A Java System for Copy-Protected Display of Photographic Images on the Internet

Willis L. Boughton
Computer Information Systems Department
Business/Social Science Division
William Rainey Harper College
Palatine, IL 60067
847-925-6354. USA
wboughto@harpercollege.edu

ABSTRACT: *When high-resolution, copyrighted photographic images are displayed on the Internet, they must be protected from being copied by screen capture or by file. HTML and a Web browser alone do not provide protection, e .g., screen capture by a standalone program can not be prevented. This paper describes a Java system that provides copy protection for Internet display of photographic images organized into albums. Included is protection aga inst image file copy and against effective screen capture. The system involves three software components, two used by an administrator to build albums and a one used by client viewers. The system supports any number of albums, provides image annotation, zooming, and searching, requires no server-side software, maintains client privacy, and, using Java Web Start, does not require clients to install any software locally. The system is being used to maintain albums of noncommercial high-resolution historical and contemporary photos.*

Keywords: Java, Security, Image, Copyright

Received: 11 November 2011, Revised 26 January 2012, Accepted 30 January 2012

1. Introduction

Most photographic images displayed on the Internet require no copy protection, either because they are in the public domain or because they are low resolution and a copied image is of little value. High resolution, copyrighted images do require copy protection. An image potentially can be copied two ways: by screen capture and by coping the image file, e.g., by URL. Screen capture in some form is always possible since it requires only a screen capture program. Whether file copy is possible depends on the image display client/server software.

The fundamental issue with screen capture is that is can not be prevented by any client software technique. A standalone screen capture program can not be prevented from accessing video memory. To protect a high resolution image, the capture must be made ineffective using one or more of the following techniques:

a) Do not display the image in high resolution at all. Display it only in low resolution. This is probably the most common technique, but it prevents a viewer from evaluating the full quality of the image.

b) Display the image at high resolution but add disfigurement, e.g., text, geometrical pattern, or visible watermark, that practically can not be edited out. Too much disfigurement, though, may be unacceptable to viewers.

c) Display only part of the image at a time. This technique may not fully protect the image since multiple captured, partial images could potentially be combined to produce the full image.

The fundamental issue with file copy is how the client software accesses image files. There are two possibilities;

a) A custom client/server protocol, e.g., Java Remote Method Invocation (RMI) or a custom TCP/IP socket protocol, is used to transfer images from the server directly into client dynamic memory. This fully protects image files because they can be accessed only by the custom protocol and software, e.g., not by Web browser. This technique, though, requires custom software on the server as well as the client, which might not be possible. It might be that only standard Web services are available on the server.

b) URLs are used to access image files from a standard Web server. The URLs must be protected, i.e., not available to viewers, since if they were not, viewers could copy the image files directly from the server with a Web browser. HTML therefore can not be used to provide the URLs, since they would be directly in the HTML and viewable with a Web browser. Even if the "*save image*" feature of the Web browser were disabled, a viewer could get images from the browser's cache, or a viewer could use custom software, rather than a Web browser, to download image files.

Another issue is image zoom. It might be desirable to display high-resolution images in a much lower resolution screen area but give viewers the ability to zoom in, to evaluate image quality. Since screen capture can not be prevented, this zooming would enable a dedicated viewer to construct the high resolution photo from multiple zooms and screen captures, if the zooms were not disfigured. Since the full photo is already displayed in low screen resolution, it would not make sense to disfigure it, but it would make sense to disfigure the zooms.

2. Past Work and Current State

As early as 1998 [1], it was proposed that a Web browser plugin be used to deliver information securely over the Internet. Though the original focus was on documents and secure communication rather than protection of images, the browser plugin concept applies to images for both screen capture protection and image file protection. Various techniques for visual watermarking have been proposed [2, 3, 4]. These watermarks are designed to distort the image as little as possible while being difficult or at least tedious to remove. To minimize image distortion, the pixels in the watermarked areas must depend on the original image pixels. The watermark does not completely hide the original image in the watermarked areas. The point of view is that the viewer has access to only that one image, so the watermark must simultaneously protect the image while not distorting it too much. Whether the watermark can be removed successfully depends on the watermarking technique and the image itself, e.g., to what extent it has regions free of details [4]. Because the watermark pixels reflect the original image pixels, the watermark pixels might be used to recover, or at least attempt to recover, the original image.

