Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today

Level: Beginner
Works with: Domino 6
Updated: 01-Oct-2002

by
John
Chamberlain

In the previous *LDD Today* article, "**Building Web applications in Domino 6: Web site rules**," we introduced the Web Site Rule document, a document type available in the new Server\Internet Sites view in the Domino Directory. We also discussed how to use this document to create redirection and substitution rules.

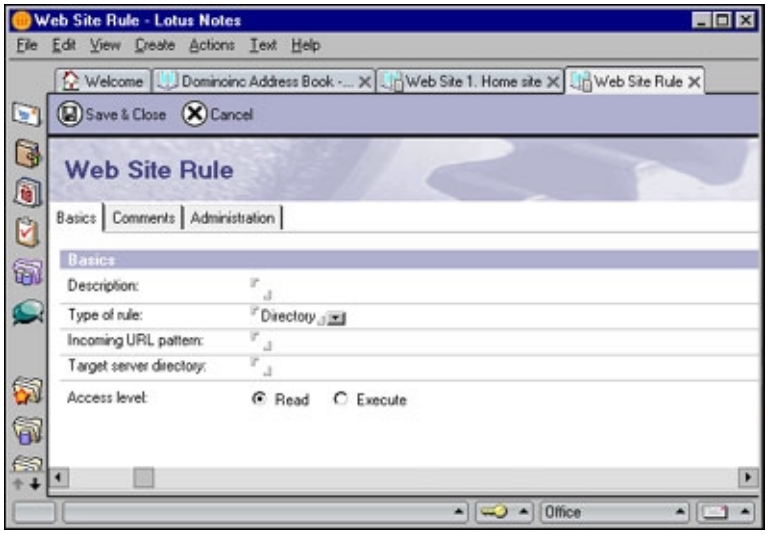
This article covers another type of Web site rule, the directory rule. If your Web applications include a large number of HTML files, images, and other file-system resources, you can use directory rules to help organize these resources. Directory rules allow Web applications to access directories located anywhere on the server, or even on another machine.

In this article, we also discuss how to control access to file-system directories with file-protection documents. This is an R5 feature that has been carried over to the new Internet Sites view. In R5, file-protection documents were applied to the entire server (or to a virtual server); but in Domino 6, every Web site can have its own file-protection scheme. However, file-protection documents do not provide the same level of security as the Domino database security model; we will explain why this is true and when each security scheme is appropriate.

This article will be of interest to all system administrators and webmasters who plan to host Web sites with Domino 6. It assumes a familiarity with the Domino Directory and administrative tasks. In addition to the article mentioned above, you may find it helpful to read the *LDD Today* articles, "**Building Web applications in Domino 6: A tutorial on Web site addressing**," which explains the basics of setting up Web sites in Domino 6.

A quick review of Web site rules

We thoroughly covered the basics of Web Site Rule documents in the previous **article**. To review quickly, Web site rules are response documents to Web sites defined in the Server\Internet Sites view of the Domino Directory. To create a Web site rule for a site, open the Web Site document, click the Web Site action button, and choose Create Rule. This opens an empty Web Site Rule document:



In the Description field, you can enter any descriptive text that identifies or explains the rule.

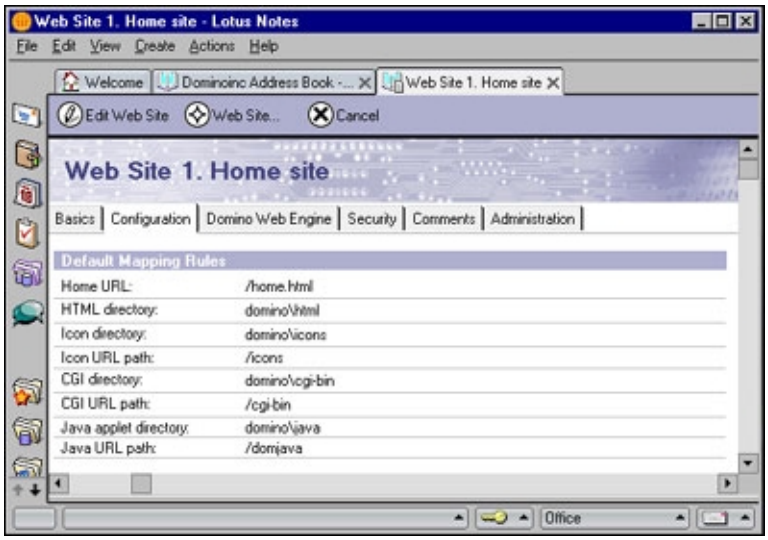
The default rule type happens to be Directory, just what we want for this discussion.

