novática
Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software). ***Novática*** co-edita asimismo **UP**GRADE, revista digital de **CEPIS** (*Council of European Professional Informatics Societies*), en lengua inglesa, y es miembro fundador de **UP**ENET (***UP****GRADE* **E***uropean* **NET***work*).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>
<http://www.cepis.org/upgrade>

ATI es miembro fundador de **CEPIS** (*Council of European Professional Informatics Societies*) y es representante de España en **IFIP** (*International Federation for Information Processing*); tiene un acuerdo de colaboración con **ACM** (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn. 963740173 <novatica_prod@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª. 08003 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B. 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia <secregal@ati.es>
Suscripción y Ventas <http://www.ati.es/novatica/interes.html>, ATI Cataluña, ATI Madrid
Publicidad Padilla 66, 3º, dcha., 28006 Madrid.
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: Malvavisco - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

sumario

Nº 215, enero-febrero 2012, año XXXVIII

Tema del próximo número: "Informática y cultura"

Análisis de la seguridad del sistema reCAPTCHA

Noemí Carranza[1], Ricardo Palma Durán[1], Gonzalo Álvarez Marañón[1], José María Gómez Hidalgo[2]

[1]Departamento de Tratamiento de la Información y Codificación, Instituto de Seguridad de la Información, Consejo Superior de Investigaciones Científicas; [2]Director de I+D de Optenet, Coordinador de la sección técnica "Acceso y recuperación de la información" de ***Novática***

<{noemi.carranza,ricardo.palma,gonzalo}@iec.csic.es>, <jgomez@optenet.com>

Resumen: *En los últimos tiempos se han popularizado extraordinariamente los sistemas CAPTCHA, que protegen servicios Web planteando al usuario una prueba destinada a verificar que se trata de un ser humano y no de un robot, o sistema automático para el envío de correo basura o difusión de malware. Estos sistemas están siempre expuestos a que spammers y hackers sean capaces de comprometer su seguridad, y abusar de los recursos subyacentes (cuentas de correo, blogs, etc.) para realizar sus actividades ilícitas. Por ello, es necesario comprobar periódicamente su seguridad usando herramientas como sistemas de reconocimiento óptico de caracteres (OCR), sistemas de análisis de imagen, y otras. En este artículo realizamos un análisis de la seguridad del sistema reCAPTCHA, que probablemente es el más usado en Internet actualmente. Para ello, utilizamos diversas técnicas de análisis de imagen orientadas a corregir las deformaciones y distorsiones realizadas por el sistema en las imágenes que muestra al usuario, así como el eficaz sistema de OCR Tesseract. Se han analizado dos versiones del sistema reCAPTCHA y se ha comprobado que la seguridad del sistema probablemente ha aumentado en la segunda versión, más reciente, aunque es posible comprometer la seguridad del sistema si se cuenta con recursos suficientes en forma de una botnet de tamaño medio (unos 10.000 ordenadores).*

Palabras clave: *CAPTCHA, correo basura, Heuristic Over Segmentation, reCAPTCHA, seguridad web, spam, Shape Context, Tesseract.*

1. Introducción

En 1997, Andrei Broder y sus colegas en el motor de búsqueda Altavista desarrollaron el primer CAPTCHA como sistema de control de acceso a una aplicación Web, el formulario de alta de URLs en el índice de dicho motor de búsqueda. Desde entonces, los CAPTCHAs (*Completely Automated Public Turing test to tell Computers and Humans Apart* – Test de Turing completamente automático para separar humanos y computadoras) se han convertido probablemente en el método más popular para la protección de servicios *online* frente a abusos por parte de robots, es decir, sistemas automáticos que rellenan formularios web de suscripción, realizan comentarios en blogs, o recolectan recursos para difundir correo basura, software malicioso, etc. [1]. Un CAPTCHA es una prueba automática orientada a confirmar que el usuario de una aplicación es un ser humano en lugar de un robot [2], y generalmente toma la forma de una palabra o una secuencia de caracteres distorsionados que el usuario debe acertar para acceder a un determinado recurso, como por ejemplo, a la posibilidad de realizar un comentario en una Red Social como Facebook. Se espera que la palabra distorsionada sea muy difícil de averiguar para un sistema automático, y al mismo tiempo sencilla para una persona. De este modo, la prueba habilita el acceso al servicio protegido sólo a los seres humanos, y evita el acceso y el abuso por parte de los robots o sistemas automáticos.

