SECURITY

Image Authentication for Lotus Notes

By John Duggan

Authentication is the process of identifying yourself so you have access to the proper files on your company's network. The default process of authentication for the Lotus Notes client is to type in a text password. This isn't the best process for authentication, but it is the most convenient because it is quick, easy to use, and simple to administer when something goes wrong.

Organizations for whom security is absolutely critical don't consider text passwords a secure form of authentication. They usually prefer alternative schemes such as smart cards, voice recognition, or even biometrics (measuring facial geometry, iris scans, hand silhouettes, and so on).

Fortunately, Lotus Notes and Domino are flexible when it comes to extending the core functionality. So, it's possible to implement many types of authentication within Lotus Notes.

Because you probably don't have a SmartCard Development Kit handy under your desk, I'll use image authentication as an example of how to implement a different authentication process for the Lotus Notes client.

Image authentication

Image authentication is a process whereby users authenticate themselves by identifying a chosen set of pictures from a large collection. You can think of it as image recollection. If the user selects the wrong set of images, the user's access is denied; otherwise, it's accepted.

There are many ways of implementing image authentication. One approach is to ask the user to select a single picture from a group of nine and repeat the process five times with different pictures.

An example of the dialog using this approach is shown in figure 1.

Why images?

There are several reasons for using images rather than text:

- Users can't write down images easily. Many users compromise security by writing down passwords and posting them near their workstation.
- Humans have a natural ability from birth to identify images, thus our recollection of images is better than text. (We can recognize a face more easily than a name.)
- Images can't be communicated easily to other individuals.
- Text passwords can be cracked by brute force, whereas images are trickier to crack.

This process isn't perfect, and has its own problems:

Resistance to change (or comfort factor)—Individuals like to write their passwords on paper somewhere, and even share their passwords. (Yes, this does defeat the purpose of authentication!)

Implementation—The images must be carefully selected, such that they should be similar or all different. In other words, if you had eight images of skies and a single image of a tree, there is a very high chance the user will almost always pick the tree because it has an obvious characteristic of being distinctly different, compared to the other images. Unfortunately, it isn't easy to decide upon a perfect set of images.

If you'd like to try a Web implementation with human faces, see http://www.passfaces.com.

If you'd like to read further about image authentication, see the paper http://paris.cs.berkeley.edu/~perrig/projects/usenix2000/. This paper highlights the different types of "visual" authentication that have been studied: (http://paris.cs.berkeley.edu/~perrig/projects/usenix2000/node19.html)

Subscriber Download

- The code used for image authentication. http://Advisor.com/Article/DUGGJ03

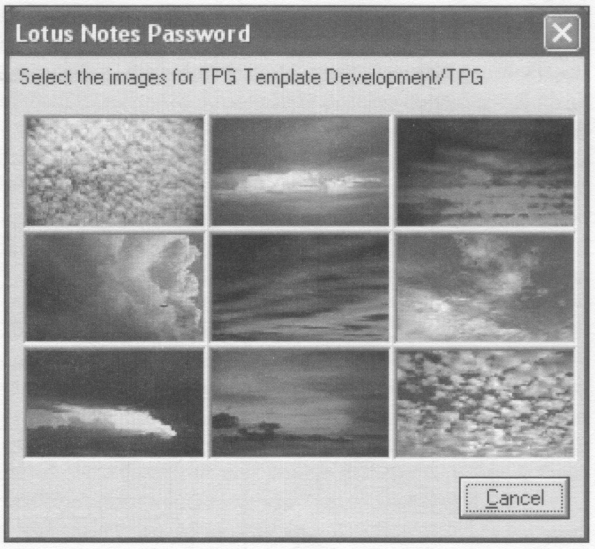

***Figure 1:* Sample image authentication**—Notes client asking the user to identify a selection of images.

The paper concludes that random art pictures are the best images to use, because they're fairly discreet (http://gs2.sp.cs.cmu.edu/art/random/).

