Design Inaccuracy
Cross Link Authoring Flaw - iPaper Platform

Aditya K Sood, - Sr. Security Researcher, Vulnerability Research Labs, COSEINC
Email: Aditya [at] research.coseinc.com

Website: http://www.coseinc.com

12/16/2009

Acknowledgement

First of all I would like to thank COSEINC for giving me enough time to do extensive research and always pushing me to do innovative and good security work.

I would like to thank Mr. Jordan Weins (http://www.psifertex.com) for giving his information driven views and appropriate guidance which results in polishing of this paper.

I also want to draw my thanks to Mr. Rodrigo Rubira Branco (Sr. Security Researcher COSEINC) for providing efficient views

Disclaimer:

The vulnerability discussed in this paper has been reported to Scribd and patched. The paper is an outcome of research conducted on the security artifacts and responsible disclosed vulnerability.

About Author:

Aditya K Sood is a Sr. Security Researcher at Vulnerability Research Labs (VRL), COSEINC. He has been working in the security filed for the past 7 years. He is also running an independent security research arena, SecNiche Security. He is an active speaker at security conferences and already has spoken at EuSecWest, ExcaliburCon , Securitybyte, FOSS, Xcon, Troopers, Owasp, Xkungfoo, CERT-IN etc. He has written a number of whitepapers for Hakin9, Usenix, Elsevier and BCS. He has released a number of advisories to forefront companies. Besides his normal job routine he loves to do a lot of web based research and designing of cutting edge attack vectors.

Personal Website: http://www.secniche.org

Personal Blog: http://zeroknock.blogspot.com

[1] Abstract:

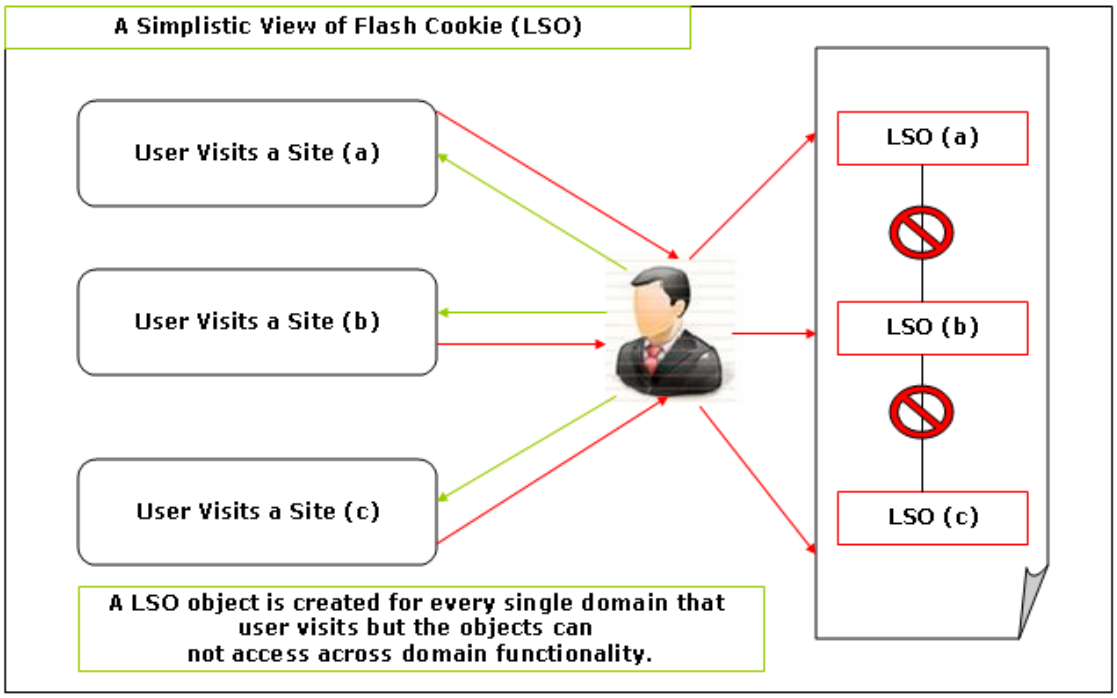
This paper sheds light on the technique of bypassing the iPaper platform for launching a number of web attacks. This iPaper platform is a new document format that is used for online document viewing and is comparatively easy to manage. It is used by a large number of websites. The best example is the Scribd network which hosts a large number of documents online. Extensive testing shows that this platform is vulnerable to a number of web attacks. The attacker is able to leverage the uploading feature and exercise greater control than the website intended. The technology is elaborate and functional but fails to sanitize the attacks appropriately despite using the flash plug-in. This paper demonstrates this factor.