The development [5, 6] of object removal algorithms and their incorporation in commercial photo editing programs has made effective visible watermarking more difficult. These algorithms can, depending on the image and region, remove even large objects in a single operation. The algorithms may make use only of pixels outside the region [5], so they might be effective against even complex image disfigurement. The algorithmically-constructed pixels in the region will not match those of the original (not disfigured) image, but this may not be apparent to someone not familiar with the original image. The algorithm work suggests that to make object removal apparent and protect the image, any disfigurement must, regardless of complexity, be spatially thick. This is demonstrated later in this paper.

3. Approach and System Requirements

The Java system described in this paper is an extension of the Web browser plugin concept combined with dynamic, opaque, zoom-dependent, size-selectable disfigurement of displayed images. This approach is distinctive as follows:

a) No custom plugin is required. The only plugin required is the Java plugin and Java Web Start, which are typically installed on computers. The viewer program, started by Java Web Start, provides the screen capture protection and file copy protection.

b) Image disfigurement is applied only when the image is maximized or zoomed. Images are displayed with no disfigurement at an initial resolution and disfigured only when the viewer maximizes the image or zooms in. A viewer is not immediately presented with a disfigured image, since this may repel some viewers. When an image is zoomed, the viewer can, by scrolling across the image, view all of it in detail even in the presence of the disfigurement.

c) The disfigurement completely hides the image in the disfigurement areas. The disfigurement does not depend on the image pixels. and the disfigured pixels can not be used to recover the original image, even if multiple images with the same disfigurement are available. The disfigurement also can be made as large as desired, to better protect against object removal filters.

d) All protection features are configurable, and images are viewable as albums with full search capability.

The system protects against screen capture and file copy and does not require custom server-side software, client software installation, or administrator programming, The specific system requirements were the following:

a) The only software required on the server must be a standard Web server.

b) The only installed software required on a client's desktop must be a Web browser and the Java runtime, including the Java browser plugin.

c) The client program must work with any Web browser and operating system, by using Java features only.

d) An administrator must be able to build and deploy photo albums to a server entirely by running programs, without writing any software, including scripts.

e) An administrator must have control, by option selection, of image screen capture protection, including initial image display resolution, whether images can be maximized, whether images can be zoomed, and the type of image disfigurement, if any. If disfigurement and zoom are both enabled, the disfigurement must be applied to zoomed images. If disfigurement and image maximize are both enabled, the disfigurement must be applied to maximized images.

f) A client user must not be able to copy a photo by file URL. Image URLs must be kept hidden and images must be saved only in dynamic memory.

g) An administrator must be able combine albums together into separate groups, e.g., put albums with historical photos into one group and albums with contemporary photos into another group. It also must be possible to put an album in multiple groups and to password protect a group.

h) An administrator must be able to specify the resolution of all image files in an album, and it must be possible to use different image resolutions for different albums.

i) A displayed photo must have annotation information, and a client user must be able to select photos by specifying annotation criteria. This search must span all albums in a viewing group.

j) A client user must be able to run a slide show of photos.

4. System Design

Figure 1 shows the system schematically. There are three system components:

a) Photo Manager program. An administrator runs this Java Swing GUI program to put photos into albums, define annotation for photos, and upload albums to a Web server by FTP.

b) Album Group Builder program. An administrator runs this non-GUI Java program to build an album group file and corresponding Java Network Launcher Protocol (JNLP) file. Each group file specifies albums viewable as a group; each JNLP file is the Java Web Start file for viewing that group. An administrator uploads these files to the Web server.

c) Viewer program. A client user runs this Swing GUI program to view an album group. The user specifies the album group JNLP file by URL in a Web browser, e.g., http://server.com/albums.jnlp. Java Web Start downloads the program specified in the JNLP file to the client's computer and starts program execution. From the Web server the program obtains the album group file specified in the JNLP file and the corresponding albums, and enables the user to select and view them.

