



LockLizard Protector

DESIGN CONSIDERATIONS

IT IS IMPORTANT THAT YOU READ THIS INFORMATION BEFORE PROTECTING ANY CONTENT

Introduction

It is essential to understand that the LockLizard Protector Viewer is not a browser into which functionality has been 'plugged.' The Viewer is specifically designed to provide a secure environment in which to handle the range of file objects that are secured. It makes use of certain browser library functions supplied by Microsoft Internet Explorer versions 6 and above that will be present on all current Windows systems from Windows 2000 onwards.

The file types supported for protection are:

htm, .html, swf*, gif, jpg, png, csv, txt.

*swf files generated by Adobe Flex are NOT supported.

File types that can be accepted by the Viewer, but which are not protected at all are:

javascript, css.

These files will be processed by Protector, but their content will not be encrypted.

Use of relative and absolute links to objects

Lizard Protector is not able to search, load and verify nested and variable path objects in javascript and css files that may be satisfied at run time. The design of Protector anticipates a static web site construction with all the links satisfied.

This means, very simply, that Protector requires that all the links to nested or embedded files are expected to be absolute – for example, <http://www.locklizard.com/Images/filename.gif> (in this context a web address is also an absolute link and references to other web locations are managed).

However, Protector will not resolve variable path objects on-the-fly, unlike the standard web browser, and even if it did, the relative objects would not necessarily have been protected because they could not have been identified correctly during the protection process since they were not present and therefore the links could not be resolved. So web page design to manage secured objects must ensure that they are accessed by absolute links and not nested in javascript or css file objects that will not have been processed.



Quality of code

It is important that the implementation of code (htm, js and so on) used for web pages must not contain any errors. Whilst the normal browser will, as standard, ignore such things as javascript errors when it is processing files, Protector Viewer will not. As a matter of security, Protector requires that code to be processed does not contain errors since they could contribute to a security breach.

Designers must evaluate their code prior to protecting information by viewing the proposed web site (before protecting) it in Protector Viewer, and stepping through it noting reported errors.

In larger sites this may be operationally inconvenient or require too much manual intervention. In those cases you can use a web validator to crawl over the proposed web site, especially checking the javascript code, in order to fully validate the code. There are a large number of freeware and paid for products to do this so we cannot make any specific recommendations for a given situation.

Performance considerations

The use of encryption in web based systems can potentially create performance constraints. You will be aware that the commonest form of web encryption, SSL, is switched on and off to that it processes the minimum possible number of pages. Also, typically on web servers, specialist hardware is required in order to 'accelerate' or reduce the impact of encryption on the web server itself.

LockLizard Protector uses a different strategy in order to reduce the impact of encryption on both the web server and on client response times. We now discuss this in more detail.

Protector uses a strategy that information is encrypted once, offline, so that when encrypted files are forwarded by the web server there is a zero impact on web server performance. This avoids the difficulties of providing specialized hardware to support encryption activities, or the possibility of server melt down under heavy load conditions. This is why Protector was developed to address static web site structures and not dynamic ones.

At the client side it is inefficient to attempt to carry out decryption on the fly as blocks of encrypted data arrive. It is possible to develop a buffered streaming approach, but that can have serious memory demands, and can require the opening of temporary files as well as requiring caching of downloaded objects. From the client perspective (the Secure Viewer) it is essential to cache all encrypted information before attempting the decryption.

Encryption and decryption are very CPU intensive processes (hence why you try to avoid carrying out those activities on web servers where it can impact on performance) which compete for machine resource (usually demanded preferentially especially on Windows operating systems) for servicing network connections.

As a result, the strategy of the Viewer is to download all the information that it will require in order to satisfy all the pages and links that it appears likely to require before it starts decrypting information to present on-screen to the client.



This has no impact at all when a secured web site has been provided to a client on a CD/DVD/Flash Drive or has otherwise been loaded onto the client's hard drive, and this guidance may be ignored by designers who are following that strategy when supplying controlled information to their customers/users. If you are operating a protected service from an online web site you MUST take into consideration the following guidance.

The web site designer must consider how they design their site so that it consists of a 'series' of sub sites and not an single entity. If the designer creates a single static web site that is a single entity, protected as a single product, then the client will not be able to view any of the site until the entire site has been downloaded over the web onto the client's hard drive! If it is a small site this may be of no importance, but if it is several hundred pages and hundreds of images then the impact will be significant.

The web site designer must consider, in order to achieve reasonable performance characteristics, how to subdivide their site up into related units of information, and make sure that each related unit of information has its own 'gateway' page by which that area is primarily accessed. Then, when protecting the site, rather than protecting the whole site (typically done by referencing the master landing page index.htm and letting Protector sort out all the links and interdependencies from there on) each sub-section of the site should be protected according to its own gateway entry, but the Product MUST be published to one single publication.

