security risks for online services by relying on reCAPTCHA

Benjamin Wegener - <info[at]wegeneredv[dot]de>

October 22, 2010

Abstract

A description of major flaws in Google's reCAPTCHA service and a proof of concept regarding successful exploitation using freely available tools in order to achieve a recognization rate of approximately ten percent.

1 Introduction

Many popular websites use the CAPTCHAs (a contrieved acronym for "**c**ompletely **a**utomated **p**ublic **t**uring test to tell **c**omputers and **h**umans **a**part"[1]) provided by reCAPTCHA to prevent the automated subscription and registration or unauthorized and extensive use of their offers and services.
reCAPTCHA was originally developed by Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham and Manuel Blum of the Computer Science Department of the Carnegie Mellon University in Pittsburgh PA to help the digitization of books by using CAPTCHAS of scanned words the OCR-Software used couldn't recognize[2]. In September 2009 Google acquired the system to use it on a wider range of projects like Google Books[3].

A challenge served by reCAPTCHA consists of two words - one already known by the system and one to be identified by the human solver. The unknown word is given to many humans in a short period of time and the most answered solution is taken. But reCAPTCHA only checks if the known word is typed in correct and also allows variety of accidental errors to be made.

2 Major Flaws

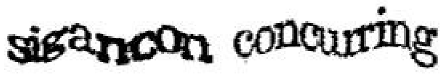

Figure 1

As mentioned before, only the already known word is taken for the check if the user is human or

1 http://en.wikipedia.org/wiki/CAPTCHA
2 Luis von Ahn, Ben Maurer, Colin McMillen, David Abraham and Manuel Blum (2008). "reCAPTCHA: Human-Based Character Recognition via Web Security Measures" (PDF). *Science* **321** (5895): 1465–1468.
3 "Teaching computers to read: Google acquires reCAPTCHA". Google. Retrieved 2009-09-16.

not. This allows the use of a dictionary attack for solving the challenge, because it is likely that this word is more common than the other one. This could be accomplished by using the by the used ORC-software recognized phrase and comparing it to the wordlist. If this word doesn't occur in the list, the most likely replacement would be used. Since reCAPTCHA takes mostly english books and sources to be digitized, a comprehensive and huge wordlist could be made out of the decompressed archive of the english wikipedia[4] using a simple algorithm to extract all the words separated by punctuation marks and order them alphabetically.
A user is allowed to enter 32 wrong answers before security measures are taken and a CAPTCHA consisting of two known words is generated which must be answered perfectly. After that the IP-address is blocked for a brief period of time.
The puzzle is monochromatic and besides a simple wave distortion no other effects are applied to the phrase.
The font used mostly is of serif type because of the sources - so a OCR-Software doesn't need to be trained for other fonts than this.

3 Preprocessing

To attack the mentioned lacks of security, the image is beeing transformed by some simple algorithms:

3.1 adjustment of contrast

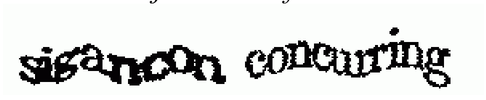
Figure 2

By using different intensities of contrast applied two problems are dealt with at once: the distinction between normal fonts and bold ones and the removal of "dirty spots" or artifacts left over by OCR.

3.2 reszize using hq2x

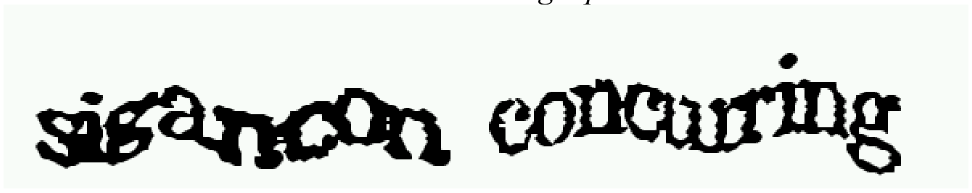
Figure 3

The opensource program *hq2x*[5] uses a pixelart algorithm to enlarge pixel based imagery which fits perfectly in the purpose here after the contrast is applied to get a better overall quality.