4. Photo Manager Program

The Photo Manager program uses the Java Image Management Interface (JIMI) for image operations. All user functionality is available by GUI operations; no programming is required. The administrator puts photos into albums by specifying annotation such as location and topic; an album is a query to a custom database to retrieve photos that match specified annotation criteria. A photo can be in any number of albums. To build an album for Internet viewing, the user selects the parameters for the Internet

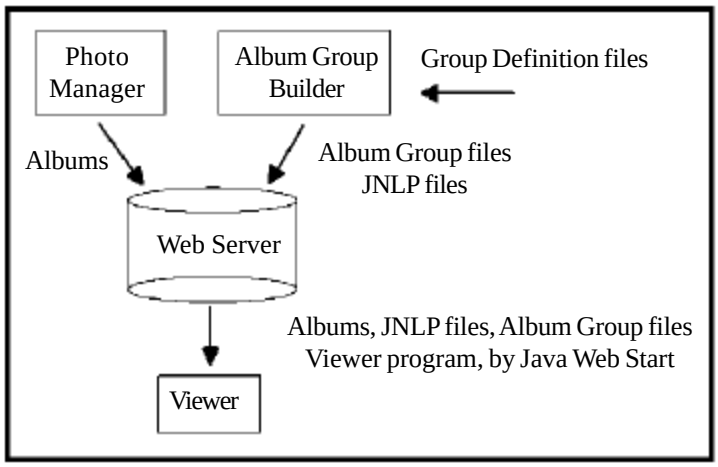

Figure 1. System Schematic

```
<?xml version= "1.0"?>
<!DOCTYPE albumGroup SYSTEM "albumgroup.dtd">
<albumGroup>
       <name> usaphotos </name>
       <dbPath> c:\apps\bpm </dbPath>
       <codebase> http://www.bluemontsw.com/ </codebase>
       <vendor> Willis L. Boughton </vendor>
       <viewSize> 0.75 </viewSize>
       <maxViewSize> 700 </maxViewSize>
       <canMaximize> true </canMaximize>
       <password> xyz123 </password>
       <copyrightMsg> Copyright % Willis L. Boughton </copyrightMsg>
       <copyProtectMark>
              <labelMark>
                     <text> Copyrighted </text>
                     <stroke> 3 </stroke>
                     <relFontSize> 0.1 </relFontSize>
                     <count> 2 </count>
                     <xPos> 0.0 </xPos>
                     <yPos> 0.0 </yPos>
              </labelMark>
       </copyProtectMark>
       <album>
              <name> America </name>
              <useMark> true </useMark>
              <zoomable> true </zoomable>
       </album>
       <album>
              <name> FSA-OWI Website </name>
              <useMark> false </useMark>
              <zoomable> true </zoomable>
       </album>
</albumGroup>
```

Figure 2. Sample Album Group Definition File

album and clicks a build button. Among the parameters are image resolution, thumbnail resolution, thumbnail annotation, and image ordering. The program uses JIMI to build all images in a folder on the administrator's computer. To upload an album, the administrator specifies the FTP parameters and clicks an upload button. For each album the program uploads the thumbnail image files for all photos, the image files for all photos, and the annotation information for all photos. The annotation information is in a custom binary file. For each album the program also uploads an empty index.html file into the specified Web server folder. This is done to prevent image file access by Web browser; see the URL security section of this paper.