[2] Understanding the iPaper Platform:

The iPaper platform is a web based platform for viewing documents online no matter what the original format is. This technology is considered as an internet based alternative to PDF. It is a full featured web based document viewing system that uses flash. The document is uploaded on the server and streamed back to flash widgets in the user's web browser. Usually, the documents are opened in a widget which uses flash player to display the files in a browser. No software needs to be downloaded as everything is displayed in the browser. With the use of flash player, certain security restrictions are applied.

[2.1] Flash Cookie Dependency – Use of Local Shared Object

The iPaper platform requires flash cookies to operate. Flash cookies are nothing but Local Shared Objects (LSO) which are used to store generic data on a client machine. Its functionality is similar to a cookie except that it stores more complex form (as well as larger amounts) of data. To maintain consistency among different operating systems and browsers, flash is used extensively. The flash player uses sandbox security to protect access to the LSOs. No user interaction is required to store LSOs on the client. Flash from one domain cannot access the LSOs of other domain a security setting defined explicitly by the Flash Player. Let's have a look at the functionality:



The creation of local shared object is based on the request by the file used to open in flash player. In maximum number of cases the local shared object is required and should be created appropriately. Browsers fail to clear flash cookies when other normal cookies are deleted. The flash cookie can impact user privacy and should be handled accordingly. So the use of local shared objects has pros and cons at the same time. The iPaper functionality is entirely based on it. A conversion process between different formats occurs during the streaming of iPaper documents across different platforms. The local shared object is created as below mentioned

local_data = SharedObject.getLocal("user_data");

local_data.data.user_name = "Richard Marx";
local_data.data.user_age = 23;
local_data.flush()

The flush command is used to write the contents to disk immediately. The date created above can be read by the other flash files on the same domain. Shared objects are stored in .sol files located in the Flash player directory of the user's profile,

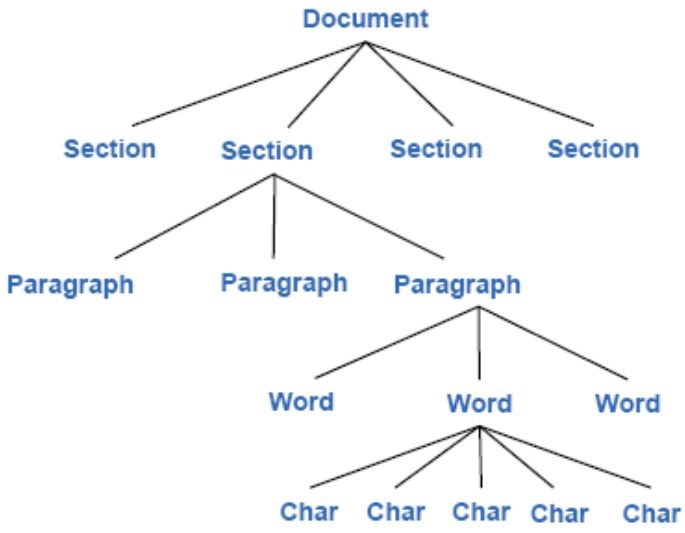
Example: *"C:/Documents and Settings/Administrator/ApplicationData/Macromedia/Flash Player*", and have their own format.

This is generic functioning of local shared objects.

[2.2] Inside iServer and iPaper – General Architecture

The integrated Server (iServer) is the main component of a well structured framework. The main functionality of integrated server is to act as a bridge between arbitrary physical documents or digital papers which are possible only by cross media linking. The iServer has well defined logical concepts that are used for structuring link definitions for a number of resources. As the iPaper architecture is based on the client server model, the flow of information is bidirectional and is directed towards one client and a specific resource at a particular point of time. Links are used for streaming documents back to the client. The links point to the resources or documents that are uploaded by the client in his repository on the web server. The functionality extends beyond just document links, but also to objects present within other resources as well. It means when a client sends a request for a particular object it can be taken easily through the selector which is defined for that object only. It makes the iServer flexible in handling a large number of documents. Due to this cross media linking, documents can be shared easily among a number of users. The general link behavior defines a number of properties for a particular set of document. As discussed above, these properties act as an object in overall document structure and the user or browser can access any specific property easily as needed.