Dado que los *spammers* y los *hackers* están altamente motivados para abusar de los servicios *online* con el fin de realizar actividades dañinas, están continuamente diseñando métodos para resolver los CAPTCHAs de manera automática. Se ha demostrado tanto de manera teórica ([3][4]) como práctica[1] que sus métodos pueden tener éxito, por lo que los diseñadores de CAPTCHAs están obligados a evolucionar sus técnicas para aumentar su dificultad sin perder usabilidad por parte de las personas. En otras palabras, los CAPTCHAs deben evaluarse periódicamente usando sistemas de reconocimiento óptico de caracteres (OCR, *Optical Character Recognition*) estándar, y técnicas de procesamiento de imagen actuales, con el fin de mantener su seguridad.

El objetivo de este artículo es presentar los resultados de una evaluación de estas características aplicada al sistema CAPTCHA más popular y difundido de la actualidad, el sistema reCAPTCHA.

2. El sistema reCAPTCHA

Hoy en día existen múltiples tipos de CAPTCHAs (de texto, de imagen o video, auditivos, cognitivos, etc. - véase [1] para una revisión exhaustiva), aunque los más populares son con diferencia los de texto, es decir, los que muestran una secuencia de caracteres en una imagen distorsionada utilizando diversas técnicas: adición de ruido, deformaciones, etc. y solicitan al usuario que introduzca dicha secuencia en un campo de texto de un formulario.

Si hay un CAPTCHA de texto singularmente importante por su difusión y su utilidad, ése es desde luego el sistema reCAPTCHA[2] [2][5]. Este sistema fue desarrollado por un equipo de investigadores de la Carnegie Mellon University, y ya en 2008 estaba desplegado en más de 40.000 sitios web, y había sido resuelto por humanos más de 1.200 millones de veces

Figura 1. Un ejemplo reciente del sistema reCAPTCHA.

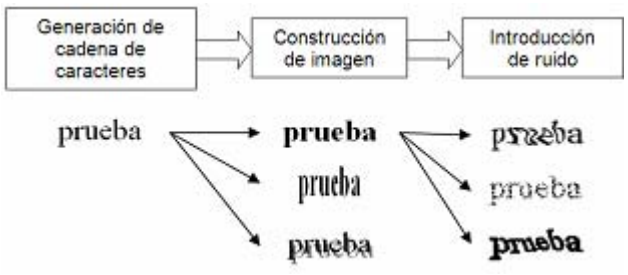


Figura 2. Proceso de generación de un CAPTCHA de texto.

en un año. Posteriormente ha sido adquirido por Google[3], lo que ha contribuido aún más a su popularidad.

El sistema reCAPTCHA funciona como un servicio que se puede sindicar en cualquier sitio Web con unas pocas líneas de código, y que muestra al usuario dos palabras que debe introducir en el campo de texto para acceder al recurso protegido (ver **figura 1**). Mientras que muchos otros sistemas CAPTCHA exigen al usuario la introducción de una sola secuencia de caracteres, este sistema exige dos, una de las cuales ha sido generada automáticamente, y la otra ha sido obtenida como resultado de la aplicación de un OCR sobre un texto antiguo. La palabra generada automáticamente actúa como texto de control, mientras que la otra se usa para corregir los errores de reconocimiento del OCR.

De este modo, se aprovecha el tiempo invertido por los seres humanos en la resolución de reCAPTCHA para digitalizar manuscritos antiguos, en lugar de desperdiciar todo ese tiempo de "computación humana". De acuerdo con [5], en un solo año se han digitalizado el equivalente a 17.600 libros transcritos manualmente con un índice de éxito superior al 99%. Esta utilidad "solidaria" hace aún más popular e interesante al sistema reCAPTCHA. Las palabras transcritas manualmente por los usuarios de reCAPTCHA pertenecen al archivo del periódico New York Times, o a libros disponibles en Google Books.

3. Comprobando la seguridad de los CAPTCHAs

Los CAPTCHAs han sido objeto de continuos ataques, muchos de ellos con éxito, durante toda su existencia [1]. De ahí que sea esencial comprobar periódicamente que un CAPTCHA particular es resistente a los ataques y que sus creadores lo perfeccionen si los ataques tienen éxito.