The effect of adopting this strategy is two fold.

From the client side the delay whilst caching all the associated files to the entry point is reduced so they get to the first viewable page in a sensible time (waiting for a whole site to download might be very similar to waiting for Godot, especially if your broadband service is not that great, and not all of us are on a T1 circuit).

Also from the client side, once file objects have been downloaded and cached they will NOT be downloaded again unless they change, so the performance at the client side will increase significantly following the original download.

However, please note that whenever a product is opened, a check is made at that time for file updates, and not on-the-fly as page items are loading. The client is advised that an update check is being made.

The reason for protecting to the same publication is a simplification for administrative purposes. It means that when allocating licenses to customers/users you only need allocate them to a publication, and the LockLizard infrastructure will automatically sort out the provisioning of decryption keys and so on, without you or the customer/user needing to take any action at all.

Please note: If you are provisioning a 'layered' web site, where different classes of customers/users have access to different parts or portions of the web site or have different information revealed, then you MUST design the site so that each layer is a separate Product. Where layering is required, each layer MUST correspond to a specific publication. That way, when authorizing customers/users you are able to connect them to the appropriate publications.



The effect of these steps is to minimize the amount of administration that will be required whether you are administering the system manually or selling complex services directly from a web site. It also minimizes the performance impact on both your web server, and on the client side, so that the client experience is as seamless as possible.

Multimedia interfaces – operational considerations

This guidance is given for flash files, but it is equally applicable to any multimedia files that are included as part of the protected site.

When running any browser or Viewer based system, a file to be displayed must be downloaded/cached before it can actually be displayed. This is particularly important where information has been encrypted, because you cannot possibly know if the encrypted file is valid until you have checked the whole of it.

Where you have complex page structures (text, images, htm and so on) then all of those elements have to be decrypted and checked before they can be displayed. This is very different from, say, an encoded movie channel, where if anything becomes corrupted then the channel just fails.

So minimizing the size of any multimedia files is also very important. When a browser encounters a multimedia file (flash, mov and so on) it pauses whilst it downloads the file before it is able to display it. It does not matter whether the browser is going to use its default display mode or invoke a client preferred plugin. Until the file is downloaded/cached in its entirety there will be a pause at the client end. So the larger the object that has to be downloaded at this time, the longer the pause that the client experiences. So whilst a 4 Mb file download will likely be tolerated by the client, a 100 Mb download will likely not be tolerated. Unless you warn them that this is not instant response, or even cup of coffee response, but a lunch break response! Some useful flash compression software can be found at http://www.hootech.com/mp3_to_swf_converter/

Because of the structure of encrypted information it cannot be verified until the whole encrypted file has been received. So unlike streaming systems (film, music) which start playing as soon as sufficient data have been received to provide a reasonable level of user experience, all the file must be downloaded before it can be opened and used.

Obviously, if the files have been distributed on a storage device (CD/DVD/flash drive) then performance will be much faster.

Multimedia interfaces – links and plugins

As with the use of links mentioned above, if javascript is being used to reference the multimedia file it needs to provide a direct link and not a relative one. Mp3 or mov files can be played, as long as the path is correct.

We recommended that you use an absolute path when calling a mp3 or mov file (so instead of trying to load /audio/file.mp3 use the form <http://domain.com/audio/file.mp3>). This is because the way plugins for mp3 or mov files operate can vary (unlike flash files, where you have to use the Macromedia plugin, and so there is a single defined interface and method of



operation) and they may not properly handle a relative path (remembering that as far as the plugin is otherwise aware the protected HTML file is being loaded directly from memory and not from any storage location, and no temporary files are opened).

Since mp3 and mov files cannot be protected unless they are converted to swf format, you may wish to open these files in a normal browser so they are not loaded from memory but instead cached from disk. This will provide an enhanced user environment with content loading quickly in the browser (rather than slowly in the secure viewer). By using the target="blank" command you can force the opening of files in a browser (rather than the secure viewer) window. Example html code to perform this action is as follows:

```
<a href="http://domain.com/audio/file.mp3" target="_blank">
```

Other items for consideration

- 1) Adobe Flex generated SWF files CANNOT BE SUPPORTED because the Adobe plugin is not supported by Protector Viewer.
- 2) Javascript menus must use absolute paths. Javascript menus that change the menu item image when moving the mouse over it, will only work if the image path is absolute.
- 3) Forms will work in the Secure Viewer software and data can be filled in by users as normal.