White Paper

TesserCap—A Visual CAPTCHA Solving Tool

By Gursev Singh Kalra
Managing Consultant
McAfee® Foundstone® Professional Services

Table of Contents

Introduction

CAPTCHAs[1] are used to prevent automated software from performing actions that degrade the quality of service of a given system, whether due to abuse or resource expenditure. Each CAPTCHA implementation derives its strength by increasing the system's complexity to perform image preprocessing, segmentation, and classification.

To analyze the strength of CAPTCHA deployments on the Internet, I conducted a research project on more than 200 high-traffic websites and several CAPTCHA service providers listed on Quantcast's Top 1 Million Ranking Websites (http://www.quantcast.com/top-sites-1). It was observed that an alarming number of CAPTCHAs (image designs) could be broken by a combination of image preprocessing[2] and optical character recognition[3] (OCR) engines. TesserCap was thus written to test CAPTCHA designs based upon these observations.

TesserCap Features

TesserCap is a GUI-based, highly flexible, interactive, point-and-shoot CAPTCHA analysis tool with the following features:

- A generic image preprocessing engine that can be configured as per the CAPTCHA type being analyzed
- Tesseract[4,5] as its OCR engine to retrieve text from preprocessed CAPTCHAs
- Web proxy and custom HTTP headers support
- CAPTCHA statistical analysis support
- Character set selection for the OCR engine

System Requirements

System requirements for TesserCap are:

- Operating system: Microsoft Windows 7, Windows XP, Windows 2003
- .Net Framework: 4.0

TesserCap Installation

The TesserCap zip file containing the Microsoft Windows installer can be downloaded from http://www.mcafee.com/us/downloads/free-tools/index.aspx.

After extracting the contents from the zip file, go to the directory where the zip file is extracted, and run TesserCap1.0_Setup.exe. This will begin the installation process. The installer is self-explanatory. Figure 1 below shows the first installation screen and Figure 2 shows a sample TesserCap run.

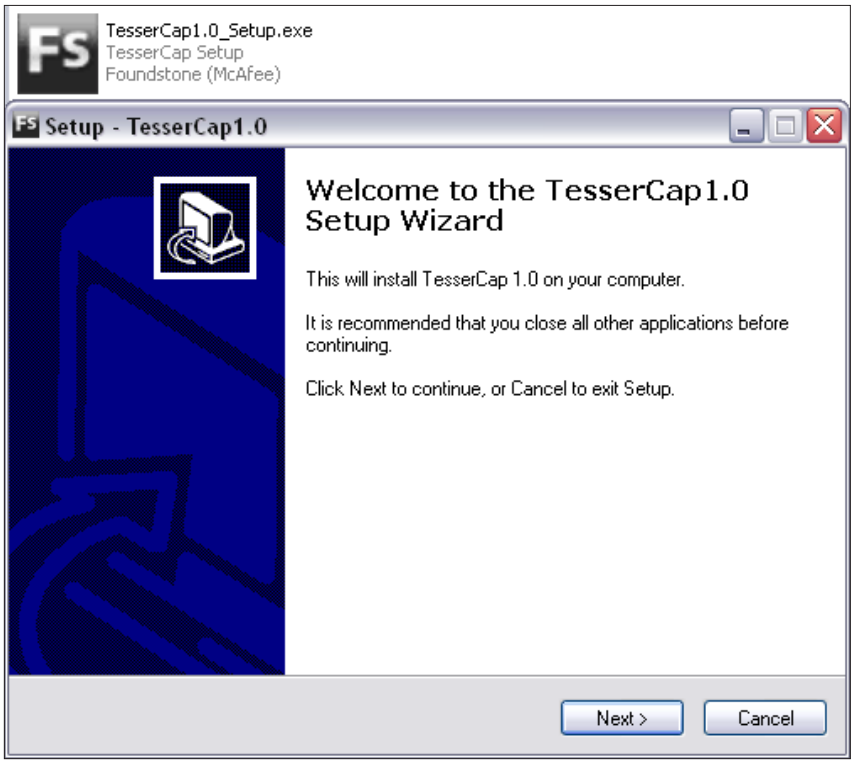

Figure 1. TesserCap installation.

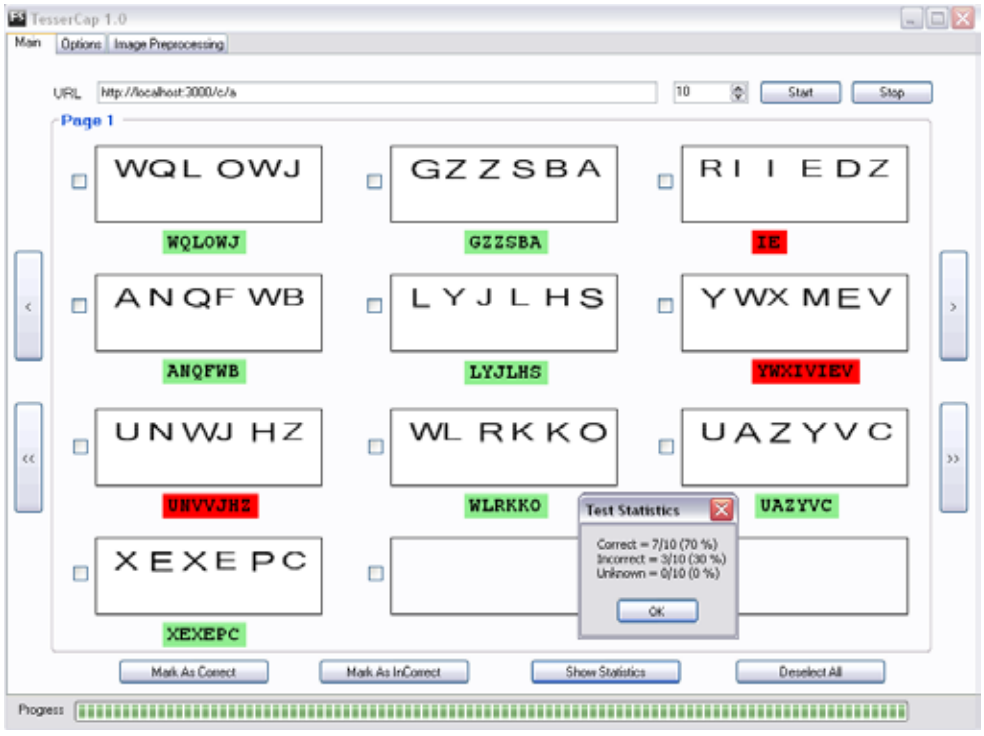

Figure 2. TesserCap sample run.

