SENDING FILE LICENSE INFORMATION THROUGH HTTP HEADERS

Milan Tair[1]

[1]School of Informatics and Computing,
Singidunum University,
Belgrade, Serbia

Abstract:

This paper describes an implementation of a method of sending file license information for files downloaded from a server. There are many files available for download on the Internet and many of them were created by authors who have published them under a certain license. In most cases, the license information is lost or is unknown by the person downloading the file. Those files may be images, video content, text documents, audio recordings, executables, compressed archive files with software source code, educational materials of different kinds etc. Currently, the only way to specify the license information is to embed the license information in a file (overlay or stamp on an image, footer text in a document, a separate file in a compressed archive of files, etc.) or to show license information on the web page shown just prior to the step where the download link is available. This proposed method does not provide for a way to embed the license information into the original file, but instead, it allows for the license information to be sent with the file from the server to the user in the same HTTP response and vice-versa, from the server to the client. The license is stored using the extended file attributes mechanism.

Keywords:

extended attributes, HTTP headers, license, concept.

1. INTRODUCTION

Simply put, a license is a permission to use the property of another person [1]. There are many types of licenses that specifically relate to electronic documents, such as images, videos, audio recordings etc. The license type used for demonstration in this paper is a group called public copyright licenses [2]. Other terms, such as free license, open copyright license are used to refer to one kind of the public copyright license or another. Licenses are important, especially for authors of original intellectual work as well as original creative such as art, photography, music etc. Licenses, as methods of giving permission to use protected intellectual and creative works, are managed by a network of treaties and conventions [3] as well as national laws in every country. For example, all countries of the European Union are signatories of the Berne Convention for the Protection of Literary and Artistic Works and Trade-Related aspects of Intellectual Property Rights Treaty [4]. Treaties such as these recognise licenses as a method of giving usage rights for otherwise protected works [5].

Correspondence:
Milan Tair

e-mail:
milan.tair@gmail.com

DOI: 10.15308/Sinteza-2017-90-96

There are multiple ways for authors of original work (licensors) to grant licenses to licensees [6]. The way which is predominant on the Internet, whenever materials which are subject of the license, is to send a copy of the license agreement document issued by the licensor in the form of a separate file, usually a text document.

In this case, the text document containing the license text, which specifies the limitations and the conditions, is supposed to accompany the licensed work at all times, even when it becomes a part of a new whole. An example of this text is the MIT license, which states that "The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software." [7] In most cases, it is not particularly hard to include the original text of the license in the end product, as it does not require much storage space. Since there are no technical limitations for making the original license accessible to interested parties, we must look at other difficulties that prevent us from complying with conditions specified in the license. Regardless of reasons for not being able to comply, they ultimately consider it irrelevant.

However, there are situations when it is impractical or impossible to include or display the license in a transparent manner. An attempt to better explain these situations and license distribution problems is made in the next section which illustrates a typical use case on the Internet.

2. WEB BASED IMAGE GALLERY USE CASE

An example which illustrates the problem of license distribution is an implementation of an image gallery of paintings or photographs by different authors. Painters and authors of photographs presented in the gallery can license their works for use in this manner of presentation under different conditions. The website hosting the image gallery would have to accompany every image with the adequate text of the license or at least with a download link where the visitor can open and view the license text for the particular image. Aside from obvious impacts on the design of the image gallery, there is another issue that should be addressed in such situations. In case of public copyright licenses, authors of works in question might allow the public to freely use their work in their products as long as they publish the license along with the derived work, in case that the license allows for derivative work to be made from the original [8]. In case the image gallery does not provide a way to download both the image and the license at once in a single container, eventually the image will be posted on a different web page and its license will be lost, as it is not immediately associated with it. For the majority of public copyright licensed work, this might not be seen as a potentially problematic situation, but for commercial users it might. This is because some licenses allow free use of licensed works for personal and non-commercial use, but disallow or require payment or purchase of works when they are going to be used commercially [9].

These kinds of licenses that allow free use to one group and limit it to another are generally not considered public copyright licenses, in case of most electronically distributable materials or open source licenses [10], in case of software.

