available at www.sciencedirect.com

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Preventing massive automated access to web resources

Alessandro Basso[*], *Stefano Sicco*

Computer Science Department, University of Torino, c.so Svizzera 185, 10149 Torino, Italy

ARTICLE INFO

Article history:
Received 15 April 2008
Received in revised form
12 September 2008
Accepted 26 November 2008

Keywords:
Anti-bot techniques
Web content protection
Massive access
Web security
Visual HIP

ABSTRACT

Automated web tools are used to achieve a wide range of different tasks, some of which are legal activities, whilst others are considered attacks to the security and data integrity of online services. Effective solutions to counter the threat represented by such programs are therefore required. In this work, we present MosaHIP, a Mosaic-based Human Interactive Proof (HIP), which is able to prevent massive automated access to web resources. Properties of the proposed solution grant an improved security over usual text-based and image-based HIPs, whereas the user-friendliness of the system alleviates the user from the discomfort of typing any text before accessing to a web content. Experimental evidence of the effectiveness of the proposed technique is given by submitting our system to a series of tests simulating possible bot attacks.

1. Introduction

In the last years, the enormous growth of the web has made indispensable the use of programs to automatize various Internet activities. This fact has led to a constant increment in the number of automated tools employed every day on the Net. However, several of these programs are developed with the unique purpose of undertaking illegal activities or achieving goals not always in line with the ethic and habits of web utilization. As a consequence, the threat to security and data integrity of online services has considerably augmented and the Internet community has shown an increasing interest in finding an effective way to stop the improper use of automated tools.

Currently, there are several situations in which using automated tools is mandatory, due to large amount of data to process. Let us consider, e.g., activities like web site mirroring, vulnerability assessment (Ollmann, 2005b) or chat and instant messaging system management. Despite the fact that such activities may be undertaken with legitimate purposes, there are cases where the use of automated tools is generally considered a serious problem. Indeed, automated web tools are also widely used to register free email accounts and harvest email addresses from web pages (Ahn et al., 2004); to extract confidential information and systematically mine databases intended for occasional use of individual humans (Coates et al., 2001); to perform massive alteration of data in web application, such as recommender systems (Chew and Tygar, 2004) or Internet polls (Basso and Miraglia, 2008); to perform brute-force guessing of authentication credentials in web applications (Pinkas and Sander, 2002; Namprempre and Dailey, 2007); and, also, to launch Denial of Service attacks (Morein et al., 2003; Kandula et al., 2005).

Devising an effective solution to address the problem of massive and automated access to web resources is not a simple goal. This is due to several practical constraints which characterize the web and do not allow to avoid multiple subsequent accesses to a specific content (Basso et al., 2004). In particular:

[*] *Corresponding author*. +39 011 6706781; fax: +39 011 751603.
E-mail addresses: alessandro.basso@di.unito.it (A. Basso), stefano.sicco@di.unito.it (S. Sicco).
0167-4048/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2008.11.002

- browsers cannot be customized, initialized or modified in any way;
- it is often practically impossible or inappropriate to distribute certificates or user names and passwords for user authentication;
- biometric approaches and techniques based on keystroke dynamics are expensive, imprecise and not always applicable;
- remote network architecture may include proxies, firewalls and Network Address Translation (NAT) nodes;
- clients may change their IP address due to user mobility and local address assignment (DHCP);
- users may connect from more than one client machine or they can share the same workstation.

In this paper, we define the concept of *security against massive and automated access*, i.e. a notion of security that could be applied to generic web applications, and we propose MosaHIP, an effective technique which can provide such form of security.

The rest of this work is organized as follows: in Section 2, we give the definition of automated tool and describe the evolution of such a class of programs. In Section 3, we show currently employed methods for stopping automated tools and, in particular, the concept of HIP, its well known properties and typical uses. Section 4 explains the MosaHIP protection scheme, whereas in Sections 5 and 6 we analyze its security properties and usability issues. A series of tests simulating possible attacks and their results are presented in Section 7 along with usability tests, to give an experimental evidence of the effectiveness of the proposed technique. Finally, in Section 8, we give our conclusions and point to directions for future research.

2. Automated web tools

An *automated tool*, also known as *scanner* or *robot* (bot), can be defined as a computer program that executes a sequence of operations repeatedly, carrying out tasks for other programs or users without the need of human interaction. A general property of all web bots is the ability to imitate human behavior surfing the web and accessing to web-based applications. However, a bot can repeatedly perform web-related activities, which have been thought and created as prerogatives of human beings, much more rapidly than a human user.

Bot's actions can be driven by legitimate purposes or can rely on malicious plans. Therefore, robots can accomplish two opposite goals: (1) help human beings in carrying out repetitive and time-consuming operations and (2) undertake hostile or illegal activities, becoming a serious threat to web application security. However, despite their dual nature, web bots are more often referred to as a source of problems rather than useful programs. Indeed, legal use of automated tools is restricted to specific ambits of application (vulnerability scan, web chat management), while malicious use is undoubtedly widespread on the Net and its consequences more easily perceptible.

Automated tools have undergone a constant evolution in order to overcome the security measures adopted in web development. According to Ollmann (2005b), we can distinguish through three generations of automated programs:

- *1st generation* – the first attempt to build web bots which automatically retrieve a set of predefined resources, without trying to interpret the content of downloaded files.
- *2nd generation* – the second generation of web bots is able to analyze downloaded HTML pages searching for links to other resources and elaborating simple client-side code in order to reconstruct the page structure.
- *3rd generation* – the last generation of automated tools is able to fully interpret client-side languages such as Javascript, VBscript or Flash and can understand web contents in a fashion more similar to what human beings do, being granted of a form of ''intelligence''.

Currently, automated tools of 1st and 2nd generations are rather common, whereas programs belonging to 3rd generation are not widespread and generally require proper tuning in order to be adapted to a specific application. However, the evolution trend of automated tools is moving towards the development of more complex and sophisticated programs, which posses an always increasing intelligence and can reproduce human actions with a high degree of fidelity.

3. Related work

In the last years, several anti-bot defense strategies have been developed with the purpose of protecting web applications from the harmful action of automated tools. Such techniques strongly rely on the relative simplicity of the majority of web bots. Thus, they cannot be considered completely secure from both a theoretical and a practical point of view. A common property characterizing many schemes is their tendency to rely on HTTP protocol features. Therefore, a simple classification can be done in HTTP *client-side* and *server-side* strategies, depending on which specific features of the protocol are exploited (Ollmann, 2005b). In addition, we can identify a third class of anti-bot strategies, which comprises schemes based on Human Interactive Proofs, also known as CAPTCHAs.

3.1. *HTTP-based strategies*

Client-side strategies rely on client-side development technologies (Javascript, Java applets, Flash, VB Script) and exploit the partial or complete inability of many automated tools to interpret and execute client-side code. They are based on the *code execution with validation purpose*, which is a two-phase procedure: 1) the client script is run during the web page loading producing an output value; 2) such a token is sent to the server-side web application to be validated (Ollmann, 2005b). The only way to generate a working token is to interpret the client-side script, therefore full support for client languages is required on the client. On the server, the only requirement needed is the support from the web application, which must check the token at each HTTP response. Depending on the way the token is computed and passed to the web application, client-side strategies can be classified into *static token appending* and *dynamic token appending*.

An improved version of token appending, known as *resource metering*, attempts to restrict the repetition frequency of data submission to an application (Ollmann, 2005a). The original idea at the basis of resource metering is the *electronic payment through a pricing function*, which has been proposed by Dwork and Naor in 1992 (Dwork and Naor, 1992). Basically, the client is forced to perform a time-consuming computation, in order to introduce a measurable time delay between two subsequent data submissions. An interesting approach to define a pricing function is through the concept of *hashcash*, defined by Adam Back in 1997, which employs the concept of *partial hash-collision* (Back, 2002). Resource metering via hashcash has been proposed to combat email spamming. However, recent studies (Laurie and Clayton, 2004) show the limits of this method in the context of spam fighting. The main issue is the possible lack of processing power for legitimate users, which prevents them from correctly using the email service.

Server-side strategies rely on specific features of the HTTP protocol which can be configured or modified only on the web server, such as headers sent in HTTP responses, or operations allowed only on the server-side of the web application. Ollmann (2005b) exhaustively discusses HTTP-based server-side strategies, identifying several protection solutions, such as ''server host renaming'', ''blocking of HEAD requests'', ''REFERER-based methods'', ''one-time links'', ''content-type alteration'', ''HTTP status code manipulation'', ''thresholds and timeouts'' and ''honeypots''.

The effectiveness of HTTP-based strategies depends on the inability of many automated tool to fully understand web development technologies. They are generally effective against automated tools of 1st and 2nd generations, but can be defeated by a sophisticated enough bot or by a properly sized botnet. In addition, information contained in HTTP headers are easily alterable (Vivo et al., 1999), therefore it is not advisable to rely only on them for creating a secure anti-bot solution.