TesserCap Explained

Providing a URL that generates CAPTCHA (see below) and clicking on the start button is all that is required to initiate a CAPTCHA test using TesserCap. Let us now look at the TesserCap GUI and its capabilities.

Main tab

The main tab houses controls that are used to start and stop a CAPTCHA test, generate test statistics, perform navigation, and select CAPTCHAs for image preprocessing.

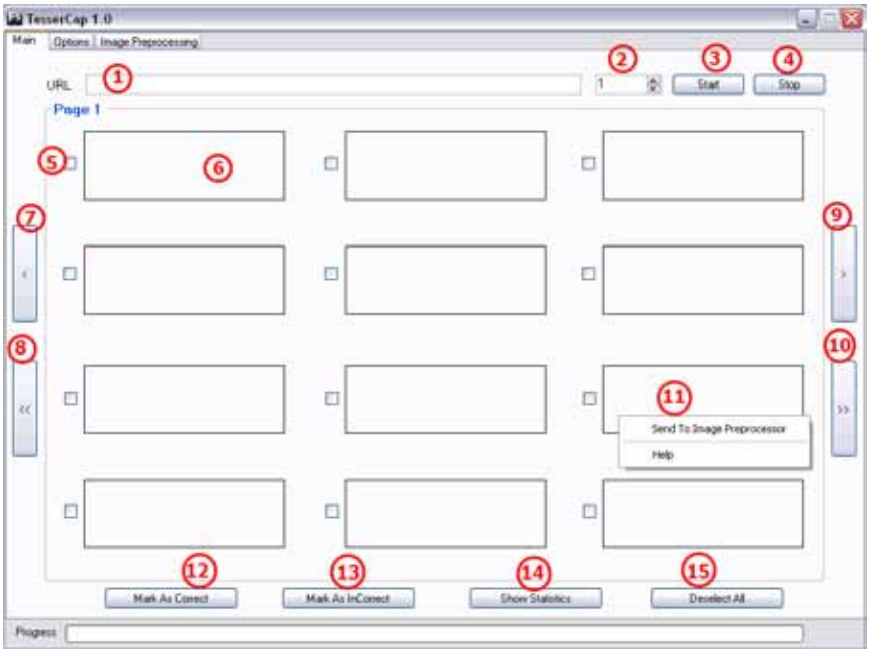

Figure 3. Image shows TesserCap Main tab.

The table below summarizes the various controls in the main tab:

Control Number	Brief Description
1	CAPTCHA URL input box
2	Provide the number of CAPTCHAs to be tested
3	Start a test
4	Stop a test
5	Select a CAPTCHA for statistical analysis
6	Left click and select a CAPTCHA for statistical analysis
7	Go to previous CAPTCHA page
8	Go to first CAPTCHA page
9	Go to next CAPTCHA page
10	Go to last CAPTCHA page
11	Right click and send a CAPTCHA for image preprocessing
12	Mark selected CAPTCHA solution(s) as correct
13	Mark Selected CAPTCHA solution(s) as incorrect
14	Display test statistics
15	Deselect all selected CAPTCHAs

Starting and stopping a test (controls 1, 2, 3, and 4)

The URL (control #1) input field should be provided with the exact URL that is used by the web application to retrieve CAPTCHAs. The URL can be extracted by right clicking on the CAPTCHA and copying the URL or viewing page source and extracting the URL from the `src` attribute of the image `<img>` tag.

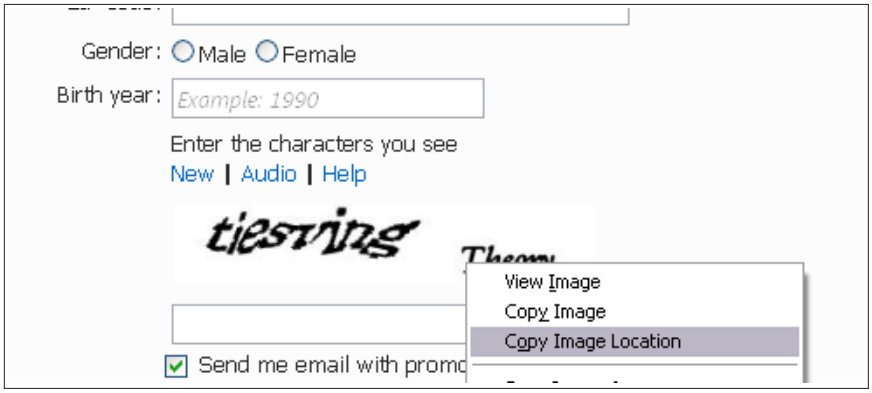

Figure 4. Right clicking on the CAPTCHA image and copying the URL helps extracts the URL that generates CAPTCHA.

Figure 5. Image shows src attribute of <img> tag in the HTML source.

Control 2 is used to provide the number of CAPTCHAs to be retrieved from the target URL. Start and stop buttons (control 3 and control 4) are used to start and stop the test.

Here are a few important considerations:

- Tests can be run against valid HTTP/HTTPS URLs only
- If the content returned by the provided URL is not an image, TesserCap will error out and the test will halt
- TesserCap does not follow redirects by default. If the test URL needs to follow redirects in order to retrieve an image, please check the relevant option in the Options tab.

CAPTCHA navigation (controls 7, 8, 9, and 10)

TesserCap displays 12 CAPTCHAs at a time. Use the buttons marked 7, 8, 9, and 10 to navigate through various CAPTCHA pages if the test configuration requires more than 12 CAPTCHAs for analysis.

Performing statistical analysis (controls 12, 13, 14, and 15)

Once the TesserCap run is complete, the next step is to perform statistical analysis of the CAPTCHA solutions. Statistical analysis can be performed as mentioned below:

1. After a test is complete (or in progress), select any number of CAPTCHAs by clicking on the check box next to the CAPTCHA or the CAPTCHA itself.
2. Mark the selected CAPTCHAs as correct (control 12) or incorrect (control 13) by clicking on the appropriate button.
3. Bulk selection across various pages can be performed before assigning correct/incorrect status to the CAPTCHAs.
4. To view the statistical analysis, click on Show Statistics (control 14).
5. To cancel CAPTCHA selections, click on Deselect All (control 15).