Before I discuss the implementation, it will help you understand Notes Object Services.

Notes Object Services

Notes Object Services (NOS) are a library of C/C++ functions grouped into distinct service areas that lie at the heart of Notes. This library of functions implements all the services that create and maintain Notes databases.

You can access the functions using the Notes application programming interface (API).

You can find an in-depth overview of NOS in the white paper at "Inside Notes: The Architecture of Notes and the Domino Server." (http://www.notes.net/notesua.nsf/6cbb500587e5dd44852566c2004e94ec/ec73cbf1c6392ba385256856005bd224?OpenDocument)

Figure 2 gives a high-level overview of how NOS fit in with Lotus Notes and Lotus Domino.

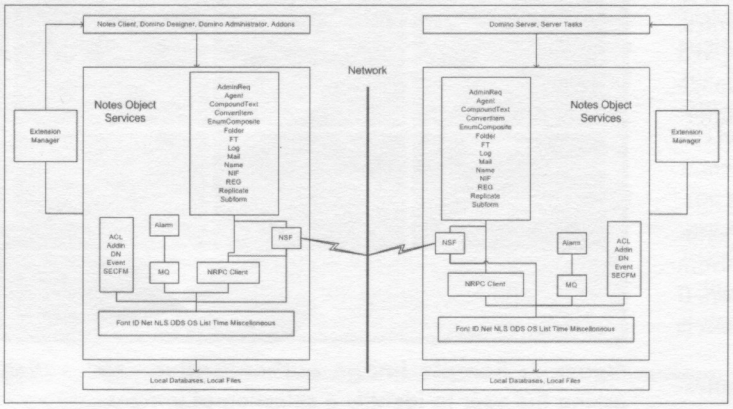

***Figure 2:* Notes Object Services**—This overview shows how NOS fits with Notes and Domino.

Extension manager

An extension manager lets you call a routine (or callback) within an executable program library before or after Domino (or Notes) performs selected functions within the NOS. In other words, this means you can extend the Notes and Domino internal operations.

Implementing the executable program library requires the Lotus C API Toolkit for Domino and Notes, available from the Lotus Web site. The toolkit is a collection of include files, samples, libraries, and documentation.

On the Windows platform, you use Microsoft Visual C/C++ to compile the applications that can call the NOS C functions directly.

The Lotus C API Toolkit doesn't provide access to all the NOS functions, and you can only use the extension manager to extend a subset of those functions. The header file "extmgr.h," included with the toolkit, defines which functions can be extended.

The toolkit is powerful, and it can perform operations you can't achieve within Notes or Domino itself.

What can you do with the extension manager?

Anti-viral products that monitor outgoing and ingoing e-mail typically use extension managers. You can use extension managers for single sign-on for NT or OS/400. Or, you can use them to provide an additional layer of security to deny access to a particular database depending upon a setting outside of Notes.

The C API toolkit includes example extension managers that help with tasks such as handling replication conflicts, converting a non-Domino database into a Domino database on creation, modifying the body of all e-mail messages, and even replacing the Password dialog. Even the AdminP server task relies on the extension manager to perform its own tasks.

You can extend more than 100 events, ranging from calendar and scheduling, full text, opening Notes, and running agents to admin process requests. The NOS Extension Manager (EM) Service makes it easy to implement extensions for the Lotus Notes client or Lotus Domino server.

Implement image authentication for the Windows Notes client

Creating a Windows extension requires the latest Notes API toolkit and Microsoft Visual C/C++ version 6.0. The example in this article shows you how to create a standard Windows Dynamic Link Library (DLL) to hold all the routines for the image authentication.

You must implement three functions for the application extension to work:

- Register the extension
- Unregister the extension
- Initiate the extension

Register the extension

The application must register its extensions with the extension manager and indicate which NOS functions it's extending and whether it should be called before or after the NOS function is called.

