

Search ...

Home (/joomla/index.php)  
About Us (/joomla/index.php/about-us)  
News (/joomla/index.php/news)  
Contact Us (/joomla/index.php/contact-us)



## Noticias

### Details

Written by Joomla  
Category: Uncategorized (/joomla/index.php/2-uncategorised)  
Published: 04 January 2012  
Hits: 502



### **Informe del estado de la seguridad en Android (/joomla/index.php/news/3-article-1-title)**

Google ha publicado un informe ([https://static.googleusercontent.com/media/source.android.com/en/us/devices/tech/security/reports/Google\\_Android\\_Security\\_2014\\_Report.pdf](https://static.googleusercontent.com/media/source.android.com/en/us/devices/tech/security/reports/Google_Android_Security_2014_Report.pdf)) el que resume **el estado de seguridad de su plataforma móvil Android**.

El documento recoge un año de correcciones y mejoras del sistema operativo móvil en materia de seguridad. Implementación de nuevos mecanismos y parches que cierran agujeros de seguridad conocidos. El informe gana peso en un importante aspecto que Android tiene como asignatura pendiente: El Malware.

Vamos a hacer un repaso de los puntos destacados del informe, teniendo siempre en cuenta que no se trata del informe de un tercero independiente, por lo que las conclusiones aportadas en el siempre estarán contrapesadas con dosis apropiadas de optimismo dogmático.

Comentar que **las nuevas medidas de seguridad, adoptadas durante 2014, van orientadas a su uso en la versión 5.0 del sistema operativo**. 4.4 representa ya el pasado y es costumbre el borrón y cuenta nueva. Tabula rasa.

### SELinux

En primer lugar tenemos la activación por defecto del modo "enforced" para todos los dominios en SELinux. Hasta la versión 4.4, SELinux solo se aplicaba en este modo a los dominios asociados al sistema. Este modo es el que permite a SELinux funcionar denegando los accesos a recursos si no se cumplen los requisitos adecuados.

Hasta ahora solo los procesos del sistema se beneficiaban de esta medida, estableciendo una capa extra para obstaculizar la explotación o minimizar su impacto. A partir de Android 5.0 todos los procesos pasarán por el filtro de SELinux.

### Cifrado de disco

Hasta la versión 3.0 de Android, el cifrado de disco tomaba como clave el patrón o contraseña de desbloqueo de la pantalla de acceso. A partir de la versión 5.0 la contraseña del usuario es usada para obtener una clave derivada usando una implementación de 'scrypt'. Es dicha clave derivada la que se usará para cifrar el disco. Esto permite mejorar la resistencia frente a ataques de fuerza bruta sobre la clave de cifrado. Adicionalmente, en sistemas que dispongan del hardware adecuado, la clave se almacenará en un chip dedicado a almacenamiento de claves.

### Perfiles de usuarios

A partir de Android 5.0 se habilita en teléfonos inteligentes la capacidad para añadir perfiles y un modo invitado que no permite acceso a datos ni aplicaciones. Hasta ahora y desde la versión 4.2 solo los dispositivos Tablet permitían múltiples perfiles.

### Autenticación "mejorada"

De nuevo, solo en la versión 5.0, se permite el desbloqueo automático del terminal si se encuentra cerca de un dispositivo autorizado o se efectúa un reconocimiento positivo del rostro del usuario.

Tal y como es descrito no se entiende como esto es presentado como medida de seguridad. A pesar de los avances en identificación y autenticación biométrica esta tecnología suele usarse como segundo factor. Recordemos la evasión del lector de huellas del iPhone 5s (<http://unaaldia.hispasec.com/2013/09/el-grupo-chaos-computer-club-consigue.html>) o la de reconocimiento facial que ya se empleaba en la versión Jelly Bean del propio Android.

Respecto del desbloqueo por proximidad radio (NFC, Bluetooth...) entendemos que se trata de una funcionalidad que **aumenta la usabilidad por comodidad del equipo pero naturalmente la moneda de cambio habitual suele ser un detrimiento de la seguridad**. Un desbloqueo automático suena mal, muy mal, desde el punto de vista de la seguridad. ¿Necesitas acceder al teléfono de alguien? ¿Está en una reunión? ¿Se ha dejado el teléfono y las llaves del coche sobre la mesa? ¿Necesitamos poner otra interrogativa a modo de pregunta retórica?

## Parches

En total, según el equipo de seguridad de Android, se han corregido 30 parches de gravedad alta, 41 moderados y 8 de gravedad baja. Los parches de seguridad no son publicados directamente en el repositorio de código público. En vez de ello, existe una rama con acceso a esta clase de parches para los fabricantes, con el fin de que corrijan las vulnerabilidades en sus propias versiones de Android antes de que sean reveladas.