Image preprocessing for CAPTCHAs (control 11)

The first step in image preprocessing requires the user to configure various image filters and modifiers in the Image Preprocessing tab. The steps below outline the process to create an image preprocessing template for a given CAPTCHA type.

1. Initiate a small test, and retrieve one/multiple CAPTCHAs from the application under test.
2. Identify any one CAPTCHA to be used for the creation of an image preprocessing template.
3. Right click on the CAPTCHA and select Send to Image Preprocessor.
4. Go to the Image Preprocessor tab, and configure the filters as per the requirement. Detailed information on configuring the filters is available in the Image Preprocessing Tab section of this paper.
5. Once an image preprocessing template is created, go to the Options tab and check Enable Image Preprocessing. From this point on, the image preprocessing template will be applied to all CAPTCHAs upon retrieval.
6. Start a new test.

Options tab

The options tab houses various configuration controls for TesserCap. These configuration controls allow users to provide an OCR character set, web proxy settings, HTTP redirect configuration settings, image preprocessing selection, and custom HTTP headers.

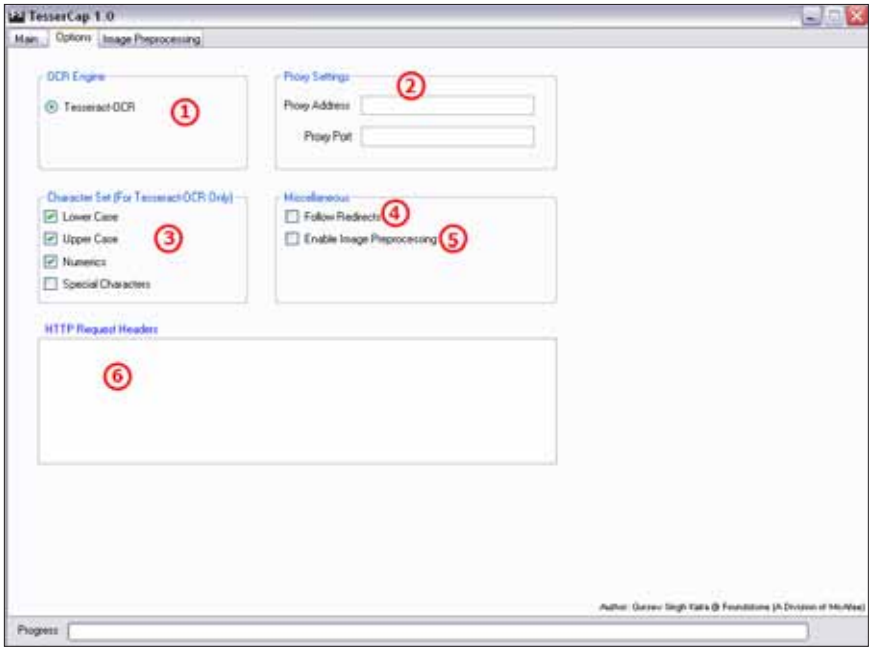

Figure 6. Image shows TesserCap Options tab.

The table below summarizes the various controls in the Options tab.

Control Number	Brief Description
1	Select the OCR engine.
2	Enter Internet proxy configuration
3	Select the CAPTCHA character set for higher accuracy
4	Allow TesserCap to follow redirects
5	Allow TesserCap to preprocess all CAPTCHA images
6	If CAPTCHA retrieval requires HTTP headers, enter them here

CAPTCHA character set (control group 3)

TesserCap utilizes Tesseract-OCR's text recognition capabilities to retrieve text from CAPTCHAs. It is recommended that you use the checkboxes in control 3 to communicate the CAPTCHA character set for higher accuracy. Based on the character set selected in this control, the appropriate configuration file with preset `tessedit_char_whitelist` parameter will be picked by Tesseract while solving CAPTCHAs. The custom character set files can be seen in the following path:

```
<ProgramFiles>\Tesseract-OCR\tessdata\configs
```

It is not recommended that you modify or remove these files.

Internet proxy settings (control group 2)

If a web proxy is required for Internet access, relevant settings can be provided in control 2.

HTTP redirects (control 4)

TesserCap does not follow HTTP redirects by default. If the test URL needs to follow redirects in order to retrieve an image, please check the Follow Redirects checkbox (control 4).

Enabling image preprocessing (control 5)

Image preprocessing is disabled by default. Users are expected to first configure image preprocessing filters as per the CAPTCHAs under test and then activate this module. Once the Enable Image Preprocessing checkbox (control #5) is checked, all CAPTCHAs downloaded from that point onwards will be preprocessed upon retrieval and then presented to Tesseract-OCR for text extraction and PictureBox for display.

Custom HTTP headers (control 6)

Certain websites expect HTTP headers like `Accept`, `Cookie`, `Referrer`, and others to be present in HTTP requests before they return any content or CAPTCHAs. Using a web proxy (Fiddler, Burp, Charles, WebScarab, Paros, or others), capture the request headers and enter them in the textbox marked control #6.

Image Preprocessing tab

The Image Preprocessing tab houses the various image preprocessing stages along with a verification component. Each image preprocessing stage reflects the changes made on a CAPTCHA to its own picture box and to the picture boxes in the subsequent processing stages. The following two figures show image preprocessing to use for solving a CAPTCHA.