3.3 distortion removal

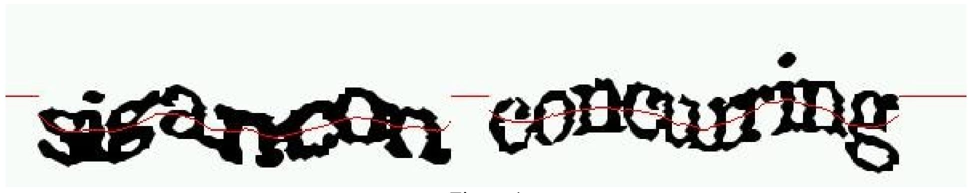
Figure 4

An algorithm to determine the average centre line of the phrase served by reCAPTCHA is applied. This is accomplished by subtracting the bottom least black pixel's y-position from the top least ones'

4 http://download.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
5 http://en.wikipedia.org/wiki/Hqx

for everey column of the image. The origin is in the top left corner as usual. To get a nice line without too huge "bumps" in it, a exponential smoothing with a factor of of 0.2 is applied to the line in both directions. So 1. every pixel is taken and the y-value is multiplied with 0.8 and then added to the following pixel's y-value multiplied by 0.2 from left to right and afterwards from right to left.
Then every column is moved by (height_of_image/2-centre_value(x)).

3.4 partially removal of letter intersections

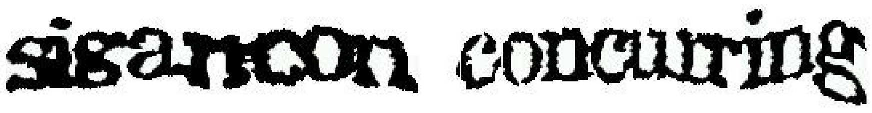

Figure 5

This part is still experimental and tries to find a solution to one of the difficult security aspects of the reCAPTCHA system. It basically deals with the intersecting letters and is based on the idea, that two letters don't intersect if the only black pixels for this column are in an distict area around the centre line. The only positions in this example as shown in figure 5 is the first letter of the second word - the "c" of "concurring" - and the first line is a negative example of what can happen and the other white line succesfully seperates the first "c" and second "o".

4 Training the OCR-Software

The OCR-Software used in this proof of concept is tesseract-ocr[6], a opensource ocr-engine developed by the Hewlett-Packard Labs[7] between 1985 and 1995 and is now revised by Google itself. The tesseract is the 4-dimansional analog to the cube.
The original trained language of tesseract-ocr obviously is English, so it doesn't need to be retrained for this special purpose. But the engine in current version 3.0 is only supporting wordlists to a limited total count - so it's essential to use the wikipedia wordlist before the recognization and train the engine with a list of most common words in the english language[8]. It is also useful to reject some sugestions made by tesseract-ocr - e. g. if the answer includes non-standard characters such as $%&!"`.- etc.
I expect a recognization rate for antirecaptcha[9] of over fifteen percent once the training procedure is finished.

5 Conclusion

Major internet services such as facebook, twitter, single-click hoster and most public boards and forums rely on the security of reCAPTCHA. Using proxies and/or dynamic IPs one could automate for example the registration on twitter to gain the maximum number of (fake) followers.
One year ago Jonathan Wilkins made suggestions[10] to strengthen reCAPTCHA (and others) and Chad Houck explained[11], how to bypass the annoying ellipse and the black bar which both do not appear in reCAPTCHA images as of today. This might allow some humans to decipher these puzzles mor easily but it also lowers the security level regarding to automated approaches.

