciones y traducción de direcciones.

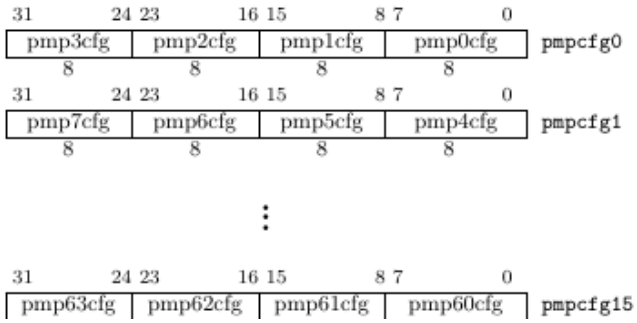
**Modo supervisor** (S-mode) – restringe las interacciones con el hardware para permitir una virtualización limpia. Ciertas facilidades a nivel de supervisor, incluidas las solicitudes de interrupciones de temporizador e interprocesador, son proporcionadas por mecanismos específicos de implementación.

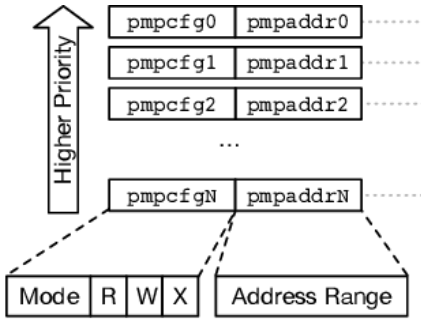
**Modo máquina** (M-mode) – se utiliza para el acceso y gestión de bajo nivel de una plataforma de hardware y es el primer modo que se ingresa al reiniciar. Este modo se puede usar para implementar funciones que son demasiado difíciles de implementar en el hardwarey tiene la prioridad más elevada, gestionando las interrupiciones a menos que el gestor de interrupciones del usuario (N-mode) este implementado.

# Seguridad por separación

*RISC-V – Aislamiento de la memoria*

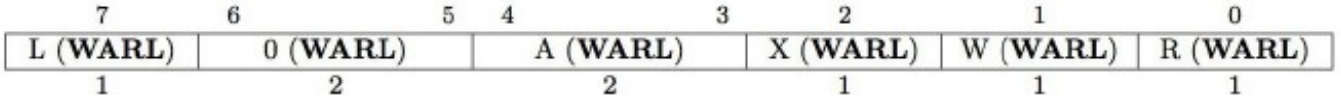
La **unidad de protección de memoria física** (PMP) proporciona un registro de configuración por cada hilo (hart) para **especificar los privilegios de acceso a la memoria física** (lectura, escritura y ejecución) de cada región de memoria física y un registro para **definir dicha region**.




**Solo son accesibles en modo máquina** y están localizados dentro de los CSR de dicho modo.

Los registros de configuración son registros 8 bits que especifican los privilegios de acceso a la memoria física: lectura (R), escritura (W) y ejecución (X) y el modo de direccionamiento (A) de cada región de memoria, así como y el bloqueo de escritura del registro (L)




Si L = 1 la PMP está bloqueada, es decir escrituras en el registro de configuración y los registros de direcciones asociados se ignoran. Las entradas de PMP bloqueadas permanecen bloqueadas hasta que se restablece el hilo.

Si no esta bloquado, se puede **intercambiar dinámicamente la configuración** de PMP para ejecutar diferentes contextos de seguridad en cada hilo.

# Seguridad por separación

*RISC-V – Aislamiento de la memoria*

El campo A en el registro de configuración de una entrada PMP codifica el modo de coincidencia de direcciones del registro de direcciones PMP asociado.

| A | Name | Description |
|---|---|---|
| 0 | OFF | Null region (disabled) |
| 1 | TOR | Top of range |
| 2 | NA4 | Naturally aligned four-byte region |
| 3 | NAPOT | Naturally aligned power-of-two region, ≥8 bytes |

Se admiten otros dos modos de direccionamientos:

* - **Regiones con tamaño de potencia de 2** alineadas naturalmente (NAPOT), cuyos bits de orden inferior del registro de dirección asociado se usa para codificar el tamaño de la region