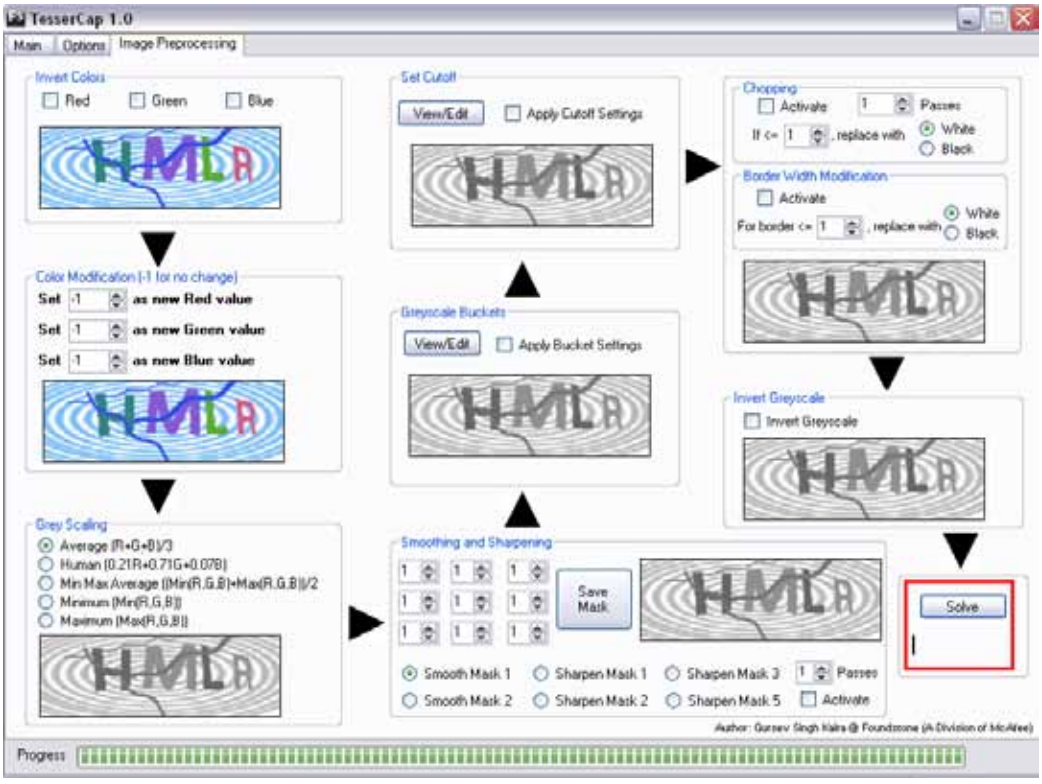

Figure 7. An OCR failure while extracting text from a given CAPTCHA with color complexities and spatial irregularities.

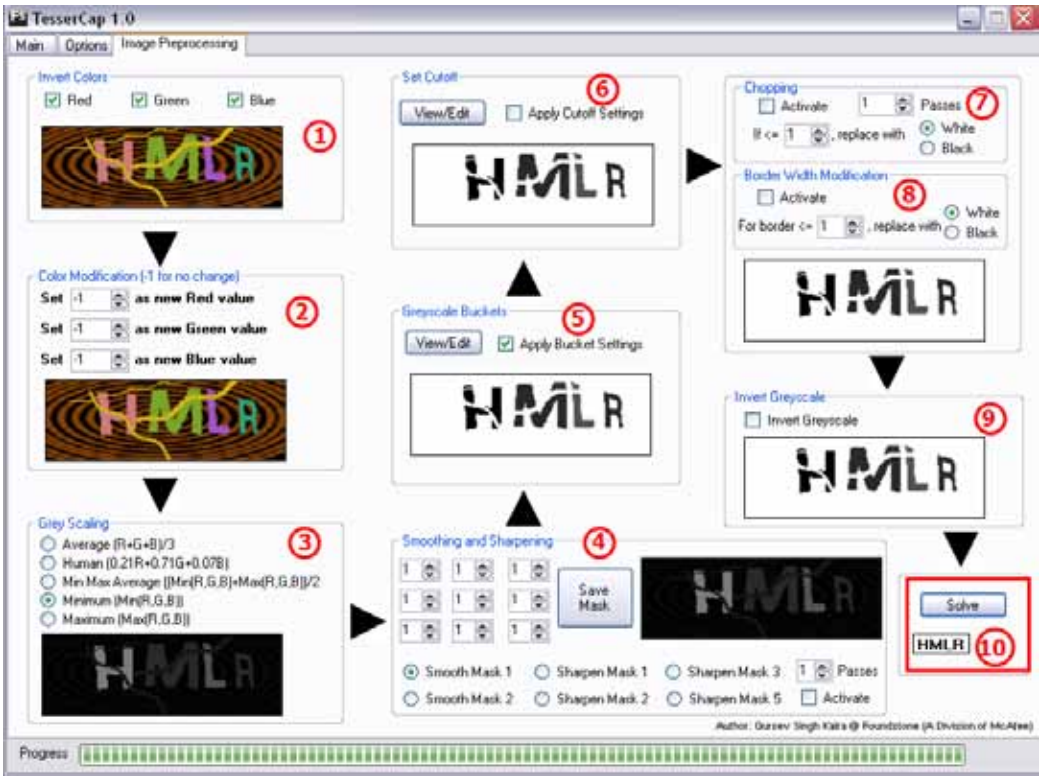

Figure 8. Successful text extraction after applying image preprocessing filters.

The table below summarizes the various controls groups in the image preprocessing tab.

Control Number	Brief Description
1	Invert red, green, or blue component of each pixel
2	Set particular value for red, green, or blue component of each pixel
3	Convert the CAPTCHA to grayscale
4	Apply smoothing or sharpening filter to the CAPTCHA
5	Set pixel value range to black or white
6	Set a grayscale cutoff and substitute the grayscale values with black or white.
7	Remove granular noise
8	Modify border color scheme for CAPTCHA
9	Invert the CAPTCHA grayscale
10	Solve the CAPTCHA to test effectiveness of image preprocessing

The various image preprocessing stages are explained below.

Stage 1: color inversion (control group 1)

Pixel color values for CAPTCHAs are inverted using this control group. Pseudo-code below shows how this preprocessing filter works.

```
for(each pixel in CAPTCHA)
{
     if (invertRed is true)
           new red = 255 – current red
     if (invertBlue is true)
           new blue = 255 – current blue
     if (invertGreen is true)
           new green = 255 – current green
}
```

Inverting individual or multiple colors often provides a new perspective to look at the CAPTCHA being tested.

Stage 2: color modification (control group 2)

Color components for all image pixels can be modified using this control group. Each numeric box can take 257 (-1 to 255) possible values. For RGB color components of each pixel, the following actions are performed based on the value in the box:

- If the value is -1, no change is performed for that color component for any pixel.
- If the value is not -1, all occurrences for the particular color component (red, green, or blue) are set to the value specified in the relevant numeric boxes. A value of 0 effectively removes the color; a value of 255 sets it to the maximum value, and likewise.