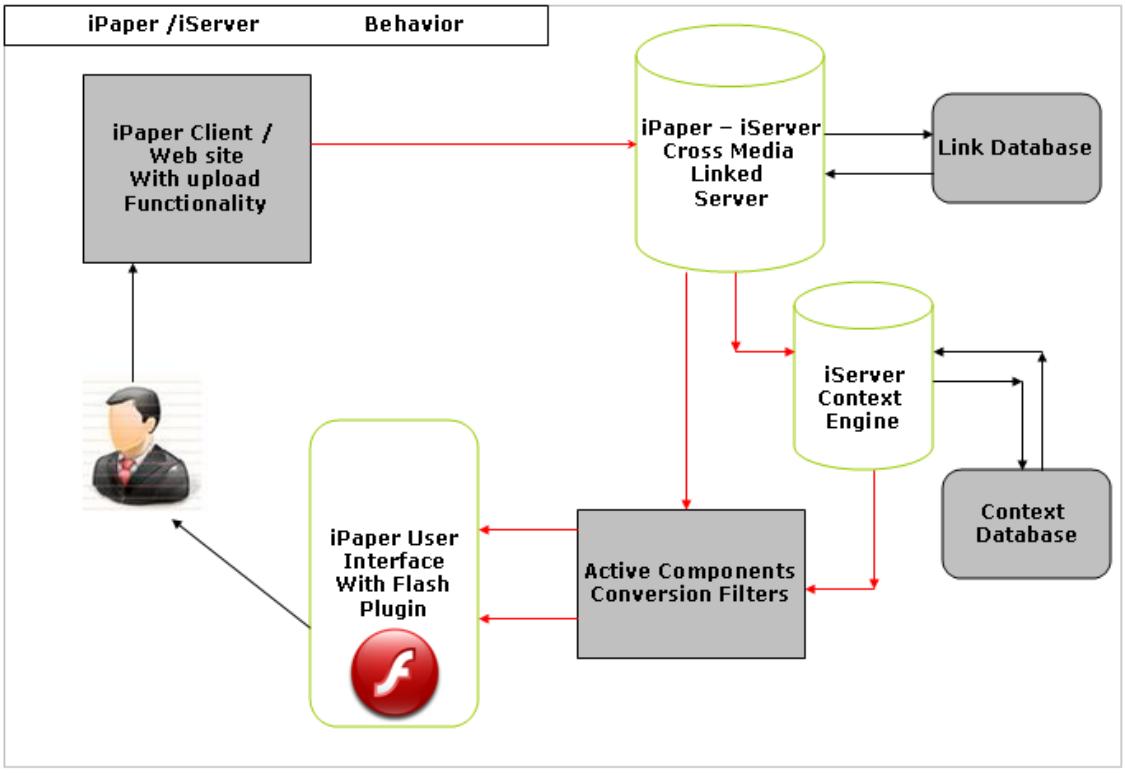
Let's have a look at the document model and the placement of various objects in it.



[Overall Document Structure]

There is no such possibility of Search Engine Optimization through iPaper. This is because of the fact that flash platform is used to display the contents of the paper. While some simple static Flash sites can be crawled by some browsers, iPaper still cannot. It makes impossible for any search engine to crawl for maintaining cache of those iPaper documents. There are certain API's which require the documents' text to be added for displaying it on the website. This is the only place where search engine works effectively to trace the characteristic of the documents uploaded on the integration server.

Let's see how the normal process works.