Una pregunta esencial es qué porcentaje de éxito debe tener un ataque para que se considere un CAPTCHA inseguro. Supongamos que un sistema automático tiene la capacidad de resolver un CAPTCHA con un porcentaje de éxito de un 0,01%, o en otras palabras, es capaz de resolver automáticamente un CAPTCHA una de cada 10.000 veces. Esto significa que si el atacante dispone de una *botnet* (red de ordenadores bajo su control infectados por medio de un virus) de 10.000 unidades[4], con 10 hilos por unidad, es capaz de resolver unos 10 CAPTCHAs por segundo, lo que equivale a la creación de 864.000 cuentas de correo falsas al día si se está atacando el formulario de registro de un sistema de Webmail como Gmail o Hotmail [4]. En otras palabras, una tasa de resolución automática que intuitivamente se puede considerar muy moderada, puede en la práctica convertir a un sistema CAPTCHA en tremendamente inseguro.

Existen múltiples formas de intentar resolver automáticamente un CAPTCHA, que dependen obviamente del tipo de sistema utilizado (imágenes de texto, vídeo, audio, etc.), así como de otros factores como la propia seguridad de su implementación. Por ejemplo, para atacar el sistema reCAPTCHA no es necesario intentar reconocer las palabras de la imagen, ya que es posible atacar su funcionalidad de audio, que se presenta en la **figura 1** a través del símbolo del altavoz, para resolver el reto planteado por el sistema. Esta funcionalidad, destinada a hacer el CAPTCHA accesible para sus usuarios con dificultades visuales, podría en consecuencia debilitar de manera crítica la seguridad del sistema si es comprometida.

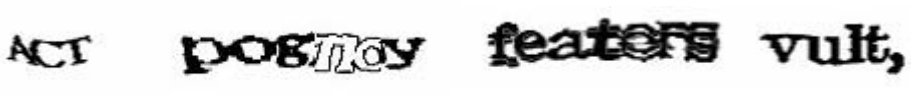
Dado que los CAPTCHAs más populares son los de texto, y que el sistema particular en el que nos concentramos utiliza esta modalidad como vía principal de uso, trataremos el ataque a estos sistemas. El ataque a un sistema CAPTCHA de texto consiste, en la práctica, en una inversión de su proceso de generación. En la **figura 2** se muestran los pasos habituales para la generación de un CAPTCHA de texto, que serían los siguientes:

1) Generación de la cadena de caracteres que se utilizará como test para el usuario. Esta cadena puede ser aleatoria, constar de números, letras y otros caracteres, o puede ser obtenida a partir de un diccionario.
2) Obtención de una imagen inicial a partir de la cadena, que puede incluir ya perturbaciones a nivel de carácter, como el solapamiento de los caracteres que la forman, creación de sombras artificiales, o uso de distintos colores para cada carácter.
3) Introducción de ruido en la imagen generada, como ruido gausiano (pixeles de distintos colores que disminuyen el contraste entre la cadena y el fondo), formas en negativo (como la que aparece en la cadena "gerYou" de la **figura 1**), deformaciones y rotaciones, etc.

Por analogía inversa, el proceso de ataque consta de los siguientes pasos:
1) Reducción del ruido de la imagen, como la aplicación de un algoritmo de binarización (paso a dos colores) para la identificación de áreas de interés (posibles caracteres), eliminación de píxeles aislados, etc.
2) Segmentación de la imagen en potenciales caracteres aislados, proceso que se puede realizar de diversas formas en función de las perturbaciones. Por ejemplo, se puede segmentar usando líneas imaginarias verticales que mantienen el color de fondo, si los caracteres no están solapados y no hay líneas que los unan.
3) Identificación de los caracteres aislados en cada fragmento de imagen, y su combinación para formar una palabra con sentido si la imagen original se ha generado a partir de un diccionario lingüístico.

Obviamente, los pasos que se dan para atacar un CAPTCHA dependen de los pasos que se han dado para crearlo. Por ejemplo, si no se han utilizado colores en su creación (como es el caso de reCAPTCHA), tampoco es preciso el proceso de binarización o paso a dos colores.