6 http://code.google.com/p/tesseract-ocr/downloads/list
7 http://en.wikipedia.org/wiki/Tesseract_%28software%29#History
8 http://www.wordfrequency.info/files/entriesWithoutCollocates.txt
9 http://wegeneredv.de/arc
10 http://bitland.net/captcha.pdf
11 http://n3on.org/projects/reCAPTCHA/

I strongly advise Google to bring back things like the ellipse mentioned to increase security of reCAPTCHA again. Otherwise the great idea of "stop(ping) spam. read(ing) books."[12] will become useless very soon.

6 Appendix

Source Code:

```
------------------------------------------------------------------------------------------------------------
<?php
// anti-recaptcha v0.2a (c)opyleft 2010 http://wegeneredv.de/arc **************
// ****************************************************************************
$debug_mode=false;
$input_image="input.jpg";
$contrast_min=122;
$contrast_max=134;
$mid_exponential_smoothing_factor=0.1;
$intersection_threshold=15;
$h2qx_commandline='hq2x.exe contrast.bmp resized.bmp';
$vietocr_commandline='VietOCR.exe out.jpg vietoutput -l eng';
$shit="!§$%&@";
// function imagesetpixel_rgb ************************************************
// ****************************************************************************
function imagesetpixel_rgb($im,$x_pos,$y_pos,$r,$g,$b){
	imagesetpixel($im,$x_pos,$y_pos,65536*$r+256*$g+$b);
}
// function readfile_chunked *************************************************
// ****************************************************************************
function readfile_chunked ($filename) {
	$chunksize = 1*(1024*1024); // how many bytes per chunk
	$buffer = '';
	$handle = fopen($filename, 'rb');
	if ($handle === false) {
		return false;
	}
	while (!feof($handle)) {
		$buffer = fread($handle, $chunksize);
	}
	fclose($handle);
	return $buffer;
}
// function imagecreatefrombmp ***********************************************
// ****************************************************************************
// save Bitmap-File with GD library
// written by mgutt of http://www.programmierer-forum.de/function-imagecreatefrombmp-laeuft-mit-
allen-bitraten-t143137.htm
// based on the function by DHKold of http://www.php.net/manual/de/function.imagecreate.php#53879
if (!function_exists('imagecreatefrombmp')) { function imagecreatefrombmp($filename) {
	// version 1.00
	if (!($fh = fopen($filename, 'rb'))) {
		trigger_error('imagecreatefrombmp: Can not open ' . $filename, E_USER_WARNING);
		return false;
	}
	// read file header
	$meta = unpack('vtype/Vfilesize/Vreserved/Voffset', fread($fh, 14));
	// check for bitmap
	if ($meta['type'] != 19778) {
		trigger_error('imagecreatefrombmp: ' . $filename . ' is not a bitmap!',
E_USER_WARNING);
```