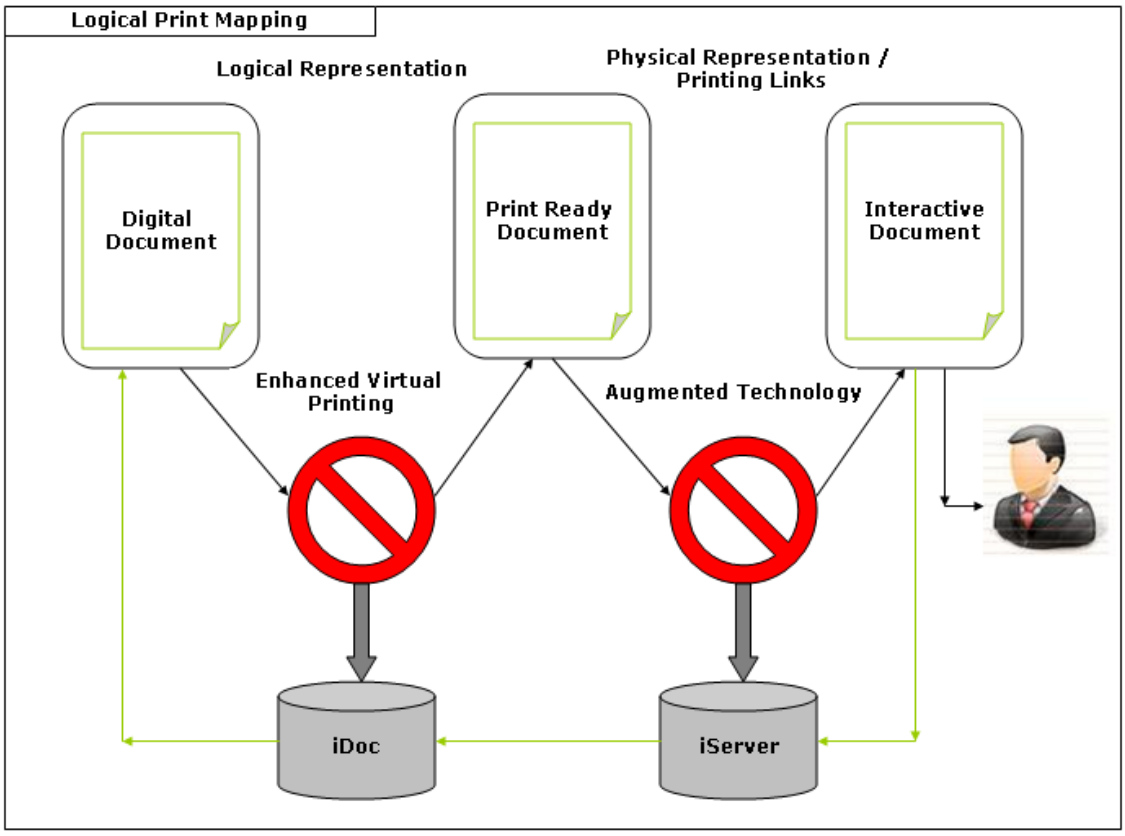


There is a well defined structure for the conversion process of iPaper too. A number of components that come together to provide the desired functionality. These components are:

1. Logical Representation
2. Virtual printing
3. Physical representation
4. Augmenting Technology / Interactive Documentation

The digital document is represented logically. Once the document characteristics are defined after uploading on the platform, the document undergoes virtual printing to generate an intermediate document. This phase comes under physical representation components. At this point, the overall model of the document is constructed but the links and others are not yet mapped . An additional layer of technology fixes the links during the final conversion on the iServer. After that the document is streamed it is finally displayed in the viewer. The overall transition takes place in converting a digital dynamic document to a static physical document and vice versa. All the objects that are defined in the document are converted per a set structure and style. The overall process of generating such a document is termed "Print Mapping". It depends a lot on the filters and security components to convert

the documents without any errors. The overall scenario can be visualized and a protocol is structured below:



[3] XML Based Authoring

To maintain a platform independent architecture, XML based authoring is used by the iPaper platform. All the content has to be transformed to and published from XML. This reduces the effort of converting content for every single platform. The following steps are followed for XML-based Authoring:

3.1 An abstract document structure is is used as the standard base for all type of projects. Different file formats have varied abstract document structures, so structures are prepared on a per file format basis.

3.2 Defining the document adaptation, extension and exchangeability based on the structures and attributes defined for a particular format.

3.3 Defining the Link notation to be used across all element relations in a XML document. It revolves around defining inline linking and Cross references in the document. The inline linking is usually implemented by W3C standard by using XLINK objects.