3.2. *Human Interactive Proofs*

An effective solution to counter automated attacks is based on the concept of *discriminating between human and computer actions*. Currently, there exist a class of tests that have been developed with this specific purpose. They are known as *Human Interactive Proof* (HIP), a term which defines a specific class of challenge–response protocols whose purpose is to allow a human being to perform a secure authentication process in order to be recognized as a member of a group (Hopper and Blum, 2001). When the purpose of the HIP is to distinguish between humans and computers, it is called CAPTCHA, an acronym for *Completely Automated Public Turing Test to Tell Computers and Humans Apart*, also known as *Automated Turing Test* (Ahn et al., 2004). However, unlike the traditional Turing test (Turing, 1950), the problem is not proving that one is a human to another human, rather a computer has to decide whether one is human or not.

According to the definition given in Ahn et al. (2003), a CAPTCHA is a cryptographic protocol whose underlying hardness assumption is based on an Artificial Intelligence (AI) problem. Basically, a CAPTCHA exploits the possible existing gap between human and computer ability with respect to a specific hard-to-solve AI problem, defined in terms of the consensus of a community.

A typical CAPTCHA must possess the following properties (Ahn et al., 2004):

- it should be taken quickly and easily by human users;
- it should accept all human users, without any discrimination;
- virtually no machine should be able to solve it;
- it should resist attacks even if the algorithm and its data are known.

The intrinsic side effect of using such a test is the possibility of inducing advances in the field of AI when a CAPTCHA is successfully bypassed by an automatic system. Ahn et al. (2003) define this advantage as a *win-win situation*: either the CAPTCHA is not broken and it realizes an effective anti-bot protection, or the CAPTCHA is broken and a hard AI program is solved.

Depending on the specific type of task they require, CAPTCHAs can be based on: *text* recognition, *image* recognition and *speech* recognition (Chew and Tygar, 2004).

3.2.1. *Text-based test*

CAPTCHAs belonging to this category exploit computer programs' inability to recognize extremely distorted and corrupted textual contents embedded inside pictures, characterized by low quality and quite strong degradation. These images are generally easily readable for human beings, but their content is usually illegible even to the best OCR softwares (Coates et al., 2001).

Two of the most famous examples of such tests are *EZ-Gimpy* and *Gimpy*. Originally proposed by Blum and von Ahn on the CMU's CAPTCHA web site (The CAPTCHA project, 2000), they are currently broken (Mori and Malik, 2003). Recently, a more secure type of text-based HIP, called reCAPTCHA (2007) has been proposed by the same authors. Other interesting CAPTCHAs are known such as ''BaffleText'' and ''ScatterType'' (Chew and Baird, 2003), which are however quite difficult due to their low legibility.

Text-based HIPs must be built to be resistant to ''segment-than-recognize'' attacks (Baird, 2006). With this term, we refer to a way of bypassing a HIP which is based on two phases: initially, the word is segmented into individual characters by using a proper algorithm. Then, each symbol is compared to a set of previously classified letters, with the purpose of finding a match.

3.2.2. *Image-based test*

CAPTCHAs belonging to this category require the user to solve a visual pattern recognition problem or understand concepts expressed by images. An example of such test was initially proposed by Blum and von Ahn with the name ''PIX'' and it is now known as ''ESP-Pix'' (The CAPTCHA project, 2000). The main problems of this type of CAPTCHA, which may lead the user to fail the test, are *misspelling* while writing the answer and *synonyms* of the correct answer (Chew and Tygar, 2004).

Other famous image-based HIPs are Microsoft ''Asirra'' (2007) (Animal Species Image Recognition for Restricting Access) and ''KittenAuth'' (2007). They both challenge the user to prove his humanity by selecting all animals of a specific species among the proposed pictures. Their main issue is the absence of any image transformation in order to avoid automatic picture classification. Moreover, Asirra is not a proper CAPTCHA, since only 10% of its labeled database is public.

An interesting image-based HIP is presented in Datta et al., 2005: ‘‘IMAGINATION’’ (IMAge Generation for INternet AuthenticaTION) uses a database of images of simple concepts and is composed of two subsequent tests, based on different AI problems. This CAPTCHA appears quite effective, but requires a human user to solve two distinct tests, with a consequent increased effort and probability of failure. A further example is constituted by ‘‘Bongo’’, which asks the user to solve a pattern recognition problem (Ahn et al., 2004). This CAPTCHA is however vulnerable to the random guessing attack with a probability of success of 50%.

Despite that the security level of image-based tests is superior than text-based HIPs, they do not possess the same level of diffusion of textual CAPTCHAs. This is mainly due to their intrinsic higher difficulty and the larger screen area required to display images. In addition, the creation of a large, labeled database may also be problematic as well as the possible inconsistency with the specific topic of a web site.

3.2.3. *Audio-based test*

CAPTCHAs belonging to this category focus on machine difficulty in understanding spoken language in presence of distortion effects, degradation and background noise (Kochanski et al., 2002). The first audio-based test was implemented by Nancy Chan and afterwards presented in Nancy, 2002. Further research was conducted by Lopresti, Shih, and Kochanski, who exploited the known limitations of Automatic Speech Recognition (ASR) techniques in order to generate sounds that humans can understand but automatic programs fail to recognize.

Audio CAPTCHAs are generally considered more difficult to solve, hard to internationalize and more demanding in terms of time and effort if compared to visual ones. Thus, they are quite common as alternative to text-based CAPTCHAs, to avoid discrimination of visually impaired people.

3.2.4. *Usability and accessibility*

Usability and accessibility are the major problems affecting current CAPTCHAs, as it has been correctly pointed out by the World Wide Web Consortium (W3C) (May, 2005). Indeed, the Automatic Turing Test poses accessibility problems to blind, visually impaired or dyslexic users and, in general, users with disabilities, since it does not correctly recognize them as humans.

A possible solution is to give the user the ability to switch to a new challenge involving different sensory abilities, when he or she does not feel comfortable with the test currently displayed. Apart some criticisms, such a solution has been proven to be an effective method to limit the discriminatory tendency of the Automatic Turing Test.

4. Preventing automated access

Currently, the most effective technique for human–machine discrimination is through the use of CAPTCHAs, which are mostly text-based. However, advances in computer vision and pattern recognition have made textual HIPs less effective and more prone to suffer specific attacks (Chellapilla and Simard, 2005; Chellapilla et al., 2005b; Moy et al., 2004). Recently, new attacking techniques have been able to solve several of the most common CAPTCHAs found on Internet with a rather high efficacy (Mori and Malik, 2003; Yahoo, 2008; Google's CAPTCHA, 2008). It is reasonable to suppose that as computer vision research advances, computers get faster and attackers get sophisticated, existing text-based HIPs will soon become completely ineffective and new challenges will have to be created.

Currently, computers are still unable to accomplish many vision-related processes that we usually consider easy tasks, like understanding a specific concept expressed by an image or seeing and perceiving the world around.[1] Therefore, the proposed method has been devised to exploit the human ability in recognizing generic objects displayed by a picture never seen before.

The main contribution of this work is the attempt to devise a protection scheme which is more secure than current text-based HIPs and, at the same time, provides a simple and intuitive user interface. Indeed, a HIP must not only be secure but also human-friendly. This not only comprises the visual appeal and annoyance factor of a HIP, but also how well it utilizes the existing gap between humans and machines at solving difficult vision-related problems, such as segmentation and recognition tasks (Chellapilla et al., 2005a).

In our scheme, a user is required to pass a proper CAPTCHA at each attempt to access a resource. Therefore, any attack demanding the automatic download of *n* resources necessitates the adversary to correctly answer *n* CAPTCHAs. Compared to common text-based HIPs which involve a check just once, our approach allows a more accurate control over the whole browsing process and a consequent higher level of security. To limit the overhead caused by the higher number of tests and to provide users with an experience as similar as possible to typical browsing activity, we also carefully considered usability issues of the new solution.

The proposed scheme, named ‘‘MosaHIP’’, exploits the current computer's difficulty in performing:

- image segmentation in regions of interests, in presence of complex background;
- recognition of specific concepts from background clutter;
- shape matching when specific transformations are applied to pictures.

