Pointer Authentication Codes (PACs)

Alcántara Hoyos José Carlos
Pérez Candanoza Roel Alejandro

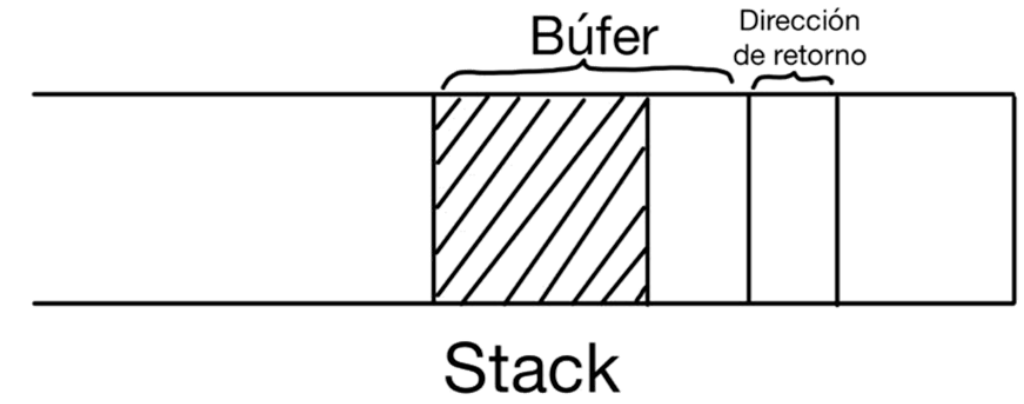
Sistemas Operativos

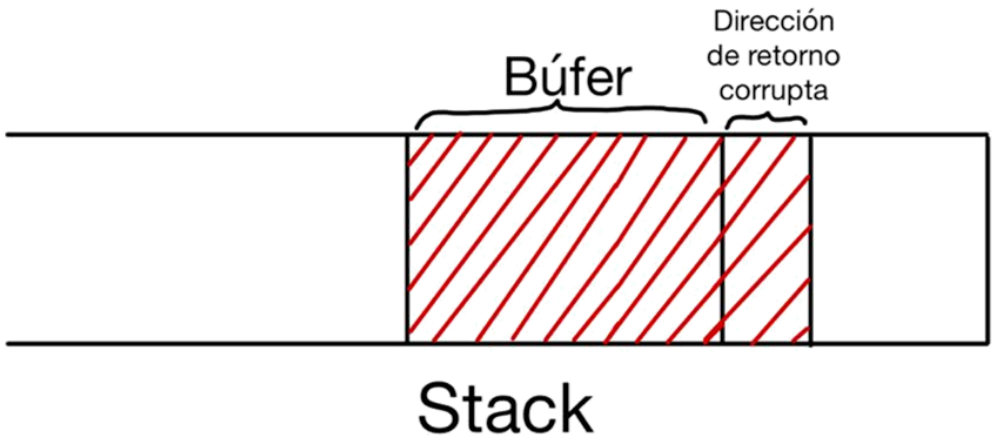
Problema

En el campo de seguridad informática, un problema común es la vulnerabilidad por corrupción de memoria.

- Return Oriented Programming
- Jump Oriented Programming
- Buffer Overflow

Buffer Overflow





Soluciones

- Almacenar datos sensibles y apuntadores en memoria de solo lectura.
- Verificar la integridad de un apuntador antes de utilizarlo.
  - Software Stack Protection (SSP)
- Aleatorizar algún aspecto de la administración de memoria.
  - Adress Space Layout Randomization (ASLR)