Stage 3: greyscaling (control group 3)

All images are converted to grayscale by this control. *This is the only image transformation stage that is mandatory and cannot be skipped*. Based on the radio button selected, one of the following actions is performed on color components of each pixel:

1. Average → (Red + Green + Blue)/3
2. Human → (0.21 * Red + 0.71 * Green + 0.07 * Blue)
3. Average of minimum and maximum color components → (Minimum (Red + Green + Blue) + Maximum (Red + Green + Blue))/2
4. Minimum → Minimum (Red + Green + Blue)
5. Maximum → Maximum (Red + Green + Blue)

Depending on CAPTCHA color component values and distribution, any one of these filters may come in handy and help extract the best image for further processing.

Stage 4: smoothing and sharpening (control group 4)

To increase the complexity of text extraction, noise in the form of single or multipixel dots, extraneous lines and random protrusions/incursions are added on the CAPTCHAs. Image smoothing spreads the random noise and then the spread-out noise can be filtered out using the Bucket or Cutoff filters. The Passes numeric box allows user to choose the number of times a given image mask is to be applied before passing it on to the next stage. Let us now look at the components that make up the smoothing and sharpening filter.

Image masks

The smoothing and sharpening filter comes with six canned image processing masks. Two types of image masks are available:

- *Canned masks*—By default, TesserCap comes with the six most popular image masks. These masks can perform image smoothing or sharpening (Laplace[6] transforms) actions. Selecting a mask via various radio buttons brings it into immediate effect.
- *Custom image masks*—A user can also configure a custom image processing mask by directly entering values to the number boxes and clicking on the Save Mask button. If the sum of coefficients provided in these boxes is less than 0, an error is returned and the mask is not applied. The Save Mask button is not required when using canned masks from the radio boxes.

The three images below show removal of extraneous dots from an image using Smooth Filter 2 and Cutoff Filter.

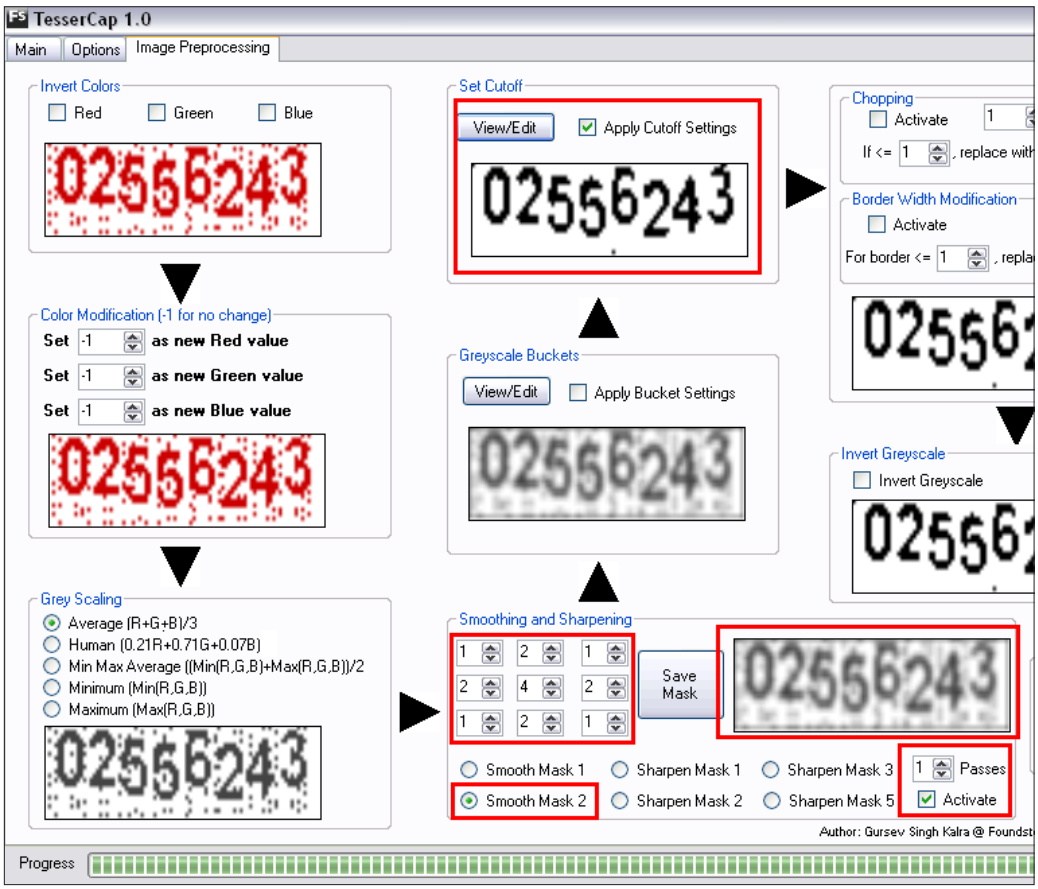

Figure 9. Image shows removal of dotted noise using image softening and cutoff filters.

Figure 10. Pixel value distribution before smoothing was applied. It was observed that the grayscale numeric value for noise was same as of the text.

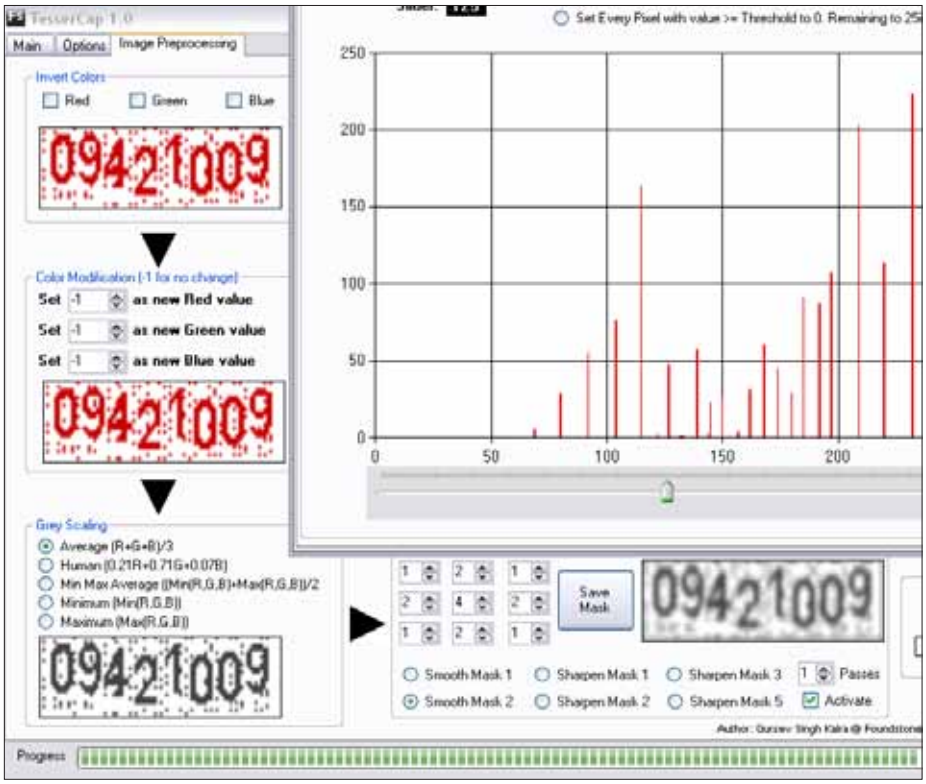

Figure 11. Pixel value distribution after smoothing filter is applied. The noise gets redistributed and makes it easy to filter out and to keep the main text with relatively little loss in clarity.