Llama la atención un dato. Solo han observado una explotación "significante" de una vulnerabilidad: CVE-2014-3153. Usada para elevar privilegios en local o lo que es lo mismo, usada para rootear el dispositivo.

De la misma forma comentan que no han observado explotación de vulnerabilidades asociadas a SSL y que lo poco que han visto "parece" estar vinculado a labores de investigación.

Sobre FakelD, la vulnerabilidad presentada en la conferencia BlackHat USA de 2014, que permitía instalar una aplicación sin necesidad de que el usuario concediese permisos, indican que han detectado una aplicación subida a Google Play y otras 258 procedentes de fuentes de terceros que hiciesen uso de esta vulnerabilidad.

## Android y los fabricantes

La relación entre fabricantes y Android (Google) es de una tensa calma. Por un lado los fabricantes introducen modificaciones y aplicaciones propias que les permite aportar diferenciación entre la competencia. Por otro lado, ese dechado de buena voluntad e individualización abre la puerta a nuevos y únicos defectos que son aprovechados como vectores por los atacantes.

De manera característica los fabricantes demoran la publicación de parches, esto se traduce en una ventana de exposición larga y en cierta medida una percepción negativa de Android en su conjunto para el usuario final. Esto mismo no ocurre en la misma medida con los dispositivos administrados por Google, que suelen experimentar actualizaciones de seguridad más frecuentes. Por supuesto, siempre en el caso de que la versión de Android no esté condenada al ostracismo.

El equipo de seguridad de Android mantiene una observación de este tipo de vulnerabilidades y sostiene que la mejora de operación de SELinux permitirá contener los daños del impacto en caso de explotación. Un palo en medio de mar montañosa, sin embargo no pueden hacer mucho más si el fabricante tiene el canal de actualización del sistema por el mango.

## Verify Apps



([http://4.bp.blogspot.com/-n6bJ\\_AVCtkA/VSLBcL35KWI/AAAAAAAADG0/KzjfEjsxkFk/s1600/VerifyApps.PNG](http://4.bp.blogspot.com/-n6bJ_AVCtkA/VSLBcL35KWI/AAAAAAAADG0/KzjfEjsxkFk/s1600/VerifyApps.PNG))

Android incluía un sistema para detener y avisar al usuario de aplicaciones maliciosas. Una suerte de antivirus básico. Su ámbito se circunscribía en las aplicaciones descargadas a través de Google Play. En abril del pasado 2014 se anunció que ese ámbito estaba siendo ampliado a todas las aplicaciones, incluidas las de otros mercados. El usuario también tiene la opción de subir aplicaciones fuera de Google Play a Google para su análisis.

## Actualización de componentes fuera del OTA

Interesante. Google ha habilitado un canal de actualización para que ciertos componentes pueden ser parcheados sin tener que esperar a una actualización completa del sistema a través de OTA (Over The Air). De momento solo puede efectuarse sobre una versión de SSL mantenida por Google y el infame componente WebView que podrán ser actualizados sin necesidad de esperar a un ciclo de publicación de actualizaciones. Por supuesto, a partir de Android 5.0.

Google asume que ciertos componentes pueden y deben ser tratados con prioridad. Además le toman la iniciativa a los fabricantes. Los usuarios podrían ver como ciertas vulnerabilidades son corregidas sin necesidad de esperar meses o indefinidamente a que el fabricante publique un parche.

## Aviso a desarrolladores

Desde el pasado mes de julio, las aplicaciones subidas a Google Play son examinadas en busca de vulnerabilidades conocidas. **Evidentemente es un proceso automatizado**, pero se agradece que los desarrolladores sean avisados de la supuesta presencia de errores de seguridad conocidos en sus aplicaciones a fin de que puedan corregirlos.

**Decir proceso automatizado y búsqueda de vulnerabilidades en el mismo párrafo es lo mismo que hablar de falsos positivos.** Naturalmente se corre el riesgo de alarma sobre algo que no existe pero es siempre preferible comprobar algo que darlo por bueno.

## Estadísticas de infección

Los datos proceden de su servicio, el comentado "Verify Apps". Las aplicaciones de Google Play son escaneadas cuando son subidas y periódicamente durante el tiempo que están alojadas. Con ello, Google cuenta con datos directos de las aplicaciones alojadas en Google Play y aquellas que están instaladas en los terminales Android.