In the Incoming URL pattern field, you specify a template, or pattern, that URLs sent from browsers should match. The pattern can include one or more asterisks as wildcard characters. For example, the pattern /*/catalog/*.htm will match the URLs /petstore/catalog/food.htm, /clothing/catalog/thumbnails.htm, and so on. Patterns for directory rules, like patterns for substitution rules, must include at least one wildcard; if you do not explicitly specify a wildcard in a pattern, Domino 6 will internally append /* to it. For example, if you specify the pattern /shop, Domino 6 will treat it as /shop/*.

The rest of the fields in the Web Site Rule document tell Domino 6 what to do with URLs that match the pattern. The redirection and substitution rules that we covered in the previous article cause Domino 6 to transform incoming URLs into different URLs before requests are processed. Directory rules have a completely different purpose. A directory rule maps a file-system directory to a URL pattern. When the Web server receives a URL that matches the pattern, the server assumes that the URL is requesting a resource from that directory.

Built-in directory rules

Before we discuss the specifics of directory rules, let's look at some that are built-in. When you install the Domino 6 Web server, a number of file-resource directories are automatically created. These default directories are mapped by directory rules that are defined on the Configuration tab of the Web Site document:



There are four default directories created:

- A directory for non-graphic files (such as HTML pages), shown in the HTML directory field
- A directory for graphic images such as GIFs, shown in the Icon directory field
- A directory for CGI programs, shown in the CGI directory field
- A directory for the default set of Java applets

When the Web server starts up, it automatically creates internal rules to map these directories to URL patterns. As a site administrator, you have some control over these rules, but you cannot completely disable them.

The default directory for non-graphic files is created below the data directory, at C:\Lotus\Domino\Data\domino\html (for consistency, throughout this article, we will assume a default installation of the Domino 6 server on a Windows machine unless otherwise specified). The default value of the HTML directory field in the Web Site document points to this directory (it actually specifies domino/html, which is relative to the data directory). You can change this field to point to another directory if you wish. However, you *cannot* change the URL pattern that is mapped to this directory, which is hard-coded to /*. The pattern is hard-coded because this rule acts as a fallback for all URLs; that is, if a URL cannot be interpreted in any other way, it is assumed to specify a readable file in this directory. For example, the URL /welcome.html specifies the file C:\Lotus\Domino\Data\domino\html\welcome.html; and the URL /product/specs/widget.htm specifies C:\Lotus\Domino\Data\domino\html\product\specs\widget.htm.

The Icon directory field in the Web Site document points to the default directory for graphics files (such as GIFs), created at C:\Lotus\Domino\Data\domino\icons. The URL pattern is defined in the Icon URL path field; the default pattern is /icons. We strongly suggest that you do *not* change either the directory or the URL pattern, because this directory is pre-loaded with a standard set of GIF files, such as navigation icons, view column icons, and other images that are used on Domino database responses.

The CGI directory field in the Web Site document points to the default directory for CGI programs, created at C:\Lotus\Domino\Data\domino\cgi-bin. The URL pattern is defined in the CGI URL path field; the default is /cgi-bin. You may change the directory and pattern if you wish.

The Java applet directory field points to the default directory for the standard set of Java applets installed with the Domino server, such as the view and action bar applets, created at C:\Louts\Domino\Data\domino\java. The default URL pattern is /domjava.

If these four default directories are sufficient for your Web site, great—you're all set! But if not, you can add as many directories as you want by creating your own directory rules.

Creating a directory rule

It's important to understand that directory rules can *only* be used to map the location of two types of resources: files that are to be read directly (such as HTML files and graphics files) and executable programs to be loaded and run by the operating system (CGI programs). Directory rules *cannot* be used to map the location of other types of resources, such as Domino databases or servlets. These resources have their own rules for specifying locations.

To create a directory rule, you create a new Web Site Rule document (as described **above**). Enter a description in the Description field and make sure the Type of rule is set to Directory. Set the Incoming URL pattern field to whatever template, or pattern, you want URLs sent from browsers to match in order to be mapped to the target directory.

