Security Guard for the Internet

by

Resource Development Associates

How it works:
Since running the Corel Web OCX in Paradox means you are running all your web users in a single Paradox session, conferring table level password rights to anyone would give all web users access to tables under the least restrictive rights in the running session. Fortunately, the BDE can handle many sessions from a single instance of Paradox. Each session can have its own configuration including different passwords and, therefore, different access rights to tables based on those passwords. The trick is to start multiple BDE sessions and keep them separate, keep the users sorted and in their own session and not let rights bubble up to the session running the server form itself. This is what Security Guard for the Internet accomplishes for you while hiding the complexity from you.

Also, since users can not see your Paradox GUI they would not know they are being prompted for a table password. Such a prompt would stop your server OCX. So using password protected tables in the OCX means you need a way to issue the table passwords. As with Paradox on a net you do not want users to have real table passwords since you would then have to restructure your tables and change passwords when users left your organization if you wished to maintain security. So a means of creating and managing intermediary user passwords which issue table passwords is important. In addition, you may wish to add Secure Socket Layers (SSL) for transmission of data on the Internet. Information on how to do this is also available on our web site on the Paradox resources page. Security Guard for the Internet is fully compatible with SSL.

Security Data:
Security data should be installed in a separate directory on the host system. Your system must have at least change level access to the data directory as users are able to modify their passwords from the Internet. The Security Data consists of the files:

Pwords.db
Pwords.px
Pwords.val
Rightslst.db
Rightslst.mb
Rightslst.px
Rightslst.val
Rights.db
Rights.px
Rights.val
Security.db
Security.px

Security.val
Security.xg0
Security.yg0

All access to these files in Security Guard for the Internet is via a tcursor so the data is never available to end users. Staff granted access in the Management Console to the security data can only be granted Insert and Delete access. It is recommended that few staff be granted this access. Initially, one user exists in the database. The User name is Newuser and the password is Newuser (case is important for both). You are strongly encouraged to create a new account for the administrator and grant the administrator access to the Security Guard system and then delete this Newuser account as soon as you begin using the system. Otherwise your system will remain open to anyone who has read this document. The uses of the various files are explained in the Security Guard documentation file (Sgdoc.pdf) which you can download from our web site if you wish. If you have the LAN version of Security Guard you already have these tables set up and should not change the existing ones. You can skip to the section on Running the Server.

To set Security Guard up you need to create table level passwords in your table structure as you would for any password protection at the table level. You then create system names for groups of tables that share passwords for the same rights. You must, as always, be sure not to use one password to confer one type of rights in one table and another type of rights in another table unless that is what you specifically intend to do. Once a password is issued to a session it will confer rights at the level defined by any table that has that password.

Once your tables have passwords and you have defined groups of tables as systems you then put those system names, the right level that confers and the password into the Security Guard Password table. As with all Security Guard's internal tables these are already password protected.

You then create users in the Security Guard Security table. These users are listed by a user name, first and last names, middle initial if wanted, the user's personal password (which is not a table password), last password modification date and expiration interval in days.

Once users exist and systems exist you fill the Security Guard Rights table with combinations of the User name, System name and Rights level, one record per combination for every system you wish to grant that user access to on your web site or LAN (the same Security Guard tables should be used for both if you are running both). All of these tasks except setting up the actual passwords in your own application tables can be accomplished simply through the Security Guard for the Internet Management Console over the Internet or your Intranet.

Running the Server:
In the web environment you run the SGServer.fsl in Paradox to run your site. (For details on setting this up see the Paradox Internet help or Tony McGuire's Paradox Web Server lessons which are available on RDA's Paradox Resources page on our web site.) When a user accesses your site they are presented with a startup screen that tells them they must log in to proceed and asking them if they wish to do so. If they submit the login they are presented with the standard browser User and Password challenge. When they submit it is evaluated and if the presented user

name and password are in your Security Guard system then a session is started for them. They are then sent back a form which lists the systems they have been granted access to (based on the rights you have assigned them) and they can pick one. Whichever they pick will be sent back as a string which you can handle to move them in to your systems. There are a few rules which you need to follow to successfully use Security Guard for the Web. These are listed below. The included demo application demonstrates how to plug in an application to Security Guard for the Internet so you should take some time looking over the code in there as well as these instructions. You may wish to use the demoapp.lsl as the base for your own applications. If so you should use a copy. There are also several useful code segments in the demo applications which you will probably want to use for all your applications. Pay special attention to the sggetfieldvals and sgerrorproc functions which are quite generic and very useful. You can see how they are applied in the code in the sgdemochoice method. All these methods are in the source code in the demoapp.lsl.