12 http://www.google.com/recaptcha

```
                return false;
        }
        // read image header
        $meta +=
unpack('Vheadersize/Vwidth/Vheight/vplanes/vbits/Vcompression/Vimagesize/Vxres/Vyres/Vcolors/Vi
mportant', fread($fh, 40));
        // read additional 16bit header
        if ($meta['bits'] == 16) {
                $meta += unpack('VrMask/VgMask/VbMask', fread($fh, 12));
        }
        // set bytes and padding
        $meta['bytes'] = $meta['bits'] / 8;
        $meta['decal'] = 4 - (4 * (($meta['width'] * $meta['bytes'] / 4)- floor($meta['width'] *
$meta['bytes'] / 4)));
        if ($meta['decal'] == 4) {
                $meta['decal'] = 0;
        }
        // obtain imagesize
        if ($meta['imagesize'] < 1) {
                $meta['imagesize'] = $meta['filesize'] - $meta['offset'];
                // in rare cases filesize is equal to offset so we need to read physical size
                if ($meta['imagesize'] < 1) {
                        $meta['imagesize'] = @filesize($filename) - $meta['offset'];
                        if ($meta['imagesize'] < 1) {
                                trigger_error('imagecreatefrombmp: Can not obtain filesize of ' .
$filename . '!', E_USER_WARNING);
                                return false;
                        }
                }
        }
        // calculate colors
        $meta['colors'] = !$meta['colors'] ? pow(2, $meta['bits']) : $meta['colors'];
        // read color palette
        $palette = array();
        if ($meta['bits'] < 16) {
                $palette = unpack('l' . $meta['colors'], fread($fh, $meta['colors'] * 4));
                // in rare cases the color value is signed
                if ($palette[1] < 0) {
                        foreach ($palette as $i => $color) {
                                $palette[$i] = $color + 16777216;
                        }
                }
        }
        // create gd image
        $im = imagecreatetruecolor($meta['width'], $meta['height']);
        $data = fread($fh, $meta['imagesize']);
        $p = 0;
        $vide = chr(0);
        $y = $meta['height'] - 1;
        $error = 'imagecreatefrombmp: ' . $filename . ' has not enough data!';
        // loop through the image data beginning with the lower left corner
        while ($y >= 0) {
                $x = 0;
                while ($x < $meta['width']) {
                        switch ($meta['bits']) {
                                case 32:
                                case 24:
                                        if (!($part = substr($data, $p, 3))) {
                                                trigger_error($error, E_USER_WARNING);
                                                return $im;
                                        }
                                        $color = unpack('V', $part . $vide);
                                        break;
                case 16:
                    if (!($part = substr($data, $p, 2))) {
                        trigger_error($error, E_USER_WARNING);
                        return $im;
                    }
                    $color = unpack('v', $part);
                    $color[1] = (($color[1] & 0xf800) >> 8) * 65536 + (($color[1] &
0x07e0) >> 3) * 256 + (($color[1] & 0x001f) << 3);
                    break;
                case 8:
                    $color = unpack('n', $vide . substr($data, $p, 1));
                    $color[1] = $palette[ $color[1] + 1 ];
                    break;
                case 4:
                    $color = unpack('n', $vide . substr($data, floor($p), 1));
                    $color[1] = ($p * 2) % 2 == 0 ? $color[1] >> 4 : $color[1] & 0x0F;
                    $color[1] = $palette[ $color[1] + 1 ];
                    break;
                case 1:
                    $color = unpack('n', $vide . substr($data, floor($p), 1));
                    switch (($p * 8) % 8) {
                        case 0:
                            $color[1] = $color[1] >> 7;
                            break;
                        case 1:
                            $color[1] = ($color[1] & 0x40) >> 6;
