Securing a Livelink System

A White Paper for OpenText's Livelink

Version	**1.1**
Release Date	**11 June 2001**
Author	**Greg Griffiths**
Contact Email	**livelink-whitepapers@greggriffiths.org**
Website	**http://www.greggriffiths.org/livelink/whitepapers**

Introduction

This White Paper discusses a variety of approaches that can be taken to improve the security or a Livelink implementation. Using the ideas presented here you should be able to provide a robust and easily manageable security model for your installation.

This White Paper is a basis of a good overall security model which should include the webserver and database software, the server Operating Systems and all other software and hardware involved in providing Livelink to your users.

About the Author

The white paper's author is **Greg Griffiths**. He is currently the eBusiness Technical Consultant for a European based Food Solutions Provider. He has been a Livelink Administrator and Developer for the past two years and has produced many addons, white papers and customised templates for the product.

More Information about the Author can be found at :
http://www.greggriffiths.org/livelink/authors/greggriffiths.html

More tools, customisations, white papers etc can be found at :
http://www.greggriffiths.org/livelink/

More information about Livelink and OpenText can be found at :
http://www.opentext.com/livelink/

Contents

Livelink Permissions – an Introduction

Introduction

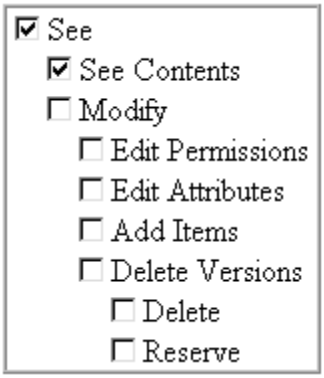
This section introduces the inherent security model in Livelink to give you a better understanding as we delve into some of the more advanced topics later in this White Paper.

Access Permissions

Each object has the following permissions, some of which have pre-requisites permissions :

Permission	Description
See	Has rights to see that the object exists.
See Contents	Has rights to view the contents of the object
Modify	Has the rights to change certain things about the object
Edit Permissions	Has the right to change the permissions on the object
Edit Attributes	Has the right to change the attributes associated with the object
Add Items	Has the ability to add other objects to this object (if possible, e.g. for a folder or compound document)
Delete Versions	Has the right to delete pervious versions of the object
Delete	Has the right to delete the object
Reserve	Has the right to reserve the object

The permissions hierarchy looks as follows :



Thus, to **Edit the Permissions** of an object, you need the **See** and **Modify** permissions as well.

3 Access Groups

Livelink offers 3 groups of access to any object within Livelink. The first two are the **User** and **Group** - which are your standard users and the groups in which they belong - and the third is **Public Access**. The following table explains this approach :

	User	A named user with a login, such as Joe Bloggs, John Doe and your Livelink Administrator.
	Group	A collection of users who share a common identity, such as Mannagers, HR staff, Bowling Club members etc.
	Public Access	Everyone. This group provides access for everyone within your Livelink environment.

Access Rights are cumulative

You, as a user get the most rights that you can have to an object. For example if we have the following arrangement :

HR Staff
John Doe
Peter Rabbit
Christopher Robbin
Managers
John Doe

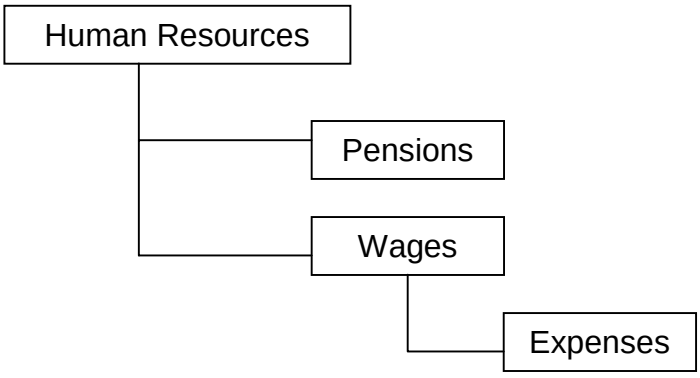
If the **HR Staff** groups was given only *see* and *see contents* rights to an object, but the **Managers** group had *delete* rights, then John Doe would have rights to delete that object.

This fact becomes increasingly important as we look at more complex models and 'sensitive' documents later on.

Beware of 'Apply to All Subitems'

This option of the permissions screen is a very powerful tool when applying a complete profile of permissions to a large number of objects under the current one. However, be careful as the function replaces ALL existing permission models in ALL objects underneath the object with the complete set defined for that object.