Because there are versions of public copyright licenses that allow and those that disallow derived works to be made from the original work which the license applies to, when an individual wants to use the work with modifications or in combination with others, which is treated as derived work [8], he or she should familiarise with the type of license under which the work is published. If the file was downloaded and re-uploaded to different sources multiple times, it can be hard to find its original source and retrieve a copy of the license. Also, it may sometimes be hard to even locate the author or an original work in order to request a license directly.

3. IMPLEMENTATION

Although there are different methods of achieving described license distribution, this paper presents a method that does not require modifications to the application layer communication protocol used for data transfer between the server and the client and vice-versa. Instead, this method uses an already described mechanism available in HTTP (The Hypertext Transfer Protocol). Since HTTP is an application layer protocol used to access content from different locations on the web it is also capable of transmitting certain meta-data about the content being delivered [11] [12].

HTTP Request and Response Headers

This meta-data is usually transferred within a HTTP response header [13]. HTTP headers are part of responses that are always delivered first. Headers contain multiple lines of text that describe the content that follows in the response body. Among these lines of text is the information such as the total size of the incoming response body as well as its type (image, text, application, spreadsheet document etc) as well as some supplementary information

used by the browser for content caching etc. [13] [14] The header mechanism allows for defining arbitrary header lines [13]. Taking advantage of this feature is a key part of the license distribution method from the server to the browser. Just like the server can use HTTP response headers to embed the license information, the browser can use an analogous mechanism available in HTTP Requests. HTTP Requests are sent by the browser to the server. They carry information such as the web page that the browser wants to open or a file that the browser wants to download from the server at the specified path. This information is stored within the header section of the request, which is always delivered first, just like the response. The header consists of lines of text that describe the request as well as the browser used to make the request. For example, these header lines store information about the browser vendor, name, version, operating system type, version, build, available support for different scripts, languages, encodings etc. [15] Whether a request has a body or not depends on the HTTP method used. If data is being sent using the POST method, the request contains the data within its body section. When sending files, they are sent as base64 encoded text of their content along with leading lines of text specifying their original name and extension. The protocol cannot be modified to include license information in this section. This is why the method presented in this paper utilises the possibility of adding user defined lines to both the request and response headers in order to send additional information about file licenses from the server to the client and vice-versa.

Extended File Attributes

When uploading a file from a browser to the server via HTTP, its content is sent, but it does not include license information. Presumably, there is no way to embed the license information into the file. This is true for most file formats and the simplest example that proves that there is at least one type of file that cannot have any additional information embedded is a plain text document. All data within it is principal and would not be ignored or skipped by a parser or viewer. Based on this presumption, license information must be stored elsewhere. Additionally, when a file is being uploaded by the browser, it should be able to extract this additional license information from its storage space and include it into the request header as explained in the previous subsection.

For this method, the storage space for information about the license for a file is implemented using extended file attributes. The ability to store extended file attributes is a mechanism available in most major file systems used today, such as EXT, NTFS etc. Although implementations vary, system level application programming interfaces provide support for handling extended file attributes [16] [17].

File handling on the server side

For the demonstration of this method, the server and the web page used to upload the file are configured in such a way that the server side application that handles file reception and sending is written in such a way to read and write license information about the file into extended file attributes.

The server is configured in such a way that the server side application handles its dispatch to the browser when a file is requested for download. Instead of merely sending the content of the file as the server, the application reads the extended file attributes of the requested file to find license information. As the implementation is done using PHP, appropriate PHP functions are used [18]. If license information is found for the requested file, it sends an additional response header line with the appropriate content, specifying license information.

When receiving a file from the browser, the application also reads the license information from the request header. If license information is found, upon storing the file on the file system, it additionally stores license information in an apt extended file attribute. The next time when this file is requested for download, the stored license information is delivered in the response header along with its content. Just like with reading the extended file attributes, the application uses appropriate PHP functions to write the attribute [19].

This way, the preservation of the license information is achieved on the side of the server.

Delegating request handling to the server-side application

In order to have the server-side application handle all requests as explained earlier, the server is configured to pass all requests to it for processing. The web server is an instance of the Apache2 Web Server application. There is an apache server module called the rewrite module which allows for request rewriting. Using the Apache2 server's per-directory configuration overwriting mechanism, all requests are routed to a single server-side PHP application that will handle them. Request routing is done by

specifying a request rewriting rule in the file named .htaccess [20] in the root directory of the web application. The sample content of the .htaccess file used for the testing of this concept is listed below.

```
<ifModule mod_rewrite.c>
        RewriteEngine On
        RewriteRule ^(.*)$ index.php [L]
</ifModule>
```

Listing 1. The content of the .htaccess file used for the demonstration.