Specifying the target directory

Set the Target server directory field to the file-system directory path being mapped. You can specify a relative path or a fully qualified path. You don't have to worry about using forward slashes on Unix and backslashes on Windows; both delimiters are valid on all platforms.

Remember that relative paths do not start with a drive letter or a slash. A relative path is always treated as relative to the Domino data directory; for example, the directory C:\Lotus\Domino\Data\Websites\pages can be specified just as Website\pages.

If you want to map a directory that isn't under the Domino data directory, specify a fully qualified path. The directory can be located on any file system that is accessible to the server; a good rule of thumb is that if you can get a directory listing from an operating-system command prompt, the directory can be accessed by Domino 6.

Directory rules also support Windows UNC paths (for example, \\server-2\pages). However, the server machine must already have the UNC share mapped with the proper level of access; the Domino 6 Web server can use existing UNC mappings but cannot initiate new ones.

Setting the access level
The final field in the Web Site Rule document is Access level. There are two choices:

- Read access allows browser users to read files from the directory. When a user requests a file in the directory, Domino 6 will send the contents of the file back to the browser. Read access is appropriate for directories that contain HTML files, graphic images, downloadable files, and other content.
- Execute access allows browser users to load and run CGI programs in the directory. Domino 6 will relay the output from the CGI program to the browser. Execute access is *only* appropriate for CGI directories.

It is critically important that you choose the correct access. Only directories that contain CGI programs should be marked for Execute access; all other directories should have Read access. If you specify the wrong access level, unexpected results will occur. If you mistakenly mark a CGI directory for Read access, then when a browser user sends a URL for a CGI program, Domino 6 will return the *source code* of the program rather than executing it. This could be a serious security breach!

You should also keep in mind that the access level is inherited by all subdirectories under the specified directory. For example, if you specify Execute access for a CGI directory, that rule will also give Execute access to all subdirectories under that directory.

Also, don't try to be tricky and mix CGI programs and regular files in the same directory. Since directory rules are mapped to URL paths, you might be tempted to put everything in one directory and create two rules with different URL paths, one with Read access and the other with Execute access; for example "/read-dir" and "/cgi-dir." This is a huge security hole! If a malicious user wants to download the source code of the CGI program http://dominoinc.com/cgi-dir/account.pl, all she has to do is switch the URL to http://dominoinc/read-dir/account.pl.

Keep in mind that directory rules cannot override file-access permissions enforced by the operating system. The Domino 6 server itself runs under a user account specified by the administrator, and accesses files and directories as that user. Therefore, all files accessed by the Web server must allow the appropriate level of access to the Domino 6 account—Execute access for CGI programs and Read access for everything else.

Example: Accessing files on another volume
Suppose you are setting up a Web site that includes a product catalog containing thousands of HTML files and JPEG images. Rather than maintain all of this material on the Web server machine, you want to place it on a large file server. The file server's disk subsystem is mapped on the Web server machine as drive F. You can access the directories on the disk by setting up a couple of directory rules. Here are the settings for the directory rule for HTML files:

Description:	HTML pages on file server
Type of rule:	Directory
Incoming URL pattern:	/catalog/pages
Target server directory:	f:\web-storage\html
Access level:	Read

Here they are for images:

Description:	Images and graphics on file server
Type of rule:	Directory
Incoming URL pattern:	/pic
Target server directory:	f:\web-storage\images
Access level:	Read

With these rules, the user's request for this URL:

http://dominoinc.com/shop/catalog/pages/trucks.htm

causes Domino 6 to return the file f:\web-storage\html\trucks.htm. If trucks.htm contains an image link for /pic/axle.jpg, Domino 6 will return f:\web-storage\images\axle.jpg.

We'll show more examples of directory rules in a few minutes, after discussing file-protection documents.