So, for example, if we had the following structure set up within a Livelink system :



Alongside this we have the following Users :

John Doe, Christopher Robbin, J R Hartley, A N Other, Peter Rabbit

These users are organised into the following groups :

HR Staff
John Doe, Christopher Robbin
HR Manager
A N Other
Finance Staff
J R Hartley

The HR Staff group has full rights to all the HR area, J R Hartley has *see* and *see contents* rights to the top level only and Peter Rabbit only has full rights to the Expenses section.

If we hit 'Apply to All subitems' at the top level, J R Hartley **gains** *see* and *see contents* rights to the entire tree and Peter Rabbit **loses** all rights to the Expenses section. This change is probably not a good move.

'Apply to All Subitems' can be used to quickly and easily replace the permissions model of all child nodes with that of the current node. In some cases can be an extremely useful, timesaving and powerful tool, it just needs a little thought before being used to avoid giving the wrong access to the wrong people or removing access from others.

Securing your Base System

Introduction

This section is intended to give a brief look over the security of the base system – Operating System, Webserver etc - that will be running your Livelink implementation and provide some suggestions on how to set up this environment to be as secure, safe and supportable as possible.

Operating System and WebServer

Livelink currently runs on both Windows and Unix systems. Many people are aware of the security issues affecting both platforms, and given the increasingly frequent attacks on webservers you must take some precautions in this area, even if the server is only internally facing.

In most cases, your company will have a Server Administrator and possible even an IT Security Officer. These two people will be able to provide you with a valuable resource to assist you when setting up your servers.

Always ensure that your servers are patched with the latest service pack for both the Operating System and the webserver and keep an eye out for security alert and bug fixes on the vendors websites. Don't forget to check out these changes on a test system before implementing them on your Live server !!!

Secure Environment

Most companies place all their servers in a 'secure' server room, and your Livelink server should be no exception to this, even your Test and Development servers should be in here if possible.

Each server should also have a unique password for the computer itself as well as for the Livelink account on the machine to enhance the protection already offered.

Backup

A good backup of all the servers is important as this provides a safety net for any problems that you encounter such as upgrade failure, hardware failure or system crash.

The backup of your Livelink application servers should remain unchanged for long periods – with the exception of the logs and indexes – so you could potentially use an older backup if required. You should obviously use the most recent stable backup of your database that you have.

Most backup tools require applications to be switched off to back them up. However, you may want to run your Livelink system on a 24x7x365 approach, in which case this could be a problem. However, most backup tools now have a 'Live' or 'hot' backup option, although an regular offline backup is also advisable.

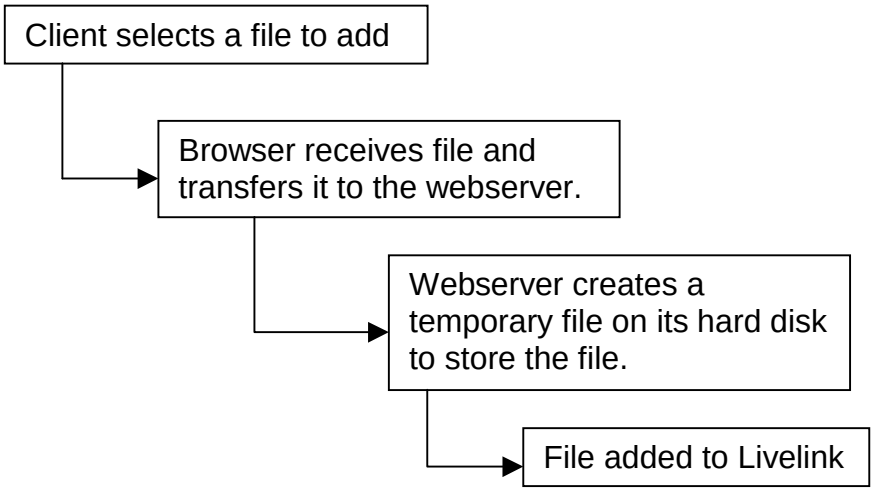
Shared Drives

Be careful of sharing drives on any of the servers involved in your Livelink implementation. Shared drives give a user the opportunity to directly access the hard disk of that server and some changes may not take effect until Livelink is restarted.

Antivirus Software

As viruses become more common, the need to defend against them becomes more important. Livelink can not be 'infected' by a virus, but documents stored within can contain a virus. This can cause major headaches if you are trying to eradicate a virus and find it reappearing on 'cleaned' machines.

However, OpenText most strongly advise you not to scan the application or index folders to avoid issues such as corruption or file conflicts. That said, an up to date virus checker can be very useful in trapping viruses before they reach Livelink. The standard process for file uploads for webserver's is as follows :



If an anti virus program was installed and configured correctly, it would trap the virus at the third step as the temporary file was being created on the webserver. The software may clean the file and then allow it to be added to Livelink if cleaned, but in all cases should alert the Admin user to its activity to allow them to track down the source of the virus and prevent it spreading.