As shown in Listing 1, all requested paths are dispatched to index.php for further processing. In the PHP application, the original requested path can be found in the server information associative array [21] at index REQUEST_URI. The HTTP request method can be found in the server information associative array at index REQUEST_METHOD. Custom headers sent with the request can be found at indices starting with HTTP_ [21, p. #89567]. A part of the PHP application's code that gathers information about the original request path and method is shown in the listing below.

```
$url = filter_input(INPUT_SERVER, 'REQUEST_URI');
$mtd = filter_input(INPUT_SERVER, 'REQUEST_METHOD');
```

Listing 2. Part of the application that acquires request information.

Note that the server information index is not sanitised with an appropriate filter in Listing 2. Ideally, the filter_input function should be called with a proper filter for the third argument, like the FILTER_SANITIZE_STRING filter [22].

If the received request's method is POST, the application checks if there are files in the request body. If so, these files are uploaded to a private directory on the server and the application checks for license information in the request header, matching the HTTP field names under which these files were uploaded. Data about uploaded files is located in the $_FILES global array in a standardised format [23].

The proper name of the index in the server information associative array, sent via HTTP headers, containing uploaded file's license information is formed by prefixing HTTP_ to the HTTP POST data field name of the particular file. If each file that is being uploaded is assigned a unique field name, their field names become keys of the $_FILES array in PHP.

The code shown below illustrates a way to retrieve license information from the request header for the second uploaded file whose field name is not known by the application.

```
$index = 1; # The second file (counting from 0)
$fileFieldNames = array_keys($_FILES);
$fieldName = $fileFieldNames[$index];
$licenseLine = 'HTTP_License-' . $fieldName;
$license = filter_input(INPUT_SERVER, $licenseLine);
```

Listing 3. Retrieving license information about the second uploaded file.

If there is no license information for the particular file, the variable remains empty. This indicates that the browser did not include license information for the particular file.

After completing the upload process for a particular file to its final destination on the server's file system, the application will execute a function that will write the retrieved license information into extended file attributes space for the particular file. For this demonstration, a simple user space key name is used. The key name is user.license. The following code shows how the application stores license information about a recently uploaded file. The uploaded filename and license information is stored in appropriate variables called $destination and $license, for the purpose of this example.

```
xattr_set($destination, 'user.license', $license);
```

Listing 4. Storing license information in extended file attributes storage.

Note that for the above shown code to work the xattr Pear package must be installed on the server and the Apache2 system user must be granted Access Control Lists permissions to the directory where files are being uploaded [24].

Finally, when the file retrieval scenario is being executed, the application would read the license information from the requested files extended file attributes. If this information available, it adds a license information header line into the HTTP response header, thus sending the license information to the browser along with the content of the file. A sample code demonstrating this

process is shown in the following listing. The path to the file being sent is stored in the variable $path.

```
$license = xattr_get($path, 'user.license');
if ($license !== false) {
  header("File-License: " . $license);
}
readfile($path);
exit;
```

Listing 5. Adding license information for the file being sent to the header.

The code shown in Listing 5 follows all other codes written in preparation for sending the requested file, such as content type, size, download file name, flag to force download and other header information specification. The output buffer must remain empty at the time this code gets executed to prevent contamination of the file content sent in the response body. Preferably, the output buffer should be cleaned before executing this code. This can be done using the ob_clean function in PHP [25].

File handling on the client side

The client-side is more difficult to implement as it requires the Internet browser to be aware of the additional response header lines when receiving a file and should be aware of the need to send additional request header lines when uploading a file or a number of files. Also, the browser should store extended file attributes along with the file when saving it as well as reading the attribute when uploading it to the server. Implementation of this functionality is currently not possible without completely rewriting the source code of the web browser and having it natively support the license information retention via extended file attributes.