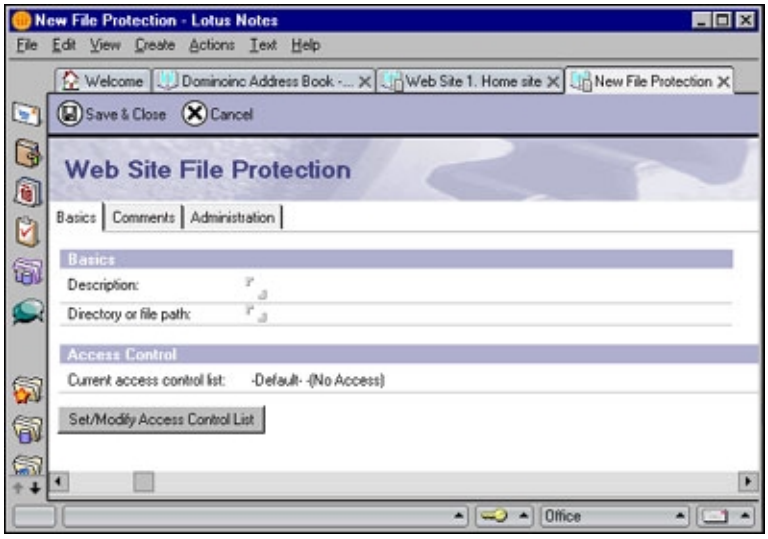
Setting up file-protection documents

A file-protection document associates an access control list (ACL) with a directory or file. Whenever a browser user sends a URL that attempts to read or execute a file, the path to the file is checked against the file-protection documents. If a match is found, the browser user's credentials are checked against the ACL. If the user is not in the ACL, access is denied and the server returns a 401 (Unauthorized) error.

File-protection documents only apply to HTTP requests that are mapped by directory rules, including both the built-in rules defined in Web Site documents as discussed above, and additional directory rules defined by the administrator. File-protection documents are *not* applied to Domino database requests (that is, any request that starts with a database file name or replica ID); access control for database requests is completely handled by the mechanisms native to databases such as the database ACL, Authors and Readers fields in documents, and so on.

There is one subtle restriction: you cannot apply file-protection to the default icons directory. Domino 6 will ignore a file-protection document that specifies the icons directory because that directory contains images that must always be available.

To create a file-protection document, open a Web Site document, click the Web Site action button, and choose Create File Protection. This opens an empty Web Site File Protection document:



In the Description field, you can enter any text that describes the document.

In the Directory or file path field, enter the path that is to be protected. The path follows the same rules as the Target server directory field in the Web Site Rule document for directory rules described **above**: the path can be absolute, relative to the data directory, or a Windows UNC path.

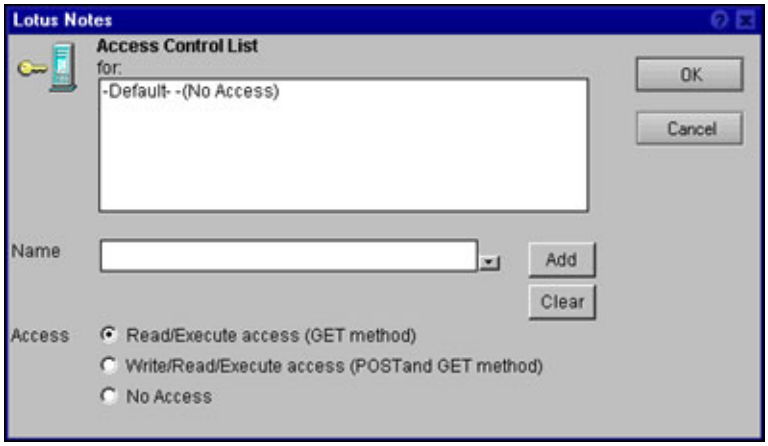
It is very important to realize that file-protection documents do not map directories to URLs. If you want to use a directory that is not under one of the server's default directories, you must first create a directory rule to map the directory to a URL path. Then you can create a file-protection document to protect that directory.

The protection defined for a directory is inherited by all of its subdirectories. If more than one file-protection document matches a requested path, the most-specific protection "wins." For example, if a user requests the file c:\website\pages\welcome.htm and there are two file-protections, one for c:\website and one for c:\website\pages, the protection for c:\website\pages is used. Therefore, if the user is denied access to c:\website but is allowed access to c:\website\pages, her request will succeed. You should avoid setting up conflicting protections like this,

however, because it can become a nightmare to administer.

A file-protection document can also specify a particular file. For example, if you have a directory of HTML pages, c:\website\pages, and you only need to protect the particular page salaries.htm, you can specify the file path c:\website\pages\salaries.htm in the file-protection document. Wildcards are *not* supported, so you must create a different file-protection document for every file you want to protect in this way. However, mixing secure and non-secure objects in the same directory can be an extremely dangerous practice that is prone to administrative error. (I would strongly suggest that you *not* try to protect individual files. Instead, you should move secure objects into their own directory and protect that entire directory.)