Advanced RISC Machine

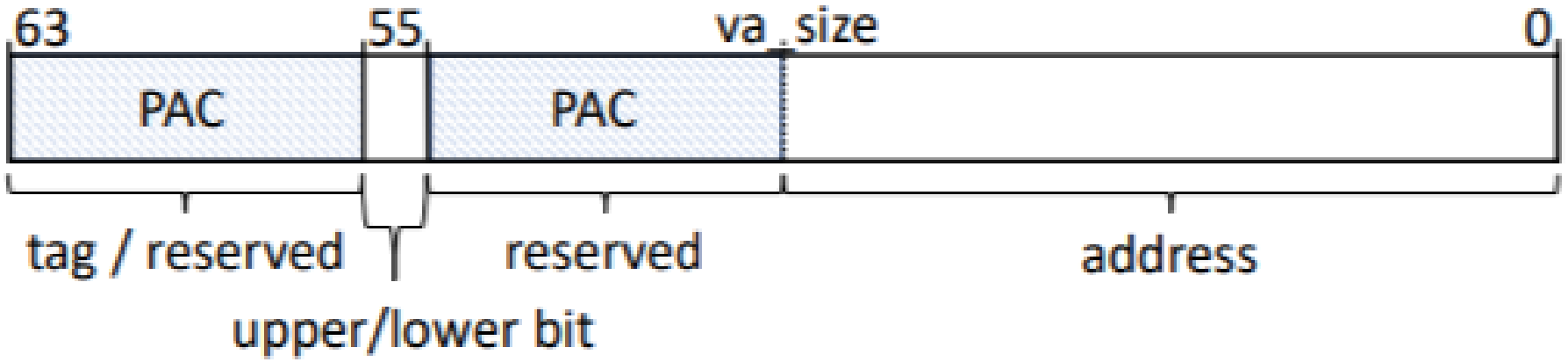
Pointer Authentication

Qualcomm añadió la funcionalidad llamada *Pointer Authentication* desde la versión ARM v8.3 con el objetivo de detectar apuntadores creados por alguna entidad externa a la ejecución.

Se agrega una firma de cifrado al valor de los apuntadores y este valor es validado antes de ser usado.

Estructura del PAC en el apuntador

El PAC se guarda en la parte no utilizada del apuntador (bits de extensión). El tamaño del PAC puede ser de entre 3 y 31 bits.



Autenticación vs. Cifrado

- Cifrado: Se oculta un conjunto de datos. Solo se puede leer si se tiene la clave de descifrado.
  - ❑ No permite la predicción de saltos
  - ❑ Dificulta la depuración

- Autenticación: Se valida la integridad de un conjunto de datos (que no haya sido alterado o corrompido).
  - ✓ Permite distinguir un error de una corrupción

Funcionamiento

Conjuntos de instrucciones:

- PAC*: Calcula e inserta el PAC en el apuntador.
- AUT*: Valida el PAC en el apuntador antes de usarlo.

Después de hacer la verificación con AUT*, si el apuntador es válido se remueve la firma del apuntador; en caso contrario se modifica el apuntador a una dirección inválida y de este modo se lanza una excepción.

Un apuntador con PAC no se puede desreferenciar; los bits de extensión de signo son necesarios para que se le considere una dirección de memoria válida.

Protección de stack utilizando PACs

	No stack protection	With Pointer Authentication

Function Prologue	`SUB sp, sp, #0x40`
	`STP x29, x30, [sp,#0x30]`
	`ADD x29, sp, #0x30`
	…
		`PACIASP`
		`SUB sp, sp, #0x40`
		`STP x29, x30, [sp,#0x30]`
		`ADD x29, sp, #0x30`
		…

Function Epilogue	…
	`LDP x29,x30,[sp,#0x30]`
	`ADD sp,sp,#0x40`
	`RET`
		…
		`LDP x29,x30,[sp,#0x30]`
		`ADD sp,sp,#0x40`
		`AUTIASP`
		`RET`

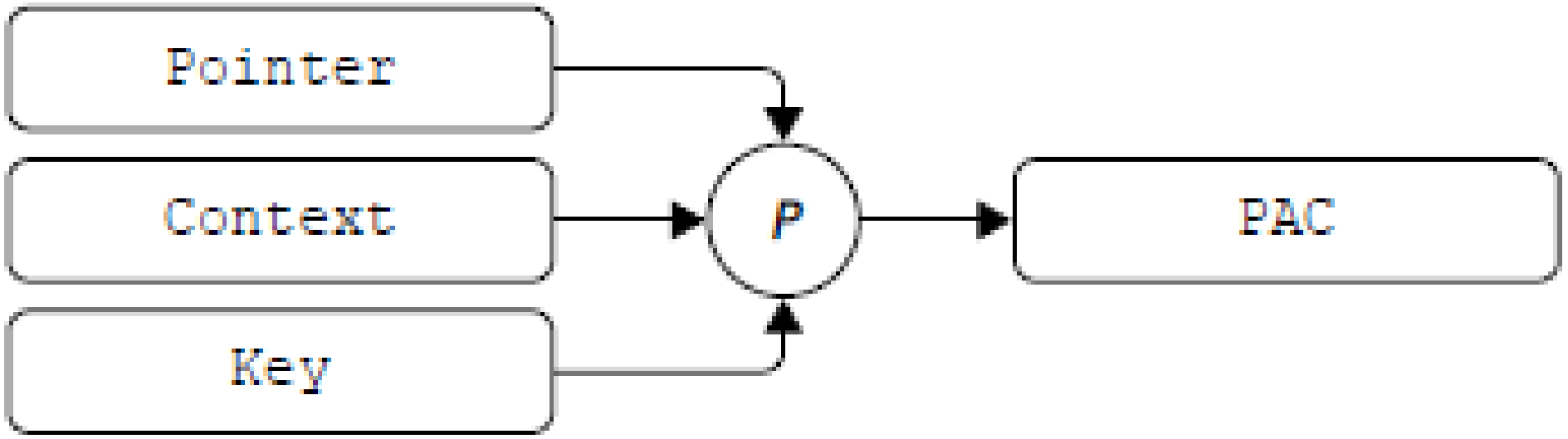
Cifrado

Para generar el código de autenticación, se desarrolló QARMA, una familia de unidades de cifrado por bloque, el cual es usado por las instrucciones PAC* y AUT*. El algoritmo de cifrado se caracteriza por ser fuerte, aún cuando es truncado, y ágil.

