Solución a las pruebas 11 y 12 del reto Secutel 2009

Mayo 2009 © Daniel Kachakil

Introducción

Este documento describe una solución a las dos últimas pruebas del reto de criptografía Secutel 2009 de la IEEEsb-UPV, publicado el 30 de abril de 2009 en la siguiente dirección web:

`http://www.ieee.upv.es/cripto/concurso.php`

El reto constaba de 12 pruebas de criptografía y esteganografía, todas ellas independientes y cada una con una puntuación acorde a su nivel de dificultad. Tras la finalización del tiempo oficial asignado para poder puntuar en el mismo (las soluciones debían enviarse antes del 09/05/2009), quedaron dos pruebas sin resolver por parte de ninguno de los participantes, que son precisamente las que se solucionan aquí.

Cuando supe de la existencia del reto, el plazo oficial ya había finalizado, los premios habían sido repartidos e incluso se habían publicado las soluciones oficiales al resto de las pruebas (disponibles aquí y aquí).

Pistas y documentación

En la misma página donde estaban publicadas las pruebas, había un espacio reservado para que los organizadores fueran dejando pistas cuando lo consideraran oportuno. Ya que en nuestro caso o no había o no resultaron de gran ayuda, usaremos como principal fuente de información la propia presentación de la conferencia Secutel 2009 en la que se dio a conocer el reto, cuyos ponentes (Amine Taouirsa y Javi Moreno) son también los organizadores de estas pruebas.

El resto de soluciones publicadas también nos podrían servir para hacernos una idea de cómo estaban planteadas las otras pruebas y tal vez para poder descartar algún algoritmo de cifrado o técnica que ya hubiera sido usada previamente (aunque evidentemente nadie garantiza que las fases no se puedan repetir).

Prueba 11 (Matrioska: get the password!)

Partimos de un fichero de imagen en formato JPEG (qrcode.jpg), que contiene un código de puntos bidimensional en un cuadrado de 1000 píxeles de lado. A simple vista (incluso sin habernos fijado en el propio nombre del fichero), podemos intuir que se trata del formato QR-Code, puesto que se aprecian en él todas las características típicas de este formato (marcadores de posición, sincronización, etc).

Para poder decodificarlo podemos usar cualquier escáner que soporte este formato o algún teléfono móvil con cámara y software adecuado (hoy en día muchos vienen ya con este tipo de aplicaciones preinstaladas), pero si no tenemos a mano uno de estos dispositivos y tampoco queremos complicarnos la vida instalando programas desconocidos, simplemente buscaremos alguna aplicación web que lo haga.

Por ejemplo, podemos usar esta aplicación, en la que no tenemos más que adjuntar y enviar el fichero, obteniendo el contenido decodificado de forma instantánea.

TEXTO DECODIFICADO:

el cifrado verdadero se abre con la contraseña ***elbigbangesunyoyo***

Si nos guiamos una parte del título de la prueba (*get the password*), podemos pensar que ya tenemos la contraseña que estábamos buscando, pero si tenemos en cuenta su nivel, podemos estar seguros de que nos hemos dejado algo por el camino...

Para no entretenernos demasiado analizando el contenido del fichero con un editor hexadecimal, podemos utilizar la herramienta jhead, que nos mostrará toda la metainformación incrustada en el fichero JPG:

```
jhead -st miniatura.jpg qrcode.jpg
```