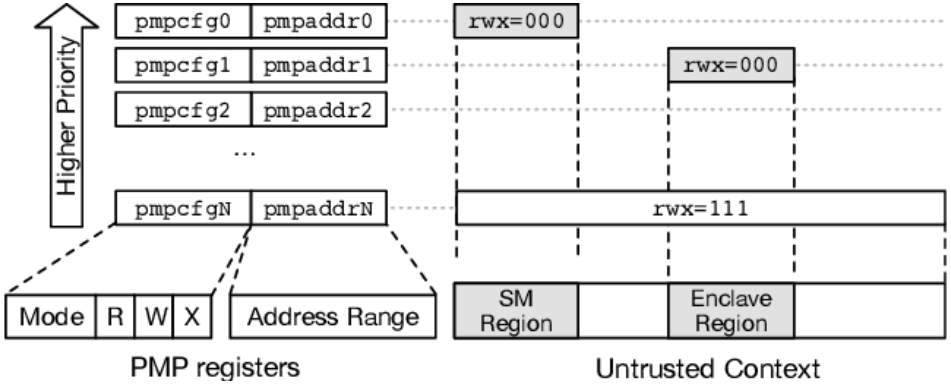
| pmpaddr | pmpcfg.A | Match type and size |
|---|---|---|
| yyyy...yyyy | NA4 | 4-byte NAPOT range |
| yyyy...yyy0 | NAPOT | 8-byte NAPOT range |
| yyyy...yy01 | NAPOT | 16-byte NAPOT range |
| yyyy...y011 | NAPOT | 32-byte NAPOT range |
| ... | ... | ... |
| yy01...1111 | NAPOT | $2^{XLEN}$-byte NAPOT range |
| y011...1111 | NAPOT | $2^{XLEN+1}$-byte NAPOT range |
| 0111...1111 | NAPOT | $2^{XLEN+2}$-byte NAPOT range |
| 1111...1111 | NAPOT | *Reserved* |

* - **Límite superior de un rango arbitrario** (TOR), donde los registros de direcciones asociados al PMP y el anterior forman la parte superior e inferior del rango de direcciones de la region.

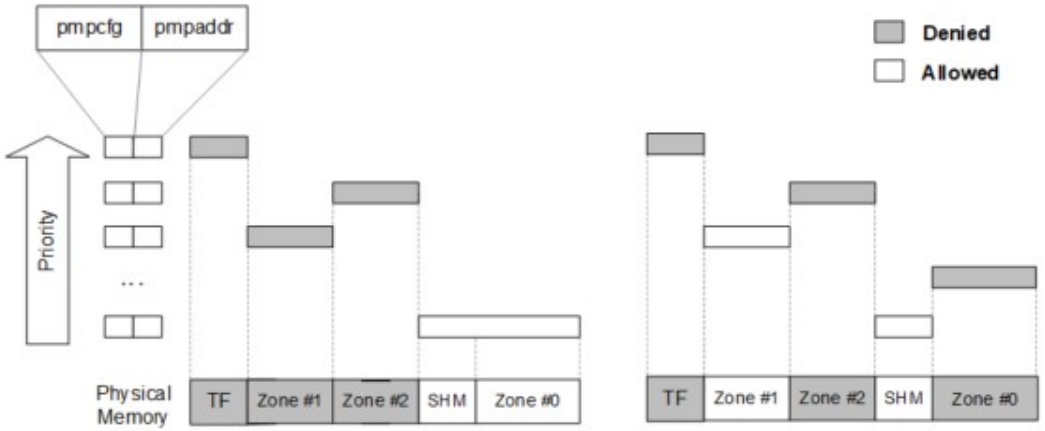
# Seguridad por separación

*RISC-V – Aislamiento de la memoria*

La unidad de protección de memoria física proporciona un mecanismo para especificar los privilegios de acceso a una región de la memoria física para aislarla del acceso de otro hilos.




Además, permite el uso compartido de una región de memoria entre varias zonas al mismo tiempo. Los permisos de acceso de la memoria requerida se escriben en la configuración de PMP de cada zona, y luego serán actualizados al cambiar de zona.



