

Un dispositivo USB de seguridad para contraseñas de un solo uso

STICK SECRETO

IOANN Mikhaylov, 123RF

Todo aquel dispuesto a experimentar, podría encontrar en OpenKubus la solución a sus problemas de contraseñas.

**POR BENEDIKT SAUTER, MICHAEL HARTMANN
Y NILS MAGNUS**

A la hora de autenticar usuarios, la mayoría de los administradores de Linux suele optar por usar contraseñas. Claro está que una costumbre así implica un sinnúmero de problemas: Si se deja a los usuarios elegir sus propias contraseñas, lo más probable es que sean cortas y fácilmente predecibles. Si las creamos con algún generador de contraseñas aleatorias, probablemente serán tan complicadas que los usuarios las olvidarán o, lo que es peor, las irán dejando por ahí, escritas en cualquier sitio. La privacidad es algo que cuesta conseguir, sobre todo en entornos hostiles en los que abundan los keyloggers y los mirones. Las contraseñas de usar y tirar, formalmente denominadas OTPs (One-time passwords), son la solución a este dilema: Puesto que las de un solo uso dejan de ser válidas tras usarse por primera vez, a nadie le importa que una tercera persona las pueda capturar. Un generador de contraseñas crea una enorme lista de contraseñas y guarda una copia de esa lista en el servidor al que el usuario pretende acceder. La idea consiste en que el usuario tenga una segunda copia de la lista de contraseñas y use en

cada autenticación la siguiente contraseña de su lista. Un ejemplo de esta técnica es el método TAN usado por muchos bancos para las transferencias seguras en banca en línea.

Al acceder un usuario, el sistema puede instarle a introducir, digamos, la décima contraseña de la lista. Si dicho usuario dispone de la lista de contraseñas correcta, podrá introducir la correcta para autenticarse. El principal inconveniente de este método es que a veces es poco práctico. Al usuario no le queda más remedio que llevar consigo la lista en todo momento; cada vez que quiera acceder, tendrá que cogerla y escribir la contraseña adecuada. Está claro también que esta técnica no evita que la lista caiga en las manos equivocadas. Algo más prácticos son los sistemas que generan contraseñas dinámicas. Las aplicaciones cliente para generar contraseñas funcionarán bien sólo cuando no se van a cambiar las ubicaciones, pero se trata de un método poco portable. Una solución mejor pasa por generar contraseñas de un solo uso mediante un token externo por hardware. Hay varias formas de tokens por hardware para contraseñas de un solo uso

(para saber más, ver el artículo sobre “Contraseñas de usar y tirar” publicado en el número de 45 de Linux Magazine [1]). Muchos de los tokens por hardware para contraseñas son privativos; sin embargo, también hay disponible un adaptador universal de programación basado en la plataforma abierta de hardware *USBprog* para la implementación de soluciones de contraseñas de un solo uso. Esta plataforma utiliza el popular microcontrolador *ATmega32* – una alternativa económica que cuenta con un montón de aplicaciones libres para desarrollo. También hay un segundo componente, el *USB9604*, que soporta acceso por USB. Esta variante puede crearse fácilmente añadiendo un solo botón (ver Figura 1). El PCB, la lista de componentes y el esquema electrónico están disponibles bajo licencia GPLv2 license [2].

Desde el punto de vista de la máquina, el stick es un teclado USB, y está soportado en muchos sistemas operativos. Cuando el usuario pulsa el botón, el stick devuelve una contraseña generada según el método mostrado en la Figura 2. El proyecto *OpenKubus* ha desarrollado un segundo diseño que es funcionalmente

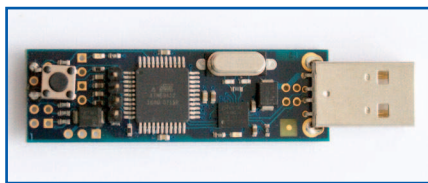


Figura 1: Una variante del proyecto hace uso del adaptador USBprog con un botón. Al pulsarlo se transfiere la contraseña.

equivalente al token abierto anterior. El token OpenKubus cabe dentro de una carcasa aún más pequeña y consta de menos elementos (ver Figura 3). El microcontrolador es un *ATmega16U4*, que es un *ATmega16* con un controlador para USB integrado. Lo mejor de la opción OpenKubus es que se puede adquirir en su web una versión prefabricada del hardware por unos 25 euros [3]. Disponiendo de un token prefabricado como éste se puede experimentar con contraseñas de un solo uso basadas en hardware sin tener que crear el hardware uno mismo.