Logs

Most webservers and some Operating Systems provide a logging option. This will provide with useful data to determine usage demographics such as most common browser, section most visited, period of most activity etc which will assist you in enhancing the offering to the user.

Secondly, the logs may show some up some strange oddities, for example a computer trying to login multiple times in a short period of time may just be a user who has forgotten or mistyped their password, but it could equally be a hacker.

Logs also provide additional audit trial information to that of the Livelink logs. Using both sets of logs it is possible to identify not only the logged in username of a given user, but which machine they are using. So a user in the next room as you who apparently deleted you annual report may be proved innocent if the webserver logs show that the originating IP address is on the other side of the world.

Checking the logs for anomalies should be a regular part of your monitoring process of your Livelink installation. IIS provides the option to place it logs into a database table rather than text files, which may make analysis easier.

Admin.index

This Livelink function provides access to the backend of your Livelink installation, where system configuration changes can be made. The URL is normally in the form :

http://www.livelink.company.com/livelink.exe?func=admin.index

If your webserver supports it, then access to this URL should be barred from all machines except a few such as the server itself and your machine.

Internal Permissions

Introduction

This section intends to investigate approaches to providing a manageable and effective permissions model for Livelink. We shall look closely at nested groups, group v user permissioning, guest accounts, group accounts and Public Access among other things.

Why do we need a permissions model ?

Any other System Administrator, be it for Windows NT, Unix, or any other system or even an application itself will be able to demonstrate the need for a permissions model. But, why does Livelink specifically need one ?

Livelink is a document and knowledge management system for your company. Thus it is likely to contain several different types of information :

Public	Anyone inside or outside the company can see this, e.g. Company Address.
Company Confidential	Should only be available to employees and not to others, e.g. staff rota.
Restricted	Information that is only to be available to a limited number of people such as Company Directors.
Personal	Information that is private and should only be available to you, e.g. bank balance.

It is this requirement to protect certain files that is the basis for a permissions model.

Permissions model also limit the access users have to a system. For example, you would not want someone from your Sales team to be able to alter their salary, or for a guest user to add or delete files from your system.

User / Group creation

Users should be created in a controlled manner (perhaps even convert the process into a Workflow). Each user should be given the access rights that they need and added to the correct groups.

Similarly 'system' groups such as **Managers**, **HR Staff** should be created and managed in a controlled manner so that you know who is in which groups. This assists when assigning permissions to groups as you are aware of who the members are and this may have a bearing on the access you give them.

With these groups, you have the additional headache of managing the issue of change within them. For example, a new member of staff, someone leaving, someone transferring departments etc all may require a change in their membership of groups. This should also be supported by some sort of Change Control process that shows the change, who requested it, who performed it and why.

You may give your users rights to create groups themselves. This may not need the same rigid control as for 'system' but you should keep an eye on them from time to time to ensure that they are correctly configured.

The relationship of users to groups is one to many, i.e. one user may be in many groups. The same is true of groups, one group may be in many groups itself, this is where the complexity comes in and the need for a clear and manageable permissions model begins to become more important.

Single User or Single User Groups ?

Why use a group that would only contain a single user when you could just assign the user themselves ? For example you may have **Peter Rabbit** as a **HR Manager**, so you set up this user to have all the rights and privileges that they need. That may seem like a better way to do things initially and saves you having to create a group AND a user.

However, what happens a few months later if he leaves, or you hire another HR Manager ? How do you ensure the smooth transfer of all the permissions that Peter Rabbit has are given to this new user and removed from Peter Rabbit if required ? As well as all the documents, projects and folder permission changes there is also the workflow steps that will need altering. If you used a group, then you simply change the membership of that group and it would all be over.[1]

Nested Groups

A group can be a member of another group, so you could potentially have multiple groups in a group which itself is a member of another group etc. This when properly organised can provide a useful structure that can save time in permissions. For example if you give access to a project group called **Project Team A**, this team has a technical group and a business group, therefore you would have the following arrangement :

<table>
<tr><td colspan="2">Project Team A Group</td></tr>
<tr><td>Technical Group</td><td>Business Group</td></tr>
<tr><td colspan="2">Users</td></tr>
</table>

Thus, we could assign either the **Technical** or the **Business** or the **Project Team A** group to object. If more than one group is assigned, each could have different permissions.