However, there are two ways to test the method in order to prove it as a valid concept. The first way is to use an unsecure Internet browser with all security features reduced to lowest settings in order to allow execution of commands on the client side. These commands would be used to write extended file attributes for the downloaded file. The second way is to write an Internet browser simulator as a client side application that would simulate file upload and download. This application would have the ability and permissions to execute system commands to write extended file attributes. Both methods are unlikely in the production scenario, especially the first which requires lowering of security settings and using ActiveX WScript.Shell objects to execute shell commands on the client. This functionality is no longer supported by any major Internet browser other than Microsoft's Internet Explorer. Even in the most current version of this browser, this functionality is considered deprecated. [20] As it is much simpler to write an application to simulate a browser, this way is used to prove the concept of the licence information preservation method presented in this paper.

An Internet Browser Simulator

The Internet Browser Simulation application used to demonstrate the proof of concept of the licence preservation method provides two usage scenarios. These scenarios are file retrieval and file upload. In the first scenario the application sends out an HTTP GET method request for a file at a specified path. If the server includes license information in the response header, the application stores the license information into extended file attributes for the downloaded file. In the second scenario the application sends out a multipart/form-data HTTP POST method request with the file included in the request and the license information for the file sent in the request header.

When the file retrieval scenario is executed, the HTTP response is received and processed. The header is parsed and a line starting with File-License is located, if present. File license information is extracted from the header and is kept in memory until the file download is complete. When the file is downloaded, it is stored at an appropriate location on the file system. After this, the program executes an adequate, platform dependant, command that stores license information by setting extended file attributes or file properties for the downloaded file. Implementation of this process varies depending on the operating system and the used file system. Some file systems do not support extended file attributes or similar mechanisms of storing additional meta-data about files [27]. In these cases, alternative methods for storing file license information are possible, but their implementation would go outside of the scope of this paper and the method presented herein.

When the file upload scenario is executed, the program reads file license information from extended file attributes and creates a header line formed as a key-value pair separated by a colon. The key part is formed by adding the HTTP POST data field name for the file being uploaded to License-. The value of the pair is the license identifier. As mentioned earlier in the paper, open copyright licenses are supported and used for this imple-

mentation. An example of the license information header line is shown in the following listing.

```
License-picture_1: CC 4.0 BY-ND
```

Listing 6. An example of a file license information request header line.

Listing 6 specifies that the file uploaded under the field name picture_1 is published under a Creative Commons, Attribution without Derivatives license, version 4.0 for use in internationally available content. This means that anyone who uses this file can share it freely, but they cannot modify it in any way. Every type of license can be abbreviated to a short string indicating the type, version and any generalised restrictions that the license imposes on the user.

Currently, this method is limited to a predefined set of licenses that can be identified by parsing the license string. This way, specialised software can show the full text of the license, which is available on-line. A possible solution to this limitation is the introduction of a specialised license type for own licenses, where the user would specify a URI with license details. The presented method does not allow this kind of content to be set as a license information value of the file's extended file attribute because of security concerns.

4. CONCLUSION

In this paper, a method of storing and retrieving file license information is presented. Also, it explains a working concept of a mechanism for sending the license information over the network via the HTTP application layer protocol by including it in HTTP request and response headers. The proof of concept implementation utilised extended file attributes mechanism of modern file systems for storing license information for files on the server and the client side. This mechanism is supported by major platforms and file systems, but in some configurations it is not enabled by default. It is explained that the major issue in making this method common practice is the inability to have modern browsers and operating systems supporting it and making it achievable without the use of additional software and special platform configurations on both client and server sides. The browser simulation application is programmed to add license information in additional header lines and to retrieve them from response headers. However, browsers cannot be easily modified to perform this without rewriting their source code.

At this point, it is unlikely that this method can be implemented, even through plug-ins or extensions installed on the system or the browser, but if the method were to be considered for adoption into a standard supported by major browser vendors, its practical use would be possible.

It is the author's opinion that there is potential for further development of this technology, as well as obvious use for it in securing rights of creators of original works who wish to impose certain limitations to the way their work may be used. It is in no way the author's wish to exclusively support copyright and the expansion of restrictions inflicted by copyright laws and regulations. Instead, the aim of this work is to encourage the use of public or open copyright licenses and to provide a way to preserve information about the license under which original work contained within a file was published by its author.

REFERENCES

[1] S. French, *License*, Encyclopædia Britannica, inc., 2003.