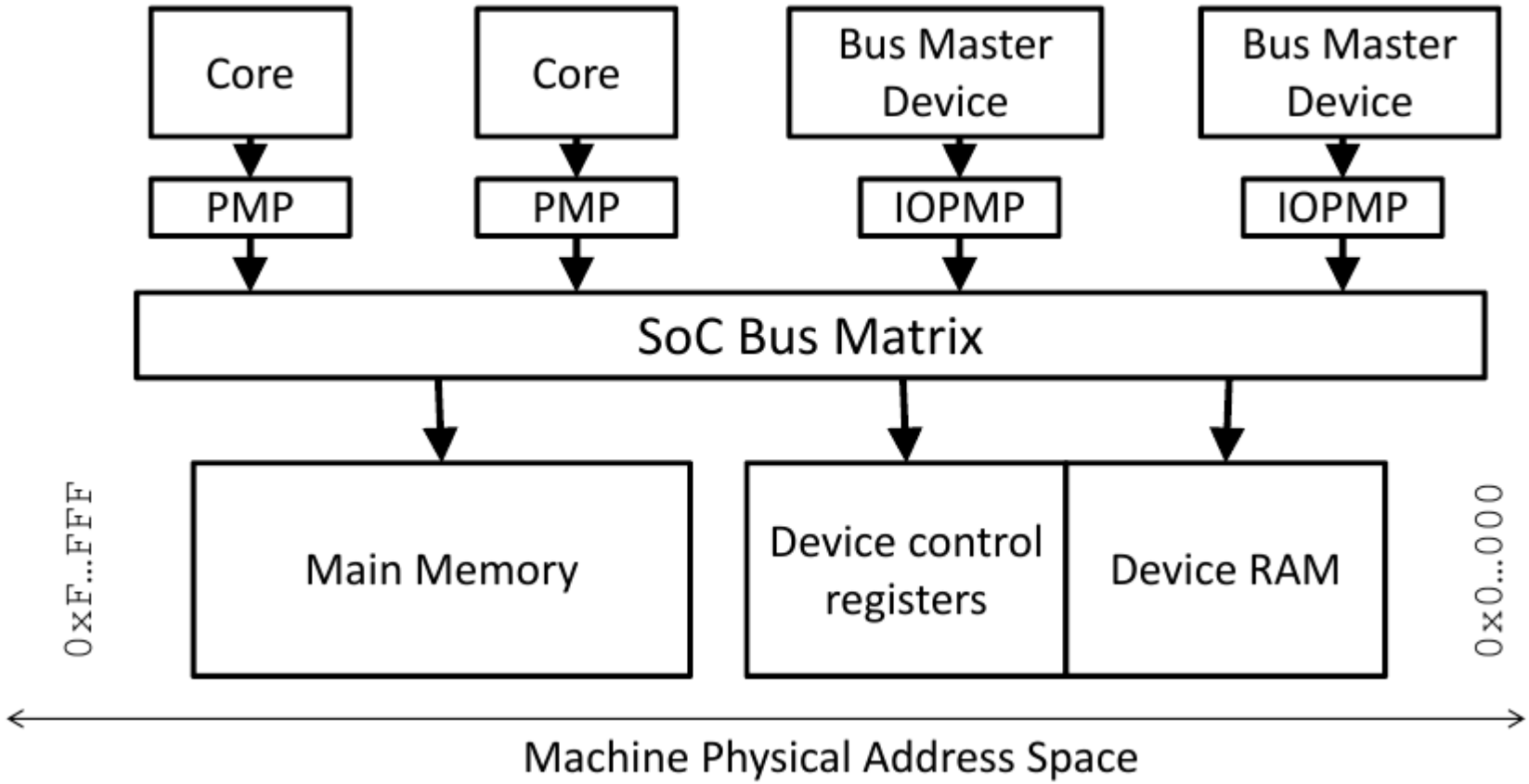

Otra característica destacada es que refuerza los permisos estrictos en modo M de una región de memoria específica, lo que reduce la exposición a ataques.

Esto también se conoce como prevención de acceso a la memoria del supervisor y prevención de ejecución de la memoria del supervisor.