To modify the Access control list field, click the Set/Modify Access Control List button. This opens the following dialog box:



The top box contains the list of names in that ACL, but you can't directly edit the box. Instead, enter a name into the Name field, select the appropriate Access level (which we'll discuss in a minute), and click the Add button. The Name field also has a drop-down button to its right; clicking this button brings up a typical Names dialog box that lets you choose entries from your address books.

ACL entries follow the general Domino ACL rules. An entry can be an individual name or group, or the special identifiers -Default-, which applies to all users who don't match any other entry, and Anonymous, which is used for requests that don't have any user credentials at all. Name entries can be hierarchical (Joe Smith/dominoinc). There is one important difference between file-protection ACLs and other ACLs: wildcards are *not* supported in file-protection ACLs. For example, you *cannot* use */dominoinc to match all users in the dominoinc organization.

You must assign an access level to every ACL entry. The access level indicates what HTTP methods will be allowed on requests that access the protected directory. There are three access levels:

- Read/Execute access only allows requests that use the GET and HEAD methods (the HEAD method acts just like GET but tells the server to only return the response headers and not the response body). The requests can be intended to read a file (http://dominoinc.com/welcome.htm) or execute a CGI program (http://dominoinc.com/cgi-bin/search.pl?product=widget). Keep in mind that if the protected path has been mapped by a directory rule, then the directory rule further restricts the request. For example, if the directory rule specifies Read access, then a user cannot execute CGI programs from that directory even if the file-protection document gives the user Read/Execute access.
- Write/Read/Execute access allows the POST method in addition to GET and HEAD. The Domino 6 Web server does not support uploading files directly to the file system (it does support uploading files into databases), so the only common use of the POST method in a file-system request is to send data (such as a filled-out form) to a CGI program.
- No Access means just what you think. Any request that targets a file in the protected directory is rejected with a 401 (Unauthorized) error.

When you create a new file-protection document, the default ACL has the -Default- entry set to No Access; in other words, *nobody* can access the directory for any request. For greatest security, you should keep the -Default- entry in the list and add specific entries for users that should be allowed access. (To delete an entry from the list, highlight the entry and click the Clear button.)

If you are protecting CGI programs, keep in mind that Domino 6 only enforces file-protection for the CGI program

file itself. If the CGI program accesses other files while it runs, Domino 6 is not involved and cannot verify file-protection for those files.

Putting it all together

Now let's put what we've learned into practice by taking a look at some complete examples of accessing and protecting directories.

Example 1: Protecting the default CGI directory

As we discussed above, the Web server always uses an internal rule for the default CGI directory, so you don't need to create a directory rule. However, you can still restrict access to programs in that directory by creating a file-protection document. Because the CGI directory is located under the Domino data directory, you can use a relative path in the file-protection document. If some of the CGI programs process forms, you will need to allow POST access. Here are the settings for a sample file-protection document for the default CGI directory that restricts access to the RegisteredUsers group:

Description:	Only registered users can run CGI programs
Directory or file path:	domino\cgi-bin
Access control list:	-Default- -(No Access) RegisteredUsers -(POST and GET)

Example 2: Enabling and protecting a directory outside the data directory

In this scenario, your Web designers maintain a large number of HTML pages in their own directory tree. You need to access one of the directories in this tree in a Web application. Since the target directory is outside the Domino data directory hierarchy, you need to create a directory rule:

Description:	HTML pages for shopping catalog
Type of rule:	Directory
Incoming URL pattern:	/catalog
Target server directory:	c:\external\website\catpages
Access level:	Read

In addition, you want to allow only two groups of Web users to access this directory: registered external users and Web site administrators (who have hierarchical IDs). Therefore, you create a file-protection document with the following settings:

Description:	Shopping catalog pages
Directory or file path:	c:\external\website\catpages
Access control list:	-Default- -(No Access) RegisteredUsers -(GET) jsmith/admin/dominoinc -(GET) rjohnson/admin/dominoinc -(GET)

With these documents in place, an authorized user can access a catalog page with a request like this:

http://dominoinc.com/catalog/books.htm

and it will map to c:\external\website\catpages\books.htm.