Según se puede leer en el informe, Google sostiene que menos de un uno por ciento de dispositivos (entiéndase, con Verify Apps activo) contiene una PHA (Potentially Harmful Application) instalada. Es más, reduce ese porcentaje al 0,15% si se trata de dispositivos que instalan aplicaciones exclusivamente desde Google Play.

Destaca la tasa por encima de la media de dispositivos Android en Rusia y China. Explicable en parte por la no presencia (Google Play fue prohibido en China) o preferencia por mercados de terceros en el ámbito nacional.

El problema quizás no es ese "menos del 1%" en su imagen y sí ese "más de 90%" comparado con otras plataformas móviles.

## Tipos de malware



(<http://4.bp.blogspot.com/-Zcq209y06H8/VSLCFKKSSxI/AAAAAAAADG8/qTu9kEp0L7s/s1600/RansomwareAndroid.PNG>)

Bajan los clásicos: spyware y los SMS senders. Mientras el malware tipo Ransomware es visto "rentable" a ojos de los desarrolladores de malware y comienza a despuntar progresivamente.

## Safety Net

Por último cierra el informe los datos relativos a Safety Net. Una suerte de hookeo de ciertas API marcadas como posible "uso por abuso". Dicho sistema comprueba cómo son llamadas estas funciones y reacciona si detecta un abuso.

Llama la atención el apartado de "*Hombre en el medio*". Desde Android 4.2 fue introducido el "Certificate pinning", a partir de la versión 4.4 el sistema avisa de aquellos certificados instalados en la CA del sistema y que son usados para efectuar capturas de comunicaciones cifradas en texto claro. Una ventanita que

venimos "padeciendo" quienes hacemos auditoría de aplicaciones cuando instalamos un certificado en la CA.

En la parte positiva se ven los esfuerzos y los frutos conseguidos por la inversión de seguridad. Más medidas de seguridad, menores tasas de infección. Sin embargo el **malware sigue creciendo en Android**, se siguen detectando más tipos de familias y técnicas de infección más avanzadas que permiten ir por debajo de la línea del radar de la detección automatizada o evadir ciertas medidas de seguridad.

Lo dicho anteriormente. Android reduce la tasa de infección a ojos de los datos expuestos por el informe de Google. Ese "1%" es cada vez menos uno por ciento. Sin embargo ese "más del 90%" (que evidentemente no dice) cuando se compara con el resto de sistemas móviles pesa mucho, muchísimo.

# Las tiendas físicas, objetivos de los cibercriminales



(<http://muyseguridad.net/wp-content/uploads/2015/05/POS.jpg>)

Los denominados **POS**, terminales de punto de venta (<http://www.muycanal.com/2015/05/26/malware-pos-tienda-fisica>), son un objetivo bastante sabroso para los cibercriminales. Con toda la información financiera de clientes y tienda, el **malware POS ha visto un punto clave** donde intentar robar datos de gran valor, [según informan en MuyCanal](#). (<http://www.muycanal.com/2015/05/26/malware-pos-tienda-fisica>)

El **pasado 2014** se reportaron **198 brechas de seguridad** en terminales de punto de venta. Una cifra se multiplicará por tres durante 2015, según las previsiones de [ABI Research](#) (<https://www.abiresearch.com/>). Un crecimiento ya se ha notado en la actividad por encontrar agujeros con malware POS.

Trend Micro comenta que en los **pasados 6 meses** se han encontrado **más variantes** de malware específicamente creados para las tiendas físicas que en varios años anteriores. Estos son los cinco más destacados:

## Alina y sus ediciones

Existente desde 2012, este malware POS ha ido creando diferentes versiones de sí mismo para conseguir esconderse. De hecho, el código moldeable para equipos con Windows [se vendía ya en octubre de 2013](#) por 2.000 dólares (<http://www.kernelmode.info/forum/viewtopic.php?f=16&t=1756&p=21100&hilit=alina+source#p21100>). Las últimas ediciones son Spark o Eagle, esta última lanzada en septiembre de 2014.

Aunque cada versión tiene sus propias funcionalidades, el malware actúa de forma más o menos similar instalando un archivo ejecutable y una carpeta en el sistema para después actuar. [Desde Nuix nos explican mejor](#) (<http://www.nuix.com/blog/alina-continues-spread-its-wings>) cómo funciona la versión Eagle.

## LogPOS

Nacido en 2014, este malware también posee sus reediciones durante este 2015. LogPOS aprovecha los agujeros de seguridad de la memoria para introducir código que abre todas las carpetas que contienen la información interesante para el criminal. [Aquí nos explican mejor](#) (<http://morphick.com/blog/2015/2/27/mailslot-pos>) el proceso de ejecución de este malware.