Object Creation Permissions

If a specified user that you create will never need to create a Live Report for example, then why do they have access to do so ? Once you have created the user, check what objects they have the right to create an amend the restrictions list as required.[2]

[1] If you do need to do it the hard way, then simply change the user profile of user A and reinvent them as user B. Or run a SQL statement to change all the references to user A to user B or just give user B the password to user A's account. None of these are really good practice.

[2] The **Restrictions** section is contained in the Administration side of the system accessed via **admin.index**.

Guest Accounts

Introduction

Many companies have so called 'Guest' accounts for various systems, these accounts are not assigned to a single user, but are to be used to gain access to your system. They are normally used for a specific function, such as initiating a workflow or partaking in a product demonstration, or just showing a non-user content which is inside the 'system'.

The problems of Guest Accounts

These accounts need closer monitoring than normal accounts as you are unlikely to know who is using them.

On the surface of it, a Guest account is a quick and easy resolution to the problem of getting users into your system for a specific reason. However, there are plenty of hidden dangers that you will need to consider and prepare for.

A Guest user account has the same permissions as the lowest user in your permissions model, quite often this is the **Public Access** grouping, thus anything that I can see because I have the rights from **Public Access** so can a guest user.

The same problem extends to other functionality such as Search, adding / deleting / modifying object within your system, setting Change Agents etc. This could potentially result in the user being able to add a document, create and then run a Live Report, set up a Change Agent to alert them when anyone adds a document, with the email address being an external one. They may even have he ability to create a user.

It is likely that you as a company could be held legally responsible for the content of your Livelink server, imagine what could happen if an external person added some illegal content, such as a slanderous document or pornography to your server without the knowledge or agreement of your company ?

Just as worrying is the prospect of this person being able to add, amend or delete some important information from your system. This could result in major problems if other people such as Purchasing or traders on the stock market used incorrect information.

Securing a Guest Account

How do you go about securing a Guest Account ? In reality it is just like you would permission any other. Start by reducing the ability of this account to create objects within your Livelink environment by using the XXXX option, you may even want to remove their rights to ALL objects.

Secondly, go in as Admin and browse that users **Personal Workspace** then remove the accounts permissions to see their own personal workspace. This stops then following any obvious links from the **Go To** menu or similar.

If the account is used for a routine purpose such as an onsite Sales Demo, then lock the account until it is needed, then unlock it for the duration of the demo. This can not be done if the purpose of the account is to initiate a workflow on a regular intermittent basis.

Another important question is that of need – *why does the content need to be held within Livelink ?* – if all you are displaying is contact info, an about us page and some diary information for example, then why not use a standard webserver – like the one you need to run Livelink – to handle this and keep it out of Livelink completely, thus removing the need for that guest account.

User Passwords

Introduction

In this section, we will take a brief look at the settings for user passwords and how these impact other Livelink related products and other existing non-Livelink systems.

Password Creation

What are the current requirements for a users password in other systems that your company uses, and how does this reflect to Livelink ? If you have a company standard then it may be possible to define this using the XXX section of the Admin pages.

For most systems a password should :

- Be at least 8 alphanumeric characters
- Contain at least one numeric
- Not be a common word such as 'password1'
- Not be something that is easy to guess such as your name, date of birth etc

Your company may require some different requirements for creating passwords. The first two of these points can be set using the XXX section of the Admin pages.

On Time

The diary and calendaring application of Livelink entwines itself with the users passwords and they need to be kept in synch – this is usually done automatically, but can be e-applied by the **Admin** user.

It is worth exploring the impact of the password control settings in On Time in conjunction with those of Livelink.

Directory Services

This is the current name of the LDAP module for Livelink. This allows a user to login once and once only, but still have access to things - such as Livelink – that normally require a separate login action.

If you are using this module, then you may have preset requirements on passwords that are enforced by the LDAP server due to other systems or a documented specification.

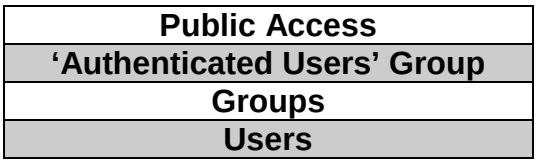
An ideal permissions model

Introduction

This section intends to provide and describe a permission model that addresses the issues raised above. This model attempts to be flexible and manageable while offering a robust security model.

Outline Diagram

The following diagram illustrates the permissions model that we shall exploring in this section :



This tiered user model offers a flexible, powerful and manageable permission model to a Livelink implementation.

Public Access Tier

This is the first tier of access, this is the permissions set assigned to the **Public Access** entity – signified by the symbol. For genuinely 'Public' information this should be set to **See** and **See Contents** only, and set to **Nothing** everywhere else in the system. This limits what 'any' user has access to within the system.