The term ‘‘MosaHIP’’ is an acronym for *Mosaic-based Human Interactive Proof*, which refers to the idea of a HIP based on a mosaic of pictures. This is indeed the basic characteristic of the scheme under analysis, which challenges the user with a single large image (mosaic image), composed of smaller and partially overlapping pictures. These are taken from two different categories:

1. images representing *real*, existing concepts;

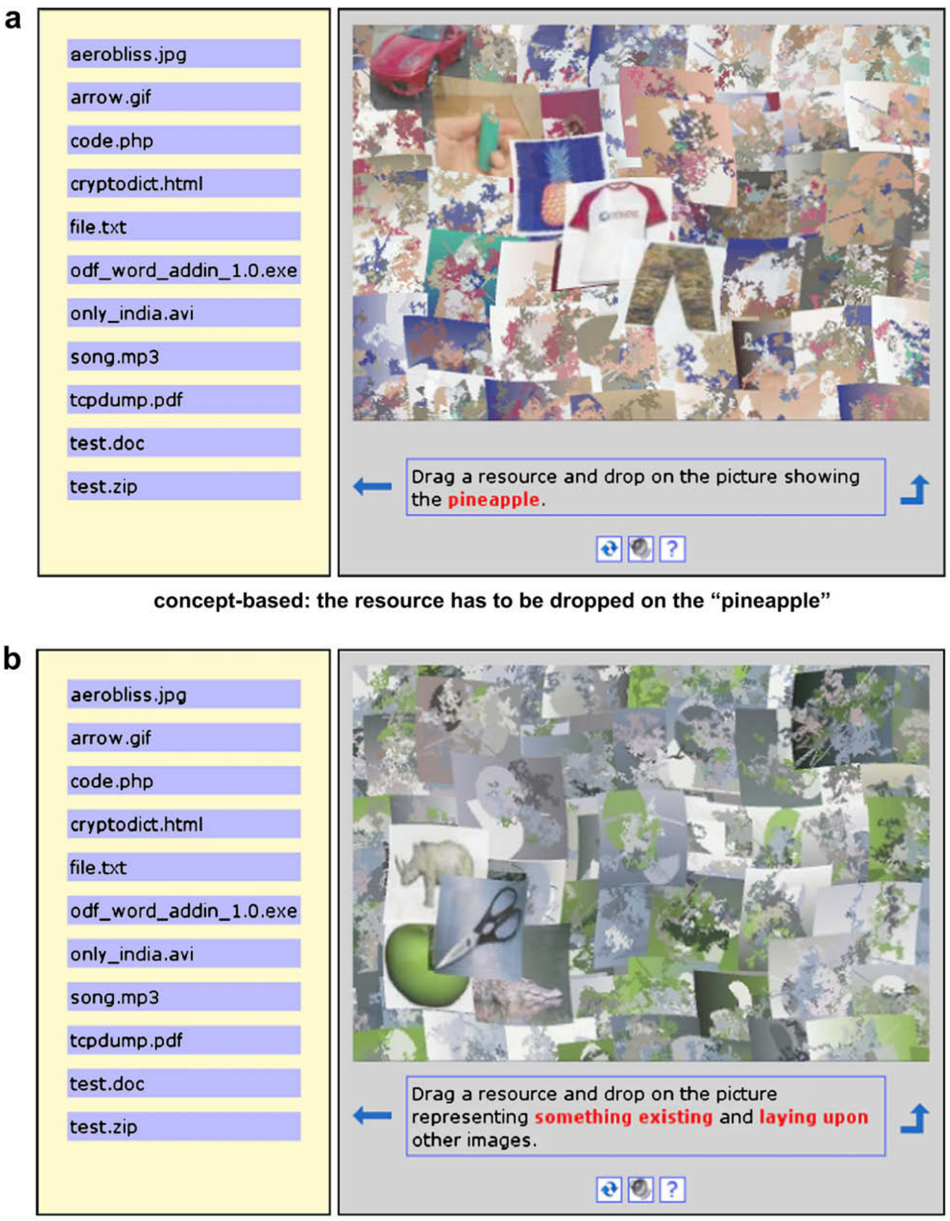
[1] *Right now there's no machine that can look around and tell a dog from a cat* (Minsky, 1998).

2. *fake* images, expressing artificially created, non-existent concepts.

Only a small subset of pictures, labeled *real*, belongs to the first category and are those the user is required to identify. They are pseudo-randomly positioned within the mosaic image and are also partially overlaying each other, so that separating them into subimages is not an easy task for a computer. The other pictures, the *fake* ones, are created by using randomly chosen colors from the color histogram of the real pictures. Moreover, a mix of randomly created shapes, lines and effects is added to each of them. They are used only to generate background clutter, whose purpose is to make difficult the recognition of images expressing real concepts to non-human users.

An example of the MosaHIP scheme is presented in Fig. 1, where two different versions of the CAPTCHA are shown: Fig. 1a displays the "concept-based" MosaHIP test, where users are required to identify the picture indicated by the CAPTCHA and contained within the mosaic image. In Fig. 1b is shown the "topmost" MosaHIP test, which asks the user to identify the image representing "something existing" and "laying upon other pictures", not overlapped by anything else.

In both tests, a user has to drag a resource, expressed in form of movable text object rather than web link, and to drop it onto the area of the mosaic picture containing the image indicated in the CAPTCHA (reference image). If this is done correctly, the resource is sent to the web browser. Otherwise, a new test is presented to the user, with a different set of images. In case the number of failed attempts from a single IP



topmost: the resource has to be dropped on the "scissors"

Fig. 1 – The MosaHIP scheme.

address exceeds a maximum threshold, specific countermeasures are taken to slow down or stop the activity of a possible attacker (see Section 5). Despite the fact that the visual appearance of two versions of the MosaHIP tests is similar, they rely on different ideas. The concept-based one challenges the user to identify a specific picture (indicated in the test page) hidden among several other images expressing real and nonexisting concepts. The topmost version of MosaHIP, instead, asks the user to identify a real image which is located in a specific position in relation to other pictures.

The algorithm for the creation of the mosaic image in both versions of the MosaHIP scheme is the following:

(Step 1)
- Select n images from the database of pictures P and add them to the set of real pictures $R = \{i_1, \ldots, i_n\}$.
- IF the test is concept-based, randomly choose a reference image $i_r \in R$ and retrieve its category g_r;
  ELSE, $i_r := i_n$.

(Step 2) For each image $i_j \in R$:
- Randomly choose a scaling factor s_j, with $s_{MIN} \le s_j \le s_{MAX}$, and apply a scaling function to i_j reducing it by s_j.
- Determine whether it has to be rotated and randomly choose the rotation angle θ_j, with $\theta_{MIN} \le \theta_j \le \theta_{MAX}$. Then rotate i_j by θ_j.
- Determine the percentage of transparency to apply to i_j, by randomly selecting a transparency factor t_j, with $0 \le t_j \le t_{MAX}$. Then, apply the transparency function to i_j, if $t_j \ne 0$.

(Step 3)
- Create the composite image c of size m by n with transparent background.
- Randomly select a starting location on the image c and position $i_1 \in R$ on c. Make sure that i_1 does not exceed the boundary of c.
- IF $i_1 = i_r$, save the top-left and bottom-right coordinates of i_1 in set D.

(Step 4) For each image $i_j \in R$, with $2 \le j \le n$:
- Determine the positioning location within the area of i_{j-1}, so that i_j only partially overlaps i_{j-1}. For this purpose, divide the i_{j-1} image in 4 quadrants of equal size and place i_j within the borders of a randomly chosen quadrant, overlapping only that one.
- IF the test is concept-based, i_j must not overlap any previously placed picture i_k, with $1 \le k < j$.
- IF $i_j = i_r$, save the top-left and bottom-right coordinates of i_j in set D.

(Step 5)
- Compute the color histogram of image c, hist(c).
- Create the background image b of size m by n.
- Fill b with a color gradient between colors randomly chosen in sets RGB_h and RGB_l, where RGB_h contains the k most frequent colors in hist(c) and RGB_l the leftovers.

(Step 6)
- Create a fake image f similar in size to real images and fill its background with a color gradient between randomly chosen colors from RGB_h and RGB_l.
- Draw on f various shapes and lines of colors chosen from RGB_h.
- Randomly change the pixel color of some areas of f using colors from RGB_l.
- Add f to the background image b, starting from the top-left corner.
- Repeat (Step 6) until b is completely covered with fake images.

(Step 7)
- Reduce the number of colors of image b by applying the Floyd–Steinberg dithering algorithm.

(Step 8)
- Overlap the composite image c to the background image b. Since c has a transparent background, now b contains both real and fake images.

(Step 9)
- Apply a distortion function (with pseudo-randomly chosen input parameters) on the image b.

(Step 10)
- Return the image b and the coordinate set D.
- IF the test is concept-based, also return the category g_r.

4.1. *Database of pictures and categorization*

The database of pictures used in the MosaHIP scheme contains a very large number of different images and a considerably high number of categories, required only in the concept-based version. A simple method for obtaining such a huge collection of images is to extract frames from different videos, making sure that only rather different frames are considered in the extraction. Otherwise, extracted pictures are likely to be very similar and this may introduce weaknesses in the scheme or lead to possible comprehension problems.

A further technique, proposed in Chew and Tygar (2004), suggests to download image thumbnails from an image directory which categorizes indexed images (e.g. "Google Image"). Such images can be retrieved dynamically or prefetched, in order to increase the creation speed of the HIP. However, the use of a dynamic database over a static one is preferable, since it is easier to maintain and, due to frequent updates of image indexes, it also possesses better anti-classification properties. To improve security, several directories and indexing services may be used, so that the total size of the database is extremely large and variable.