## FighterPOS, el temerario

El autor de FighterPOS es AlejandroV, uno de los cibercriminales más prolíficos. [Según nos cuentan desde Trend Micro](#) (<http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/wp-fighterpos.pdf>), este experto en encontrar agujeros de seguridad ha conseguido vender su panel de control por unos 18.000 Bitcoins, unos 5.250 dólares al cambio.

Este malware POS es uno de los más cruentos dedicados a los terminales de punto de venta. Ya el año pasado entró en los sistemas de 200 empresas en Brasil, México, Italia y Reino Unido recolectando más de 22.000 tarjetas de crédito en un solo mes.

## Punkey, la evolución

Este malware POS es una evolución de la familia NewPOStings. [Descubierto por Arbor Networks](#) (<http://asert.arbornetworks.com/lets-talk-about-newpostings/>) en septiembre de 2014 ha conseguido optimizarse y configurarse de forma diferentes dependiendo el objetivo a seleccionar.

## El último grande, NitlovePOS

El equipo de FireEye ha detectado ([https://www.fireeye.com/blog/threat-research/2015/05/nitlovepos\\_another.html](https://www.fireeye.com/blog/threat-research/2015/05/nitlovepos_another.html)) al último de la saga. Se trata de NitlovePOS, el último malware especialmente creado para atacar los sistemas de cobro de las tiendas físicas. Este malware POS escanea los procesos de la máquina infectada para capturar información de tarjetas de crédito.

Sin embargo, ante de llegar al terminal de punto de venta hay más. La campaña de infección comenzó con un ataque masivo de spam que comenzó hace unos días. El email ofrecía un puesto de trabajo e incluía un documento donde se integraba el malware.

# BUENAS PRÁCTICAS DE SEGURIDAD EN DOCKER

## (HTTP://WWW.SECURITYBYDEFAULT.COM/2015/05/BU PRACTICAS-DE-SEGURIDAD-EN-DOCKER.HTML)



Docker (<https://github.com/docker/docker>) es una plataforma abierta que permite construir, portar y ejecutar aplicaciones distribuidas (<https://youtu.be/ZzQfxoMFH0U>), se basa en contenedores que corren en Linux y funcionan tanto en máquinas físicas como virtuales simplemente usando un runtime. Está escrito en Go (<https://golang.org/>) y usa librerías del sistema operativo así como funcionalidades del kernel de Linux en el que se ejecuta. Consta de un engine con API RESTful y un cliente que pueden ejecutarse en la misma máquina o en máquinas separadas. Es Open Source (Apache 2.0) y gratuito.

Los contenedores existen desde hace muchos años, Docker no ha inventado nada en ese sentido, o casi nada, pero no hay que quitarles mérito, están en el momento adecuado y aportan las características y herramientas concretas que se necesitan en la actualidad, donde la portabilidad, escalabilidad, alta disponibilidad y los microservicios en aplicaciones distribuidas son cada vez más utilizados, y no sólo eso, sino que también son mejor entendidos por la comunidad de desarrolladores y administradores de sistemas. Cada vez se desarrollan menos aplicaciones monolíticas y más basadas en módulos o en microservicios, que permiten un desarrollo más ágil, rápido y a la vez portable. Empresas de sobra conocidas como Netflix, Spotify o Google e infinidad de Start ups usan arquitecturas basadas en microservicios en muchos de los servicios que ofrecen.

Te estarás preguntando ¿Y no es más o menos lo mismo que hacer un chroot de una aplicación? Sería como comparar una rueda con un coche. El concepto de chroot es similar ya que se trata de aislar una aplicación, pero Docker va mucho más allá, sería un chroot con esteroides, muchos esteroides. Por ejemplo, puede limitar y controlar los recursos a los que accede la aplicación en el contenedor, generalmente usan su propio sistema de archivos como UnionFS o variantes como AUFS, btrfs, vfs, Overlayfs o Device Mapper que básicamente son sistemas de ficheros en capas. La forma de controlar los recursos y capacidades que hereda del host es mediante namespaces y cgroups de Linux. Esas opciones de Linux no son nuevas en absoluto, pero Docker lo hace fácil y el ecosistema que hay alrededor lo ha hecho tan utilizado.