El código de autentificación es calculado con tres valores:

- El apuntador
- La clave secreta del PAC
- Otro valor del entorno. Por ejemplo el apuntador actual del stack.

Cifrado



Cifrado

ARMv8.3 define cinco claves de autenticación:

- Dos para uso de apuntadores de funciones.
- Dos para uso de apuntadores de datos.
- Una de uso genérico.

Cuando un proceso es creado, el kernel genera una clave asociada al contexto del proceso, la cuál es usada para validar los códigos. Por seguridad, el proceso no puede leer la clave por si solo.

Propiedades

La autenticación de apuntadores no es sensible tanto a lecturas como escrituras arbitrarias de memoria.

- Si un atacante tuviera lectura arbitraria de memoria, obtener códigos de apuntadores con firma no ayudaría a falsificar códigos, ya que el algoritmo de cifrado es fuerte.
- Si un atacante tuviera escritura arbitraria de memoria, modificar un apuntador causaría fallos, dificultando los ataques de fuerza bruta.

iOS y Jailbreak

El sistema operativo de los iPhone y iPad se llama iOS, el cual es un sistema cerrado al no dejar que el cliente acceda al *usuario administrador* del sistema.

Cuando se remueve este bloqueo, se dice que se ha llevado a cabo un *jailbreak* . Esto se hace con una modificación de iOS.

Los PACs se presentan como un problema para los jailbreak developers ya que estos procedimientos actualmente utilizaban buffer overflows , integer overflows, y otras técnicas de modificación de memoria.

Debilidades del Pointer Authentication en iOS

Ataque mas sencillo es traer las claves de cifrado a la memoria del kernel y leerlas desde ahí.

Al parecer los dispositivos iOS recientes parecen no tener los niveles EL2, EL3, por lo tanto el kernel es el único que administra los PAC, esto facilita el ataque.

Dificultades:

- Se requiere del algoritmo de cifrado empleado.
- Se necesitan permisos de lectura de memoria del kernel.

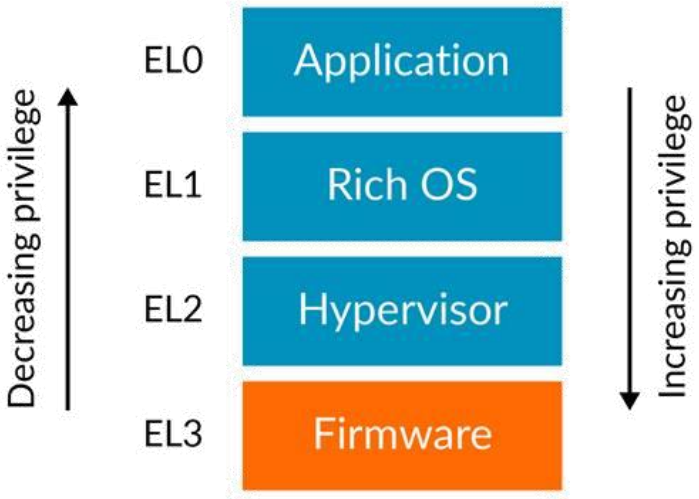


Diagrama de niveles de privilegio

Referencias

- ARMdeveloper.(s.f.). Privilege and exception levels. Descargado de https://developer.arm.com/architectures/learn-the-architecture/exception-model/privilege-and-exception-levels
- *Jailbreak: romper las limitaciones del fabricante en dispositivos iOS*. (2020). Descargado de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/jailbreak-ios/
- Liljestrand, H., Nyman, T., Wang, K., Perez, C. C., Ekberg, J.-E., y Asokan, N. (2019). *PAC it up: Towards pointer integrity using arm pointer authentication*. Descargado de https://arxiv.org/pdf/1811.09189.pdf
- Qualcomm. (2017). *Pointer authentication on ARMv8.3.* Descargado de https://www.qualcomm.com/media/documents/files/whitepaper-pointer-authentication-on-armv8-3.pdf
- R., I. (2018). *What do pointer authentication codes mean for iOS jailbreaking?* Descargado de https://ivrodriguez.com/pointer-authentication-on-armv8-3
- R., I. (2019). *Examining pointer authentication on the iPhone Xs*. Descargado de https://googleprojectzero.blogspot.com/2019/02/examining-pointer-authentication-on.html
- Rutland, M. (2017). *Pointer authentication in aarch64 Linux*. Descargado de https://www.kernel.org/doc/Documentation/arm64/pointer-authentication.rst5