6. Album Group Builder Program

Input to this program is an album group definition file specifying albums viewable as a group and information required for a matching JNLP file. The definition file is in XML format. Figure 2 shows a sample definition file, and Table 1 describes the fields. For each definition file the program writes an album group binary file and matching JNLP file. These files are uploaded to the server by FTP. Figure 3 shows the JNLP file that matches the Figure 2 definition file. The resources fields in the JNLP file specify the Java version, the memory required by the client viewer program, and the name of the client program file, bpmviewer.jar. The JNLP application-desc fields are the two input parameters to the client program. They are the root URL on the Web server for images and the name of the album group file. The full URL for the album group file is the root URL plus the album group file name. This URL is derivable from the JNLP file, which has consequences discussed in the URL security section of this paper.

```
<?xml version='1.0' encoding='UTF-8' ?>
<jnlp spec='1.0' codebase='http://www.bluemontsw.com/'
href='usaphotos.jnlp'>
<information>
   <title> Photo Album Viewer </title>
   <vendor> Willis L. Boughton </vendor>
</information>
<resources>
   <j2se version='1.5+' initial-heap-size='256M' max-heap-size='384M'/>
   <jar href='bpmviewer.jar' main='true'/>
</resources>
<application-desc main-class='viewer.Viewer'>
   <argument> http://www.bluemontsw.com/ </argument>
   <argument> usaphotos.bpm </argument>
</application-desc>
</jnlp>
```

Figure 3. Sample JNLP File

7. Viewer Program

The Viewer is a standalone program executed by Java Web Start. The user starts the program by entering a URL in a Web browser, but the program executes in its own window, separate from the Web browser. Figure 4 is a screen capture of the Viewer displaying a high-resolution photo with no disfigurement. The program executes with Java applet security; it has no access to the local filesystem or clipboard and can only read files from the Web server that supplied the viewer program. There are no privacy issues for the user. The program obtains all images from the Web server by HTTP. Image files are read directly into dynamic memory, and the user has no access to image files. When the user selects an album, the program obtains the thumbnails for all photos from the server and displays them in user-selectable tabbed pages. To display a photo the user double-clicks the thumbnail, which opens the full image in a dialog window. The dialog window has buttons for moving to the next or previous photo and starting a slide show at the photo. Depending on how the administrator defined the album group, the dialog also may have buttons for maximizing the image display size and resizing it, and the user may be able to zoom and scroll an image. The user always can select how the photos are ordered, can select a set of photos by tag matching from the currently-viewed album, and can do a search with "*" wildcards across all albums in the group. Figure 5 shows an example of a search, in which the user is selecting only photos taken in Illinois in the year range 1940 through 1960. The Viewer uses multiple threads to maximize responsiveness. The user can display a full image while thumbnails of other images are being loaded, and double-buffering is

used to retrieve each full image. When a photo is displayed, the program starts retrieving the next one, so it may be immediately displayed when the user clicks next. Annotation values which are common to all photos in an album and which more appropriately belong in the album's name are excluded from the displayed annotation for all photos in that album, to avoid repetition. For example, if all photos in an album were taken by the same person, that person's name does not appear in any image annotation for that album.

Field	Count	Description
name	1	The name for the Album Group file and JNLP file
dbPath	1	The folder containing the Photo Manager database
codebase	1	The JNLP codebase, which specifies the server folder containing the Viewer program
Vendor	1	The name displayed when Java Web Start starts the viewer.
viewSize	0 or 1	The initial maximum dimension of the image display window, as a fraction of the screen size, defaulted to 0.5. Images displayed larger than this (maximized) or zoomed in this window are disfigured if disfigurement is specified.
maxViewSize	0 or 1	The maximum dimension of the image display window, in pixels, defaulted to 700. If this value results in a size less than viewSize, it overrides viewSize.
canMaximize	0 or 1	If true, the viewer can maximize the image display window. If false, the maximum window size is set by viewSize and maxViewSize.
password	0 or 1	A password enabling a viewer to override the canMaximize setting..
copyrightMsg	0 or 1	A copyright message to be displayed at the bottom right corner of displayed images, defaulted to none
copyProtectMark	0 or 1	A disfigurement to be displayed , in gray color, on each image. The options are a labelMark, the default, or a geometricalMark.
labelMark	0 or 1	A text mark with a specified text, line stroke, repeat count, font size relative to the image display window, and relative position in the image display window. If either position is zero, the text is centered in the image display window. If the repeat count is 1, the text is drawn once in gray. If it is 2, the text is drawn twice, once in gray and once in black, slightly offset.
geometriclMark	0 or 1	A specified number of ellipses drawn on images, either outward from the center or inward from the edge, with a specified stroke.
album	1 or more	An album, consisting of name in the Photo Manager database, whether a copyProtectMark should be used, and whether the album images are zoomable.