Adicionalmente, la flexibilidad, comodidad y ahorro de recursos de un contenedor es mayor a la que aporta una máquina virtual o un servidor físico, esto es así en muchos casos de uso, no en todos. Por ejemplo, tres servidores web para un cluster con Nginx en una VM con una instalación de Linux CentOS mínima ocuparía unos 400MB, multiplicado por 3 máquinas sería total de uso en disco de 1,2 GB, con contenedores serían 400MB las mismas 3 máquinas corriendo ya que usa la misma imagen para múltiples contenedores. Eso es sólo por destacar una característica interesante a nivel de recursos. Otro uso muy común de Docker es la portabilidad de aplicaciones, imagina una aplicación que solo funciona con Python 3.4 y hacerla funcionar en un sistema Linux con Python 2.x es complicado, piensa en lo que puede suponer en un sistema en producción actualizar Python, con contenedores sería casi automático, descargar la imagen del contenedor y ejecutar la aplicación de turno.

Solo por ponernos en situación de la envergadura Docker, unos números alrededor del producto y la compañía (fuente aquí (<http://www.businesswire.com/news/home/20150414005387/en/Docker-Secures-95M-Series-Funding-Meet-Unrivaled#.VVIFntNViqq>)):

- 95 millones de dólares de inversión.
- Valorada en 1.000 millones de dólares.
- Más de 300 millones de descargas en 96 releases desde marzo de 2013

Pero un contenedor no es para todo, ni hay que volverse loco "dockerizando" cualquier cosa, aunque no es este el sitio para esa reflexión. Al cambiar la forma de desarrollar, desplegar y mantener aplicaciones, también **cambia en cierto modo la forma de securizar estos nuevos actores**.

Docker aporta seguridad en capas, aísla aplicaciones entre ellas y del host sin usar grandes recursos, también se pueden desplegar contenedores en máquinas virtuales lo que aporta otra capa adicional de aislamiento (estaréis pensando en VENOM (<http://www.zdnet.com/article/venom-security-flaw-millions-of-virtual-machines-datacenters/>) pero eso es otra película que no afecta directamente a Docker). Dada la arquitectura de Docker y usando buenas prácticas, aplicar parches de seguridad al anfitrión o a aplicaciones suele ser más rápido y menos doloroso.

### Buenas Prácticas de Seguridad:

Aunque la seguridad es algo innato en un contenedor, desde Docker Inc. están haciendo esfuerzos por la seguridad, por ejemplo, contrataron hace unos meses a ingenieros de seguridad de Square (<https://blog.docker.com/2015/03/secured-at-docker-diogo-monica-and-nathan-mccauley>), que no son precisamente nuevos en el tema. Ellos, junto a compañías como VMware entre otras, han publicado recientemente un extenso informe de sobre buenas prácticas de seguridad en Docker ([https://benchmarks.cisecurity.org/tools2/docker/CIS\\_Docker\\_1.6\\_Benchmark\\_v1.0.0.pdf](https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.6_Benchmark_v1.0.0.pdf)) en el CIS (<http://www.cisecurity.org/>). Gracias a este informe tenemos acceso a más de 90 recomendaciones de seguridad a tener siempre en cuenta cuando vamos a usar Docker en producción. En la siguiente tabla podemos ver las recomendaciones de seguridad sugeridas, algunas son muy obvias pero un check list así nunca viene mal:

#### 1. Recomendaciones a nivel de host

- 1.1. Crear una partición separada para los contenedores
- 1.2. Usar un Kernel de Linux actualizado
- 1.3. No usar herramientas de desarrollo en producción
- 1.4. Securizar el sistema anfitrión
- 1.5. Borrar todos los servicios no esenciales en el sistema anfitrión
- 1.6. Mantener Docker actualizado
- 1.7. Permitir solo a los usuarios autorizados controlar el demonio Docker
- 1.8. Auditar el demonio Docker (auditd)
- 1.9. Auditar el fichero o directorio de Docker - /var/lib/docker
- 1.10. Auditar el fichero o directorio de Docker - /etc/docker
- 1.11. Auditar el fichero o directorio de Docker - docker-registry.service
- 1.12. Auditar el fichero o directorio de Docker - docker.service
- 1.13. Auditar el fichero o directorio de Docker - /var/run/docker.sock
- 1.14. Auditar el fichero o directorio de Docker - /etc/sysconfig/docker
- 1.15. Auditar el fichero o directorio de Docker - /etc/sysconfig/docker-network
- 1.16. Auditar el fichero o directorio de Docker - /etc/sysconfig/docker-registry
- 1.17. Auditar el fichero o directorio de Docker - /etc/sysconfig/docker-storage