Siguiendo procesos similares a los anteriores, la seguridad de muchos CAPTCHAs ha sido comprometida. Por ejemplo, Mori y Malik [3] utilizaron una secuencia de pasos similar a la anterior para comprometer el sistema EZ-Gimpy, utilizado por Yahoo!, con un éxito de un 83% (158 sobre 191 casos de prueba), lo cual acabó obligando a dicha empresa a descartar el sistema.



(a) (b)

Figura 3. Versiones de reCAPTCHA analizadas en este trabajo.

Figura 4. Proceso de corrección de la deformación del texto: imagen original, proceso para estimar la ondulación, e imagen final.

4. Método de análisis de reCAPTCHA

4.1. Versiones de reCAPTCHA analizadas

Analizaremos aquí dos versiones recientes de reCAPTCHA, concentrándonos en el ataque a las imágenes proporcionadas por el sistema, que se muestran en la **figura 3**. En la primera versión (**3.a**), los caracteres utilizados en el texto están difuminados de manera que no se puedan separar, y en ocasiones, se muestra sólo el borde de las letras de las palabras. En la segunda versión (**3.b**), los caracteres usados también se unen con el fin de dificultar el proceso de aislamiento, y una de las palabras se "duplica" formando una sombra variable. En esta versión, la palabra no duplicada es usada para la corrección de OCR, mientras que la duplicada es el texto de control generado automáticamente.

4.2. Descripción general del proceso

Con el fin de analizar las versiones anteriores de reCAPTCHA, se ha procedido a implementar un proceso inverso al de generación de las imágenes, consistente en corregir las deformaciones y distorsiones introducidas y luego detectar los caracteres utilizados en las mismas. Cada palabra individual se ha tratado siguiendo los procesos siguientes:

- Aplicación del reconocedor óptico de caracteres Tesseract directamente sobre la imagen.
- Corrección de las deformaciones e inclinaciones de la imagen, y aplicación de Tesseract.
- Alternativamente, utilización de los algoritmos *Heuristic Over Segmentation* (HOS) y *Shape Contexts* (SC) sobre la imagen original, para la detección de los caracteres utilizados en ella.

Tesseract[5] (v.3.01) es un sistema reconocedor óptico de caracteres de software libre considerado actualmente entre los más eficaces. Una de sus características más importantes es la utilización de múltiples diccionarios, siendo uno de los más extensos el del idioma inglés.

El proceso de corrección de deformaciones e inclinaciones de la imagen, desarrollado con el fin de aumentar la eficacia de Tesseract, y los algoritmos HOS y SC han sido implementados en MatLab, y se explican en las próximas secciones.

4.3. Pre-procesamiento de la imagen

El pre-procesamiento de la imagen, orientado a corregir las deformaciones más básicas, consta de los siguientes pasos:

1) Se intenta corregir la deformación global de las palabras de la imagen, calculando la línea horizontal central de la imagen, y transformando la imagen para que dicha línea sea recta. Este proceso se puede ver en la **figura 4**. Para ello, se obtiene la línea de máximos en la imagen, la línea de mínimos y la línea central, que será la que dé la pauta para corregir la ondulación de la figura. A la línea central se le aplican varios procesos de filtrado para suavizarla, debido a que letras como "p" o "l" que son más altas o bajas que el resto pueden dar lugar a deformaciones erróneas. De esta forma, se logra de forma suavizada la deformación de la palabra en global.

2) Se procede a corregir la inclinación de los caracteres, para lo que se construye una serie de imágenes rotadas en distintos ángulos. De estas imágenes obtenidas, se realiza una proyección horizontal de todas ellas, y se escoge aquella en que esta proyección tiene un menor número de píxeles (en este caso, la primera de las cuatro que se muestra en la **figura 5a**). Tras elegir esta proyección, se obtiene el ángulo que se ha rotado y se puede corregir la imagen de la misma forma en que se modificó la ondulación de los caracteres, corrigiendo este ángulo de rotación, de forma que se bajan las letras en función del ángulo obtenido.

El resultado final de corregir ondulaciones e inclinaciones de las letras es el que se muestra en la **figura 6**.

4.4. Aislamiento de caracteres con el método HOS

Tanto en la primera versión de reCAPTCHA, pero sobre todo en la segunda, las correcciones anteriores son insuficientes ya que se ha introducido el duplicado de la palabra para dificultar en gran medida el aislamiento de los caracteres individuales, que es el paso fundamental en el ataque a un CAPTCHA. Este proceso en cuestión se suele denominar segmentación, y es habitualmente el proceso más complejo en los procesos de OCR.