**SHM indica el espacio de memoria compartida.**

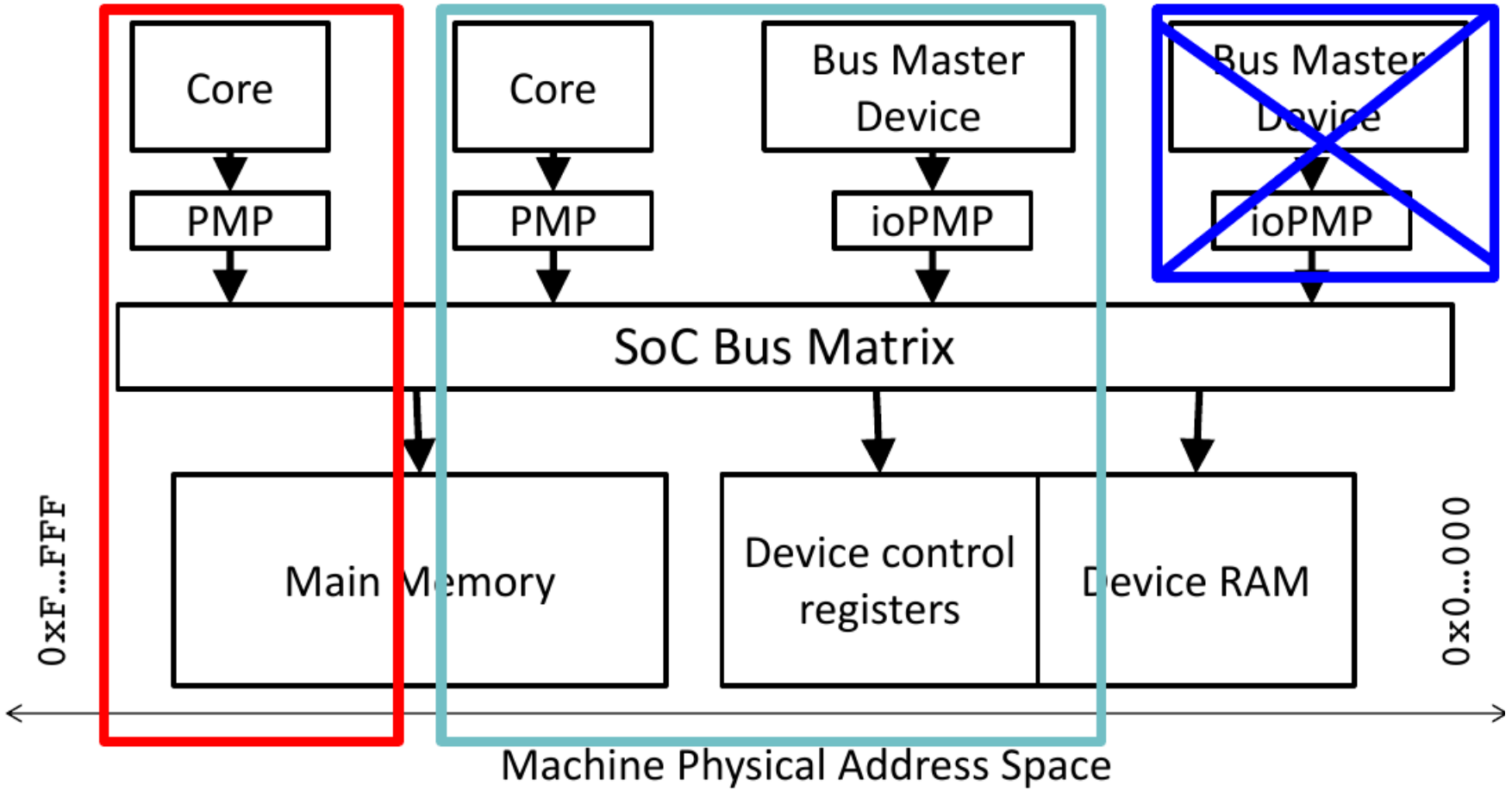
# Seguridad por separación

*RISC-V – Seguridad multizona*



# Seguridad por separación

*RISC-V – Seguridad multizona*



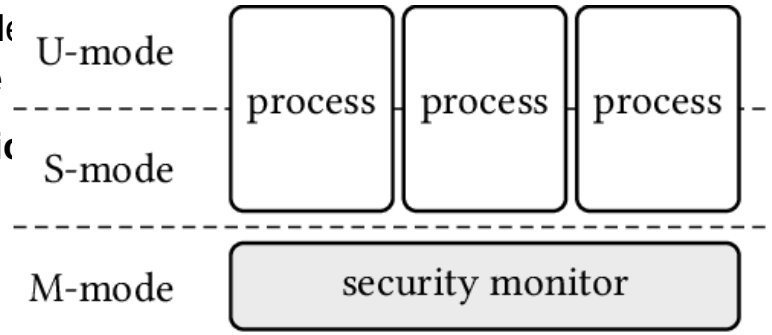
# Seguridad por separación

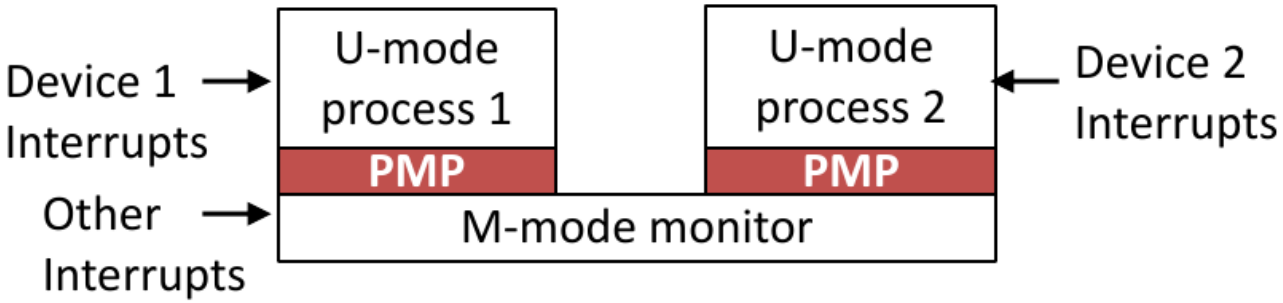
*RISC-V – Seguridad multizona para sistemas embebidos*

Cuando los códigos de una aplicación se implementan en un sistema embebido utilizando RISC-V solo se de dos modos de operación:

- El modo M se usa para ejecutar el arranque seguro y un monitor de ejecución, los cuales garantizan integridad de la ejecución;
- El modo U se utiliza para ejecutar las aplicaciones;
- Los accesos a la memoria en modo U son controlados por unidad de Protección de memoria física (PMP e ioPMP);
- El manejo de interrupciones se puede delegar al código de de interrupciones a nivel de usuario (Extensión N), sino se