Aliases:
Aliases must be created for Security Guard to function. Normal path rules must be followed for accessing the data and the data must be controlled by the same .Net file for all users as with any shared Paradox data. If you have the LAN version of Security Guard use the existing aliases you have set up.

The Alias to the Security Data must be :SecurityData:. It must exist in the IDAPI.CFG file the copy of Paradox which is running the SGServer.fsl is using. The security data files listed above must be in that alias. In addition, to run the demo application you will also need to place the following files there.
demoapp.db
demoapp.px
demoapp.val

The alias SecurityWeb will be created in the same directory as the server if it does not already exist. The following files must be in the SecurityWeb alias:
demoapp.ldl
demoapp.lsl
server.db
server.px
sghtml.db
sghtml.fam
sghtml.mb
sghtml.px
sghtml.tv
sgmcweb.ldl
sgweb.ldl

You must also put the following files in the folder \PAGES below the folder in which you have the sgserver.fsl located:
rdalogo1.gif

sgcons.htm
sglogin.htm
sgpw.gif
sgpwldg.htm
sgtimout.htm

It is more secure to place the SecurityWeb alias in a directory above or in a different path from the Server since it will be unable to be accessed directly from the web. The other Security Guard aliases will also need to exist. SecurityWeb can be pointing to the same directory as SecurityMain if you are using the LAN version of Security Guard. SecurityWeb is also where the Server.db for Security Guard for the Internet must be located. You can add your Server.db to the one that comes with Security Guard or add the one that comes with Security Guard to yours. You could, if you wanted to, create separate tcursors to separate server.db files but this will add an unnecessary level of complexity into your applications. The sghtml.db also must be located in the SecurityWeb Alias. You can also add your templates to this table or have separate HTML template tables for your applications.

To modify the path to the SecurityWeb alias either edit the line that sets it to the server path in the DoStartup of the web OCX in the SGSERVER.FSL form or add a permanent alias. The line is currently strAliasPath = strPath. You can replace strPath with the actual path you wish to use. Be sure to use double \’s if needed.

If you wish to add project aliases they must be added here as well. Security Guard for the Web will use all Public and Project Aliases in its processing of code. But only project aliases created in the SGSERVER.FSL session will be visible when they are needed. User rights and managing user access can only be handled from aliases which exist as public aliases in your idapi.cfg file because of how database variables are assigned from the aliases. This is only important for aliases holding your data. You must have aliases to your tables and those aliases will need to be passed in calling the session and database variables as you are using Security Guard to provide access to your systems. This will be described in more detail below.

Passwords:
As with the LAN versions of Security Guard, the user can change their own password if it has expired or about to expire. Users are notified if they present valid but expired credentials and are told they must change their password (in fact they can reinitialize the same password). If a user is within 7 days of their password expiring they are prompted to change it as well. You set the length of time a password lasts before it must be changed. It can be any number of days up to 999 or you can leave it blank which will mean it never expires. You can disable a user account by setting the expiration to 0 or by using the Disable User Account option on the web console which does the same thing. Disabled accounts can be enabled again by resetting the expiration time or by using the Enable Disabled User Account option on the web console.

Access Granted:
Once a user has been granted access, all the systems that they have access to will have the user’s rights level conferred to their session variable in the running version of Paradox. If you unload the

library you will kill the session variables which will remove all users' current access. They will need to log in again at that point. This is unavoidable as the security info must be maintained in memory to make it inaccessible to hackers and to manage the multiple session connections to the BDE.