Indexing services, however, are likely to produce classification errors, since the categories are estimated by means of the resource name, a phenomenon known as "misspelling" (Chew and Tygar, 2004). A further problem affecting picture-based HIPs is "polysemy" (Chew and Tygar, 2004), i.e. the ambiguity of an individual word that can be used in different contexts to express two or more different meanings. For example, if the requested category is "cup", both images with cups and mugs are acceptable. However, only selecting the picture showing the cup allows to pass the CAPTCHA.

Only the concept-based version of MosaHIP is affected by misspelling and polysemy, since it requires picture categorization. MosaHIP solves both problems by giving the user the possibility to obtain a new set of images upon request and allowing the failure of one or more tests before more restrictive measures are taken.

4.2. *Fairness*

The proposed scheme can be considered *fair* with respect to users and operating situations, since users are always allowed to access to a resource, without any restriction due to IP address filtering or other discriminating parameter. This is due to the CAPTCHA-based nature of MosaHIP, and it holds even if multiple users share the same IP address or use the same workstation.

Being a visual CAPTCHA, MosaHIP may pose problems for blind, visually impaired people and users with cognitive and learning disability. However, the suggested scheme can be considered fair even under these circumstances. Indeed, it can be easily integrated by means of an audio CAPTCHA, which is generally considered a suitable alternative in these specific situations. The idea is to embed an audio challenge inside the web page in place of the visual one, upon user request, and display the resources as a checkbox list. By selecting one of the checkboxes and correctly typing the characters heard from the audio stream, the user can access to the chosen resource.

5. Security analysis

The main property of the MosaHIP method is the ability to stop automated tools from accessing all the resources contained within a web page. In order to show that the protection scheme effectively possesses this property, it is important to define the concept of *security against massive access*.

Definition 1. *A protection scheme is secure against massive access if programming and using a bot to massively download web resources is an inconvenient task in terms of time, human resources and process efficacy*.

From the above definition, it comes out that a CAPTCHA does not have to be 100% resistant to computer attack in order to be considered secure. If it is at least as expensive for an attacker to break the challenge by machine than it would be to pay a human to take the CAPTCHA, the HIP is secure (Chew and Tygar, 2004).

By exploiting different hard-to-solve problems in computer vision and image retrieval area, MosaHIP grants an effective resistance to operations like content extraction and identification, which may be used to attack the scheme. In the proposed method, a specific content is accessed only when the visual challenge is passed. This is possible only understanding the concept expressed by a generic image. However, even employing the most recent techniques in the area of machine vision, modern computers cannot yet understand the content of generic images. In general, performing pattern recognition in presence of complex background or when the concepts to recognize are taken from a large domain and segmenting an image to extract its internal components are some of the most difficult challenges for a computer program (Lew et al., 2006; Chellapilla et al., 2005b).

We can therefore reasonably state that the proposed scheme satisfies the requirements of security against massive and automated access and it is therefore resistant to bot attacks. In the rest of this section, we will discuss more in detail the MosaHIP's security properties which motivate our claim.

5.1. *Resistance to segmentation through edge detection*

Current research in the image retrieval field shows that one of the most important challenges yet to be solved is *visual concept detection* in the presence of *complex background* or *clutter* (Lew et al., 2006). Therefore, an important security measure adopted by the MosaHIP scheme exploits the difficulties of Content Based Image Retrieval (CBIR)[2] methods to extract the semantic content expressed by a picture in presence of complex background. Segmenting an image in regions of interest is indeed required to attempt an identification procedure in order to discriminate among images containing real concepts from fake ones. A possible method for executing such a task is to employ an edge detection algorithm, in order to delimit the contours of subpictures which compose the mosaic image. The procedure of contour separation is intended for fast reduction of image data volume and simplification of subsequent recognition steps.

We experimentally tested image segmentation on MosaHIP pictures applying two effective and well-known techniques to detect edges: the *zero-crossing Laplacian operator* (Gonzalez and Woods, 1992) and the *Canny operator* (Canny, 1986). The result of the application of both algorithms on a MosaHIP picture is shown in Fig. 2. It is clear from the example that the problem of separating real images from the background noise created by means of the fake pictures is extremely difficult to solve. The application of an edge detection algorithm has reduced the amount of information present in the MosaHIP image and extracted relevant features of each subpicture. However, the number of connected components within the resulting image is still very high and does not allow to easily detect the shapes belonging to real images, which are needed to perform a CBIR process.

5.2. *Resistance to segmentation through thresholding*

In computer vision, *thresholding* provides an easy and convenient way to separate regions of an image corresponding to objects of interest from background. This segmentation process can be performed on the basis of different intensities of colors in the foreground and background regions of an image (Fisher et al., 2004). The segmentation through thresholding is determined by comparing each pixel in the image with an intensity threshold and, if its intensity is higher than the threshold, the pixel is set to white in the output, otherwise it is set to black.

Not all images can be neatly segmented into foreground and background using simple thresholding. A simple way to determine this possibility is to look at the intensity histogram of the image. If it is possible to separate the foreground of an image from the background on the basis of pixel intensity, then the intensity of pixels within foreground objects must be distinctly different from the intensity of pixels within the

[2] Content Based Image Retrieval analyzes the actual contents of the image focusing on visual features to represent and index the image (Long et al., 2003).

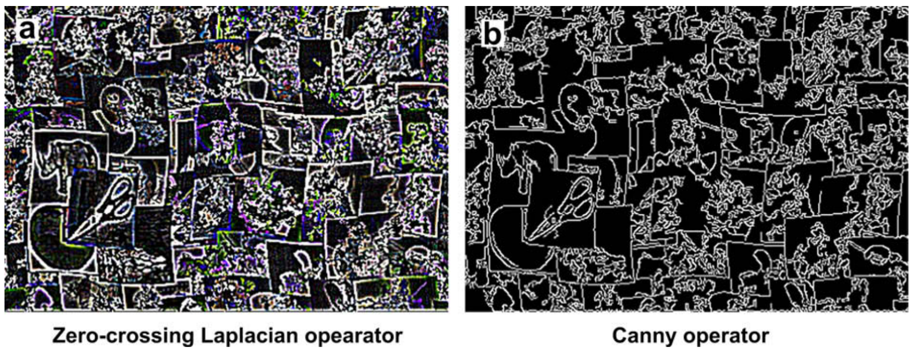


Fig. 2 – Edge detection on the MosaHIP image of Fig. 1(b).

background (Fisher et al., 2004). In this case, we expect to see one or more distinct peaks in the histogram which correspond to foreground objects. They can be isolated by properly choosing one or more thresholds, as shown in Fig. 3a,b. If it is not possible to distinguish such distinct peaks, then it is unlikely that simple thresholding will produce a good segmentation, as it happens in Fig. 3c. An alternative approach to single thresholding is *adaptive thresholding*, which changes the threshold dynamically over the image by selecting an individual threshold for each pixel based on the range of intensity values in its local neighborhood. This allows for thresholding of an image whose intensity histogram does not contain distinct peaks.

Images generated by the MosaHIP scheme are not easily segmentable by means of color thresholding. This is due to the use of randomly chosen colors from the histogram of the image containing the real pictures. In fact, it is not possible to discriminate between the real and fake pictures only by focusing on pixels whose color is more or less than a specific threshold, because colors of real pictures are widespread throughout the MosaHIP image and characterize also the contents of fake pictures.

An example of segmentation through single thresholding on a generic MosaHIP image is shown in Fig. 4a. The reader should notice that it is not possible to set up two thresholds *T*1 and *T*2 which can isolate the regions of interest, i.e. the real subimages from the artificially created background, because the image histogram contains the aggregation of two peaks. In fact, the chosen thresholds produce an incomplete segmented image, which includes only partially the contents of the real pictures. Even when the image histogram displays a single steep peak, the segmentation process does not grant the expected result, as it is shown in Fig. 4b. Also in this case, the two thresholds *T*1 and *T*2 determine a partial loss of information. This is due to the tendency of the MosaHIP scheme to spread colors of real images throughout the mosaic picture. Therefore, removing background information to isolate real subpictures also determines a partial removal of information which should be preserved.

Approaches based on adaptive thresholding are also not successful in segmenting MosaHIP pictures. Examples of Fig. 5 clearly show this fact. Both mean and median functions are used, with different neighborhood windows and threshold reduction. The resulting images are more accurate than in the case of single thresholding, but do not allow to select specific regions of interest, i.e. real subpictures. By changing the values to obtain less detail and possibly a better segmentation, the thresholding process uniformly removes information throughout the mosaic image. However, also the content of real subpictures is partially or completely removed.