Table 1. Album Group Definition Fields

8. Image Disfigurement

Figure 6 shows the three disfigurement options: text, outer elliptical marks, and inner elliptical marks. The font size, stroke size, position, and repeat count can be specified for the text, and the stroke size and number of lines can be specified for the elliptical marks. The magenta text in the bottom right corner is the optional copyright label. Figure 7 shows the effectiveness of a large and thick text disfigurement against the object removal filter of a commercial photo editing program. On the left is the original, disfigured image, and on the right is the image with the disfigurement removed with the filter. The removal was done in one operation by selecting the entire disfigurement. The filter effectively removes the disfigurement in the region with no features (sky) but severely distorts the regions with features (fence and building). Similar distortion occurs when only the disfigurement on the building is removed.

9. Image URL Security

A Viewer user can not obtain image file URLs from the program itself since they are not displayed and there is no HTML. This

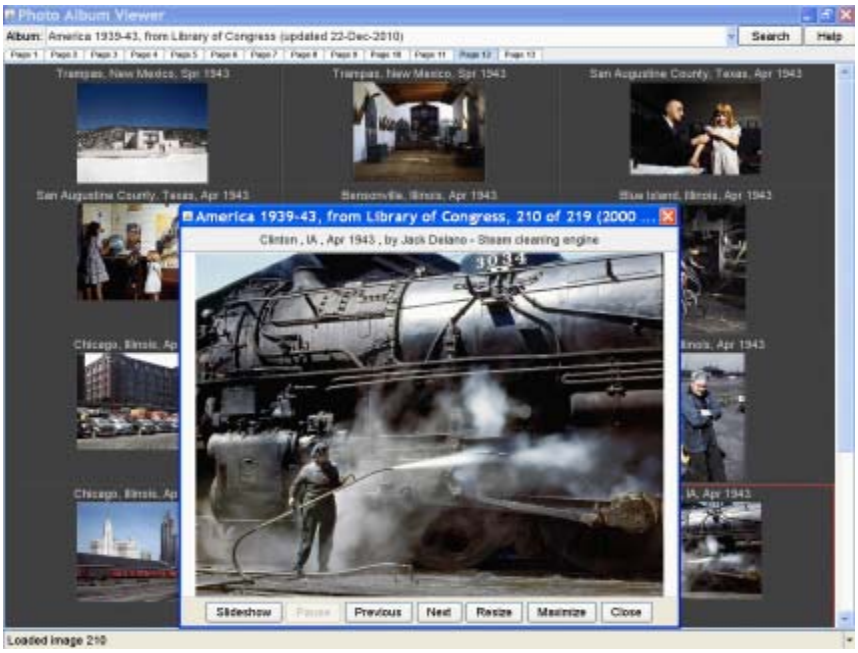

Figure 4. Viewer Program Screen Capture

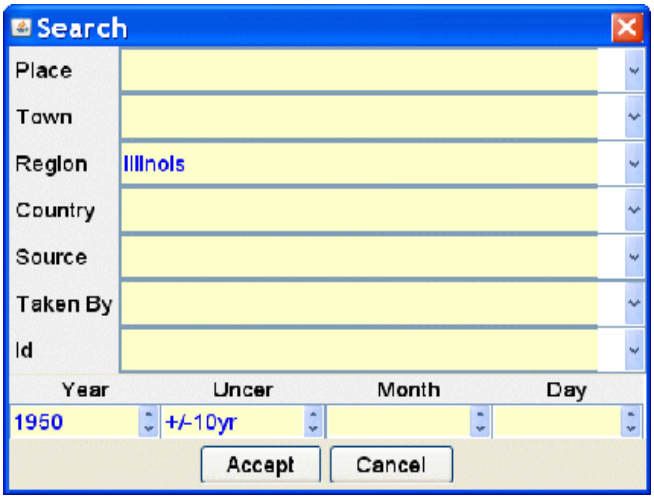

Figure 5. Sample Multi-Album Search

Figure 6. Disfigurement Options

does not, however, fully protect the URLs. The complicating issues are:

a) If the URL for a Web server folder is entered in a browser, it will display a file listing for that folder if it does not contain an index.html file. If a Viewer user could obtain the URL for a Web server folder that contains images but does not contain an index.html file, he could get the image files by a save directly from the browser's file listing. The Viewer user needs to know only the folder URL, not the image URLs.