This does mean that if you plan to bring down your libraries to make changes you should start another Paradox instance, shut down the SGSERVER.FSL in the current Paradox instance and immediately start a copy of the SGServer.fsl in the second instance. Also, if you have more than the Security Guard Libraries running, which you generally will, you may have to shut down the instance of Paradox to clear all of the libraries. This is because not only does the SGSERVER.FSL open libraries, but the sgweb.ldl and sgmcweb.ldl open each other and open your application's libraries and your libraries will open the Security Guard libraries as well. This interconnected opening is required to protect the security of your password tables and because there is no way for us to know which libraries, nor how many libraries, you will be adding to Security Guard for the Internet so we use a scheme similar to the one used by the web server OCX form.

Once the user has rights granted, they will have a record placed in the in memory tcursor which you can access to check their login time stamps and to see who is logged in. Unloading the libraries will kill the tcursor as well. While creating the session and issuing the passwords for the system to the session, Security Guard for the Internet will also list the systems to which the user has been granted rights. These will be listed just as you have listed them in Security Guard. The Log Out option will also be added to the list as the default option.

These will be presented to the user and they are allowed to pick the one they want. Whichever application they pick will be returned as a string. Log Out is handled internally by Security Guard for the Internet. Any other response string will be the fetchrequest with the system name returned which will be the startup screen for the application to be called. This template to start any system will have to be added to the sghtml.db. Those templates will have to include the response action that will then be called from the server table. What gets called is actually:

```
strResponse = string(Fetchtemplate(strSystemRequest))

Response.ResultString = strResponse
return TRUE
```

If Security Guard itself is called the Management Console startup is already in the sghtml.db. If you leave the demo application in place this will also be an option for anyone you grant rights to it. You can use the demo application form in the sghtml.db as a model for how to call your applications.

Library Methods you can access:
You have direct access to 8 methods in the sgweb.ldl and sgmcweb.ldl. You can, and should, use them from your application libraries to verify users and sessions. In some cases you must use them so your application will function with Security Guard for the Internet. It is suggested you use the

included demoapp.lsl as a model. There are also some library methods such as sggetfieldvals which you can use in your libraries or modify if you wish. You add them to your libraries as you would any other library methods. In the Uses at the library level add:

```
Uses ObjectPal
sgverifyaccess(strName string) string
sgisuserloggedin(strUserName string) logical
sglastaccessstamp(siTimecheck smallint,strUserName string ) String
sglogout(strAuthName string, strAuthPW string) string
sgreturndbvars(strLibCalling String, strAliasCalling String, strPassedName string) logical
sgreturnsessionvars(strLibCalling String, strPassedName string) logical
sgsetsessionpws(strLibName String, strPassedName string) logical
openapplibrary(strLibName String, strLibAlias string ) Logical
endUses
```

In the Var at the library level add:

```
Var
loLibSGopen         logical
loLibSGMCopen       logical
lbSecurityGuard     Library
lbSecurityGuardMC   Library
dynDBs              Dynarray[] database
dynSessions         Dynarray[] session
tcTemplate          tcursor
tcSectable          tcursor
strPath             String
strName             string
strPW               string
strErr              string
Endvar
```

In the library’s open add:

```
method open(var eventInfo Event)
var
dynFile             DynArray[] String
endVar

loLibSGopen   = false
loLibSGMCopen = false

splitFullFileName(getFileName(), dynFile)
strPath = dynFile["DRIVE"] + dynFile["PATH"]

        if (NOT tcTemplate.open(":SecurityWeb:sghtml.DB")) then
```

```
            eventInfo.setErrorCode(CANNOTARRIVE)
            endIf

if not lbSecurityGuard.isassigned() then
   if not lbSecurityGuard.Open(":SecurityWeb:sgweb.ldl") then
            eventInfo.setErrorCode(CANNOTARRIVE)
   endIf
endif

if not lbSecurityGuardMC.isassigned() then
            if not lbSecurityGuardMC.Open(":SecurityWeb:sgmcweb.ldl") then
            eventInfo.setErrorCode(CANNOTARRIVE)
   endif
endIf
endMethod
```

You need to open these libraries to make calls to them and both must be present as they make calls to each other and the opening login screen depends upon them. The logical variables are used in the first method you call in your library. They must be initialized to false on opening.