5.3. *Resistance to shape matching*

The proposed scheme attempts to counter the task of performing image retrieval by picture comparison, by using different techniques. First, the MosaHIP algorithm makes difficult the separation of the sequence of real images in subpictures by *partially overlapping* real and fake images. As a further security measure, different levels of *transparency* are applied to every real picture, in order to partially visualize the content of images lying beneath the overlapping area. This fact implies an increased number of false positives in the

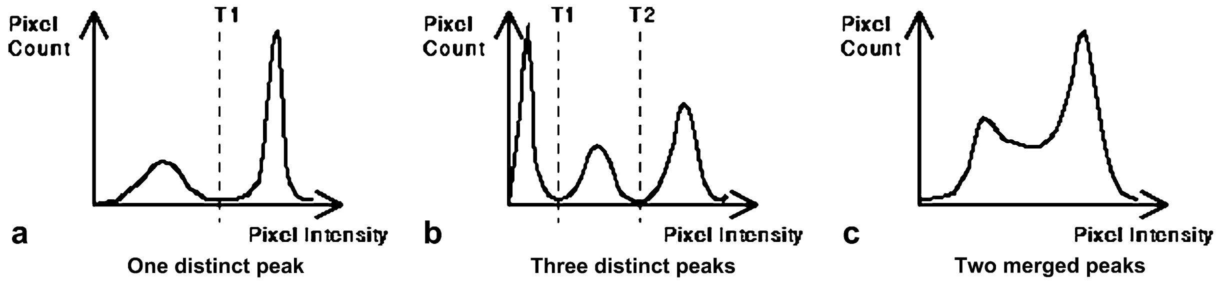


Fig. 3 – Different intensity histograms.

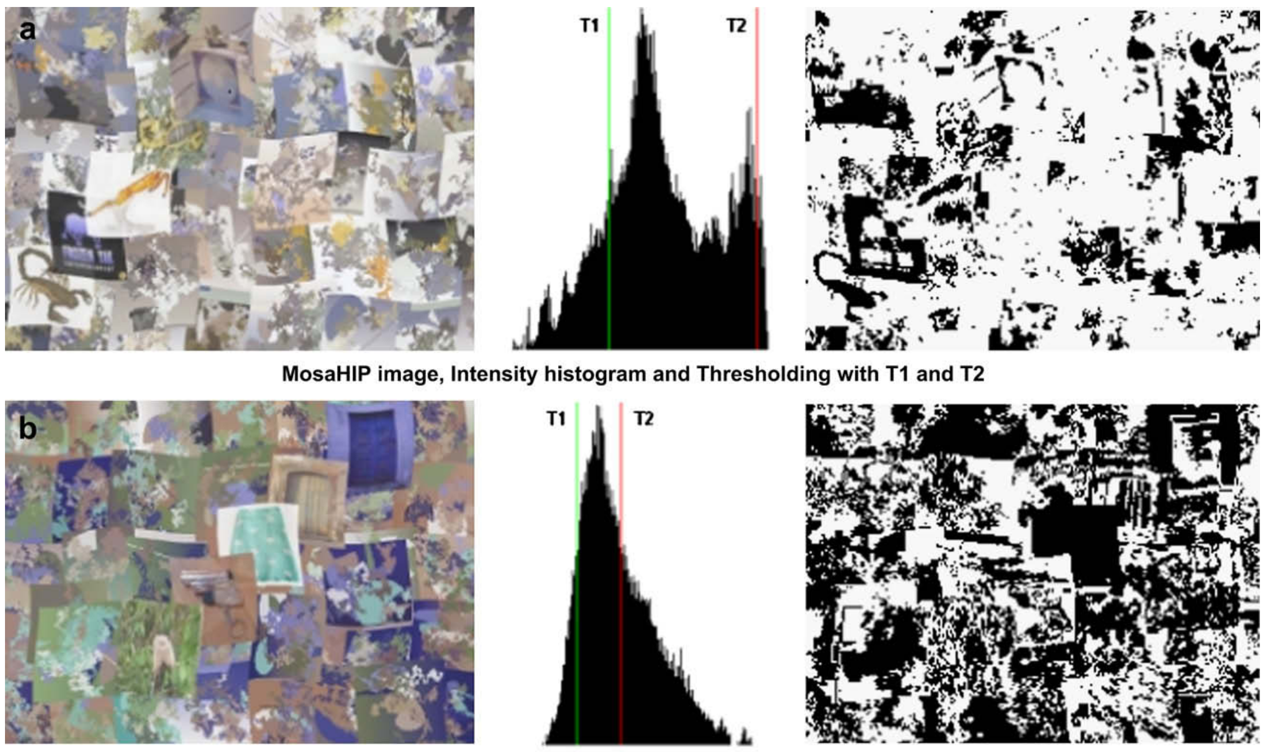


Fig. 4 – Single thresholding on MosaHIP images.

image segmentation process. The overall result is an increased difficulty in performing shape matching through CBIR methods.

In addition, the MosaHIP algorithm applies a number of transformations and effects on both real and fake images, with the purpose of making their automatic identification more difficult. The scheme employs *rotation*, *scaling* and *transparency* functions, with input parameters randomly chosen within specific ranges. A *controlled distortion* on the mosaic image is also applied. Obviously, image transformations should be applied without excessively affecting the quality and clarity of the input picture, in order to allow an easy recognition by humans.

A further measure to reduce the effectiveness of automated recognition techniques is the *color reduction* through the application of the Floyd–Steinberg dithering algorithm. This step ensures that the task of automatically determining image borders remains challenging, whereas human recognition abilities still allow a correct identification of single pictures. Moreover, the overall size in bytes of the generated image can be considerably decreased.

5.4. *Resistance to random guessing*

A simple way to attack the proposed scheme involves random guessing of the area belonging to the correct image. We refer to it as "blind attack". Given a MosaHIP image of size m by n, such an attack attempts to locate the subarea of size x by y, with $x < m$ and $y < n$, corresponding to the image indicated in the CAPTCHA (reference image). The attacker then chooses a specific resource and drops it on the selected area.

If we suppose that every pixel in the CAPTCHA image has the same chance of being chosen to position the reference picture, we can compute the success probability of the blind attack:

$$P_{suc} = \frac{A_{ref}}{A_{mos}} = \frac{xy}{mn}, \tag{1}$$

where A_{ref} is the area of the reference picture and A_{mos} is the area of the MosaHIP image.

The MosaHIP scheme, however, can be considered resistant also against attacks based on the random guessing model. This is possible because the probability of success of the blind attack can be arbitrarily reduced, in order to make the attack ineffective. This is done by properly determining the size of real pictures and the mosaic image. Basically, defined the area of a real picture A_{ref} and a probability of success P_{suc}, we can compute the area of the MosaHIP image A_{mos} through the relation:

$$A_{mos} = \frac{A_{ref}}{P_{suc}}. \tag{2}$$

In our tests, we used real pictures of size comprised between 65 and 70 pixels. Considering a success probability $P_{suc} = 4.1\%$, according to Eq. (2) the area of the CAPTCHA image should be equal to nearly 110,000 pixels, which correspond to an image size of 400×275 pixels. Since the MosaHIP image area cannot be increased indefinitely, we can assume that a size of 400×400 pixels is adequate for many web

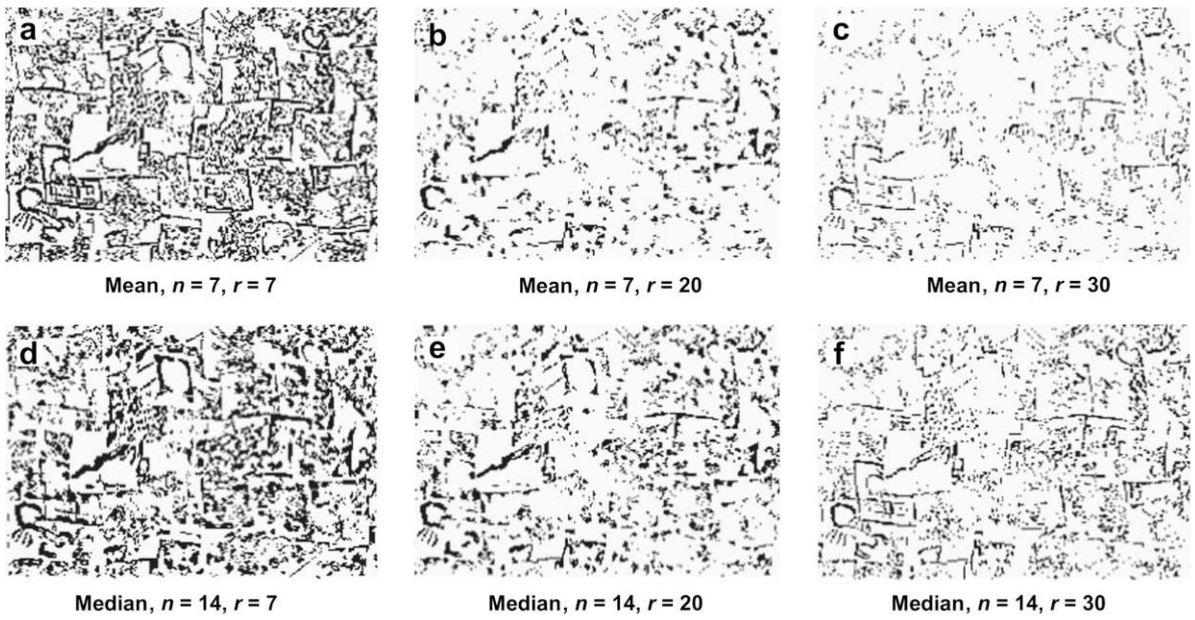

a
b
c
Mean, $n = 7$, $r = 7$
Mean, $n = 7$, $r = 20$
Mean, $n = 7$, $r = 30$
d
e
f
Median, $n = 14$, $r = 7$
Median, $n = 14$, $r = 20$
Median, $n = 14$, $r = 30$

Fig. 5 – Adaptive thresholding with different functions, neighborhood windows (*n*), and threshold reductions (*r*).

applications. The estimated probability of success related to such dimensions is 2.8%.