Stage 5: grayscale buckets (control group 5)

When a CAPTCHA reaches this stage, its pixels may have a wide range of grayscale values. This filter shows pixel grayscale value distribution in 20 buckets/ranges. The percentage of pixels with grayscale value between 0 and 12 is maintained in bucket 0, percentage of pixels with grayscale value between 13 and 25 is maintained in bucket 1 and henceforth. The user can select one of the following actions for each grayscale value range:

1. Leave unchanged (Leave As Is radio button)
2. Substitute with white (White radio button)
3. Substitute with black (Black radio button)

These options provide good control on various grayscale ranges and help reduce/remove noise by changing its grayscale value to white or black. The two images below show application of this filter and how the image is changed when the filter is applied.

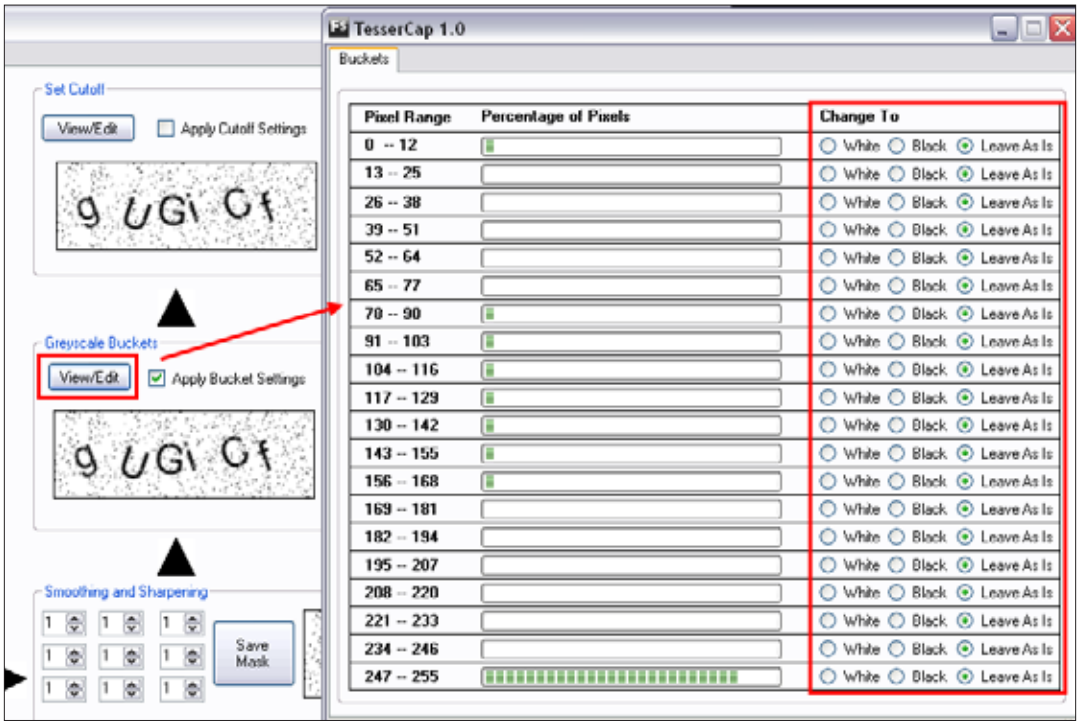

Figure 12. Image shows number of pixels in each pixel value range.

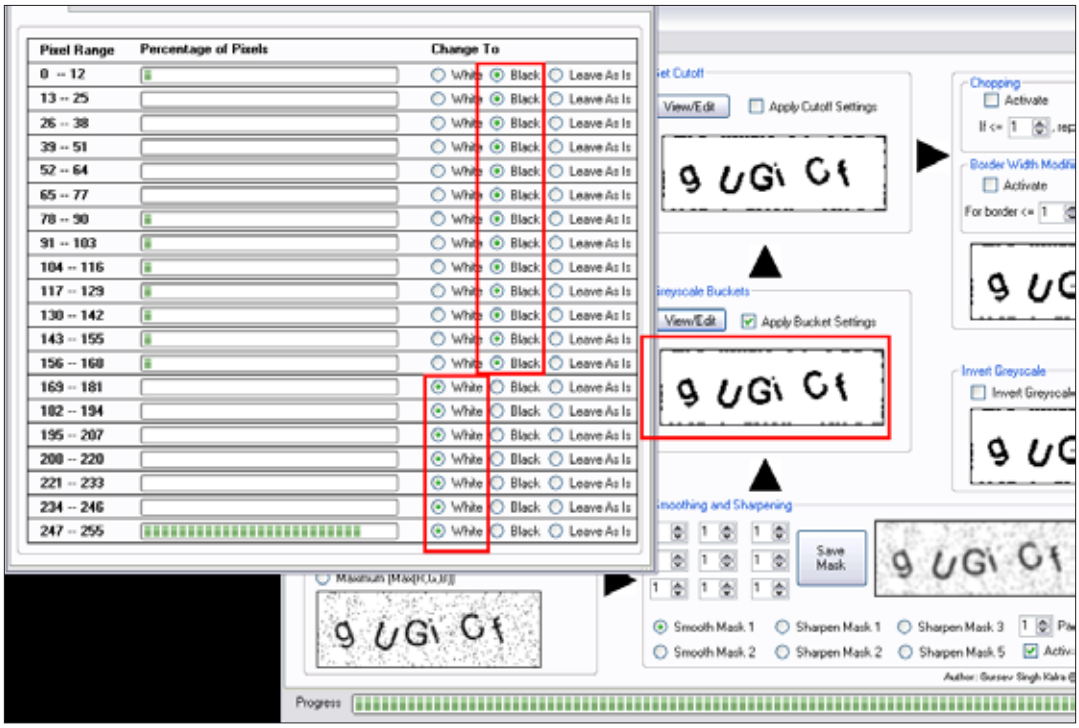

Figure 13. Image shows image noise change when various pixel value ranges were changed to black or white.