Su Funcionamiento

El programa que se ejecuta en el stick usa un bloque de datos de 15 bytes que comprende dígitos aleatorios o un ID de stick. El microcontrolador antepone un número de serie de dos bytes. Cuando un usuario pulsa el botón, el microcontrolador concatena el número de serie y el bloque de datos y los cifra ambos utilizando una clave de 256 bits que está almacenada dentro del stick. La EEPROM del chip controlador guarda las tres partes. En el último paso, OpenKubus convierte los 16 bytes que componen la clave cifrada de un solo uso en caracteres legibles y los envía al driver de teclado USB (ver Figura 2). Debido a que el stick no sabe qué variante de teclado es la que está activa en la máquina a la que está conectado, envía el keycode de la letra z para distinguir si se trata de un teclado alemán o de uno inglés.

¡Déjame Entrar!

Para comprobar estas credenciales, el otro extremo necesita el bloque de datos, la clave AES y un número de serie. Convierte la entrada de teclado de nuevo a los 16 bytes de contraseña cifrada y la descifra usando la clave AES. Los dos primeros

objetos cifrados deben tener un valor mayor que el almacenado en el lado del servidor (de este modo se permite al sistema detectar potenciales ataques de repetición). Si OpenKubus necesita autenticarse ante más de un dispositivo, los servidores tienen que sincronizar los números de serie. Tanto el servidor como el stick incrementan entonces el número de serie. En la instalación actual, el servidor agota sus posibilidades después de 65636 aplicaciones, momento en el que el firmware actual vuelve a empezar desde el principio.

OpenKubus usa *AVR-Crypto-Lib* como librería criptográfica; la librería incluye una gran recopilación de funciones criptográficas para microprocesadores Atmel y tiene en cuenta los precarios recursos de flash y RAM. El elevado número de algoritmos soportados permite el uso de métodos criptográficos distintos a todo aquel que esté dispuesto a experimentar. Se pueden hacer también otros cambios, como utilizar un método diferente para la gestión del reinicio del firmware anteriormente comentado.

Preparación

Una vez reunido el hardware, o abierto el paquete en el caso de quienes encarguen la versión OpenKubus, hay que compilar el firmware y transferirlo al dispositivo. Con la última versión de OpenKubus (ver Figura 4) se puede omitir este paso, puesto que el software ya viene instalado. El comando

```
svn checkout http://openkubus.googlecode.com/svn/trunk/openkubus-read-only
```

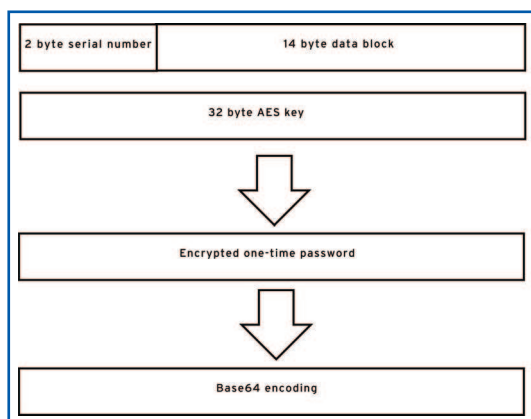


Figura 2: El token referencia al bloque de datos y un número de serie para calcular la contraseña de un solo uso, la encripta con AES y realiza algo parecido a una codificación en Base64.