[2] Wikipedia contributors, "Public copyright license," Wikipedia, The Free Encyclopedia, 19 03 2017. [Online]. Available: https://en.wikipedia.org/wiki/Public_copyright_license. [Accessed 24 03 2017].

[3] M. S. Denniston, "International Copyright Protection: How Does It Work?," Bradley Arant Boult Cummings LLP, 03 04 2012. [Online]. Available: http://www.mondaq.com/unitedstates/x/171306/Copyright/International+Copyright+Protection+How+Does+It+Work. [Accessed 11 02 2017].

[4] K. Rainer, *Copyright Issues in the European Union - Towards a science - and education-friendly copyright*, 2013.

[5] *Appendix 1 to the Berne Convention for the Protection of Literary and Artistic Works,* Paris, 1971.

[6] H. Ward Classen, A Practical Guide to Software Licensing for Licensees and Licensors: Analyses and Model Forms, Chicago: American Bar Association, 2005.

[7] Massachusetts Institute of Technology, *The MIT License*, https://opensource.org/licenses/MIT.

[8] V. Lindberg, Intellectual Property and Open Source: A Practical Guide to Protecting Code, Sebastopol: O'Reilly Media, Inc., 2008.

[9] Creative Commons, "NonCommercial interpretation," 25 09 2014. [Online]. Available: https://wiki.creativecommons.org/wiki/NonCommercial_interpretation. [Accessed 02 03 2017].

[10] C. DiBona, M. Stone and D. Cooper, Open Sources 2.0: The Continuing Evolution, Sebastopol: O'Reilly Media, Inc., 2005.

[11] R. Steinmetz and K. Nahrstedt, Multimedia Applications, Springer Science & Business Media, 2004.

[12] M. P. Clark, Data Networks, IP and the Internet: Protocols, Design and Operation, John Wiley & Sons, 2003.

[13] D. Gourley and B. Totty, HTTP: The Definitive Guide, O'Reilly Media, Inc., 2002.

[14] G. Held, A Practical Guide to Content Delivery Networks, Second Edition, CRC Press, 2010.

[15] F. Scholz, T. and M. D. Network, "Request header," Mozilla Developer Network, 19 06 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Request_header. [Accessed 11 02 2017].

[16] Noite.pl, File system extended attributes and kernel capabilities: Linux Basic. AL1-043, NOITE S.C.

[17] M. Minasi, D. Gibson, A. Finn and W. Henr, Mastering Microsoft Windows Server 2008 R2, John Wiley & Sons, 2010, p. 483.

[18] The PHP Group, *xattr_get — Get an extended attribute*, The PHP Group.

[19] The PHP Group, *xattr_set — Set an extended attribute*, The PHP Group.

[20] The Apache Software Foundation, "Apache HTTP Server Tutorial: .htaccess files," 2017. [Online]. Available: https://httpd.apache.org/docs/2.4/howto/htaccess.html. [Accessed 30 01 2017].

[21] The PHP Group, "Predefined Variables - $_SERVER," The PHP Group, [Online]. Available: http://php.net/manual/en/reserved.variables.server.php. [Accessed 04 02 2017].

[22] The PHP Group, "Filter Functions - filter_input," The PHP Group, [Online]. Available: http://php.net/manual/en/function.filter-input.php. [Accessed 04 02 2017].

[23] The PHP Group, "Predefined Variables - $_FILES," The PHP Group, [Online]. Available: http://php.net/manual/en/reserved.variables.files.php. [Accessed 04 02 2017].

[24] "Access Control Lists," Arch Linux Documentation, 05 11 2016. [Online]. Available: https://wiki.archlinux.org/index.php/Access_Control_Lists. [Accessed 06 02 2017].

[25] P. Hudson, PHP in a Nutshell: A Desktop Quick Reference, O'Reilly Media, Inc., 2005, p. 182.

[26] Microsoft, "VBScript is no longer supported in IE11 edge mode," [Online]. Available: https://msdn.microsoft.com/en-us/library/dn384057%28v=vs.85%29.aspx?f=255&MSPPError=-2147217396. [Accessed 29 01 2017].

[27] freedesktop.org, "Guidelines for extended attributes," 18 05 2013. [Online]. Available: https://www.freedesktop.org/wiki/CommonExtendedAttributes/. [Accessed 06 02 2017].