                            break;
                        case 2:
                            $color[1] = ($color[1] & 0x20) >> 5;
                            break;
                        case 3:
                            $color[1] = ($color[1] & 0x10) >> 4;
                            break;
                        case 4:
                            $color[1] = ($color[1] & 0x8) >> 3;
                            break;
                        case 5:
                            $color[1] = ($color[1] & 0x4) >> 2;
                            break;
                        case 6:
                            $color[1] = ($color[1] & 0x2) >> 1;
                            break;
                        case 7:
                            $color[1] = ($color[1] & 0x1);
                            break;
                    }
                    $color[1] = $palette[ $color[1] + 1 ];
                    break;
                default:
                    trigger_error('imagecreatefrombmp: ' . $filename . ' has ' .
$meta['bits'] . ' bits and this is not supported!', E_USER_WARNING);
                    return false;
            }
            imagesetpixel($im, $x, $y, $color[1]);
            $x++;
            $p += $meta['bytes'];
        }
        $y--;
        $p += $meta['decal'];
    }
    fclose($fh);
    return $im;
}}
```

```
// function imagebmp ********************************************************
// ***************************************************************************
// create Bitmap-File with GD library
// written by mgutt of http://www.programmiererforum.de/imagebmp-gute-funktion-gefunden-
t143716.htm
// based on the function by legend(legendsky@hotmail.com) of http://www.ugia.cn/?p=96
function imagebmp($im, $filename='', $bit=24, $compression=0) {
	if (!in_array($bit, array(1, 4, 8, 16, 24, 32))) {
		$bit = 24;
	}
	else if ($bit == 32) {
		$bit = 24;
	}
	$bits = pow(2, $bit);
	imagetruecolortopalette($im, true, $bits);
	$width = imagesx($im);
	$height = imagesy($im);
	$colors_num = imagecolorstotal($im);
	$rgb_quad = '';
	if ($bit <= 8) {
		for ($i = 0; $i < $colors_num; $i++) {
			$colors = imagecolorsforindex($im, $i);
			$rgb_quad .= chr($colors['blue']) . chr($colors['green']) . chr($colors['red']) .
"\0";
		}
		$bmp_data = '';
		if ($compression == 0 || $bit < 8) {
			$compression = 0;
			$extra = '';
			$padding = 4 - ceil($width / (8 / $bit)) % 4;
			if ($padding % 4 != 0) {
				$extra = str_repeat("\0", $padding);
			}
			for ($j = $height - 1; $j >= 0; $j --) {
				$i = 0;
				while ($i < $width) {
					$bin = 0;
					$limit = $width - $i < 8 / $bit ? (8 / $bit - $width + $i) * $bit : 0;
					for ($k = 8 - $bit; $k >= $limit; $k -= $bit) {
						$index = imagecolorat($im, $i, $j);
						$bin |= $index << $k;
						$i++;
					}
					$bmp_data .= chr($bin);
				}
				$bmp_data .= $extra;
			}
		}
		// RLE8
		else if ($compression == 1 && $bit == 8) {
			for ($j = $height - 1; $j >= 0; $j--) {
				$last_index = "\0";
				$same_num = 0;
				for ($i = 0; $i <= $width; $i++) {
					$index = imagecolorat($im, $i, $j);
					if ($index !== $last_index || $same_num > 255) {
						if ($same_num != 0) {
							$bmp_data .= chr($same_num) .
chr($last_index);
						}
						$last_index = $index;
						$same_num = 1;
					}
```

```
                                        else {
                                                $same_num++;
                                        }
                                }
                                $bmp_data .= "\0\0";
                        }
                        $bmp_data .= "\0\1";
                }
                $size_quad = strlen($rgb_quad);
                $size_data = strlen($bmp_data);
        }
        else {
                $extra = '';
                $padding = 4 - ($width * ($bit / 8)) % 4;
                if ($padding % 4 != 0) {
                        $extra = str_repeat("\0", $padding);
                }
                $bmp_data = '';
                for ($j = $height - 1; $j >= 0; $j--) {
                        for ($i = 0; $i < $width; $i++) {
