WebGuard: A System for Web Content Protection

M. Mourad[†], J. Munson[†], T. Nadeem[‡], G. Pacifici[†], M. Pistoia[†], A. Youssef[†]
[†]IBM T.J. Watson Research Center [‡]University of Maryland at College Park
{magdam, jpmunson, giovanni, pistoia, ayoussef}@us.ibm.com

ABSTRACT
In this paper, we present WebGuard, a content protection system for Web documents. WebGuard allows content owners to exercise control over usage conditions governing access to their content. We first introduce the concept of transparent digital rights management, and then show how WebGuard realizes it, using existing Web browsers.

1. INTRODUCTION
With the advent of digital distribution and the many mechanisms available for it on the Internet, it is now possible for a single person to make a perfect copy of a digital content and distribute it to millions of others. While in most cases content owners welcome this widespread distribution, in some cases the content owners may wish to enforce some control over the distribution and access to *high-value* digital content. High-value digital content can be commercially valuable content (e.g. course material, artwork) or personal and confidential property (e.g. medical records).

Digital Rights Management (DRM) technology allows content owners—such as publishers, artists and instructors— to distribute high-value digital content with the confidence that the terms and conditions they set for the use of their content will be respected. For example, the content owners may wish to specify terms and conditions that prevent the digital content from being copied or disable the content after it has been accessed a certain number of times.

These benefits of DRM technology have been available for some time in the specific industries of music and e-books, and there are several systems currently available on the market [1][2][3][4][6]. However, for general multimedia Web content—HTML, GIF and JPEG images, animations, audio—DRM technology has been less successful.

The reason for this, to a large measure, lies in the current state of the art of the DRM technology and the specific requirements it imposes on the end-user. Typically a DRM system works by encrypting the content and providing a program capable of playing (or displaying) the content to the user. This player ensures that the user does not make unauthorized use of the digital content. This, however, restricts the number of players available and requires users to obtain a separate player for each protection system. This approach is particularly problematic in the case of Web content. On the one hand, users are not willing to use a different browser to access protected content, especially if the user has to use a different web browser to access content protected by different DRM systems. On the other hand, the DRM developers would be burdened by the need of providing and maintaining a fully featured web browser, making the DRM solution cost effective only for large scale distributions and commercially viable content.

In this paper we present *WebGuard*, a content protection system that allows users to access protected content using existing web browsers and plug-ins. Our solution centers around three major components: an application certification process, a DRM-enabled http protocol handler, and an application-independent user-interface control module. We show that using these three components a complete end-to-end content protection system can be achieved. We have built a prototype of WebGuard using the Internet Explorer Browser and the Microsoft protocol handler technology.

2. WEBGUARD COMPONENTS
By "transparent DRM," we mean that DRM functions are provided to an application without requiring it to be specially DRM-enabled. Our approach has three main elements: (1) browser code verification, (2) a trusted content handler, and (3) UI control through event blocking. These are shown in Figure 1.

The trusted content handler is a protocol subsystem responsible for retrieving and decrypting the content according to the terms and conditions set by the content owner. The browser code verification mechanism ensures that the decrypted content is delivered to a known (and trusted) application, e.g. a known version of the Microsoft Internet Explorer browser. The UI control subsystem prevents users from performing actions that are not allowed by the usage terms (e.g., print, save as, etc.)

2.1. Browser Code Verification
Our approach uses a three-step process of establishing the trust of an application at call-time: (1) the application goes through an off-line certification process that generates a trust certificate containing the digital signature of the application, (2) this signature is checked against the signature of the executable image of the application at run time, and (3) this validation is remembered and checked by the Trusted Content Handler (TCH) upon each request the application makes for content.

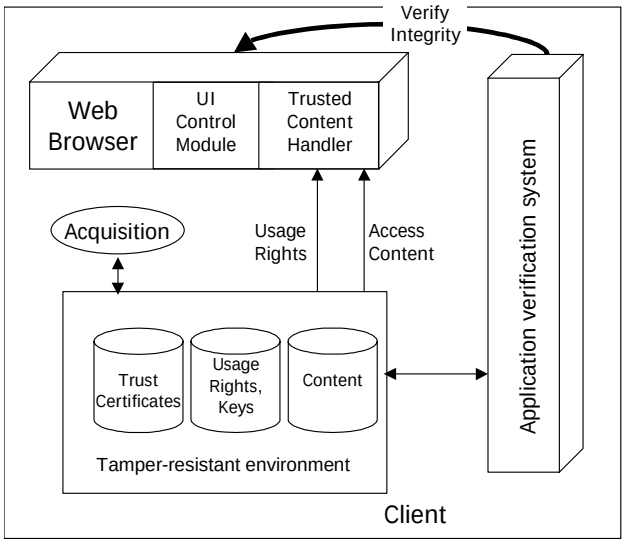

Figure 1. The WebGuard content protection system.

We have developed two variations of the run time verification process: one that verifies the application at launch time, and another that verifies the application at content-request time. We describe only the former below; please see [5] for a description of the latter.