'Authenticated Users' Group Tier

Create a group – for example **Authenticated Users** – and add into this group every user account that is not a 'Guest' type account. This gives the ability to split permissions between 'everyone' and 'every user'.

It is then recommended that instead of using the **Public Access** setting to give all the users access to a particular object, that you assign this group instead, which achieves the same result, but without granting access to every user. In most cases the permissions assigned to this tier will just be relatively small, such as **See** and **See Contents**.

Another feature of this tier is that it should the first tier to have any ability to create objects within your Livelink system. Thus all objects with creation rights set to **Unrestricted** should be restricted to this group to remove the ability of any user – such as a Guest account – being able to create any object within the system. It may be necessary to grant some access to certain Guest accounts, but each should be treated as a special case and should have a valid reason, documentation and approval from a senior manager.

Groups Tier

This is the tier where you carefully organised groups are used, they provide the opportunity to give one set of permissions to all your users – using the previous tier – and then give more rights to a select group(s).

Within this tier you could have several child groups of a single parent group assigned with different levels of permissions. For example :

Group	Sub Group	User
Project Team A		
	Technical	
		Peter Rabbit
		John Smith
	Manager	
		Joe Bloggs
	Sales	
		Christopher Robbin
		John Doe

Thus, we have the option to either assign **Project Team A** as an entire group with a single set of permissions for the entire group, or we could set up each of its sub groups up with a different set depending on the content. Thus, the **Sales** group may have all the rights and the others may only have **See** and **See Contents**.

User Tier

This is your individual named users. In most cases they will never be used directly as all their permissions will be inherited from their membership of one of the previous tiers.

It is likely that at the n^{th} level of your system – where your users have a lot more control over permissioning – that some of your permissions will be of this sort. It is recommended that wherever possible this is not done, especially within the key areas of your system.

Reworking an Existing System

Introduction

This section looks at some of the issues that you may face if you are trying to re-permission an existing system, and provides an evaluation of the options and obstacles that you may face when doing so.

User and Group Hierarchy

From your existing system, document the users and groups that already exist. This will help you to understand the current permissions model as make adjustments and also may highlight any issues or areas of confusion that need clarification from others.

You may find that you have a huge number of groups, in many cases you will find several groups have been set up for the same purpose of to represent the same 'entity'. However, it is also likely that each of these disparate groups will have a differing selection of users. These groups could probably be rationalised down to a single group.

Object Creation Permissions

Alongside the existing permissions structure it is worth investigating and documenting what object creation rights have been defined for the system or have they all be left as **Unrestricted** ? If not, it may be worth investigating who has the rights to create what within the system and you may find that some users have too much or too few rights in this area.

If you find that a user has the ability to create more objects than they need, for example if a user doesn't know anything about SQL then why do they have the ability to create Live Reports ?

You may have a problem when you are reducing someone abilities to create objects, even if they never have and probably never will use them. Thus it is probably worth having a very well reasoned argument and supporting evidence to convince them or their / your manager of the need for this change.

Permissions Structure

Document the permissions in place for the top few levels and any other key areas of your system. With this information you should be able to see who has what permissions to the key content areas within the system.

This is useful when you organise your groups to ensure that each user who previously had a certain set of rights still has them under your new model – if they actually need them in the first place of course.

Internal Organisation

In conjunction with the previous section it is worth examining what content you have within the key areas of your system. This allows you to spot any odd things such as empty folders, bad aliases, out of date content. Armed with this information and with an understanding of the users needs, you should be able to come up with a pruned, leaner and cleaner structure for these areas.

Get Buy In to Change

If you need to make any changes either as a result of the above or due to user feedback, then ensure that you have buy in from the managers within your company and that you have a well reasoned plan for making any changes to the existing system.

While this is not in one sense part of providing a secure environment from a technical point of view, it will make life easier when you come to implement this revised system.

Passwords

What are the current settings for user passwords, what is the minimum / maximum length, must they contain digits etc ? If your company has an existing standard then this could be mapped into the Livelink password settings.

Be careful if changing this on an existing system as some users may have passwords that don't fit your existing criteria and this may cause problems. It is also worth being wary if you are using any other modules such as **On Time** or **Directory Services** for example.

Conclusion

This document provided information on how to implement a securely permissioned Livelink system, as well as looking at the underlying systems that support a Livelink implementation. It also gives some thoughts on how to approach reorganising an existing Livelink installation.

I hope that everyone has been able to draw something from this white paper that is of use to them in their work with Livelink. If you have any additional comments or suggestions, please let me know.