Una técnica para realizar la segmentación, que con el tiempo se ha convertido prácticamente en un estándar, se denomina *Heuristic Over Segmentation* (HOS) [6]. Esta técnica consiste en la generación de un gran número de cortes potenciales entre los caracteres, usando métodos heurísticos de procesamiento de imagen, y la selección posterior de la mejor combinación de cortes de acuerdo con las probabilidades o resultados numéricos arrojados por un OCR. La eficacia de esta técnica depende, por tanto, de la calidad de los cortes generados, así como de la habilidad del OCR para distinguir correctamente los po-

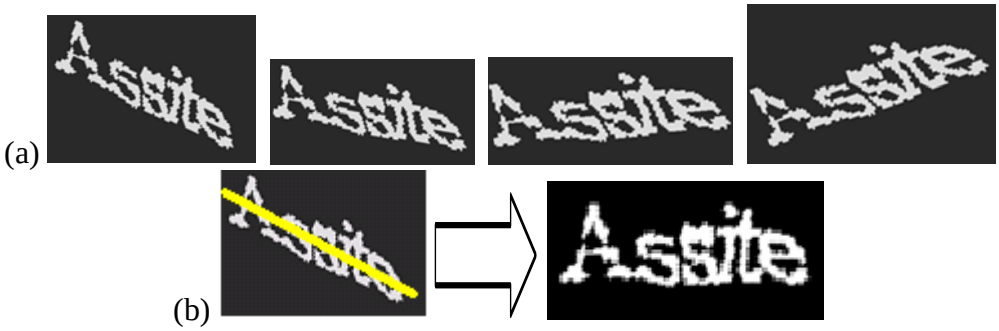


Figura 5. Proceso de corrección de la inclinación de los caracteres. Cuatro de las rotaciones obtenidas de la imagen (a), y corrección de la inclinación de la cadena a partir de la seleccionada (b).

Figura 6. Resultado global del proceso de corrección de la inclinación del texto.

tenciales caracteres segmentados a partir de sus partes.

Para determinar los posibles cortes, se analiza la palabra mostrada en la imagen y se establece un posible corte con un valor cuando se detecta un cambio abrupto en la parte superior o inferior del perfil horizontal (altura) de la misma. Cuando se encuentra al mismo tiempo un cambio abrupto en el perfil superior e inferior, el valor se duplica para ese posible carácter.

En la **figura 7** se muestran los posibles cortes para una palabra de ejemplo. Se han representado dichos cortes correspondientes tanto a cambios abruptos en el perfil superior como en el inferior, usando líneas amarillas.

El proceso heurístico de generación de cortes se ha diseñado con el objetivo de obtener más cortes de los necesarios, en la esperanza de que el conjunto correcto de cortes esté incluido entre los generados. Una vez que se han obtenido los posibles cortes, las segmentaciones candidatas se representan a través de un grafo denominado grafo de segmentación. Se trata de un grafo acíclico dirigido con un nodo de inicio y un nodo final. A cada nodo interno se le asigna un corte candidato producido por el algoritmo de segmentación. Cada arco entre dos nodos se asocia con una imagen que contiene todos los píxeles entre el corte asociado al nodo origen, y el corte asociado al nodo destino de dicho arco [6].

Cada arco lleva asociado un valor que se determina utilizando la distancia entre histogramas, que se explicará en la sección siguiente sobre *Shape Contexts*, que es la forma en que se estima la distancia de cada "corte" con una letra del alfabeto. Por ejemplo, suponiendo que hay cuatro posibles cortes como se muestra en la **figura 8**, con los distintos pesos de ir de un nodo a otro se puede computar el camino mínimo y obtener el resultado final de la palabra. En este caso, el camino mínimo sería el formado por los nodos 1-3-4, con peso total 2.

Figura 7. Cortes potenciales entre caracteres para una palabra de ejemplo.