Example 3: Windows UNC path

Suppose your server machine has mapped a UNC share that contains additional CGI programs. To allow browser users to access the directory, you need to map it with a directory rule:

Description:	CGI directory on UNC share
Type of rule:	Directory
Incoming URL pattern:	/scripts

Target server directory:	\\bighost\drive-c\web\cgi
Access level:	Execute

You want to allow all authenticated users to run these CGI programs, so you create the following file-protection document. This is probably the only common scenario where you would set -Default- to something other than No Access:

Description:	Protection for UNC share
Directory or file path:	\\bighost\drive-c\web\cgi
Access control list:	-Default- -(GET) Anonymous -(No Access)

Now, any authenticated user can run a CGI program with a request like this:

http://dominoinc.com/scripts/buy.pl

and it will map to \\bighost\drive-c\web\cgi\buy.pl.

How secure are file-protection documents?

If you are thinking about using file-protection documents, it is extremely important for you to understand that file-protection ACLs are *not* as secure as database ACLs. The reason is that in the Domino database security model, access control is a property of the database objects themselves; but for file-protection documents, access control is associated with a *path* to the resource, not with the resource itself. Domino cannot place access control directly on a file-system resource because the operating system owns the resource, not Domino. This is the same reason that directory rules cannot override operating system file-access permissions, as we mentioned previously.

Access to objects in a Domino database is completely controlled by Domino. As far as the operating system is concerned, a Domino database is just a file, nothing more. However, Domino knows that a database file is actually a highly complex container of various objects. Only Domino knows how to interpret the contents of a database file. Therefore, Domino is completely in control of access to database objects, and also of the paths to database objects. For example, suppose that a particular document has a Readers list on it that only allows access to a group called GoodGuys. The Domino URL syntax allows access to objects by name and by UNID, so assume that this document can be requested by any of these URLs (which doesn't even exhaust all possibilities):

- http://dominoinc.com/products.nsf/byname/widget?opendocument
- http://dominoinc.com/products.nsf/b86a878b9ae5ee0d85256ad8000813a6?opendocument
- http://dominoinc.com/products.nsf/8a6d147cf55a7fd385256658007aacf1/widget?opendocument
- http://dominoinc.com/8fa345bc02abd5f7/0/b86a878b9ae5ee0d85256ad8000813a6?opendocument

Suppose that Joe BadGuy, who isn't in the GoodGuy group, wants to read that document. He tries all of the URLs above. Will he succeed? No! Because Domino interprets the URLs, the server realizes that all of these URLs point to the same document; and since the Readers list is a property of the document itself, Joe BadGuy is rejected.

The situation is completely different with Web site file-protection. Domino doesn't own files and directories, the operating system does. Therefore, Domino *cannot* place access control directly on file-system objects. All that Domino can do is attempt to control the path to file-system objects.

For example, suppose you create a directory rule that maps the URL path /shop/* to the directory c:\website\shopping. You want to limit access to that directory, so you create a file-protection document that specifies an ACL for c:\website\shopping. Now, if a user sends the URL http://dominoinc.com/shop/welcome.html, Domino uses the directory rule to determine that the desired file is actually c:\website\shopping\welcome.html. Domino then checks its list of file-protection documents, finds one for c:\website\shopping, and checks the user's credentials against the ACL. The key point here is that the *path* to the file welcome.html matches the *path* specified in a file-protection document, so the ACL is applied.

The problem: alternate paths to file-system objects

But there is a serious problem with this scheme. As we mentioned, when you request an object from a Domino database, it doesn't matter how you specify the path to the object because the ACL is a property of the object. But with Web site file-protection, the ACL is a property of the path. Therefore, if more than one path can be found to the object, a malicious user can evade the file-protection. And because the operating system, not Domino, is

ultimately in charge of both supplying and interpreting paths, it is *impossible* for Domino (or any similar Web server) to comprehensively determine how many alternate paths exist at any given point in time. Domino 6 does its best to compensate for alternative paths, but it is always possible that another path may exist that Domino 6 knows nothing about.

The most familiar example of an alternate path involves path-navigation sequences. On both Windows and Unix platforms, a single dot (.) in a path represents the current directory level, and double dots (..) represent the parent level. Suppose you set up the file-protection document for the default cgi-bin directory that we showed above like this:

Description:	Only registered users can run CGI programs
Directory or file path:	domino\cgi-bin
Access control list:	- Default- -(No Access) RegisteredUsers -(POST and GET)

If a user wants to run the CGI program account.pl, you expect the user to send the URL:

http://dominoinc.com/cgi-bin/account.pl

which Domino 6 interprets as pointing to the file:

C:\Lotus\Domino\Data\domino\cgi-bin\account.pl

This path clearly matches the file-protection rule, so Domino 6 will check that the user is a member of the RegisteredUser group before running the program.