Despite that a probability of success of 2.8% is rather low, it is however sufficient to allow the complete download of the protected resources, if enough time is provided. To counter such a threat, a simple solution is to delay the attacking host by temporary locking the IP address of the possible attacker. The locking time can be exponentially increased with the number of subsequent failed attempts of downloading, until a maximum value is reached. When the locking time expires, a new MosaHIP test is presented to the user.

IP locking is an effective protection against the random guessing model, since it delays the blind attack to such an extent that it becomes a non-viable attack vector. An experimental evidence of such a claim is given in Section 7.1. However, in front of repeated test failures from a specific IP address, the IP locking might cause a partial ''Denial of Service'' (DoS). This is possible because users who share the same IP address of the adversary[3] are prevented from accessing the protected resources. Note that the DoS is ''partial'' because it is limited only to hosts sharing the locked IP address.

A solution to this problem improves the MosaHIP scheme by replacing the IP locking with a computationally intense process on the client-side, in a similar way to the resource metering technique (see Section 3.1). The activation of such a protection scheme may be triggered when a specific error threshold is exceeded, in order to avoid useless resource consumption and interruptions in web browsing activity when not effectively needed. Such an approach avoids problems related to insufficient computation capacity characterizing users with older hardware and mobile devices, which is also the main limit for practical use of the resource metering technique (Laurie and Clayton, 2004).

[3] Due to the presence of proxy, NAT boxes, reuse of dynamic assigned IP address, etc.

Following the typical hashcash computation idea (Back, 2002), the method requires that only the n most-significant bits of the hashed values of two strings m_1, m_2 are equal. Thus, the sender is challenged to produce a string whose cryptographic SHA-1 hash starts with a certain number, k, of zeros. It is possible to control the difficulty of the computation, therefore the time required, by defining how long must the matching parts be. Practical considerations related to the computational capacity of current hardware suggest to use $k = 26$ or even $k = 27$.

In the context of massive access prevention, the message used in the hash calculation should vary in relation to: the *resource requested*, the *time* and possibly other *pseudo-random data*. So, the token must be closely related to the specific content requested by taking into account the *URL* of the resource. In addition, since a computed value should not be reused to access the same resource, a *timestamp* must be added to the message, in order to consider tokens older than a certain date invalid. Finally, each token should be unique and used only once. For this purpose, a sufficiently long *random seed* must be appended to the message and each token should be checked for validity against a database containing all previously used tokens.

A generic message used to produce a hashcash token can be defined with the following format:

`URL:timestamp:seed:random_suffix`, where `random_suffix` indicates the random sequence of data repeatedly generated by the client in order to find the collision, i.e. the hash digest whose first k digits are all zeros.

A basic version of the protocol for resource metering via a hashcash adapted to the web context can be described as follows:

(Step 1)

- a client C asks for a web page w containing some resources.

(Step 2)

- the server S sends w to C, with some parameters: the number of bits to collide, the `timestamp` and the unique `seed`.

(Step 3)

- C selects a specific resource to access, r, and retrieves its `URL`.
- C combines `URL` with the `timestamp` and the `seed` to create the message prefix:

  $m_{\text{pref}} = \texttt{bits:URL:timestamp:seed}$

(Step 4)

- C generates some random data, `random_suffix`.
- C creates the hashcash message m by joining m_{pref} with `random_suffix`:

  $m = \texttt{bits:URL:timestamp:seed: random\_suffix}$

(Step 5)

- C computes the digest $d = H(m)$ by means of the hash function H (e.g. SHA-1), implemented within w in a client-side language.
- IF a sequence of k zeros is found, C sends the corresponding message m_i to S as a proof of computation, with URL;

  ELSE, go to (Step 4).

(Step 6)

- S receives m_i and verifies the validity of the token by computing the digest $d = H(m)$.
- IF the check is passed, S sends the resource to C;

  ELSE, go to (Step 2).

The delay time may be further increased by gradually augmenting the difficulty of computing the moderately hard function with the growing of the number of attacks from a single IP address. This way, the adversary host is forced to waste an increasing amount of CPU time for computing the function, slowing down the attack to such an extent that it may become impracticable or inconvenient.

Besides the possibility of avoiding a partial DoS attack, the resource metering technique makes also useless multi-threaded attacks, which is the main ability used by a web bot in order to speed up its hostile actions. Indeed, since the computational effort sustained to execute a single thread is very high, parallel attacks run from a single host lose their efficacy, due to the necessity of sharing the CPU among the various running threads (Ollmann, 2005a). A further advantage is the possible reduction of the number of hosts in a botnet. Indeed, users unaware that their host belongs to a botnet might notice the high CPU load and take proper countermeasures to stop the illegal use of their computer's resources.

6. Usability and performance

As shown in Section 5, both versions of the MosaHIP method can be considered secure against massive and automated access. However, it is a well-known fact that security is often in contrast with usability. This property holds for CAPTCHAs as well, and MosaHIP is not an exception. Indeed, the very same protection techniques, which make difficult the automatic identification of real pictures to computers, augment the complexity of the recognition task also for humans. This fact characterizes text-based and audio-based HIPs as well, and is the main reason behind most of the criticisms of the CAPTCHA idea.

The most evident effect of such an issue is identifiable in the process of recognizing the reference picture among the set of real pictures. This is due to specific properties of combined images, like quality, type and size, but it is also influenced by the overall number of real pictures used and by the way MosaHIP arranges them in the mosaic image. In particular, reducing the number of real pictures allows the user to easily identify the reference image, since the number of possible candidates is lower. However, for the same reason, it also negatively affects the security of the HIP. From our experiments, we could quantify the number of real images to be used, around 5 pictures, considering an overall MosaHIP size of 400×400 pixels. As further consideration, note that partial overlapping of images, transparency and other transformations may also increase the difficulty of identifying the correct picture and lead the user to a wrong recognition and the consequent failure of the test.

In Section 7.2, we deeply analyze usability aspects of the proposed HIP by means of a specific test, which permits to quantify the perceived complexity of a sample of different MosaHIP images. As possible countermeasure to mitigate the negative impact of such issues on simplicity and user-friendliness, the MosaHIP scheme has been devised to allow the generation of a new mosaic picture upon user request.

Another property of the proposed scheme, which contributes to increase its security but represents an issue for usability, is the dimension of the mosaic picture with respect to the size of real and fake images. As previously explained in Section 5, the security of MosaHIP towards blind attacks closely depends from the size of the mosaic image. According to Eq. (2), the bigger the size of the mosaic image, the lower the probability that a random guess succeeds. Large mosaic images may improve the easiness of the recognition process, since real pictures are likely to get placed throughout the mosaic image area, thus allowing a simpler identification by human users. On the other side, despite that the suggested value of 400×400 pixels can grant a good level of security towards blind attacks, it may be excessive under some circumstances. In fact, such an image requires a large portion of the screen, thus it affects positioning and dimensions of other page elements.

All visual HIPs are affected by this issue; in particular, picture-based CAPTCHAs are most exposed to this problem, because they are made by sequences or aggregations of pictures, which even individually require a considerable portion of the screen. Text-based CAPTCHAs are less vulnerable to this issue, although not immune, since the screen area used to display them is usually smaller than in the image case. Audio CAPTCHAs, instead, do not suffer from any display-related problem, since they require a very small region of the screen, whose purpose is only to activate the audio stream.

In Table 1, a comparison among different CAPTCHAs with respect to the screen area required to display them is

Table 1 – Comparison among different CAPTCHAs and MosaHIP, considering both the size of the sole image and the overall screen area occupied (in pixels).

Name	Image area	Total area
Text-based HIPs		
Google	200 × 70	400 × 200
Yahoo	200 × 80	450 × 120
Microsoft	218 × 48	350 × 200
reCAPTCHA	300 × 57	350 × 200
MosaHIP	400 × 275	480 × 377
Picture-based HIPs		
Asirra	356 × 164	450 × 200
IMAGINATION	800 × 600	800 × 700
KittenAuth	495 × 375	545 × 530
ESP-Pix	400 × 370	640 × 370
MosaHIP	400 × 400	480 × 485

presented. In general, text-based HIPs need a relatively small display area, which can be quantified as nearly a quarter of the size necessary to visualize a picture-based CAPTCHA. Note that, while the size of the image containing the text is rather small, the overall space needed to display the test is considerably bigger, due to the presence of captions, buttons and input boxes. On the contrary, picture-based HIPs may require a wider area of the screen, as it happens with the IMAGINATION test. The MosaHIP scheme, even in its larger version, does not exceed the average dimensions of picture-based methods; compared to common text-based HIPs, instead, the difference in size is noticeable.