Note: Not all routines can be extended before and after. For example, using GETPASSWORD and EM_REG_AFTER has no effect.

If you use an extension manager that calls itself recursively, you have to retrieve a recursion ID to prevent Domino or Notes from crashing (for example, if you call NSFBOpen within an extension to NSFBOpen).

For image authentication, you're only interested in two events: After EM_SETPASSWORD is called and before EM_GETPASSWORD is called.

The example DLL must contain the following function:

```
DLL_EXPORT_PREFIX STATUS LNPUBLIC DLL_EXPORT_INFIX
    MainEntryPoint (void)
```

Within this function, you must include a call to EMRegister telling Notes the event and the NOS function it's extending. Here are the calls from the sample program:

```
status=EMRegister(EM_GETPASSWORD,EM_REG_BEFORE,
            ExtHandler,0,&hHandler);
```

```
status=EMRegister

(EM_SETPASSWORD,EM_AFTER,ExtHandler,0,&hHandler2);
```

Unregister the extension

When an application is finished with the extension manager, you must use EMDeregister to remove all the registered routines.

The code for implementing a function that handles the clear extension event is straightforward:

```
DLL_EXPORT_PREFIX STATUS LNPUBLIC DLL_EXPORT_INFIX
            ExtClear (void)

if (0 != hHandler)
{
  status = EMDeregister (hHandler);
  status = EMDeregister (hHandler2);
}
else
  status = NOERROR;
```

The routine must use the following declaration:

```
STATUS LNCALLBACK ExtMgrCallback (EMRECORD far
*pExtRec);
```

A single argument acts as a pointer to an EMRECORD structure. This structure stores the notification ID, the type of notification, error code, and another pointer to the arguments passed to the NOS function. It's possible to change the arguments and pass the changed arguments to the NOS function.

Now you've completed the code for registration. The next step is to do some processing.

Before users can specify their sets of images, they must enter their existing text password. The sample extension provides a standard dialog for the text password (figure 3). After the text password has been accepted and the images are selected, the ID will be protected by the image-based authentication.

Notice this password prompt doesn't have any hieroglyphics, compared to the standard Notes prompt (figure 4).

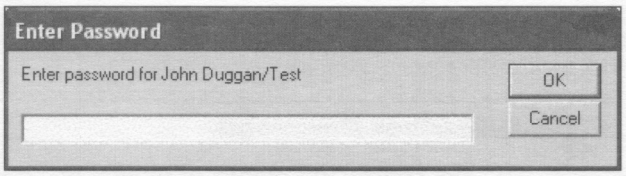

***Figure 3:* Password prompt from the sample extension manager**—Users must enter their password before selecting their images.

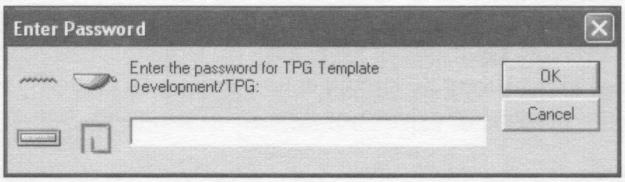

***Figure 4:* Egyptian encryption**—The familiar Notes prompt with hieroglyphics.

Why don't you use hieroglyphics within your own application? You can't. This is intentional to avoid the use of Trojan horses. Unless you know the full algorithm to reproduce the same hieroglyphics, you can't fool the user into thinking you haven't created a password capture program. You might be able to duplicate something similar, but users often subconsciously recognize whether the pattern has changed, thus knowing their password process has been tampered with.

After the initial password has been accepted, the user is now presented with a set of images. He chooses his required images, and that will be his new form of authentication. A variable within the Notes.ini is updated to indicate the password has now changed to "visual." In a live environment with multiple IDs on a single machine, you can't switch between text and visual passwords because the DLL assumes all the IDs are now image-based. It's possible to cater for text and images, but that's beyond the scope of this article. If you only have a single ID on a Notes client, you have nothing to worry about!

