



# Bruteforcing Movable.SED KeyY

Paloma Candocia y Daniel Pedrosa

01

# Nuestro objetivo



Lleno de ritmo.

Ilustraciones extraídas de <https://taiko.namco-ch.net/taiko/en/>





# Region-Lock

Queremos jugar al famoso videojuego “Taiko no Tatsujin”  
en nuestra consola 3DS.


Únicamente se vende en Japón, y nuestra consola impide  
jugar juegos de una región diferente al de la consola.



**02**

**Para llegar hasta  
nuestro objetivo...**

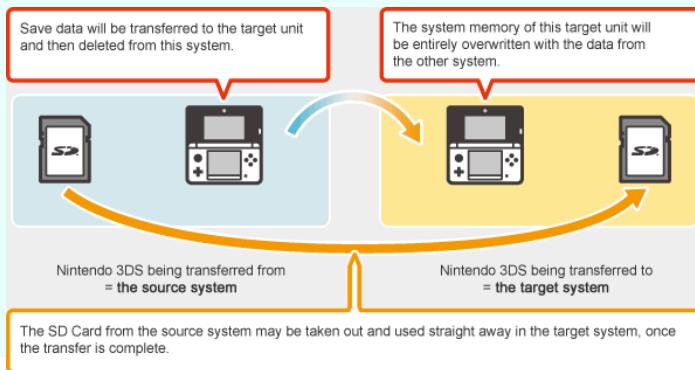
Tendremos que poder ejecutar código  
arbitrario en nuestra consola



# 03

## El exploit

Summary	Description	Successful exploitation result	Fixed in FIRM system version	Last FIRM system version this flaw was checked for	Timeframe this was discovered	Public disclosure timeframe	Discovered by
DSiWare_Exports CTCert verification	Just like DSI originally did, 3DS verifies the APCert for DSiWare on SD with the CTCert also in the DSiWare .bin. On DSI this was fixed with system-version 1.4.2 by verifying with the actual console-unique cert instead(stored in NAND), while on 3DS it's still not(?) fixed.  On 3DS however this is rather useless, due to the entire DSiWare .bin being encrypted with the console-unique movable.sed keyY.	When the movable.sed keyY for the target 3DS is known and the target 3DS CTCert private-key is unknown, importing of modified DSiWare SD .bin files.	Unknown, probably none.	?	April 2013		Yellows8



De izquierda a derecha, extraído de <https://zoogie.github.io/web/34%E2%85%95c3/#/4> y <https://www.nintendo.co.uk/Support/Nintendo-3DS-2DS/Transfer-between-Nintendo-3DS-systems-including-Nintendo-3DS-XL-/Understanding-the-transfer-process/Understanding-the-transfer-process-242847.html>






# Movable.SED

Para evitar tener que re-encryptar todos los datos de una consola al realizar una transferencia de datos de una consola antigua a una nueva, Nintendo utiliza una llave única por cada consola para este propósito.

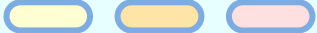


Conociendo esta clave podemos aprovechar el *exploit* anterior.







u64 (Local Friend Code Seed)	u64 (Completely Random)




# Movable.SED Key.Y

Se descubre que los primeros 64 bits se pueden extraer de espacios de memoria no protegidos.

Los últimos 64 bits... Quedaría sacarlos por fuerza bruta. Pero no tenemos un superordenador y no queremos esperar varias décadas.





u64 (Local Friend Code Seed)	u32 (Completely Random)	u32 ( <u>Not</u> Random)
---------------------------------	----------------------------	-----------------------------




# Movable.SED Key.Y

¡Descubrimos que los últimos 64 bits no son completamente aleatorios!

Existe una gran correlación entre el Local Friend Code Seed y los últimos 32 bits.

Ahora Sí es factible encontrar la clave secreta mediante fuerza bruta.






A stylized illustration of a white computer mouse with a blue cord, positioned on a light blue surface. A small white star with a blue outline is located above the mouse. The background is a light pink color with a blue border.

**04**

# Tiempo promedio para encontrar la clave por bf.

Porque de elegantes no nos falta.

A yellow star with a blue outline and a white shadow, located in the bottom right corner of the image.



# En una CPU


La mejor implementación consigue un tiempo de 20 días  
aproximados para  $2^{42}$  combinaciones...

No son tantas iteraciones para un ordenador, pero el algoritmo  
de encriptación es computacionalmente costoso.







# ¡Aceleramos con OpenCL!

Ahora, gracias a que utilizamos los circuitos específicos de nuestra GPU para nuestros propósitos y a su gran grado de paralelización SIMD, se consigue un tiempo de descryptado de 1 a 5 minutos.



Podemos encontrar más información en <https://github.com/Jimmy-Z/bfCL>



05

Fin.



Do Don Ka Katsu DON



# Basado en

- <https://zoogie.github.io/web/34%E2%85%95c3>