3.4 The overall structure is implemented in three steps as mentioned below:

3.4.1 Conceptual Level – Describing the content units.

3.4.2 Structural Level – Describing content units for logical structuring.

3.4.3 Media Level – Describing the content units with resource elements and link target elements.

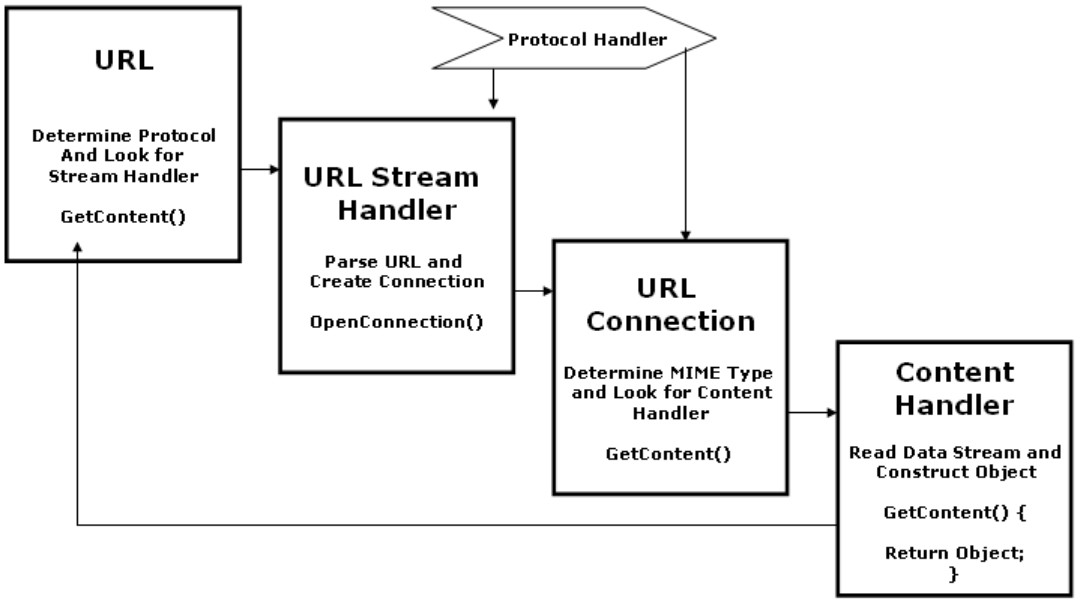
Metadata is used for content (can be media or text) and media content simultaneously.

[4] Protocol Handler Conversion Inaccuracy

There is an inherent flaw present in the iPaper platform's conversion of protocol handlers. Pluggable protocol handlers are defined as communication objects based on a particular set of protocols that can be used directly in the context of running process. As the target is a web based document platform, at this point of time, it means that the protocol object that can be accessed on the web. The protocol object is executed when the entity is rendered in the browser itself. So the pluggable protocol handlers call the functionality of a specific protocol through web browser resulting in another application handling the request. Protocol handlers are primarily URL based. These are the rules by which data is extracted over the links comprised of protocol handlers. Browsers support some of the protocol handlers mentioned below:

- mailto:
- file:
- ftp:
- Telnet:
- view-source:
- chrome:
- http:
- https:
- JavaScript:
- news:
- res:

The vulnerability arises when a protocol handler (primarily for web based attacks JavaScript is the mainelement) is inserted with some argument in the base paper which is to be converted and the server does not appropriately sanitize the link. As a result, when the iPaper is opened in the Flash player or in the browser, the protocol handler object lies hidden in it and gets persistent after completion of conversion process. Let's analyze the generic working of protocol handlers.



Every protocol handler is discreet in its working. For example: inserting links as http and https is correct because it is suitable for application layer that serves the World Wide Web. But protocol handlers like JavaScript, Telnet etc call the specific application directly from the iPaper. The flash player security is not related to this conversion process. The flash player acts as an editor and works only as the destination software to render the content of iPaper. No doubt the there is a lot of difference in security aspects when the iPaper is opened in flash player and the browser itself. But this is a secondary aspect because once the protocol handler is not filtered prior to the building of iPaper; the links become persistent and can result in a lot of web based attacks. Consider the JavaScript protocol handler which renders all the calls made to DOM (Document Object Model) through JavaScript which is indirectly allowed in flash and browser itself resulting in DOM based injections directly in the context of running a website. The Web 2.0 model is a centralized framework for reading and writing content directly on the web. If the framework design allows all types of protocol handlers, then it is a high security risk. The attacker can steal ample amount of information from the web as well as from the client side due to this oversight in the iPaper platform. The vulnerability is discussed in the next section.