  **Esta configuración proporciona un número arbitrari**



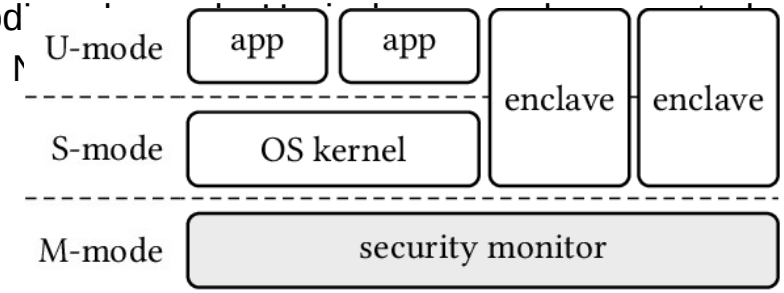



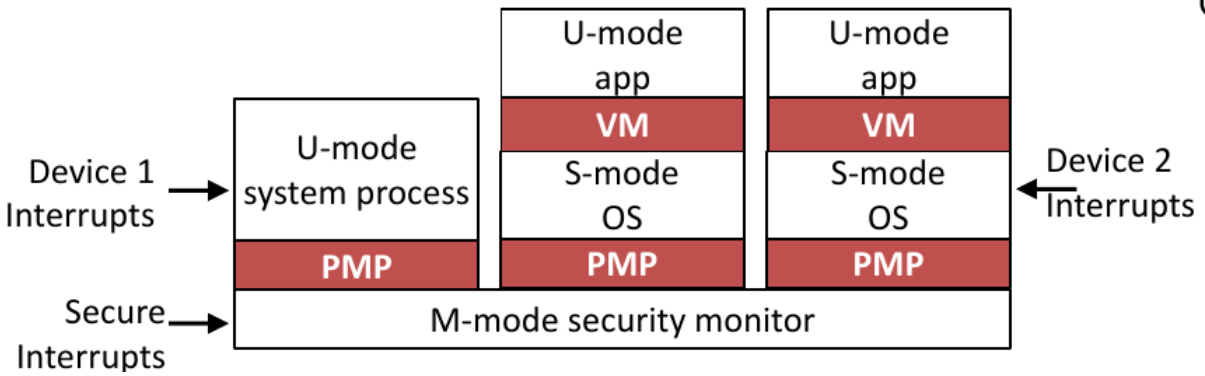
# Seguridad por separación

*RISC-V – Seguridad multizona para sistemas embebidos*

Cuando los códigos de una aplicación se implementan en un sistema embebido utilizando RISC-V disponen de todos los modos de operación:

- El modo M se usa para ejecutar el arranque seguro y un monitor de tiempo de ejecución, los cuales garantizan integridad de la ejecución;
- El modo S se utiliza para ejecutar el sistema operativo;
- El modo U se utiliza para ejecutar las aplicaciones;
- Los accesos a la memoria en modo S son controlados por unidad de Protección de memoria física (PMP e ioPMP) y en modo U son controlados por la gestión de memoria virtual;
- El manejo de interrupciones se puede delegar al cód[illegible] gestión de interrupciones a nivel de usuario (Extensión N[illegible]