But suppose a malicious user wants to get access to account.pl. One method he might try is to use path-navigation sequences in the URL (historically, many Web applications have proven to be vulnerable to this type of attack). Therefore, he sends the URL:

http://dominoinc.com/html/../cgi-bin/./account.pl

hoping that Domino 6 will interpret the path as:

C:\Lotus\Domino\Data\domino\html\..\cgi-bin\.\account.pl

which points to the same file, but does not literally match the file-protection rule. Will this work? Good try, but it won't work because Domino 6 immediately simplifies all path-navigation sequences when a URL is received, so the incoming URL path /html/../cgi-bin/./account.pl is simplified to /cgi-bin/account.pl before any processing on the request is done (for details, see the *LDD Today* article "**Building Web applications in Domino 6: Web site rules** ").

Another example of an alternate path is specific to Windows. In order to maintain compatibility with the original MS-DOS operating system, Windows automatically creates an MS-DOS alias for a file whose file name is greater than eight characters or whose extension is greater than three characters. For example, the file c:\website\pages\productlist.htm is given the alias c:\website\pages\produc~1.htm. So, if you create a file-protection document for c:\website\pages\productlist.htm, can an attacker evade it by requesting the URL http://dominoinc/pages/produc~1.htm? Once again, it's a good try but Domino 6 won't be fooled. On Windows platforms, Domino 6 automatically checks your file-protection documents for possible MS-DOS aliases.

These examples of alternate paths are well-known and documented, but unfortunately there are occasional reports of undocumented path syntaxes or strange operating system parsing behavior. This is a real problem for the Domino 6 file-protection feature, and indeed any Web server or application that relies on the operating system to interpret paths. This problem has become more acute in recent times because OS vendors are releasing patches more and more frequently, and any OS patch could introduce a new path syntax vulnerability. Therefore, the best that we can do is to say that the latest releases of Domino 6 and R5 compensate for all *known* path vulnerabilities at the time of release. At any time after release, a new OS vulnerability could be discovered that would allow attackers to evade Domino 6 file-protection. Even if the OS vendor issues a patch to fix the vulnerability, you must still rely on the site administrator to actually download the patch.

Better ways to secure file resources

What are the alternatives to file-protection for securing file resources? In R5 you could import your HTML and image files into database pages or documents, either directly or as file attachments. This lets you secure the resources by using the database ACL and/or document Readers fields.

Domino 6 has an even better solution: we have added new database object types for File, Image, and Style Sheet resources. You can create these objects from disk files by using the Domino Designer or WebDAV-enabled clients. The objects can be accessed in a Web application by using new URL commands such as OpenFileResource. Since the objects reside in a database, you can control access to them with the full power of the database ACL. For example, if you want to control access to a directory of HTML files, rather than using file-protection documents, you can move the files into a database created just for this purpose. If you use a WebDAV-enabled client such as Windows File Explorer, you can move the entire directory into the database with a single drag-and-drop operation. You can then set the database ACL to give appropriate access to your Web users. If you are using a cluster of Web servers, you get another huge benefit, because the objects participate in database replication, which automatically keeps the resources synchronized across the entire cluster.

The bottom line

To sum all this up, here are our recommendations for the safe use of the Domino 6 file-protection feature:

File protection documents are an adequate defense against casual snooping, such as the curious visitor who wanders around your site just to see what he or she can dig up. They are also a sufficient protection against legitimate search engines and other well-behaved Web crawlers. All of these clients will use common, documented path syntaxes.

However, file-protection documents are *not* a reliable safeguard against an attacker who is able to exploit operating system vulnerabilities or unpatched systems. Therefore, all of the sensitive data on your site (such as credit card numbers, competitive information, personnel files, and so on) should be stored in Domino databases with appropriate ACLs. Do *not* store truly sensitive information in the file system, and do *not* rely on file-protection documents to protect such information.

ABOUT THE AUTHOR

John Chamberlain is a Senior Software Engineer on the Domino 6 Web Server team.