4.5. Cálculo de similitud con el método Shape Contexts

La contribución principal en el trabajo de Belongie *et al.* [7] es un algoritmo robusto y simple para encontrar correspondencias entre formas. Las formas se representan con un conjunto de puntos muestreados de las formas, normalmente unos cien puntos. El descriptor de forma, llamado *Shape Context* (SC, Contexto de Forma), se utiliza para describir la distribución del resto de la forma con respecto a un punto determinado de la misma. Encontrar las correspondencias entre dos formas equivale a encontrar para cada punto muestreado en una forma el punto en la otra forma que tiene un SC más parecido.

En nuestro caso, hemos realizado 62 modelos de caracteres (26 letras del alfabeto, 26 letras en mayúscula y 9 dígitos), con 100 puntos elegidos para cada uno de los caracteres, que se comparan con el modelo que proviene de cada carácter de la imagen obtenida en reCAPTCHA.

Para cada punto p_i de la primera forma, se quiere encontrar el punto q_j más similar que se corresponda con la segunda forma. Consideramos un conjunto de vectores originados de un punto hacia todos los otros puntos de la forma. Estos vectores expresan la configuración de la forma entera relativa al punto de referencia.

Se calcula un histograma, desde el punto p_i a los *n-1* puntos restantes, usando la siguiente fórmula:

$$h_i = \#\{q \neq p_i : (q - p_i) \in bin(k)\}$$

Este histograma se define como el SC de p_i. Se ha utilizado un sistema de coordenadas log-polar, para que el descriptor sea más sensible a diferencias en los píxeles más cercanos (ver ejemplo en **figura 9**). Además, se han normalizado los histogramas para que se puedan utilizar con un número diferente de puntos.

Para comparar distintos histogramas se ha usado la distancia X^2, tal y como se muestra en la siguiente ecuación:

$$d(h_i, h_j) = \sum_{bins\ k} \frac{\|h_i(k) - h_j(k)\|^2}{\|h_i(k)\| + \|h_j(k)\|}$$

Éste es el número que se utilizará en el ejemplo de la **figura 8** para hallar las distancias de un nodo a otro.

5. Experimentos y resultados

5.1. Primera versión: bordes y deformaciones

Para la primera versión de reCAPTCHA, con bordes y deformaciones, se han obtenido 100 ejemplares que se corresponden con 200 palabras, y se han evaluado utilizando dos métodos. Por un lado, se ha aplicado el sistema OCR Tesseract de Google a las imágenes originales, y a las imágenes pre-procesadas para corregir las inclinaciones y deformaciones del texto. Por el otro, se han aplicado los métodos HOS y SC a las imágenes originales.

En la **figura 10** se muestran los resultados de ambos métodos. En el gráfico de la izquierda se muestra el porcentaje de éxito del OCR Tesseract con las imágenes originales y preprocesadas. Como se puede observar, dicho OCR es capaz de identificar correctamente un 4,5% de las palabras presentes en las imágenes originales, y más de un 10% cuando se han preprocesado. Cabe reseñar que Tesseract hace

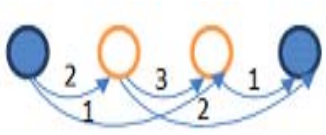


Figura 8. Ejemplo para determinar el proceso de selección de cortes y la palabra final.

Figura 9. Histograma en un punto de la letra A.

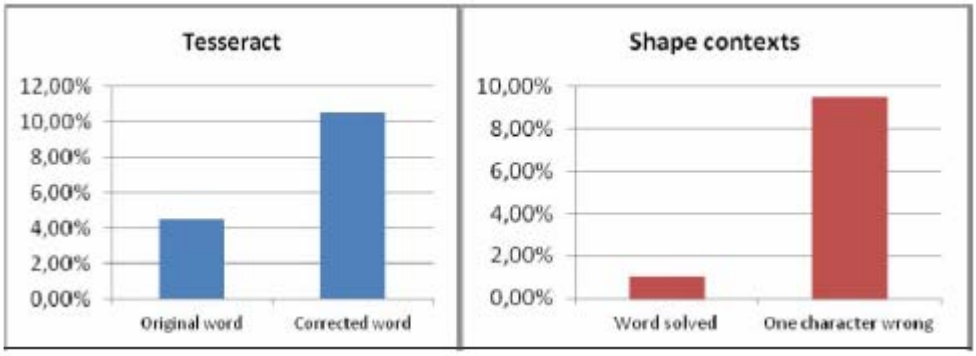


Figura 10. Resultados sobre el primer tipo de reCAPTCHA.

uso internamente de un diccionario del idioma inglés, lo que le permite identificar correctamente una palabra aunque no sea capaz de acertar todos los caracteres de la misma, y que la mayoría de las palabras usadas en reCAPTCHA están en idioma inglés.