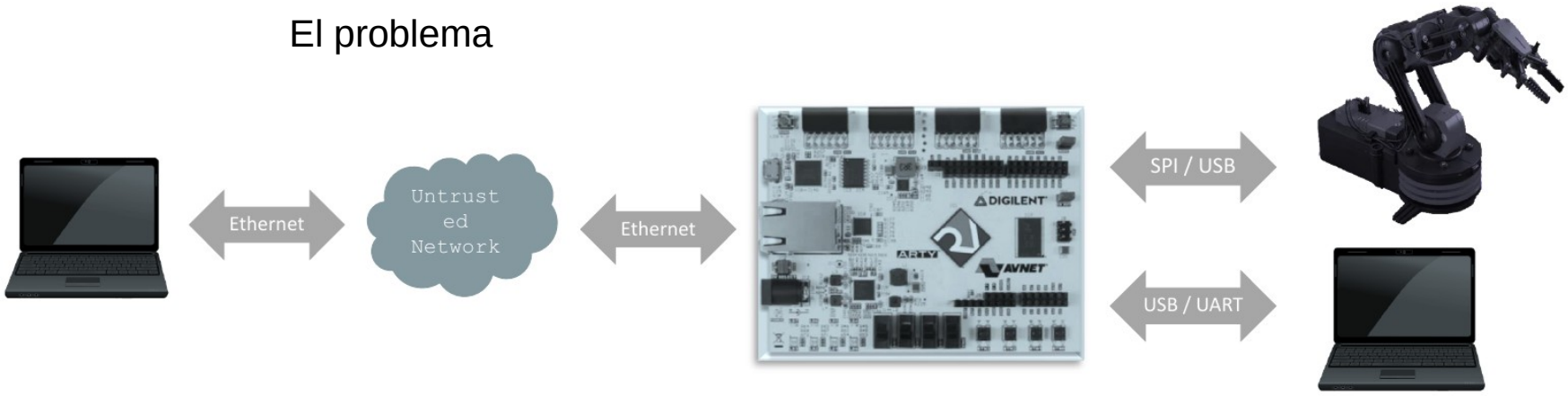


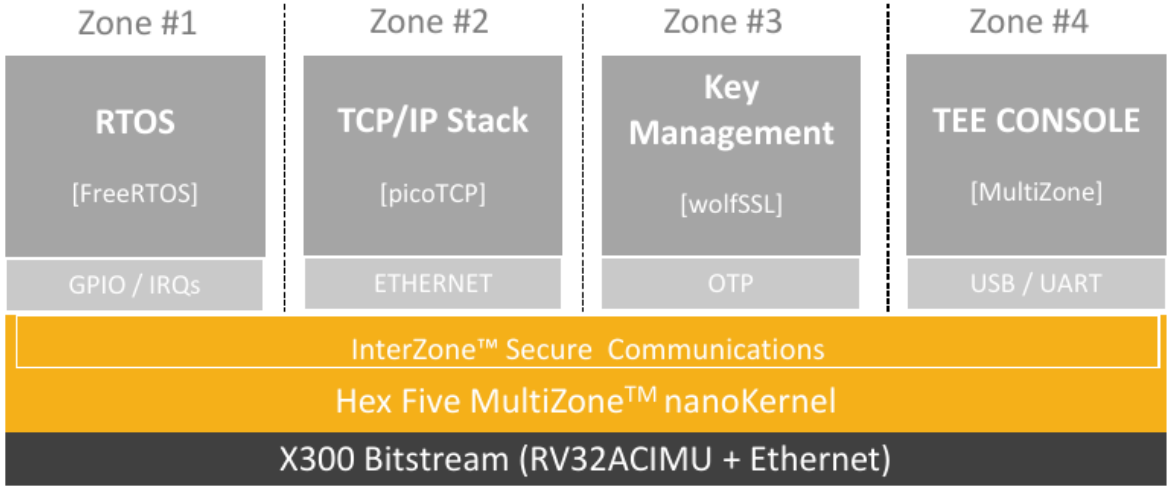
# Seguridad por separación

*RISC-V – Seguridad multizona*

El problema




La organización del software



# Seguridad por separación

*RISC-V – Seguridad multizona*

Múltiples enclaves definidos estáticamente: ram, rom, entrada/salida, interrupciones;

Mensajería segura sin memoria compartida: búferes seguros para Linux ;

Controladores de interrupciones seguros asignados a enclaves y ejecutados en modo U

Emulación de instrucciones privilegiadas y Traps, temporizadores por sofware, arranque seguro