                                $index  = imagecolorat($im, $i, $j);
                                $colors = imagecolorsforindex($im, $index);
                                if ($bit == 16) {
                                        $bin = 0 << $bit;
                                        $bin |= ($colors['red'] >> 3) << 10;
                                        $bin |= ($colors['green'] >> 3) << 5;
                                        $bin |= $colors['blue'] >> 3;
                                        $bmp_data .= pack("v", $bin);
                                }
                                else {
                                        $bmp_data .= pack("c*", $colors['blue'], $colors['green'],
$colors['red']);
                                }
                        }
                        $bmp_data .= $extra;
                }
                $size_quad = 0;
                $size_data = strlen($bmp_data);
                $colors_num = 0;
        }
        $file_header = 'BM' . pack('V3', 54 + $size_quad + $size_data, 0, 54 + $size_quad);
        $info_header = pack('V3v2V*', 0x28, $width, $height, 1, $bit, $compression, $size_data, 0, 0,
$colors_num, 0);
        if ($filename != '') {
                $fp = fopen($filename, 'wb');
                fwrite($fp, $file_header . $info_header . $rgb_quad . $bmp_data);
                fclose($fp);
                return true;
        }
        echo $file_header . $info_header. $rgb_quad . $bmp_data;
        return true;
}
for($contrast=$contrast_min;$contrast<$contrast_max;$contrast++){
        $source_img=imagecreatefromjpeg($input_image);
// contrast *******************************************************************
// ****************************************************************************
        $height=imagesy($source_img);
        $width=imagesx($source_img);
        for($x=0;$x<$width;$x++){
                for($y=0;$y<$height;$y++){
                        $rgb=imagecolorat($source_img,$x,$y);
                        $r=($rgb>>16)&0xFF;
                        $g=($rgb>>8)&0xFF;
```

```
				$b=$rgb&0xFF;
				if(($r>$contrast)&&($r>$contrast)&&($r>$contrast)){
					imagesetpixel($source_img,$x,$y,16777215);
				}else{
					imagesetpixel($source_img,$x,$y,0);
				}
			}
		}
// resize image **************************************************************
// ***************************************************************************
		imagebmp($source_img,"contrast.bmp");
		$h2qx_output=shell_exec($h2qx_commandline);
		$img=imagecreatefrombmp('resized.bmp');
		$height=imagesy($img);
		$width=imagesx($img);
// reaglignment **************************************************************
// ***************************************************************************
		$temp=@imagecreatetruecolor($width,$height) or die('Cannot Initialize new GD image
stream');
		for($x=0;$x<$width;$x++){
			for($y=0;$y<$height;$y++){
				imagesetpixel($temp,$x,$y,16777215);
			}
		}
// calculate top and bottom **************************************************
		$top=null;
		$bottom=null;
		for($x=1;$x<$width-1;$x++){
			for($y=0;$y<$height;$y++){
				$c=imagecolorat($img,$x,$y);
				if($c==0){
					$top[$x]=$y;
					break;
				}
			}
			for($y=$height-1;$y>0;$y--){
				$c=imagecolorat($img,$x,$y);
				if($c==0){
					$bottom[$x]=$y;
					break;
				}
			}
		}
// calculate mid *************************************************************
		$mid=null;
		$mid_1stpass=null;
		$mid_2ndpass=null;
		for($x=0;$x<$width;$x++){
			if($bottom[$x]-$top[$x]==0){
				$mid_1stpass[$x]=abs($height/2);
			}else{
				$mid_1stpass[$x]=abs((1-
$mid_exponential_smoothing_factor)*$mid_1stpass[$x-1]+
$mid_exponential_smoothing_factor*(($top[$x]+$bottom[$x])/2));
			}
		}
		for($x=$width-1;$x>=0;$x--){
			if($bottom[$x]-$top[$x]==0){
				$mid_2ndpass[$x]=abs($height/2);
			}else{
				$mid_2ndpass[$x]=abs((1-
$mid_exponential_smoothing_factor)*$mid_2ndpass[$x+1]+
$mid_exponential_smoothing_factor*(($top[$x]+$bottom[$x])/2));
```

```
				}
		}
		for($x=0;$x<=$width;$x++){
				$mid[$x]=(($mid_1stpass[$x]+$mid_2ndpass[$x])/2);
				$vector[$x]=(($height/2)-($mid[$x]));