Stage 6: set cutoff (control group 6)

This filter plots a chart of "pixel grayscale value" against the "number of occurrences" and prompts the user to choose a cutoff. The working principle of the cutoff filter is explained with pseudo-code below:

```
if (pixel's grayscale value <= Cutoff)
      pixel grayscale value = (0 OR 255) ← as per the radio button selected
```

The chart allows the user to view the CAPTCHA pixel value distribution in great detail and helps remove noise by means of grayscale value cutoff. The image below shows the cutoff filter applied to a sample image and its output.

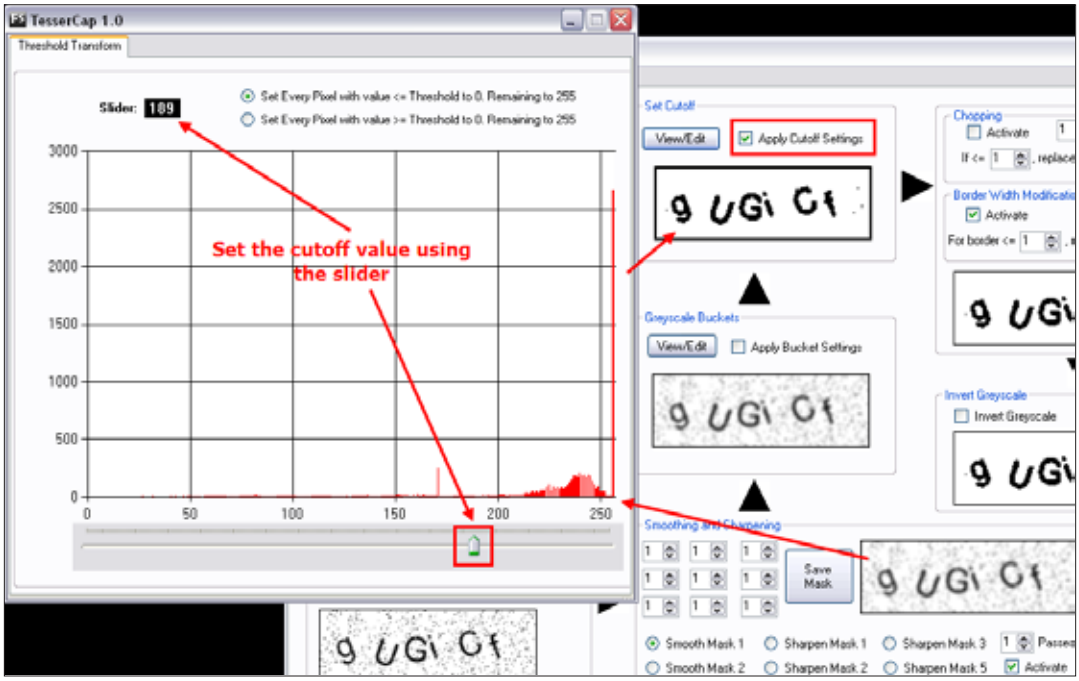

Figure 14. Image shows cutoff filter applied to a given CAPTCHA.

Stage 7: chopping (control group 7)

After applying the smoothing, cutoff, bucket, and other filters, CAPTCHA images may continue to have noise in the form of single or multipixel dots, extraneous lines, and random protrusions/incursions on the CAPTCHA text. Operation of chopping filter is explained below:

If the contiguous number of pixels for given grayscale values are less than the number provided in the numeric box, the chopping filter replaces these sequences with 0 (black) or 255 (white) as per user choice. The CAPTCHA is analyzed in both horizontal and vertical directions and corresponding changes are made.

The two images below show the Chopping filter in action.

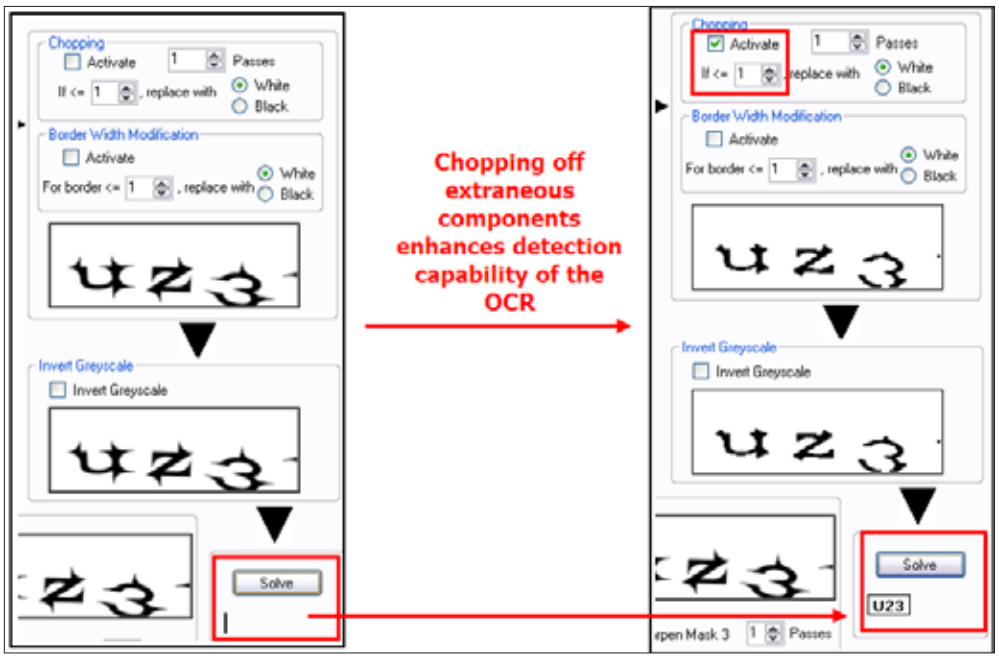

Figure 15. Image shows the extracted CAPTCHA value changing from blank to near correct.