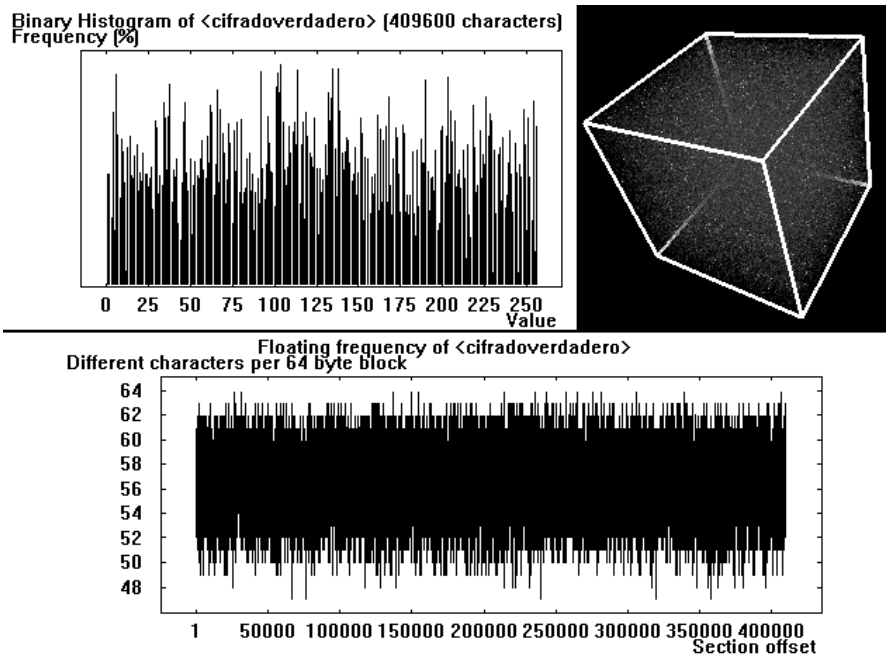
Como se puede apreciar de un simple vistazo, la miniatura que contiene el fichero no se corresponde con la imagen original, sino que contiene un texto (en formato gráfico) con la siguiente frase: "*El password del stegoculta es* ***arpanet***"

Además de revelarnos otra contraseña (que tampoco es la definitiva), este texto nos indica que hay una parte de esteganografía (información oculta en la imagen) y que además la aplicación de esta técnica está combinada con algún tipo de cifrado.

Si nos remitimos a la diapositiva 64 de la presentación, encontramos un listado de algunas de las herramientas de esteganografía más conocidas. Entre ellas, destacamos las que utilizan el formato JPEG como portador de información. Por tanto, si utilizamos la herramienta steghide con la contraseña indicada, obtendremos un fichero oculto que contiene el siguiente texto: "*para seguir bajate el archivo http://www.ieee.upv.es/cripto/pruebas/11/cifradoverdadero*"

Tras proceder a la descarga del fichero indicado en esa URL, observamos que se trata de un fichero de 409.600 bytes, sin ningún tipo de extensión ni de cabecera que indique de qué formato se trata. Por la primera pista, asumiremos que está cifrado con la contraseña *"elbigbangesunyoyo"*, pero desconocemos con qué algoritmo está cifrado.

Podemos utilizar la herramienta CrypTool para analizar e intentar descifrar el fichero con los diferentes algoritmos que tiene implementados. Lo primero que se observa es que el fichero tiene un grado de dispersión y de aleatoriedad muy elevado, lo que suele ser un indicador de algún tipo de cifrado moderno. Habiendo descartado el cifrado XOR y el de desplazamiento de bytes, nos quedan unos cuantos algoritmos por probar (IDEA, RC2, RC4, DES, AES, etc.), los cuales usan diferentes longitudes de clave, pero ninguno de ellas es de 17 bytes (136 bits). Podríamos intentar recortar o rellenar con ceros la clave hasta que cuadre con las longitudes, pero no lograríamos nada. Tampoco logramos descifrar nada usando como clave diferentes hashes de la contraseña (por ejemplo, MD4, MD5, SHA-1, SHA-256, SHA-512, etc.)



Análisis del fichero cifradoverdadero con CrypTool

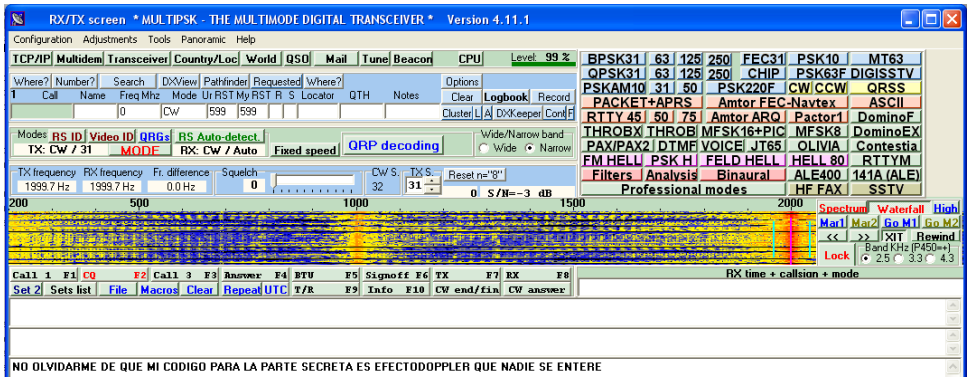
Si tenemos en cuenta el propio nombre de la prueba (*matrioska*, como las famosas muñecas rusas), podríamos deducir que se trata de un cifrado múltiple, a base de aplicar varias veces el mismo algoritmo, o incluso diferentes algoritmos, lo cual aumentaría bastante la dificultad de la prueba. Si tenemos en cuenta que la contraseña es de 17 caracteres (número primo) y que el único algoritmo (de los implementados en CrypTool) que admite una clave de un byte es el RC4, cabría la posibilidad de que esté cifrada 17 veces con este algoritmo, pero resulta que tampoco es así...