[5] iPaper Framework Security Design Flaw - Plug-ins

There is an inherent design flaw in most of the iPaper platforms. For usability and mobility of the content, the cross linking between the data is allowed by this framework. The base entity used for data transference is XML as described above in the form of authoring. The basic functionality of iPaper is divided into two parts as mentioned below:

1. Automatic authoring of iPaper Applications on Web.
2. Support of link definitions in XML

Links may be used for cross-media content. The integration server supports plug-ins for various documents formats such doc, ppt, pptx, and many more. The content in the iPaper is divided into two basic functional entities as presented below

1. Adding links as a static content
2. Links execution as a dynamic content

Once the links are added, it is static content. The integration server supports the active content too as a part of the functionality. The active content is considered a small section of code which is executed as the link target. The program executes on the client-side only when the link is active. . The active components and defined protocol handlers can be used on different platforms irrespective of the base system. It supports the rapid prototyping of new iPaper applications as a part of the extended functionality. The enhancement of different features has potential ramifications on the security of the software.

In the authoring process the active areas defined in the resource file are linked directly to the templates set for iPaper supported by the iServer resource plug-in. Plug-ins used by the iServer therefore plays a critical role in determining the active content to be rendered in the browser. Consequently, any data present in the resource file is not filtered or scrutinized appropriately by the integration server before writing it to the iPaper. The plug-ins convert the content directly without applying any custom filters or sanitization for hidden data present in the links. It is XML based data transference and the links can be easily injected into the final iPaper which is displayed in the flash player embedded in the browser.

The iPaper framework supports a specific set of plug-ins for cross media content access. This framework uses plug-ins for web pages, movie clips, flash files and other resources. The critical point which should be analyzed is that for accessing these specific set of files, HTTP/HTTPS protocol handlers are sufficient

for setting links through URL in XML. The prime insecurity arises from permitting other types of protocol handlers as discussed in previous section. Due to this, the integration server is relying heavily on the pre installed plug-ins. If the integration servers match that the resource file has a installed plug-in it simply starts the automated authoring process based on XML. The resource file is converted into the iPaper directly. This is the major security deficiency in the design of iPaper framework. Flash security itself is not a major concern because the player is only used as reader software and no Action Script operations can be carried out directly without meeting specific set of conditions which are way too hard in the context of iPaper platform. So this design issue can be exploited on the web.

[6] Persistent Link Injections

Injections are one of the foremost exploitation techniques in the web 2.0 model. Any application running on the web which allows a user to provide input values has to be more cautious about the security. It is the foremost principle in web application security of not trusting the user input data before validation--which much occur both on the client-side and the server-side. In this case, for securing iPaper, care should be taken to check the integrity of uploaded file.

The input vector is not consistent across all iPaper platforms because there are a number of files that can be uploaded on the customized integrated server for further processing and conversion of data. The security component comes into play whenever the content has to be scrutinized. If an attacker uses a stealth technique of setting a persistent injection in hyperlinks in a Word document or other Microsoft Office application, the integrated server should remove the hyperlink.

Reference: http://secniche.org/papers/SNS_09_01_Evad_Xss_Filter_Msword.pdf

Content placed in hyperlinks should not be allowed to be transformed when displayed in the flash player and can be further checked by clicking the link directly. This technique of conducting cross site scripting through office documents has already been discussed previously. We will discuss one real time case study to show the presence of this design flaw.