1.18. Audit el fichero o directorio de Docker - /etc/default/docker

## 2. Recomendaciones a nivel de Docker Engine (daemon)

- 2.1 No usar el driver obsoleto de ejecución de lxc
- 2.2 Restringir el tráfico de red entre contenedores
- 2.3 Configurar el nivel de logging deseado
- 2.4 Permitir a Docker hacer cambios en iptables
- 2.5 No usar registros inseguros (sin TLS)
- 2.6 Configurar un registro espejo local
- 2.7 No usar aufs como driver de almacenamiento
- 2.8 No arrancar Docker para escuchar a una IP/Port o Unix socket diferente
- 2.9 Configurar autenticación TLS para el daemon de Docker
- 2.10 Configurar el ulimit por defecto de forma apropiada

## 3. Recomendaciones a nivel de configuración de Docker

- 3.1 Verificar que los permisos del archivo docker.service están como root:root
- 3.2 Verificar que los permisos del archivo docker.service están en 644 o más restringidos
- 3.3 Verificar que los permisos del archivo docker-registry.service están como root:root
- 3.4 Verificar que los permisos del archivo docker-registry.service están en 644 o más restringidos
- 3.5 Verificar que los permisos del archivo docker.socket están como root:root
- 3.6 Verificar que los permisos del archivo docker.socket están en 644 o más restringidos
- 3.7 Verificar que los permisos del archivo de entorno Docker (/etc/sysconfig/docker o /etc/default/docker) están como root:root
- 3.8 Verificar que los permisos del archivo de entorno Docker (/etc/sysconfig/docker o /etc/default/docker) están en 644 o más restringidos
- 3.9 Verificar que los permisos del archivo /etc/sysconfig/docker-network (si se usa systemd) están como root:root
- 3.10 Verificar que los permisos del archivo /etc/sysconfig/docker-network están en 644 o más restringidos
- 3.11 Verificar que los permisos del archivo /etc/sysconfig/docker-registry (si se usa systemd) están como root:root
- 3.12 Verificar que los permisos del archivo /etc/sysconfig/docker-registry (si se usa systemd) están en 644 o más restringidos
- 3.13 Verificar que los permisos del archivo /etc/sysconfig/docker-storage (si se usa systemd) están como root:root
- 3.14 Verificar que los permisos del archivo /etc/sysconfig/docker-storage (si se usa systemd) están en 644 o más restringidos
- 3.15 Verificar que los permisos del directorio /etc/docker están como root:root
- 3.16 Verificar que los permisos del directorio /etc/docker están en 755 o más restrictivos
- 3.17 Verificar que los permisos del certificado del registry están como root:root
- 3.18 Verificar que los permisos del certificado del registry están en 444 o más restringidos
- 3.19 Verificar que los permisos del certificado TLS CA están como root:root
- 3.20 Verificar que los permisos del certificado TLS CA están en 444 o más restringidos
- 3.21 Verificar que los permisos del certificado del servidor Docker están como root:root
- 3.22 Verificar que los permisos del certificado del servidor Docker están en 444 o más restringidos
- 3.23 Verificar que los permisos del archivo de clave del certificado del servidor Docker están como root:root
- 3.24 Verificar que los permisos del archivo de clave del certificado del servidor Docker están en 400
- 3.25 Verificar que los permisos del archivo de socket de Docker están como root:docker
- 3.26 Verificar que los permisos del archivo de socket de Docker están en 660 o más restringidos

## 4. Imágenes de Contenedores y Dockerfiles

- 4.1 Crean un usuario para el contenedor
- 4.2 Usar imágenes de confianza para los contenedores
- 4.3 No instalar paquetes innecesarios en el contenedor
- 4.4 Regenerar las imágenes si es necesario con parches de seguridad

## 5. Runtime del contenedor

- 5.1 Verificar el perfil de AppArmor (Debian o Ubuntu)
- 5.2 Verificar las opciones de seguridad de SELinux (RedHat, CentOS o Fedora)
- 5.3 Verificar que los contenedores esten ejecutando un solo proceso principal
- 5.4 Restringir las Linux Kernel Capabilities dentro de los contenedores
- 5.5 No usar contenedores con privilegios
- 5.6 No montar directorios sensibles del anfitrión en los contenedores
- 5.7 No ejecutar ssh dentro de los contenedores
- 5.8 No mapear puertos privilegiados dentro de los contenedores
- 5.9 Abrir solo los puertos necesarios en un contenedor
- 5.10 No usar el modo "host network" en un contenedor
- 5.11 Limitar el uso de memoria por contenedor
- 5.12 Configurar la prioridad de uso de CPU apropiadamente
- 5.13 Montar el sistema de ficheros raíz de un contenedor como solo lectura
- 5.14 Limitar el tráfico entrante al contenedor mediante una interfaz específica del anfitrión
- 5.15 Configurar la política de reinicio 'on-failure' de un contenedor a 5
- 5.16 No compartir PID de procesos del anfitrión con contenedores
- 5.17 No compartir IPC del anfitrión con contenedores
- 5.18 No exponer directamente dispositivos del anfitrión en contenedores
- 5.19 Sobre-escribir el ulimit por defecto en tiempo de ejecución solo si es necesario