		}
// clear intersections ***********************************************************
		for($x=0;$x<$width;$x++){
				if((($bottom[$x]-
$top[$x])<=$intersection_threshold)&&($bottom[$x]>=$mid[$x])&&($top[$x]<=$mid[$x])){
						$bottom[$x]=0;
						$top[$x]=0;
						for($y=0;$y<$height;$y++){
								imagesetpixel_rgb($img,$x,$y,255,255,255);
						}
				}
		}
// debug *****************************************************************************
		if($debug_mode){
				$debug=imagecreatefrombmp('resized.bmp');
				for($x=0;$x<$width;$x++){
						imagesetpixel_rgb($debug,$x,$mid[$x],255,0,0);
				}
				imagejpeg($debug,"debug.jpg");
		}
// remove distortion ***********************************************************
		for($x=0;$x<$width;$x++){
				if($bottom[$x]-$top[$x]>0){
						for($y=floor($mid[$x]);$y<$height;$y++){
								if($y<=$bottom[$x]){
										$c=imagecolorat($img,$x,$y);
										imagesetpixel($temp,$x,$y+$vector[$x],$c);
								}
						}
						for($y=floor($mid[$x])-1;$y>0;$y--){
								if($y>=$top[$x]){
										$c=imagecolorat($img,$x,$y);
										imagesetpixel($temp,$x,$y+$vector[$x],$c);
								}
						}
				}
		}
// output to ocr *******************************************************************
//*************************************************************************************
		imagejpeg($temp,"out.jpg");
		$vietocr_output=shell_exec($vietocr_commandline);
		$viet_output=readfile_chunked('vietoutput.txt');
		if($debug_mode){
				echo "<img src=input.jpg><br><img src=contrast.bmp><br><img
src=debug.jpg><br><img src=out.jpg><br>Output:<br>".$viet_output;
				imagedestroy($debug);
		}else{
				imagedestroy($temp);
				imagedestroy($img);
				unlink('resized.bmp');
				unlink('out.jpg');
				unlink('contrast.bmp');
				unlink('debug.jpg');
		}
		$word1_disqualified=false;
		$word2_disqualified=false;
		if($viet_output=='~'){
				$word1_disqualified=true;
```

```
			$word2_disqualified=true;
		}
		$word1=strtolower(strstr($viet_output,' ',true));
		$word2=substr(strstr($viet_output,' '),1,strlen(strstr($viet_output,' '))-3);
		$word1_array=str_split($word1);
		$word2_array=str_split($word2);
		for($pos=0;$pos<count($word1_array);$pos++){
			$letter=ord($word1_array[$pos]);
			if(($letter<45)||(($letter>45)&&($letter<48))||(($letter>57)&&($letter<97))||($letter>122)){
				$word1_disqualified=true;
				break;
			}
		}
		if(!$word1_disqualified){
			$word1_words[$contrast-$contrast_min]=$word1;
		}
		for($pos=0;$pos<count($word2_array);$pos++){
			$letter=ord($word2_array[$pos]);
			if(($letter<45)||(($letter>45)&&($letter<48))||(($letter>57)&&($letter<97))||($letter>122)){
				$word2_disqualified=true;
				break;
			}
		}
		if(!$word2_disqualified){
			$word2_words[$contrast-$contrast_min]=$word2;
		}
	}
	if(count($word1_words)==0){$word1_words[0]=$shit;}
	if(count($word2_words)==0){$word2_words[0]=$shit;}
	$word1_word_count=array_count_values($word1_words);
	arsort($word1_word_count);

	$word1_best_found=false;
	for($word1_word_count_index=0;$word1_word_count_index<count($word1_word_count)-
1;$word1_word_count_index++){
		$current=current($word1_word_count);
		next($word1_word_count);
		$next=current($word1_word_count);
		if($current>=$next){
			$word1_best_found=true;
			prev($word1_word_count);
			$found_word1[]=key($word1_word_count);
			break;
		}
	}
	$word2_word_count=array_count_values($word2_words);
	arsort($word2_word_count);
	$word2_best_found=false;
	for($word2_word_count_index=0;$word2_word_count_index<count($word2_word_count)-
1;$word2_word_count_index++){
		$current=current($word2_word_count);
		next($word2_word_count);
		$next=current($word2_word_count);
		if($current>=$next){
			$word2_best_found=true;
			prev($word2_word_count);
			$found_word2[]=key($word2_word_count);
			break;
		}
	}
	if($word1_best_found){
		$word_output=$found_word1[0];
	}else{
```

```
	$word_output=$shit;
}
if($word2_best_found){
	$word_output.=' '.$found_word2[0];
}else{
	$word_output.=' '.$shit;
}
$output_file=fopen('output.txt','w');
fwrite($output_file,$word_output);
fclose($output_file);
if($debug_mode){echo $word_output;}
?>
```