[7] Case Study: Scribd iPaper Platform

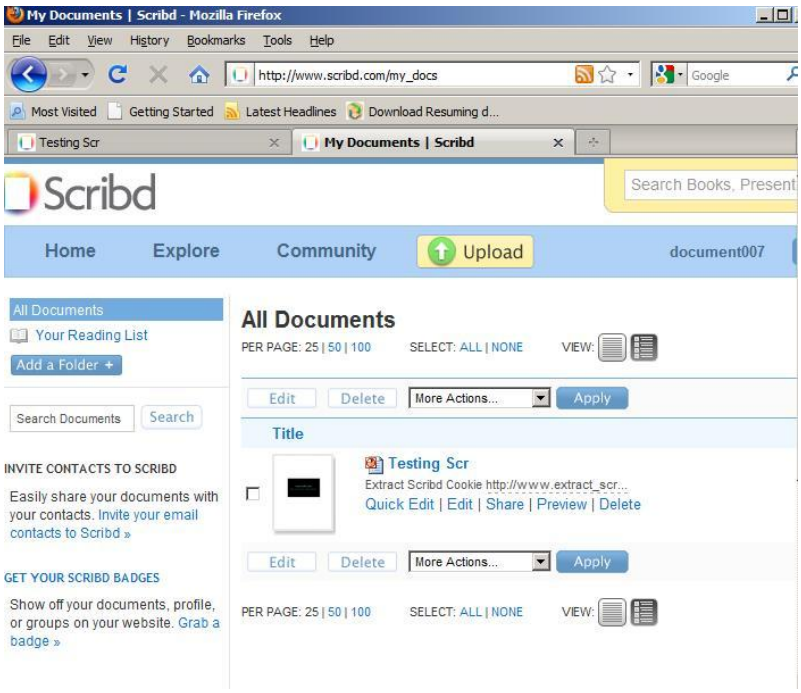
*About Scribd: document-sharing Web site, Scribd.com, with more than 50 million users and more than 50,000 documents uploaded daily, aims to be the YouTube for print. Scribd's main feature is its iPaper application—a Flash reader that allows creators to upload original writings—which permits both uniform publication display of documents and content to be embedded from the Scribd.com site and easily shared. Documents can be downloaded for free, and iPaper allows users to embed text on their Web sites, blogs and on Twitter***.**

Source : http://www.publishersweekly.com/article/CA6640708.html?q=scribd/

This case study demonstrates the design bugs discovered above which leads to serious security vulnerability in the shared environment. There is a possibility of launching persistent cross site scripting attacks due to a design flaw discussed above in detail. Let's analyze the issue in simple steps:

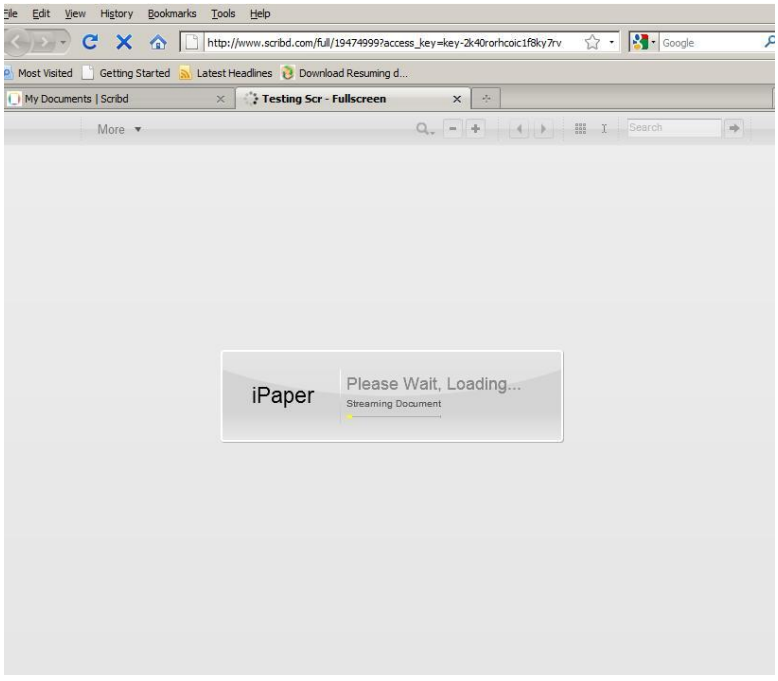
Note: The document can be designed with same attack pattern as discussed previously.

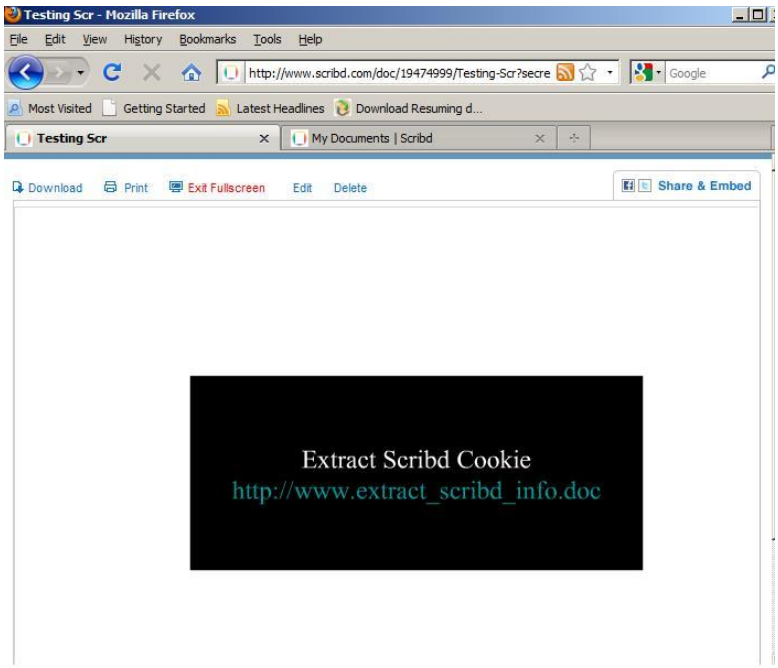
Step 1: Design a document with persistent injection in it and upload it on the scribd portal.