An unwanted side effect of using a large mosaic image is the negative impact on performances, in terms of image creation and transfer. While the latter is rather obvious and closely depends on the bandwidth available for transfer, the former is mainly due to the complexity of the MosaHIP algorithm, which requires to apply several image processing operations on both subimages and the mosaic picture. In particular, the procedure for random clutter generation, used to create fake images, and transforms like image rotation or distortion are among the main reasons for this issue. Furthermore, since the number of subimages used is strictly related to the available space in the mosaic picture, increasing its size affects the number of subimages to be generated. In consequence, the total number of effects and transforms applied is also augmented.

From our tests, we could identify some possible improvements to the image generation process, which may result in better performances:

- decrease the overall number and type of effects and transforms applied to both subimages and mosaic picture;
- increase the size of subimages, in order to decrease their number;
- pre-generate a large number of mosaic images, e.g. by using a different system or when the web server load is low;
- improve the implementation of the algorithm by using optimized image processing functions and preferring compiled languages rather than interpreted ones.

However, we suggest to carefully evaluate the reduction of the number and type of tranforms, as well as the increment of subimage size, in order to avoid possible security issues.

7. Tests and results

In order to evaluate the correctness of our solution, we performed robustness and usability tests on a prototype of both versions of MosaHIP (concept-based and topmost), which can be found at: http://secnet.di.unito.it/massivedemo/. Note that the current MosaHIP implementation, despite being mostly complete in its functionality, is just a demonstration of the proposed scheme, thus it has not been optimized for practical use.

7.1. *Robustness test*

Robustness tests were executed by means of an automated tool simulating the blind attack. The main goal of the bot is to randomly identify the correct area of the composite image onto which the resource has to be dropped. This way, we want to obtain a practical evidence regarding the inefficacy of attacks based on the random guessing model. Considering the security properties of MosaHIP, we strongly believe that blind attacks are the only effective and simply way of attacking the proposed scheme.

We performed two different simulations, to test the effectiveness of MosaHIP under different conditions: one with images of 400 × 275 pixels, and another one using images of 400 × 400 pixels. Both sessions lasted for 22 days and results were recorded and plotted on a graphic.

As we can observe from Fig. 6a, which shows the test on a CAPTCHA of size 400 × 275, only 8 attempts allowed to successfully download a resource (cross-hatched line), whilst the number of failed attempts is 207 (continuous line). For each of them, the bot activity was forcefully delayed (square-marked line). We observed a success rate of nearly 3.8%, which is slightly less than the previously estimated success probability of 4.1%.

It is clear that the blind attack is not convenient in terms of actions executed over the time. In fact, the bot spends most of the time waiting for the end of the delay caused by the protection scheme, instead of actively attempting to download the resources.

In Fig. 6b, the result of the second simulation is shown, which used a picture of size 400 × 400 pixels. Such an image might not be suitable in some situations since it requires a large portion of the screen, despite the fact that it considerably improves the security of the protection scheme.

From the graph, we can observe only 6 successful attempts (cross-hatched line), whereas 204 times the bot failed (continuous line) and was thus delayed (square-marked line). In this second simulation, our expectation was also completely fulfilled, since we obtained a success rate of 2.9%, which is very close to the previously computed success probability (2.8%). In both simulations, the bot was able to download only a tiny number of resources during a very long period of time. We can therefore conclude that the blind

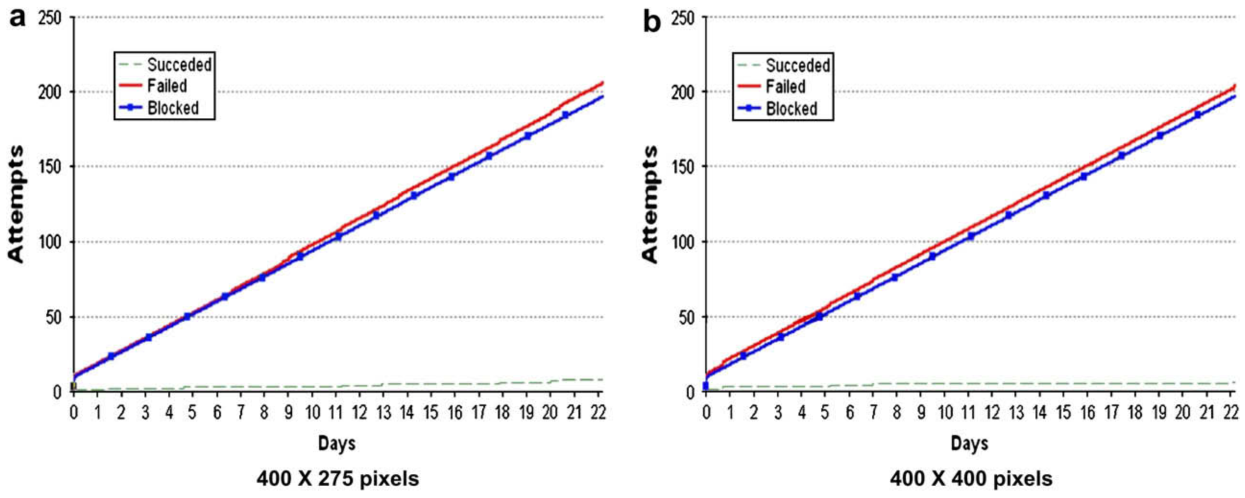


Fig. 6 – Blind attack on MosaHIP images.

attack cannot be considered successful when the MosaHIP scheme is used to protect a set of resources from massive automated access.

7.2. *Usability test*

During the creation of the MosaHIP test, we took into consideration usability and friendliness aspects of the proposed solution. Comparing the two versions of the scheme, we consider the concept-based MosaHIP test easier and more intuitive, since it suggests the correct picture category to the user. The topmost version, instead, appears to be more difficult, although it avoids the classification process, thus the occurrence of mislabeling and polysemy.

Since we wanted a more reliable feedback concerning usability aspects of the proposed CAPTCHA, we conducted a test on a sample of users. For this purpose, we set up a testing page showing a sequence of different images, taken from both versions of MosaHIP. A total of 110 users participated in the study. They were recruited in different ways, considering age, which varies from 18 to more than 60; education, which alternates from junior high school to Ph.D.; and the frequency of computer usage, which varies from less than 1 h per week to more than 40 h per week. We are aware that the results from a sample of 100 users may not predict the performance of the entire population of Internet. However, we attempted to select a sample which can be as various as possible, in order to approximate the real distribution of Internet users.

Since user participation to the test was on voluntary basis, we could not ask them to solve a large number of CAPTCHAs in each session. So, we focused on a total of 6 images, 3 taken from the concept-based test and 3 from the topmost version. They were chosen from a set of possible candidates by considering the following parameters:

- the visibility of the reference image with respect to background clutter;
- the position of the reference image with respect to other real pictures;
- the levels of transparency applied to the real images;
- the presence of fake images with colors similar to those of real pictures;
- the presence of fake images which can be easily confused with real pictures.

For each test, a set of elements was recorded: in particular, the result (passed or failed), the time required to solve it and the difficulty rate of each test. The rating process was not included in the timing measurements of each round. Also, the answers were revealed only at the end of the 6 tests, to prevent bias.

The final results are shown in Table 2(a), which displays the number of passed tests in relation to failed ones, for both versions of MosaHIP. We can observe that almost every user was able to solve the concept-based version, with 98% of correctly passed tests. Regarding the topmost version, 80% of

Table 2 – Comparison of the concept-based and topmost MosaHIP tests.

	Concept-based	Topmost
(a) Result		
Passed	323	264
Failed	5	52
No answer	2	14
Total	330	330
(b) Rate		
Very easy	191	88
Easy	98	91
Medium	29	88
Difficult	6	40
Very difficult	4	9
Not solvable	2	14
Total	330	330

participants could correctly solve it. This can be explained with the higher intrinsic difficulty of the topmost version of MosaHIP. In particular, the first impact with this test is generally hard and requires the user to carefully read and understand what must be done to pass this CAPTCHA. This fact is more evident analyzing the results related to each image of the topmost test: the percentage of passed test for the first image is 72%, that is considerably less than the values obtained with the second and third images (respectively, 87% and 80%). A further evidence of this fact is given by the rates of the topmost test: 20% of users graded the first image as difficult, whilst only 8% of participants did the same with the second and third image. In addition, also the time measurement indicates that the first test required an average time which is nearly double if compared to the second and third. The global rates given to the MosaHIP scheme are presented in Table 2(b). The general opinion of 96% of the participants is that the concept-based CAPTCHA possesses a difficulty level varying from very easy to medium, whilst 88% of users consider the test very easy or easy. Regarding the topmost scheme, 81% of users rated it from very easy to medium. However, only 54% of the testers gave a rate equal to easy or very easy. This fact confirms our supposition regarding the higher difficulty of the topmost version when compared to the concept-based scheme.