The Verifying Launcher (VL) is responsible for verifying at launch time that the viewer application is certified as a trusted application for safely handling protected content entrusted to it. As mentioned above, each trusted viewer must undergo an offline certification process, which results in a trust certificate that includes the signed digest(s) of the application code modules. Before launching the viewer, VL verifies the integrity of the code. If the code installed on the client host is identical to the one certified, it is safe to handle the content. VL then instructs the operating system to load the application from the verified code files. By virtue of its role as the application launcher, VL obtains OS-specific information, such as the process ID or the process creation date, which uniquely identifies the loaded application instance within the system. VL uses this information to compute a stamp that still uniquely identifies the application instance but is hard to guess or forge. The stamp is computed using a hashing function, which is known to TCH as well. The algorithm used by the hashing function must be deterministic (always generating the same result given the same input, for reasons described later. One such algorithm may be a common encryption algorithm using a predetermined key.

When an application makes a call on the Trusted Content Handler (TCH) to access a protected resource, the TCH first verifies that the application was launched and verified by the VL. It does this by computing the stamp for the application using the same uniquely identifying information and scrambling information that the VL did, and then contacting the VL for comparison. If the TCH-computed stamp and the VL-computed stamp are the same, then the TCH was called by the same application instance that the VL verified and launched. The TCH may then cache its stamp so that no further communication with the VL is necessary, for this session with the application. The TCH and the VL communicate through a secure connection.

2.2. Trusted Content Handler

WebGuard use Internet Explorer's protocol handler extension mechanism to implement a trusted content handler. URLs for WebGuard-protected content use the protocol names `rmfile` (for local content) and `rmhttp` (for remote content). When the browser receives URLs with these method names, the WebGuard Trusted Content Handler is invoked. The TCH invokes the necessary DRM functions, decrypts the content, and passes it back to the browser.

The interface that the DRMC offers to the IE protocol handler is similar to a file system interface in following an "open-read-close" pattern. The content identifier received from the protocol handler is a URL with a hierarchical path name identifying the content item. From this path name we identify the name of the *package* the item belongs to. Given the package name, we can locate the set of rights associated with the package.

The WebGuard rights specification mechanism allows content publishers to specify rights at an arbitrarily fine level of granularity. This is important for protected packages that contain large numbers of individual content items—e.g., courseware, photography collections, literature anthologies, to which publishers may wish to allow different levels of access. For example, they may wish to offer the first section of a course at no charge, but require users to purchase access to the remaining sections. The rights specification mechanism we developed for WebGuard allows fine-grained specification, but is space-efficient and has a rights-lookup time of O(lg *n*).

2.3. User interface control module

An application handling protected content must prevent the user from invoking unauthorized operations on the content. To provide this function for an application in a transparent manner proved to be one of the more challenging tasks in the project. We developed a mechanism that allows us to control the user-interface operations of Microsoft Windows™-based applications. We rely on the event-handler structure of Windows programs and the ability of other program modules to register themselves as listeners of the events and to receive them before the application does. In this way, if the received event represents an operation that should be blocked, according to the rights in effect for the subject content, the application can be prevented from receiving the event.

WebGuard's User Interface Control Module (UICM) operates on an application-specific event-to-action map, which tells it which application events represent which controllable actions (such as Print, Save, and Copy). This map is generated through a manual process of inspecting the execution of an application using a tool such as Microsoft Spy++ to determine which events are generated by certain user actions, and which windows they occur in. The information yielded by the inspection is used to create the event-to-action map.

Because users may acquire different sets of rights to content, the policy of the UICM is dynamic. Upon each request for content, the DRMC sets the control policy of the UICM. If, for example, the rights to a particular web page included "View:Yes, Print:No," the DRMC would return the content to the protocol handler but would set the UICM's control policy such that if the user chose the "Print…" command from the application's menu, a dialog box would pop up and inform the user that they had not acquire the "Print" right.

Using the same method of window subclassing that we have described here, an intruder could attempt to disable our security model by re-subclassing the windows which we had already subclassed. This way, the intruding program could intercept the window messages before our UICM receives them. Then the intruder could pass these messages to the original handler. In order to prevent this, we detect any such additional subclassing and terminate the application immediately.

3. CONCLUSIONS

We have presented WebGuard, a system that provides digital rights management to off-the-shelf Web browsers and browser plug-ins. WebGuard pioneers the concept of transparent DRM, and we are currently developing techniques to extend transparency to general applications, beyond Web browsers.

REFERENCES

[1] ContentGuard, http://www.contentguard.com.

[2] IBM's Electronic Media Management System, http://www.ibm.com/software/emms.

[3] Intertrust's Metatrust system, http://www.intertrust.com.

[4] Microsoft's Windows Media Rights Manager, http://www.microsoft.com/windows/windowsmedia/en/wm7/drm.asp.

[5] M. Mourad, J. Munson, T. Nadeem, G. Pacifici, M. Pistoia, A. Youssef, WebGuard: a system for web content protection, IBM Research Report RC21944.

[6] WebBuy, http://www.adobe.com/products/acrobat/webbuy/main.html.