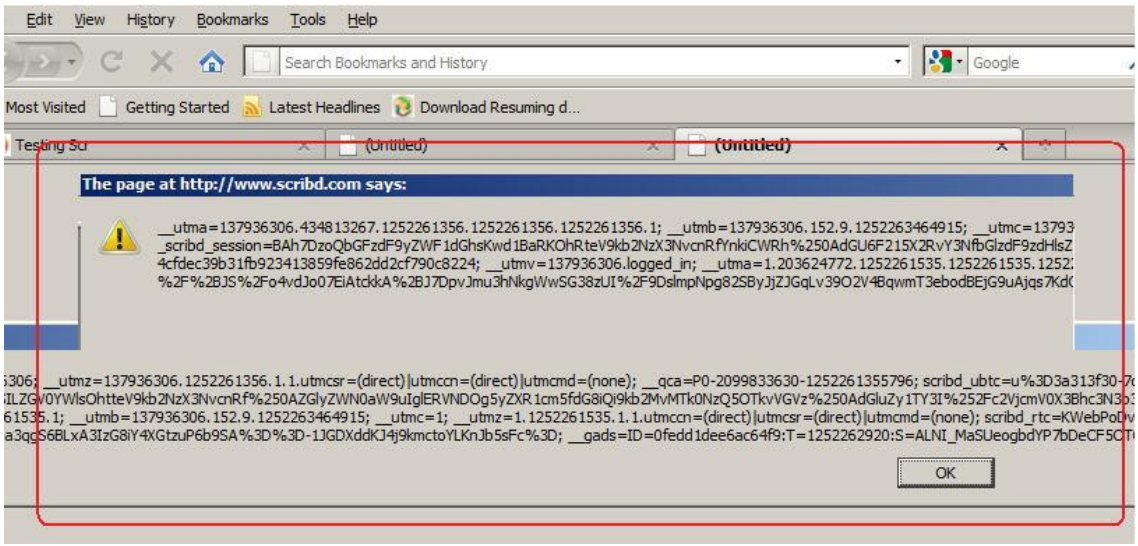
The vulnerable file is uploaded on the scribd platform.

Step 2: Viewing the Document and converting it





Step 3: Executing the Hidden Payload



Due to above discussed design flaw, the cookie from scribd.com is extracted easily--the entire sandbox security model which the Web depends on has been violated.

The case study shows the practical citation of the issues described in this paper.

[8] Conclusion

Design bugs result in severe security vulnerabilities in any running technology. The design has to be implemented carefully, being mindful of security concerns in the social web environment. Appropriate measures should be taken prior to publishing any user-generated content on the World Wide Web so that the attacks can be circumvented in a shared environment. This is possible only when the design is scrutinized as thoroughly as possible. In this iPaper technology, the conversion filters should be structured so that the streaming content is free from all types of hidden or persistent injections. This security vulnerability is a design fallacy and should be corrected in order to have robust applications and software.

References:

[1] http://www.adobe.com/products/flashplayer/articles/lso/

[2] http://www.adobe.com/products/flashplayer/articles/thirdpartylso/

[3] http://epic.org/privacy/cookies/flash.html

[4] http://www.adobe.com/devnet/flashplayer/articles/fplayer10_security_changes.html

[5] http://www.informit.com/guides/content.aspx?g=security&seqNum=276

[6] http://www.xml.com/pub/a/2001/03/21/xmlauthoring.html

[7] http://elpub.scix.net/data/works/att/210elpub2005.content.pdf