## 6. Operaciones de Seguridad en Docker

- 6.1 Realizar auditorías de seguridad tanto en el anfitrión como en los contenedores de forma regular
- 6.2 Monitorizar el uso, rendimiento y métricas de los contenedores
- 6.3 Endpoint protection platform (EPP) para contenedores (si las hubiese)
- 6.4 Hacer Backup de los datos del contenedor
- 6.5 Usar un servicio centralizado y remoto para recolección de logs
- 6.6 Evita almacenar imágenes obsoletas, sin etiquetar correctamente o de forma masiva.
- 6.7 Evita almacenar contenedores obsoletos, sin etiquetar correctamente o de forma masiva.

# HACIENDO FUERZA BRUTA A CONTENEDORES CIFRADOS EN MAC OS X

## (HTTP://WWW.SECURITYBYDEFAULT.COM/2015/05/HACIENDO-FUERZA-BRUTA-CONTENEDORES.HTML)



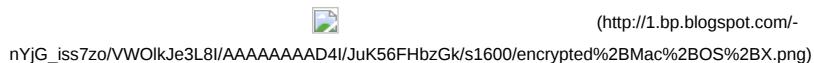
(http://2.bp.blogspot.com/-

V4dOp0mPt\_I/VWOmEjAqH4l/AAAAAAAD4Q/j82KDnJhOu0/s1600/Master\_lock\_with\_root\_password.jpg)

A todos nos ha pasado alguna vez, que guardamos algo tan seguro, tan seguro, tan seguro,... que no nos acordamos dónde lo dejamos. Hace unos días me tocó tirar de un disco duro que guardo cifrado, mediante el sistema de ficheros HFS+ de Mac, con las opciones Journaled, Encrypted (<http://osxdaily.com/2012/01/25/password-protect-external-drive-mac-encrypted-partition/>).

Cuando insertas el USB, Mac te muestra un Pop-Up, en el que se te pide la contraseña de desbloqueo del volumen. Todo esto está bien cuando usas esa contraseña a menudo, pero cuando no es el caso dices... por qué habré puesto una contraseña de cifrado tan rara.

A todo esto, el pop-up, no te permite ver la contraseña en claro, por lo que si te has equivocado en una letra, te toca volverla a escribir completa. Tampoco permite hacer copy-paste, por lo que no podemos escribirla en un bloc de notas y pegarla en la caja de texto, así que volvemos al paso 1.



(http://1.bp.blogspot.com/-

nYjG\_iss7zo/VWOlkJe3L8I/AAAAAAAD4l/JuK56FHbzGk/s1600/encrypted%2BMac%2BOS%2BX.png)

Como la contraseña era bastante complicada, me puse a investigar la opción de hacer el montaje desde un terminal.

El comando **diskutil** (<https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man8/diskutil.8.html>) tiene múltiples combinaciones que nos permiten gestionar, entre otras cosas, los volúmenes cifrados.

Con el flag "**cs list**", el CoreStorage nos muestra la estructura formada por los diferentes Grupos de Volúmenes, Volúmenes Físicos, Familias de Volúmenes Lógicos y finalmente los Volúmenes Lógicos que ve el sistema operativo. Si tenemos habilitado FileVault2 para el cifrado del disco completo (opción muy recomendable), nos aparecerá el primero de ellos, con el volumen desbloqueado (Unlocked dentro del campo Encryption Status)

```
Data hosted with ♥ by Pastebin.com (http://pastebin.com/) - Download Raw (http://pastebin.com/raw.php?i=GXD8KiwJ) - See Original (http://pastebin.com/GXD8KiwJ)
1. +- Logical Volume Group D9183FF1-7F0A-4AF8-9E3D-68BD25BCF7BF
2. =====
3. Name: test
4. Status: Online
5. Size: 7412113408 B (7.4 GB)
6. Free Space: 13357056 B (13.4 MB)
7.
8. +-< Physical Volume 7A2E4D19-DCA8-4037-BDC6-60F4449864DF
9. | -----
10. | Index: 0
11. | Disk: disk3s2
12. | Status: Online
13. | Size: 7412113408 B (7.4 GB)
14.
15. +-> Logical Volume Family 5F1E3906-6B56-4D8E-8DBC-F1682D625BC5
16. -----
17. | Encryption Status: Locked
18. | Encryption Type: AES-XTS
19. | Conversion Status: Complete
20. | Conversion Direction: -none-
21. | Has Encrypted Extents: Yes
22. | Fully Secure: Yes
23. | Passphrase Required: Yes
24.
25. +--> Logical Volume AA0E91C1-131A-4B86-9F6D-772D281C3ED6
26. -----
27. | Disk: -none-
28. | Status: Locked
29. | Size (Total): 7046430720 B (7.0 GB)
30. | Conversion Progress: -none-
31. | Revertible: No
32. | LV Name: test
33. | Content Hint: Apple_HFSX
```