How are the chosen images stored? Every bitmap within a Windows application program has a unique resource ID. The collection of resource IDs are put into a string and this string is stored as the "text" password.

For example, say an authentication scheme requires the user to select two images. If the user selects the 5th and the 20th images, the password is stored as "0520." This value is encrypted and passed to the NOS password handler. The sample program does not do any encryption; you must implement your own encryption routine for additional security. Bruce Schneier's book, Applied Cryptography: Protocols, Algorithms and Source in C (John Wiley & Sons, 1995) provides a good introduction to encryption.

The other forms of security, such as biometrics, use the same approach. The process generates a "key" you can convert to text and store within the Notes ID file as the password.

This routine now handles the EM_SETPASSWORD. Next, you have to set up the EM_GETPASSWORD.

When a user is asked for his password, the program checks whether it is still text or has been set to images and shows the appropriate dialog. The following "pseudo code" shows the structure of the program. By "pseudo code," I mean I've shortened the original code for the article. The code below won't compile, but it shows the process:

```
if (Notes.ini variable is set)
{
answer = DialogBox(Show images dialog);

  switch (answer)
  {
   case DLG_ANSWER_OK:
     copy 'key' into the password field.
     return (ERR_BSAFE_EXTERNAL_PASSWORD);
   case DLG_ANSWER_CANCEL:
     return (ERR_BSAFE_USER_ABORT);
   default:
     return (ERR_BSAFE_PASSWORD_REQUIRED);
}
else
{
answer = DialogBox(Show text password dialog)

switch (answer)
{
```

```
    case DLG_ANSWER_OK:
      return (ERR_BSAFE_EXTERNAL_PASSWORD);
    case DLG_ANSWER_CANCEL:
      return (ERR_BSAFE_USER_ABORT);
    default:
      return (ERR_BSAFE_PASSWORD_REQUIRED);
}
```

The return code from the function controls what processing the NOS function performs. These are the special return codes for passwords:

ERR_BSAFE_EXTERNAL_PASSWORD—The user has supplied the password.

ERR_BSAFE_USER_ABORT—The user cancelled the request.

ERR_BSAFE_PASSWORD_REQUIRED—The user didn't specify a password.

ERR_EM_CONTINUE—Perform the normal task (e.g., show the standard dialog).

If ERR_BSAFE_EXTERNAL_PASSWORD is supplied, the extension manager NOS checks whether the password is valid and accepts or revokes the access.

That's it! Obviously, there is some extra code to handle the dialogs, but these are standard Windows API routines. If you want to find out how these dialogs work, I recommend you read one of the many Windows API books.

Initiate the registration

After you've built and compiled the application, the next step is to place it into the Notes directory.

You must tell Notes the executable program library exists. You do so by adding an extra line to the Notes.ini:

```
EXTMGR_ADDINS=imgpwd
```

If there's more than one entry, you must separate them with commas. For the Windows platform, the application filename must be defined in the format of "nextmngr.dll," where "n" refers to the Windows platform. The first character varies for each platform.

Finally, start Notes and you now have a new process for user authentication.

Summing up

Hopefully, from this article, you've seen that extension managers are easy to write and implement. Also, implementing a totally different authentication scheme is surprisingly straightforward.

I recommend you read the Notes C API documentation regarding the security issues on extending EM_GETPASSWORD.

Be aware the API documentation says some high-level database services depend on the extended routines, which may affect the outcome of certain functions. Note that exactly which functions can vary, although it would be the server tasks, such as AdminP, Monitoring, Stats, and so on. The main problem is Notes crashing unpredictably. However, in the times I've developed extension managers, I've never come across this problem.

The source code accompanying this article has been deliberately simplified to keep the example easily understandable. You shouldn't use it in a live environment. It doesn't provide any encryption, doesn't cope with multiple IDs, and doesn't handle changing location documents.

Finally, always perform a backup of your ID file before doing any testing or experimentation! ■