Figura 3: Si no nos atrevemos a soldar, podemos adquirir un OpenKubus prefabricado. Esta versión incluye el driver para USB en su microcontrolador ATmega16U4.

hace un volcado y lo guarda en la máquina [4]. Además, se necesitan los paquetes *gcc-avr* y *binutils-avr* para el compilador cruzado AVR-GCC [5]. El paquete *avrdude* contiene herramientas para la transferencia de firmware. Todos los paquetes están disponibles a través de los repositorios de Debian; para otras distribuciones puede que haya que compilarlos.

La mayoría de subdirectorios incluyen sus propios makefiles: el firmware necesita las funciones criptográficas, que se compilan haciendo un **make** en el directorio *firmware/crypto-lib/*. Por otro lado, si se va a usar la variante USBprog, se ha de compilar el firmware en el directorio *firmware/secstick_v1/*. Para la segunda versión de hardware de OpenKubus, cambiamos al directorio *firmware/secstick_v2/*. Luego, ejecutamos *avrdude* y un programador como USBprog para transferir la imagen de firmware prefabricada, *openkubus.hex*, a la memoria flash del microcontrolador. Otra alternativa sería ayudarnos de un *Atmel MkII* para transferir el software; el puerto USB aún no está habilitado en este punto.

Cargado y Asegurado

OpenKubus ya está listo, pero aún no conoce nuestros secretos. La herramienta *stick-write*, en *firmware/create-stick*, carga en la EEPROM las tres partes de información. Hasta la fecha no existe ninguna herramienta para preparar el stick, por lo que no queda más remedio que hacer algunas cosas a mano: la opción **-p** espera 48 caracteres (los 32 bytes de la clave AES, un bloque de datos de 14 dígitos y los dos octetos de los primeros números de serie, sin separadores).

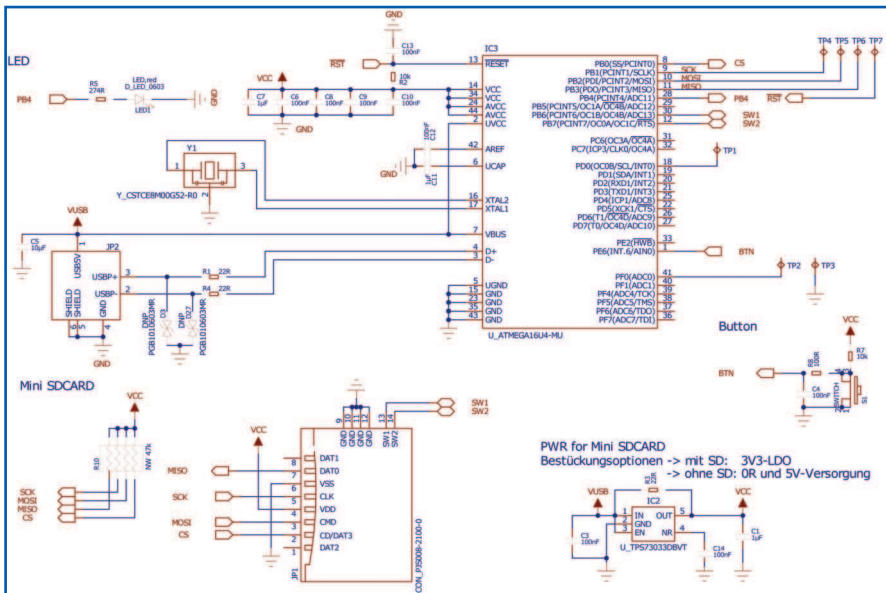


Figura 4: La nueva versión de OpenKubus utiliza un ATmega16U4, que también se encarga de la parte USB.

La herramienta utiliza de USB para transferir los argumentos de línea de comandos al stick sin convertirlos. Si se prefiere transferir datos binarios, hay que modificar el código fuente. La opción `-l` le ordena al firmware no aceptar más contraseñas iniciales. Claro está que podemos usar el programador para blindar el stick, pero implicaría la imposibilidad de actualizar el firmware más adelante.