Nos encontramos en uno de los puntos más desesperantes de la prueba, porque para poder avanzar necesitamos alguna idea feliz. Si repasamos minuciosamente todas las diapositivas de la presentación, veremos que la última de ellas nos habla de cifrado de volúmenes con TrueCrypt (¿no suena esto a "cifrado verdadero"?), así que

procedemos a la descarga de esta aplicación y abrimos nuestro fichero con ella. Al hacerlo, el programa nos pide una contraseña y si le introducimos la que tenemos en nuestro poder lograremos montar una unidad virtual de disco que contiene un fichero (cuyo nombre es " = ", sin ninguna extensión).

Si le echamos un vistazo al contenido del fichero, enseguida nos damos cuenta de que la cabecera es de tipo ID3, un estándar muy utilizado para incluir metadatos en ficheros MP3, lo que nos lleva a abrir el fichero con algún reproductor de audio para ver si podemos escuchar alguna grabación (¿con la contraseña final?), pero no es así. Parece que esto no ha llegado a su fin, porque escuchamos una serie de tonos intermitentes, que tendremos que decodificar, pero antes de ello deberemos averiguar a qué algoritmo de codificación corresponden.

Para ello contamos con una pista que podíamos leer en la misma cabecera del fichero ("*El código de los mares*"), la cual unida con las propias características de la señal audible, nos lleva a deducir que se trata del famoso código Morse. Dependiendo de nuestras habilidades, podemos intentar ralentizar la velocidad de reproducción del audio y entretenernos decodificando el mensaje de forma manual, o bien podemos optar por alguna herramienta que lo haga por nosotros, como MultiPSK, vinculando la línea de entrada a de la salida de audio y seleccionando la opción CW (Morse). Mientras se reproduce el fichero, iremos obteniendo poco a poco su contenido legible.



Estaba convencido de haber obtenido la última de las contraseñas, puesto que ya no teníamos más elementos con los que hacer nada más, así que envié mi solución. Sin embargo, al día siguiente recibí como respuesta un mensaje inesperado, indicándome que la prueba no acaba ahí. ¿Aún hay más? Parece que algo se nos ha pasado por alto...

Al igual que en la primera fase obtuvimos un mensaje oculto en la imagen JPEG, no sería de extrañar que también se podrían haber aplicado técnicas de esteganografía en el fichero MP3, así que probamos con MP3Stego (herramienta que también aparece en las diapositivas de la charla). La propia contraseña podría haber sido una pista, puesto que el efecto Doppler trata precisamente de ondas como las del sonido. Sin embargo, esta opción tampoco nos llevará a ningún sitio.

Tras repasar uno por uno todos los elementos de la prueba, vemos que tal vez la apuesta más segura sea la del disco virtual, puesto que su tamaño es de 126 KB, de los que solamente tiene ocupados 68 KB. Intentamos recuperar ficheros borrados con alguna herramienta, pero tampoco encontramos nada, así que volcamos la imagen completa del disco (p. ej., usando ADRC recovery tools) para poder analizarla a fondo.

Efectivamente, observamos que contiene datos con formato irreconocible justo al final del mismo (a partir del byte 86.016). Las características de los datos son muy similares a los del fichero del "cifrado verdadero", es decir, parece que están cifrados con un algoritmo de los llamados modernos. Nos encontramos de nuevo en una situación en la que podríamos probar un buen número de algoritmos, a ver si con alguno de ellos tenemos suerte, pero precisamente esa similitud en el formato nos puede llevar a pensar que podría tratarse de otro disco virtual, tal vez cifrado también con el mismo TrueCrypt. Si separamos ese bloque de datos y lo intentamos abrir, no conseguiremos nada, así que tal vez sea hora de repasar las diapositivas de la charla otra vez más...