b) The URL of an album group file is known to all clients from the JNLP file, as described previously. A Viewer user can download this file by entering the URL in a browser and doing a save. All contents of the album group file that identify image folders therefore must be unreadable. If they were not, a user could combine these folder names with the root URL on the server to obtain the image folder URLs.

c) A Viewer user could obtain the image URLs by using a proxy server that displays the HTTP requests.

Figure 7. Disfigurement Removal with Photo Editing Filter

The system addresses these issues as follows:

a) The administrator uses long, password-like names for the image folders on the server, to minimize the chance of a Viewer user obtaining a folder URL by guess alone. In addition, when the Photo Manager uploads an album it uploads an empty index.html file into the folder containing the images. Even if a Viewer user were able to obtain the image folder URL, the browser would display only an empty Web page, not the folder listing and image file names. The Photo Manager also uploads an empty index.html file into the server root folder, if the folder does not already contain an index.html file. This prevents a Viewer user from entering the root URL in the browser and seeing folder names from the browser file listing. This is a secondary protection, since even if a user obtains the folder names, the index.html files in the image folders prevent the browser from displaying image file names.

b) Server folder names in the album list file are encrypted.

c) The HTTP connections for reading images are opened with the Java "*no proxy*" parameter, which forces direct connections even if a proxy is being used.

10. Use

The system is being used to maintain a database of several thousand noncommercial photos, with a few hundred available on the Internet. They are arranged in multiple groups, each generally consisting of multiple albums. The image files have resolution of 2000 or more pixels in the larger dimension, higher than typically used on the Web. Albums containing images in the public domain are viewable full screen, at full resolution, zoomable, without disfigurement. Copyrighted images are viewable in a combination of limited resolution, no zoom, and disfigurement, depending on the album group. A group of two example albums, one of public domain photos and another of copyrighted photos, is at http://www.bluemontsw.com/usaphotos.jnlp.

References

[1] Jenkin, M., Dymond, P. (1998). A Plugin-based Privacy Scheme for World-wide Web File Distribution, *In:* Proceedings of the Thirty-First Hawaii International Conference on System Sciences. 7, 621-627.

[2] Braudaway, G, Magerlein, K, Mintzer, F. (1996). Protecting Publicly Available Images with a Visible Image Watermark, *In:* Proc. SPIE 2659, 126.
[3] Kankanhalli, M. S., Rajmohan, Ramakrishnan, K. R. (1999). Adaptive Visible Watermarking of Images, IEEE International Conference on Multimedia Computing and Systems, 1,568-573.
[4] Mohanty, S. P., Ramakrishnan, K. R., Kankanhalli, M. S. (2000). A DCT Domain Visible Watermarking Technique for Images, IEEE International Conference and Expp on Multimedia, 2, 1029-1032.
[5] Bertalmío, M., Sapiro, G., Caselles, V., Ballester, C. (2000). Image Inpainting, Proceedings of ACM SIGGRAPH, p. 417-424
[6] Avidan, S., Shamir, A. (2007). Seam Carving for Content-aware Image Resizing, *ACM Transactions on Graphics*, 26(3).

Author Biography

Willis L. Boughton has a Ph.D. in astronomy from the University of Illinois at Urbana-Champaign and twenty years of business experience in medical imaging, real-time embedded systems, and statistical process control. His last business position was Director of Process Measurements for Ameritech Corporate Information Services. Since 2000 he has been on the faculty of the Computer Information Systems Department at William Rainey Harper College in Palatine, IL. He is the author of a two-volume Java programming textbook. His last publication was in the Journal of Instruction Delivery Systems.