El Enlace con PAM

La mayoría de las distribuciones Linux utilizan PAM para la autenticación de usuarios. Esto significa que podemos enseñar nuevos métodos de autenticación a programas como `login` o `sshd`. OpenKubus incluye su propio módulo en el directorio `software/PAM/` que requiere los archivos para desarrollo de `libpam0g-dev`. Haciendo un `make install` se copiará al directorio `/lib/security/` y se modificarán los privilegios. Tras completar la instalación, y trabajando como `root`, configuraremos los módulos en el archivo `/etc/pam.conf` o en el directorio `/etc/pam.d/`. Al añadir

```
auth sufficient pam_openkubus.so
```

a `auth-common` le estamos diciendo a PAM que acepte el stick [6] como alternativa en un futuro.

Verificación de Contraseñas

Para verificar la autenticidad de una contraseña, el módulo PAM necesita conocer

los secretos del stick. Para ello, el administrador guardará la línea correspondiente en `/etc/openkubus-passwd` con el formato `usuario contraseña número_de_serie`. PAM compara estos datos con la entrada proporcionada por el stick. Conviene asegurarse de que el archivo sólo es accesible para el usuario `root`.

Si se desea comprobar la entrada de OpenKubus desde una aplicación propia sin tener que recurrir a PAM, la librería ligera de OpenKubus, ofrecida por el proyecto y disponible para muchos lenguajes de programación, puede ser una opción interesante a tener en cuenta. La librería propiamente dicha está escrita en C, pero el envoltorio SWIG da acceso a Python, Ruby, Java, Perl y PHP a sus funciones [7]. Ahora mismo la API sólo incluye la siguiente función:

```
int openkubus_authenticate >
(const char *pad, >
const char *pw, >
int offset, int num);
```

Ésta espera que se le pase como primer argumento la contraseña de un solo uso, y la clave AES como segundo parámetro. Los dos últimos argumentos son un offset opcional para el número de serie (que por defecto es 0) y el propio número de serie. Si tiene éxito, la función devolverá el mismo número de serie, de lo contrario devolverá un valor negativo.

La librería sólo comprueba la contraseña; no accede a ningún archivo. Los desarrolladores que usen OpenKubus necesitarán gestionar ellos mismos el número de serie actual, la contraseña inicial y el offset. El servidor de red de ejemplo ubicado en `software/server/`, que compara las contraseñas de un solo uso con las contenidas en una lista, nos da una idea de cómo se pueden gestionar estos parámetros.

Conclusión

Una de las principales ventajas de OpenKubus es la de permitirnos personalizar el hardware sin tener que disponer de un presupuesto exagerado. La mayor desventaja es que hay que sincronizar necesariamente el número de serie entre el stick y todos los servidores. Si se necesita autenticación contra varios servidores, lo más conveniente será establecer un servidor central. Las herramientas existentes para la gestión de OpenKubus en entornos de mayor envergadura con un número mayor de usuarios son aún rudimentarias.

OpenKubus no nos protegerá ante ataques MITM (*man in the middle*) [8]. El servicio al que estamos llamando ha de demostrar su autenticidad utilizando otros medios. Sin embargo, el proyecto supone una interesantísima plataforma para todo aquel al que le guste experimentar. ■

RECURSOS

- [1] "Contraseñas de Usar y Tirar" por Udo Seidel, Linux Magazine – Edición en Castellano, número 45, pág. 18.
- [2] Esquema de USBprog: <http://www.embedded-projects.net/usbprog>
- [3] Tienda de hardware OpenKubus: <http://shop.embedded-projects.net>
- [4] OpenKubus (en alemán): <http://code.google.com/p/openkubus>
- [5] Apuntes sobre la instalación en AVR-GCC: http://www.nongnu.org/avr-libc/user-manual/install_tools.html
- [6] Sintaxis de configuración de PAM: <http://kernel.org/pub/linux/libs/pam/Linux-PAM-html/sag-configuration-file.html>
- [7] SWIG: <http://www.swig.org/>
- [8] Ataques MITM: http://en.wikipedia.org/wiki/Man-in-the-middle_attack