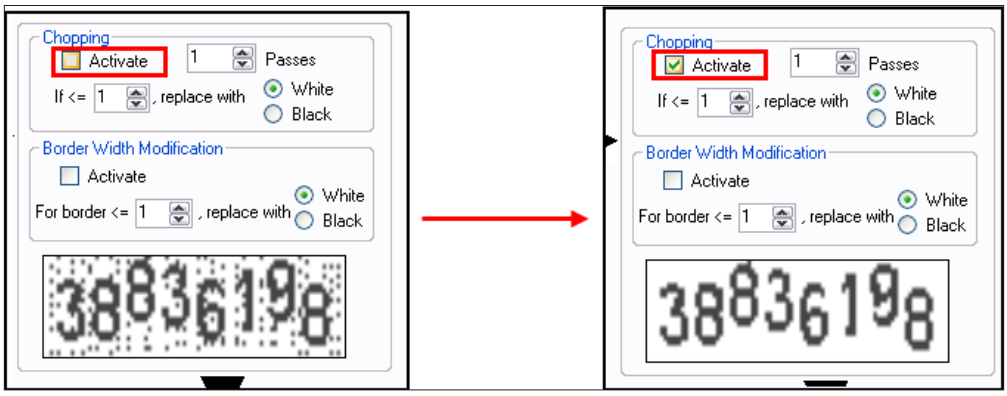

Figure 16. Image shows the CAPTCHA change after the Chopping filter is applied.

Stage 8: border width modification (control group 8)

During initial research and TesserCap development, it was repeatedly observed that when the CAPTCHAs have a thick border line and the color of the border line is different than the main CAPTCHA background, several OCR engines fail to recognize the text. This filter is designed to work on the borders and modify their appearance. The border with width provided in the numeric box is colored with black or white color, per user option.

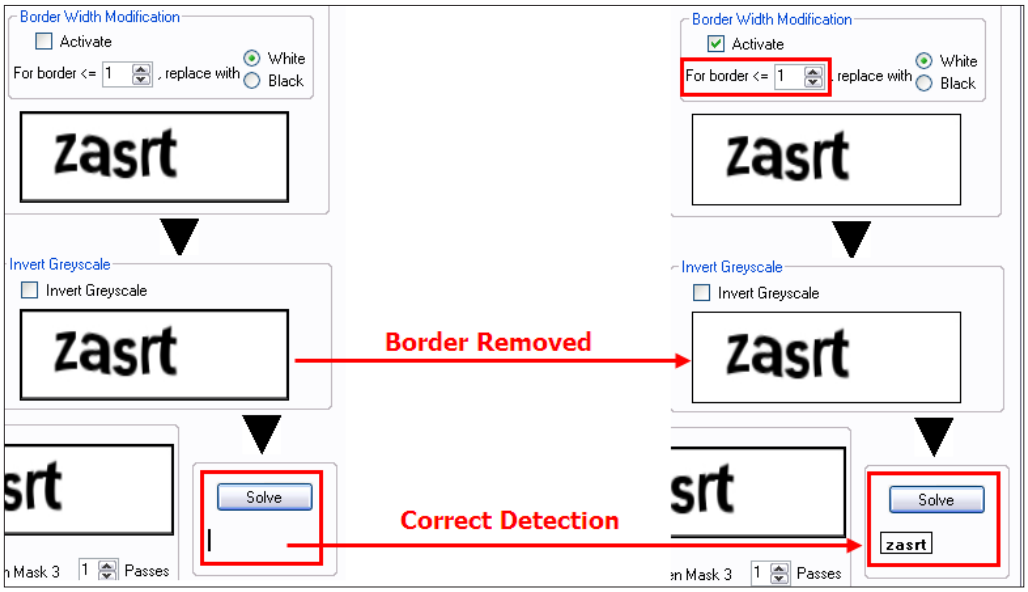

Figure 17. Image shows successful text retrieval when the border is removed.

Stage 9: grayscale inversion (control group 9)

The purpose of this filter is to go through each pixel and replace its value with a new value as shown in the pseudo-code below. The grayscale inversion accommodates OCR engine color preferences.

```
for(each pixel in CAPTCHA)
      new grayscale value = 255 – current grayscale value
```

Stage 10: verify CAPTCHA solution (control group 10)

This option is used to subject the preprocessed CAPTCHA to OCR recognition and provide instant feedback on its effectiveness. The Solve button picks the image from the output of the grayscale inversion filter (control group 9), sends it over to the OCR for text extraction, and displays the returned text on the GUI. Once a good preprocessing filter configuration is identified, the Enable Image Preprocessing checkbox in the Options tab can be checked to allow preprocessing on all CAPTCHAs downloaded from that point onwards.

Conclusion

CAPTCHAs have been one of the most potent mechanisms to protect web applications against automated form submissions. However, a weak CAPTCHA design may only protect against random robots and may not offer any protection against targeted attempts to bypass it. TesserCap aims to enable security professionals, developers, and testers to evaluate their CAPTCHA designs and make their implementations more secure. Like cryptographic algorithms, it is in an organization's best interest to rely on CAPTCHAs that have been thoroughly tested and are proven to be highly secure.

About the Author

Gursev Singh Kalra serves as a managing consultant at McAfee Foundstone Professional Services. Gursev is a speaker at security conferences like ToorCon, NullCon, and ClubHack and has authored an open source SSL cipher enumeration tool SSLSmart. Gursev has also developed several internal tools, web applications, and enjoys coding in Ruby, Ruby on Rails and C#.

About McAfee Foundstone Professional Services

McAfee Foundstone Professional Services offers expert services and education to help organizations continuously and measurably protect their most important assets from the most critical threats. Through a strategic approach to security, McAfee Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively. The company's professional services team consists of recognized security experts and authors with broad security experience with multinational corporations, the public sector, and the US military.

1. http://en.wikipedia.org/wiki/CAPTCHA
2. http://en.wikipedia.org/wiki/Noise_reduction
3. http://en.wikipedia.org/wiki/Optical_character_recognition
4. http://en.wikipedia.org/wiki/Tesseract_(software)
5. http://code.google.com/p/tesseract-ocr/
6. http://en.wikipedia.org/wiki/Discrete_Laplace_operator

McAfee
2821 Mission College Boulevard
Santa Clara, CA 95054
888 847 8766
www.mcafee.com

39301wp_file-carving_1111_fnl_ETMG