Regarding the average time to solve both versions of MosaHIP, the concept-based test requires about 7 s, whilst the topmost test necessitates an average of 10 s, which can be considered an acceptable requirement for an image-based CAPTCHA. However, we should consider that the first image demanded a time twice longer, as a consequence of the initial impact with the new scheme. Subsequent tests show that the time required to pass the CAPTCHA is about 7 s, which is also the average time for the concept-based version.

8. Conclusion

In this work we presented MosaHIP, an anti-bot defense scheme derived from picture-based Human Interactive Proofs, whose purpose is to prevent massive and automated access to web resources. The proposed technique can be considered more effective than existing text-based and picture-based HIPs from the point of view of security and ease of use. Indeed, its robustness relies on the inability of modern computer to correctly recognize generic concepts expressed by pictures, as well as sounds contained in audio streams. In addition, it also permits to verify the human nature of the requester at every attempt to use a resource. Being based on a drag-and-drop approach, it alleviates the user from the discomfort of typing any text before accessing to a web content and allows a simple integration inside Internet portals and web pages.

We gave an experimental evidence of our claims, by testing the proposed scheme with respect to security and usability. Our expectations were completely fulfilled in both test sessions, since the presented method effectively made the massive and automated access to web resources an inconvenient activity for an adversary. Unless a major advance in the area of Artificial Intelligence and, in particular, in the fields of computer vision and image recognition occurs we believe that the proposed protection scheme shall constitute a challenging problem even to the most sophisticated existing and yet to come automated tools.

Regarding future research, we believe that additional studies on usability aspects of the MosaHIP scheme would bring a sensible contribution to further reduce the initial impact with this type of CAPTCHA. Moreover, further work is required to improve the overall performance of the image creation procedure, in order to easily integrate the HIP in web applications interested in obtaining increased protection from automated tools.

REFERENCES

Ahn LV, Blum M, Hopper NJ, Langford J. CAPTCHA: using hard AI problems for security, EUROCRYPT. In: Lecture notes in computer science, vol. 2656. Springer; 2003.

Ahn LV, Blum M, Langford J. Telling humans and computers apart automatically. Commun. ACM 2004;47(2):56–60.

Back A. Hash cash: a partial hash collision based postage scheme, http://www.hashcash.org/papers/hashcash.pdf; 2002.

Baird HS. Complex image recognition and web security, Data complexity in pattern recognition. Springer-Verlag London Ltd; 2006.

Basso A, Bergadano F, Coradazzi I, Checco PD. Lightweight security for internet polls, EGCDMAS. INSTICC Press; 2004.

Basso A, Miraglia M. Avoiding massive automated voting in internet pollsSTM2007. Electron. Notes Theor. Comput. Sci. 2008;vol. 197(2). Elsevier.

Canny J. A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. 1986;8(6):679–98.

Chellapilla K, Simard PY. Using machine learning to break visual Human Interaction Proofs (HIPs). Cambridge, MA: MIT Press; 2005. p. 265–72.

Chellapilla K, Larson K, Simard PY, Czerwinski M. Building segmentation based human-friendly human interaction proofs (HIPs). In: Baird HS, Lopresti DP, editors. HIP. Lecture notes in computer science, vol. 3517. Springer; 2005a.

Chellapilla K, Simard P, Czerwinski M. Computers beat humans at single character recognition in reading-based human interaction proofs (HIPs). In: Proceedings of the second Conference on Email and Anti-Spam (CEAS), Palo Alto, CA; 2005b.

Chew M, Baird HS. Baffletext: a human interactive proof. In: Proceedings of the SPIE/IS&T document recognition and retrieval X conference, vol. 4670. Santa Clara, CA, USA: SPIE; 2003.

Chew M, Tygar JD. Image recognition CAPTCHAs. In: Proceedings of the seventh international Information Security Conference (ISC 2004). Springer; 2004.

Coates A, Baird H, Fateman R. Pessimal print: a reverse Turing test. In: Proceedings of the sixth international conference on document analysis and recognition, Seattle, WA; 2001.

Datta R, Li J, Wang JZ. Imagination: a robust image-based captcha generation system. In: Proceedings of the 13th annual ACM international conference on Multimedia (MULTIMEDIA '05). New York, NY, USA: ACM Press; 2005.

Dwork C, Naor M. Pricing via processing or combatting junk mail. In: CRYPTO '92: proceedings of the 12th annual international cryptology conference on advances in cryptology. London, UK: Springer-Verlag; 1992.

Fisher R, Perkins S, Walker A, Wolfart E. Hypermedia image processing reference; 2004.

Gonzalez R, Woods R. Digital image processing; 1992. p. 679–98.

Google's CAPTCHA busted in recent spammer tactics, http://www.websense.com/securitylabs/blog/blog.php?; 2008. BlogID = 174.
Hopper NJ, Blum M. Secure human identification protocols, ASIACRYPT. In: Lecture notes in computer science, vol. 224. Springer; 2001.
Kandula S, Katabi D, Jacob M, Berger AW. Botz-4-sale: surviving organized ddos attacks that mimic flash crowds. In: Second symposium on Networked Systems Design and Implementation (NSDI), Boston, MA; 2005.
KittenAuth, http://www.thepcspy.com/kittenauth; 2007.
Kochanski G, Lopresti D, Shih C. A reverse Turing test using speech. In: Proceedings of the international conferences on spoken language processing, Denver, Colorado; 2002.
Laurie B, Clayton R. Proof-of-work proves not to work. In: The third annual workshop on economics and information security, Minneapolis, MN; 2004.
Lew MS, Sebe N, Djeraba C, Jain R. Content-based multimedia information retrieval: state of the art and challenges. ACM Trans. Multimedia Comput. Commun. Appl. 2006;2(1):1–19.
Long F, Zhang H, Feng D. Fundamentals of content-based image retrieval. Springer; 2003 [chapter 1].
May M. Inaccessibility of CAPTCHA: alternatives to visual Turing tests on the web, W3C working group note, http://www.w3.org/TR/turingtest/; November 2005.
Microsoft Asirra, http://research.microsoft.com/asirra/; 2007.
Minsky M. Mind as society, thinking allowed: conversations on the leading edge of knowledge and discovery with Dr. Jeffrey Mishlove, http://www.intuition.org/txt/minsky.htm; 1998.
Morein W, Stavrou A, Cook D, Keromytis A, Misra V, Rubenstein D. Using graphic Turing tests to counter automated ddos attacks against web servers. In: Proceedings of the 10th ACM international conference on Computer and Communications Security (CCS), Washington D.C.; 2003.
Mori G, Malik J. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: Proceedings of the conference on computer vision and pattern recognition, Madison, USA; 2003.
Moy G, Jones N, Harkless C, Potter R. Distortion estimation techniques in solving visual CAPTCHAs. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004) 2004;(2):23–8.
Namprempre C, Dailey MN. Mitigating dictionary attacks with text–graphics character CAPTCHAs. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 2007;E90-A(1):179–86.
Nancy C. Sound oriented captcha. In: Proceedings of the first workshop on human interactive proofs. CA: Xerox Palo Alto Research Center; 2002.
Ollmann G. Anti brute force resource metering, Whitepaper – NGS software insight security research, http://www.ngssoftware.com/papers/NISR-AntiBruteForceResourceMetering.pdf; 2005a.
Ollmann G. Stopping automated attack tools, Whitepaper – NGS software insight security research, http://www.ngssoftware.com/papers/StoppingAutomatedAttackTools.pdf; 2005b.
Pinkas B, Sander T. Securing passwords against dictionary attacks. In: CCS '02: proceedings of the 9th ACM conference on computer and communications security. New York, NY, USA: ACM Press; 2002.
reCAPTCHA: stop spam, read books. Department of Computer Science, Carnegie Mellon University, http://www.recaptcha.net/; 2007.
The CAPTCHA project: completely automatic public Turing test to tell computers and humans apart. Department of Computer Science, Carnegie Mellon University, http://www.captcha.net; 2000.
Turing AM. Computing machinery and intelligence. Mind 1950; LIX (59)(236):433–60.
Vivo MD, Vivo GOD, Koeneke R, Isern G. Internet vulnerabilities related to tcp/ip and t/tcp. SIGCOMM Comput. Commun. Rev. 1999;29(1):81–5.
Yahoo! CAPTCHA is broken, http://network-security-research.blogspot.com/2008/01/yahoo-captcha-is- broken.html; 2008.

Alessandro Basso is a postdoctoral researcher at the Department of Computer Science, University of Torino, Italy, after having obtained his Ph.D. from the same department. His research interests are in the area of computer and network security and include web application security, digital watermarking of multimedia content and mobile device security.

Stefano Sicco obtained his B.Sc. and M.Sc. in Computer Science at the University of Torino, Italy. Currently, he is a temporary researcher at the Department of Computer Science, University of Torino. His research activity includes web application security, networking security and digital identity management.