Precisamente al final de la diapositiva que trata de TrueCrypt aparece un punto que no podemos ignorar: *"Volúmenes ocultos: Añaden un interesante nivel adicional de seguridad"*. Si nos ponemos a investigar cómo se montan estos volúmenes ocultos con TrueCrypt, nos daremos cuenta que es tan simple como introducir la otra contraseña cuando se nos solicite. Efectivamente, tras introducir la otra (*"efectodoppler"*), se monta la unidad oculta, que contiene un único archivo llamado *"la_meta"*.

Dicho fichero contiene una serie de caracteres ASCII y está formado únicamente por las vocales "I" y "O", así que parece que el paso más lógico es asociar la letra "I" al bit 1 y la letra "O" al bit 0. Además, su longitud es de 432 bytes (divisible entre 8), lo que nos lleva a pensar que podría tratarse de algún tipo de fichero codificado de esa manera. Veamos lo que ocurre al agruparlo e interpretarlo en grupos de 8 caracteres:

Original	Binario	Decimal	ASCII
OIOOIIOO	0100 1100	76	**L**
OIOOOOOI	0100 0001	65	**A**
OOIOOOOO	0010 0000	32	
OIOIOOOO	0101 0000	80	**P**
OIOOOOOI	0100 0001	65	**A**
OIOOIIOO	0100 1100	76	**L**
OIOOOOOI	0100 0001	65	**A**
OIOOOOIO	0100 0010	66	**B**
OIOIOOIO	0101 0010	82	**R**
OIOOOOOI	0100 0001	65	**A**
…	…	…	…

Cuando parecía que por fin habíamos terminado la última fase, resulta que el texto deja de poder leerse tan solo tras unas posiciones más adelante del texto mostrado en la tabla, así que tras analizar el resto del fichero a ojo, todo indica que también se trata de caracteres ASCII legibles, pero desplazados. Si probamos con todos los desplazamientos posibles (7 opciones más), vemos que se puede leer una pequeña parte en el desplazamiento de 3 bits y la última parte en el desplazamiento de 6 bits. Juntando todas las piezas y con algo de imaginación, deducimos que la frase era la siguiente:

```
LA PALABRA SECRETA ES Sup3rc4l1fr4g1l1st1c03sp14l1d0s0
```

Seguramente, el fichero haya sido saboteado de forma manual, habiendo eliminado del mismo 8 caracteres de forma más o menos arbitraria, pero la cuestión es que al fin, ahora sí, podemos dar por concluida esta prueba. ¿Vamos a por la siguiente?

Prueba 12 (¡Menuda pitufada!)

En esta ocasión partimos de un fichero ZIP, que contiene varios ficheros y un par de subdirectorios, con la estructura que se muestra a continuación:

- **datos_confiscados_1**
  - clave.asc
  - datos.txt
  - passwd.txt
- **datos_confiscados_2**
  - Gnupg.png
  - Uhmmmm.7z
- Instrucciones.txt
- Mensaje.txt.asc

Tras leer las instrucciones y echarle un vistazo rápido a cada uno de los ficheros, parece que la estrategia a seguir está bastante clara. Nuestro objetivo es descifrar el mensaje (asumiremos que se trata del fichero *"Mensaje.txt.asc"*) utilizando los datos que están en nuestro poder. Hay dos secciones claramente legibles en el mensaje (la cabecera y el pie), estando el resto codificado en Base64:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.11 (GNU/Linux)

hQEMA7XCRi9iJU1EAQf/f62yJUiNUCQA73MEkm61bekRHYRkwQadJrLCJG2ve51O
y4BaikrTRDVvxyx1G16jCXWGRFlri5O3pMpIRxwMTsHlybE7ZF/  ...
-----END PGP MESSAGE-----
```

Si la prueba no está diseñada para despistarnos, está claro que se trata de un cifrado usando PGP (concretamente GnuPG, tal y como aparece en la cabecera). Este cifrado es de los de tipo asimétrico, en los que existe un par de claves relacionadas entre sí (denominadas "clave pública" y "clave privada"). Sin entrar en demasiado detalle, nos quedaremos con la idea de que el mensaje que se cifra con una de ellas se puede descifrar con la otra y viceversa, así que nuestro objetivo será conseguir la clave con la que está cifrado el mensaje.

Al inspeccionar el fichero *"clave.asc"*, comprobamos que su formato guarda cierta relación con el mensaje cifrado, pero que en este caso, al contrario que antes, resulta bastante ilegible, pero deducible:

```
-----ENPIT ELK RONGNIX ONQ AGLRW-----
Dohnbno: DrzBK v2.0.11 (BGZ/Pkqwh)
...
-----IKZ HQR XFCTKKM KDP ITELK-----
```

Parece que estamos ante el fichero que almacena una de las claves y tal vez sea nuestro objetivo más claro. Veamos si logramos descifrarlo, asumiendo que el fichero original debería contener las siguientes cadenas:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v2.0.11 (GNU/Linux)
...
-----END PGP PRIVATE KEY BLOCK-----
```

Tras descartar el cifrado César (en el que cada letra tendría una única correspondencia) y el de Vigenère (ya que requeriría una clave demasiado larga), nos centramos en el análisis de las características del texto. Si obtenemos el histograma de ambas cadenas (cifrada y sin cifrar), usando CrypTool, por ejemplo, veremos que coinciden plenamente (sin tener en cuenta la parte codificada en Base64). Esto es un claro indicador de que el mensaje está cifrado con un algoritmo de transposición o permutación. Sin haber llegado a profundizar en el criptoanálisis, conseguimos descifrar el mensaje con una simple prueba con la clave "3,1,2", con las opciones por defecto de CrypTool (la entrada por líneas, la permutación y la salida por columnas):

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v2.0.11 (GNU/Linux)

lQO+BEn4SKQBCACTG61DzL7OU+Rklw5lRdLj39OFQ4hojNxZ/ByBN9TXOYHdorrZ
d+zcbbIVKDcVPQBKIGLlzE8t0p1h8pk8GLQWywUK1HCL5LU6vY6nQgik  ...
-----END PGP PRIVATE KEY BLOCK-----
```

Aunque no era estrictamente necesaria, también teníamos una pista en el fichero *"datos.txt"*, ya que en él aparece una frase bastante relacionada con la transposición e incluso podríamos intuir las siglas RSA a partir de los tres nombres que aparecen ahí.

Si importamos la clave y posteriormente intentamos descifrar el mensaje, se nos solicitará una frase de contraseña que aún no tenemos. Por lo menos sabemos que la clave privada es la que buscamos puesto que es la que ha cifrado el mensaje:

```
> gpg --import clave_descifrada.txt
  clave FA382338: clave secreta importada
  clave FA382338: clave pública "Pitufo Genio <pitufogenio@vierito.es>" importada

> gpg Mensaje.txt.asc
  Necesita una frase contraseña para desbloquear la clave secreta
  del usuario: "Pitufo Genio <pitufogenio@vierito.es>"
  clave $s de $u bits, ID $s, creada el $s(ID de clave primaria FA382338)

  Introduzca frase contraseña:
```

Como la contraseña podría estar en cualquiera de los ficheros confiscados, analizamos los demás ficheros. En primer lugar, descartamos que la imagen contenga datos ocultos, tras cotejarla con otras imágenes idénticas en Internet y por no tener paleta de colores, ni datos anexos fuera del marcador que define el estándar PNG.

Hablando de datos anexos, si analizamos el contenido del fichero comprimido en formato 7z, veremos que tiene una secuencia hexadecimal (legible en ASCII) bastante larga que no pertenece a dicho formato. Si generamos un nuevo fichero a partir de dicha cadena en binario, obtenemos una cadena en Base64. Si la decodificamos, obtenemos un fichero con la clave pública:

```
4c5330744c5331435255644a5469425152314167554656435445 6c4449457446575342435445394 45
37930744c533074445170575a584a7a615739754f694248626e5651527942324 ...
→
LS0tLS1CRUdJTiBQR1AgUFVCTElDIEtFWSBCTE9DSy0tLS0tDQpWZXJzaW9uOiBHbnVQRyB2Mi4wLjExI
ChHTlUvTGludXgpDQoNCm1RRU5CRW40UitBQkNBRElldGF6YmR1emdxWWw1MWtd6 ...
→
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.11 (GNU/Linux)

mQENBEn4R+ABCADIetazbduzgqYl51kbXau/UYjcCYZCaZZ508aF9U4MMktF1GlY
92YVMrCsI66ZjMVW11h15VBQZia7IV33Toh7to7E4ooNt3SwYg ...
-----END PGP PUBLIC KEY BLOCK-----
```

De momento, dicha clave no nos resulta de ninguna utilidad para descifrar el mensaje. De hecho, todo el contenido del directorio *"datos_confiscados_2"* no sirve para nada, más que para hacernos perder el tiempo (o para divertirnos un rato, según como se mire…).

La cuestión es que solamente nos queda el fichero *"passwd.txt"*, que contiene una secuencia de 180 dígitos. Si la observamos con atención, nos daremos cuenta de que en realidad muy probablemente se trate de una secuencia de 60 números de 3 dígitos, comprendidos entre 141 y 171 (ambos inclusive):