Por otra parte, en el gráfico de la derecha se muestran los resultados de la aplicación de la técnica SC. Con esta técnica, se identifican correctamente un 1% de las palabras originales, y en casi un 9,5% de las palabras se falla un solo carácter. En consecuencia, con la ayuda de un diccionario de inglés, sería posible identificar correctamente hasta un 9,5% de las palabras.

Es importante reseñar que estos grados de éxito no se corresponden directamente con los de resolución de reCAPTCHA, dado que éste muestra dos palabras elegidas independientemente, y sólo una de ellas es la palabra de control. Si, por ejemplo, con el uso del preprocesamiento y del OCR es posible acertar un 10% de las palabras, entonces podemos estimar un grado mínimo de acierto de las dos palabras de un 1% del CAPTCHA, pero el grado de resolución de reCAPTCHA puede alcanzar hasta un 10% si casualmente se acierta siempre la palabra de control, y en media puede alcanzar un 5% (es decir, la mitad de las veces que se acierta la palabra, ésta es la de control).

5.2. Segunda versión: palabras duplicadas

Para el análisis del segundo tipo de CAPTCHAs, también se han recolectado 100 ejemplares correspondientes a 200 palabras. En este tipo de reCAPTCHA, la palabra duplicada es siempre la palabra de control, por lo que es posible establecer una correspondencia más directa entre el número de palabras duplicadas resueltas y el de ejemplares de reCAPTCHA resueltos.

En la **figura 11** se muestran los resultados de aplicar Tessereact sobre la palabra original, sobre la palabra pre-procesada, y sólo sobre las palabras duplicadas. A la derecha se muestran los resultados de aplicar Shape Context, y en particular, el porcentaje de veces que se resuelve la palabra completa, el porcentaje en que se falla sólo en un carácter, y el porcentaje de acierto sobre las palabras duplicadas. Como en el caso anterior, Tesseract hace uso de un diccionario del inglés, mientras que Shape Context no.

Como se puede observar, el OCR es capaz de identificar correctamente un 20% de las palabras originales. Sin embargo, el pre-procesamiento no permite mejorar los resultados, ya que las palabras que se muestran no sufren otra deformación que la propia duplicación de la palabra de control. Sin embargo, si nos concentramos en los resultados sobre las palabras duplicadas, se puede observar que se obtiene un grado de acierto de un 1%, y dado que se trata de la palabra de control, se puede afirmar que el CAPTCHA se rompe en un 1% de las ocasiones.

Por lo que respecta a los resultados de Shape Contexts se puede observar que sólo es capaz de resolver un 0,5% de las palabras, alcanzando un 3% de aciertos cuando se falla sólo un carácter, y que su grado de acierto sobre las palabras duplicadas es del 0%. En conclusión, los resultados son sustancialmente peores que en el caso del primer tipo de reCAPTCHA, y no se puede concluir que esta técnica sea capaz de comprometer esta segunda versión.

6. Conclusiones

Para mantener la seguridad de los populares CAPTCHAs de texto, es necesario evaluar su resistencia a las técnicas más actuales de reconocimiento de imágenes. En este artículo se han evaluado dos versiones del sistema reCAPTCHA, uno de los más populares en la actualidad, usando dos técnicas diferentes. La primera versión, que incluye alteraciones en los caracteres para mostrar sólo su contorno, es vulnerable a la utilización del OCR Tesseract (capaz de resolver con éxito un 10% de las palabras), y a la técnica Shape Context (capaz de resolver con éxito un 1% de las palabras sin ayuda de un diccionario). Sin embargo, casi todas las palabras resueltas no