En el caso que se ve arriba, dentro de la sección Logical Volume, AA0E91.... etc... se puede ver que el Status es Locked.

Con Diskutil y los flags unlockvolume , te pediría por línea de comandos la contraseña de desbloqueo. Si además le añadimos -passphrase , ya nos permite hacerla en claro y ver si nos estamos equivocando o no, hacer copy/paste o incluso... montarnos un bruteforcer. Ya que estaba, he hecho algo muy sencillito pero que si a alguien le puede ser de ayuda, pues aquí lo tiene:

```
Data hosted with ♥ by Pastebin.com (http://pastebin.com/) - Download Raw (http://pastebin.com/raw.php?i=KqttuDKQ) - See Original (http://pastebin.com/KqttuDKQ)

1. LawMac:Desktop Lawrence$ more bruteforcea.pl
2. #!/usr/bin/perl
3. use Getopt::Std;
4.
5. getopts('u:d:');
6.
7. die "Usage $0 -u -d \n" if (!$opt_u && !$opt_d);
8.
9. #Guardamos parametros
10. my $uid=$opt_u;
11. my $dict=$opt_d;
12.
13. open (FILE,<$dict");
14.
15. #Por cada posible contraseña generada, probamos
16. while ()
17. {
18.     chomp ($_);
19.     my $resp=`diskutil cs unlockvolume $uid -passphrase $_ `;
20.
21.     #Si acertamos, indicamos la contraseña correcta
22.     if ($resp =~ m-successfully/i)
23.     {
24.         print "Password found!!! It was \"$_\"\n";
25.         print "$resp\n";
26.         exit(0);
27.     }
28. }
29. close (FILE);
30. print "Password not found in diccionary $dict. Insert coin and play again\n";
```

Tan fácil como pasarse con el parámetro -u el UID del volumen a desbloquear y con un -d un fichero "diccionario", que previamente habremos generado. En este punto, la inteligencia a aplicar es diferente y no voy a entrar. Que haya palabras concatenadas con números, caracteres especiales o lo que queráis, es otra historia diferente.

Una vez el script acierta con la contraseña, lo deja desbloqueado, pero sin montar.

```
Data hosted with ♥ by Pastebin.com (http://pastebin.com/) - Download Raw (http://pastebin.com/raw.php?i=bSn5hzZ5) - See Original (http://pastebin.com/bSn5hzZ5)

1. LawMac:Desktop Lawrence$ perl bruteforcea.pl -u AA0E91C1-131A-4B86-9F6D-772D281C3ED6 -d dict.txt
2. Error: -69749: Unable to unlock the Core Storage volume
3. Error: -69749: Unable to unlock the Core Storage volume
4. Error: -69749: Unable to unlock the Core Storage volume
5. Error: -69749: Unable to unlock the Core Storage volume
6. Password found!!! It was "securitybydefault.com"
7. Started CoreStorage operation
8. Logical Volume successfully unlocked
9. Logical Volume successfully attached as disk4
10. Logical Volume successfully mounted as /Volumes/test
11. Core Storage disk: disk4
12. Finished CoreStorage operation
```

Nos dice cuál era la contraseña, por si sólo necesitamos eso. Si queremos montarlo, tan sencillo como "diskutil mount disk4" (en este caso) y ya está!

Lo que es un axioma innegable es que la prisa que te corre recuperar lo que hay dentro de un contenedor cifrado, es directamente proporcional a las posibilidades que te pegues con una contraseña que no funciona (o que no recuerdas ;D)







You are here: Home

### Side Module

This is a module where you might want to add some more information or an image, a link to your social media presence, or whatever makes sense for your site. You can edit this module in the module manager. Look for the Side Module.

### Login Form

 Username Password

Remember Me

Forgot your username? ([/joomla/index.php/component/users/?view=remind](#))  
Forgot your password? ([/joomla/index.php/component/users/?view=reset](#))