```
156 155 145 141 147 163 165 154 153 152 163 141 164 156 144 157 141 151 154 171
165 143 141 167 141 143 163 144 160 151 151 141 142 166 163 151 164 152 144 143
165 156 146 167 151 165 151 165 165 144 143 154 156 157 157 163 150 165 141 144
```

Al principio intenté sumar o restar algún valor fijo para relacionarlos con valores ASCII. Luego intenté asignar cada uno de estos valores a una letra, secuencialmente. Sin embargo, al final me di cuenta de que a la secuencia le faltaban algunos dígitos (no contiene ningún 8, ni tampoco 9), por lo que cabía la posibilidad de que estuviera codificado en octal. Y asumiendo eso, aquí tenemos su correspondiente decodificación:

```
nmeagsulkjsatndoailyucawacsdpiiabvsitjdcunfwiuiuudclnooshuad
```

Tras analizar la frecuencia de aparición de cada carácter, la aleatoriedad de los datos y un montón de factores más, habiendo probado con todo tipo de algoritmos de cifrado clásico sin éxito, al final (tras haber malgastado un montón de horas de auténtica desesperación) su descifrado resultó ser mucho más fácil de lo que parecía y, además, tenía cierta relación con el resto de la prueba. Es bien sabido que las claves asimétricas están basadas en números primos, así que veamos lo que sucede si nos quedamos únicamente con las posiciones correspondientes a dichos números:

```
         11111111112222222222333333333444444444555555555 6
12345678901234567890123456789012345678901234567890123456789 0
nmeagsulkjsatndoailyucawacsdpiiabvsitjdcunfwiuiuudclnooshuad
 me g u   s t   a l   a     p i     t   u f   i     n     a
```

Afortunadamente, la frase parece ser coherente con la prueba y tener sentido (*"Me gusta la Pitufina"*). Si la introducimos como frase de contraseña (tal cual se obtiene, en minúsculas y sin espacios), lograremos descifrar y leer claramente el mensaje, lo que resulta ser el paso final de la prueba. ¡Lo tenemos!

```
Saludos Pitufo Genio,

Tengo una misión para tí. Vamos a pitufar la escuela de teleco. Tienes de tiempo
hasta el 08/05/09 a las 23:59 hora smurf para la finalizar con éxito la operación
Gargamel. Tendrás que colarte como espía y experto en seguridad informática,
gracias tus dotes pitufantes, entre los altos cargos de la escuela y obtener los
recursos necesarios para poder aprobar PTCs de los Erasmus y permitir una puerta
trasera para que yo pueda irme de Erasmus sin cumplir con todos los requisitos
(porque siempre me suspenden Campos). Ten cuidado con Azrael.

En tus manos lo dejo Pitufo Genio, nada de estar ahora pensando en Pitufina, ¡que
se te va la olla!

Saludos pitufantes,

Papa Pitufo
```

Agradecimientos y comentarios

A pesar de no haberme enterado de la existencia del reto hasta que éste se dio por cerrado, tengo que reconocer que todas las pruebas tenían un planteamiento bastante razonable y entretenido. Si el objetivo era el de despertar el interés por la criptografía y la esteganografía, creo que se ha logrado con creces. También podemos sacar alguna que otra conclusión de todo esto, ya que hemos visto que de nada sirve aplicar algoritmos de cifrado fuerte si al final vamos a dejar la contraseña apuntada justo al lado (ya sea en un fichero de texto o pegada en un "post-it")

Por último, solo me queda dar la enhorabuena a todos los que participasteis en el reto, por supuesto, agradeciendo también el excelente trabajo a los organizadores, Amine Taouirsa (*metalamin*) y Javi Moreno (*vierito5*), tanto por el reto como por las diapositivas y la charla. A ver si el año que viene contamos con una próxima edición.

Saludos,

Daniel Kachakil
dani@kachakil.com