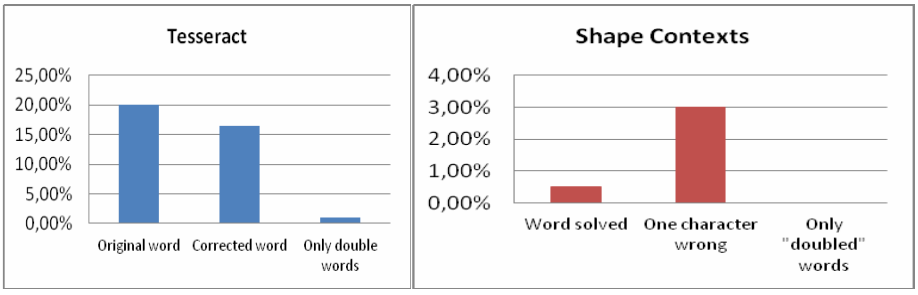


Figura 11. Resultados sobre el segundo tipo de reCAPTCHA.

habían sido alteradas con contornos, que suelen ser en torno a un 20% de las mostradas en reCAPTCHA. Además, no es posible conocer exactamente la tasa de éxito en la resolución de CAPTCHAs, porque no es posible conocer a priori cuál es la palabra de control.

En la segunda versión de reCAPTCHA, que utiliza sombras para duplicar las palabras de control, se ha conseguido resolver un 20% de las palabras usando Tesseract, y un 0,5% con Shape Contexts. Centrándonos sólo en la palabra de control, Tesseract es capaz de resolver un 1% de los reCAPTCHAs, mientras que Shape Contexts se queda en el 0% de éxito.

El proceso previo necesario para poder aplicar Shape Contexts consiste en la segmentación de la palabra en sus caracteres, que es una de las principales diferencias entre reCAPTCHA y otros CAPTCHAs, en los que se muestran los mismos completamente separados [3]. Por tanto, la técnica de segmentación HOS descrita en este artículo es esencial.

Por otra parte, Tesseract ha demostrado ser bastante robusto en las pruebas realizadas en este trabajo. Sin embargo, si las palabras utilizadas en el CAPTCHA no aparecen en el extenso diccionario de inglés usado por este programa, cabe esperar una reducción importante en la tasa de éxito de este sistema. En cualquier caso, el pre-procesamiento usado en este trabajo puede marcar la diferencia entre romper o no romper un CAPTCHA.

Finalmente, sobre la evolución de las versiones de reCAPTCHA usadas en este artículo, se puede señalar que se resuelven más palabras en el segundo caso, pero el número de ejemplos de reCAPTCHA es inferior, por lo que se puede concluir que la seguridad del sistema ha aumentado. Sin embargo, los atacantes pueden deducir fácilmente que la palabra de control es la duplicada, y ahorrar bastante tiempo tratando de resolver sólo dicha palabra, aumentando por tanto las probabilidades de comprometer la seguridad del sistema.

Referencias

[1] J.M. Gómez Hidalgo, G. Álvarez Marañon. CAPTCHAs: An Artificial Intelligence Application to Web Security. En Marvin V. Zelkowitz, ed. *Advances in Computers, Vol. 83*, Burlington: Academic Press, 2011, pp. 109-181.

[2] L. von Ahn, M. Blum, J. Langford. Telling humans and computers apart automatically—how lazy cryptographers do AI. *Comm. of the ACM 47* (2004), pp. 56-60.

[3] G. Mori, J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol.1, 2003*.

[4] Jonathan Wilkins. *Strong CAPTCHA Guidelines v1.2*, 2009. <http://es.scribd.com/doc/24497942/Strong-CAPTCHA-Guidelines-v1-2>.

[5] L. von Ahn, B. Mauer, C. McMillen, D. Abraham, M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures, *SCIENCE, Vol. 321*, pp.1465-1468, septiembre 2008.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE, Vol. 86, No. 11*, noviembre 1998.

[7] S. Belongie, J. Malik, J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 24*, abril 2002.

Notas

[1] <http://community.websense.com/blogs/securitylabs/archive/2012/01/30/trojan-caught-on-camera-shows-captcha-is-still-a-security-issue.aspx>.

[2] <http://www.google.com/recaptcha>.

[3] <http://googleblog.blogspot.com/2009/09/teaching-computers-to-read-google.html>.

[4] De acuerdo con, por ejemplo, los resultados de un estudio de la empresa de seguridad Danballa, existen *botnets* integradas por millones de ordenadores, y una cantidad apreciable de botnets con más de 10,000 ordenadores: <http://blog.damballa.com/?p=361>.

[5] <http://code.google.com/p/tesseract-ocr/>.