You should also add the following to your library’s close method:

```
method close(var eventInfo Event)

if lbSecurityGuard.isassigned() then
lbSecurityGuard.close()
endif

if lbSecurityGuardMC.isassigned() then
lbSecurityGuardMC.close()
endif

endMethod
```

Of course you will also need the Const, fetchtemplate and handlerequest methods as you would for any library you want to run on the OCX along with the usual variables etc.

The Security Guard for the Internet methods you need and what they are for are described below.

The sgisuserloggedin(strUserName string) logical method:

You can use this to verify that a user has an open record in the security tcursor. You access the users name from the request as:

```
strName = Request.AuthorizationUserid
```

You can then use the method as

```
if lbSecurityGuard.sgisuserloggedin(strName) then
;// do your stuff
else
;//they are not authorized in the security system so kill the process, force them to log in etc.
endif
```

The sglastaccessstamp(siTimecheck smallint,strUserName string) String method:

You can use this to verify that a user has accessed their security session within the last n minutes. The variable siTimeCheck is the number of minutes you will allow between accesses. It is best not to make it too short. Remember, the web is often slow and the users may also be looking at data your system has returned. Security Guard itself uses a 30 minute window before timing out. You access the users name from the request as:

```
strName = Request.AuthorizationUserid
```

You can then use the method as

```
strReturn = lbSecurityGuard.sglastaccessstamp(15,strName )
```

this method returns 3 possible values.

“OK” means the user has accessed the system within the correct time allotted to not have them timed out.

"UserNotFound" means the user name that was passed is not a currently authorized user.

“PastTime” means that the user session has timed out and they are now removed from the authorization tcursor and their session was terminated in Security Guard. If you receive this response you should remove their session from your application too.

Once you have a returned value you can do something like:

```
Switch

Case strReturn = “OK”:
;// do your stuff

Case strReturn = "UserNotFound":
;//they do not have a session running in the security system so kill the process, force them to log
;//in etc. since without a session they have no passwords to systems issued.

Case strReturn = “PastTime”:
```

```
;// they timed out. Tell them so and have them log back in.
;// and kill any running session they have in your application

Otherwise:
;//the method failed so check your code or the system

endswitch
```

When OK is returned the authorization tcursor has this users last access stamp renewed. It is suggested you call this method at the top of each process you activate from your libraries.

The sgverifyaccess(strName string) string method:

Another, generally better alternative is to call this method instead of the previous two. This method calls the other 2 although it uses the default 30 minute window for the time out period. This method is called like the others as:

```
strVerified = lbSecurityGuardMC.sgverifyaccess(strName)
```

This method returns the same 3 possible stings as sglastaccessstamp but has the fourth possibility of returning the string (not logical) false if the user does not have a current login. You can handle the response with a case statement as with the sglastaccessstamp. In Security Guard for the Internet Management Console this is called each time the handler sgchoice is called, which is generally when someone makes a menu selection. We suggest you use this method in the same way. See the demo application's sgdemochoice method as an example.

The sglogout(strAuthName string, strAuthPW string) string method:

You can and should call this method whenever someone logs out of one of your applications which you are running in association with Security Guard for the Internet. These are all the systems which you set up using the Security Guard for the Internet Management Console. These are all systems with password protection on the tables. Generally you will want to present the user with a Log Out option on each system that you have running on the web. When the response comes back as Log Out then you would have an if/endif or a case statement such as:

```
if strChoice = "Log Out" then
   strName = Request.AuthorizationUserid
   strPW = Request.AuthorizationPassword
   strResponse = lbSecurityGuard.sglogout(strName,strPW)
   Response.Resultstring = strResponse
endif
```

The openapplibrary(strLibName String, strLibAlias string) logical method:

This method is essential to plugging your application in to Security Guard for the Internet. The purpose of this method is to tell Security Guard’s libraries to open Library variables to your application library. You pass it the name of the library you are calling it from and the alias in which the library exists so that the called Security Guard library can find it and open it. You will need to call this in the first action your library takes for a user. This is also where the logical variables you initialized in the open come in. For example if you look in the sgdemochoice method in the demo application library you will see:

```
if not loLibSGMCopen then
;//this tells the Security Guard console library to open this library
        if not lbSecurityGuardMC.openapplibrary("demoApp.ldl", ":SecurityWeb:") then
        strErr = errorMessage()
        writeprofilestring(":SecurityWeb:errorlogs.ini",strval(today()),strval(time()),"demoapp
open sgmc lib"+" "+strName+" "+strerr)
        while errorPop()
        sleep(1000)
        strErr = errorMessage()
        writeprofilestring(":Securityweb:errorlogs.ini",strval(today()),strval(time()),"demoapp
open sgmc lib"+" "+strName+" "+strerr)
        endwhile
        else
        loLibSGMCopen = true
        endif
endif
```

This checks the logical variable loLibSGMCopen to see if the Management Console library has already opened this calling library. If not, it tells it to try to open it. If the method fails it writes a message to the Security Guard for the Web Error log. If it succeeds it sets the variable to true so that this will not need to be run again. You need to run this procedure against both of the Security Guard for the Internet libraries. See the demo application sgdemochoice method.

The sgreturnsessionvars(strLibCalling String, strPassedName string) logical method and The sgreturndbvars(strLibCalling String, strAliasCalling String, strPassedName string) logical method:

These two methods need to be run in conjunction with each other. The dynSessions dynarray manages the session variable which Security Guard for the Internet opens. The sessions are all named with the user name as submitted in the authorization process. If the session does not exist it needs to be passed from Security Guard’s libraries by the sgreturnsessionvars method. If it can not be created the process will end with an error message. If it can be created then the database variables for the session need to be assigned. Database variables will be assigned for each alias in your idapi.cfg. For this reason your aliases to data should be created in the idapi.cfg while you should generally use project aliases for aliasing your programs so as not to have a bunch of superfluous database variables using memory.

Each session will be assigned all the rights the user has set in Security Guard. Each user only gets one session. You want to ensure it is deleted when the user logs out. These methods should also be called from the first action the user takes in your library. They must be called after the openapplibrary methods are run and before your users try to access any data. You call these methods as shown here and in the sgdemochoice method.

```
if not dynSessions.contains(strName) then
        if not lbSecurityGuardMC.sgreturnsessionvars("demoapp.ldl", strName) then
        strErr = errorMessage()
writeprofilestring(":SecurityWeb:errorlogs.ini",strval(today()),strval(time()),"sgdemochoice
sgreturnsessionvals"+" "+strName+" "+strerr)
        while errorPop()
        sleep(1000)
        strErr = errorMessage()
writeprofilestring(":Securityweb:errorlogs.ini",strval(today()),strval(time()),"sgdemochoice
sgreturnsessionvals"+" "+strName+" "+strerr)
        endwhile
        return sgerrorproc(Request, Response)
        endif



        if not lbSecurityGuardMC.sgreturndbvars("demoapp.ldl", ":SecurityWeb:", strName) then
        strErr = errorMessage()
writeprofilestring(":SecurityWeb:errorlogs.ini",strval(today()),strval(time()),"sgdemochoice
sgreturndbvals"+" "+strName+" "+strerr)
        while errorPop()
        sleep(1000)
        strErr = errorMessage()
writeprofilestring(":Securityweb:errorlogs.ini",strval(today()),strval(time()),"sgdemochoice
sgreturndbvals"+" "+strName+" "+strerr)
        endwhile
        return sgerrorproc(Request, Response)
        endif

endif
```

The sgreturnsessionvars uses the handleSessions, sgsessionpw and setpws methods which must be placed in your library. These are generic handler methods which must be in your library exactly as they are in the demo application. You can copy them from there. Be sure they are correctly named and the variables correctly specified. HandleSessions, sgsessionpw and setpws are called by the Security Guard libraries to set up the user session and rights from the ones Security Guard has set.

The sgreturndbvars uses the handledbs method which must be placed in your library. This is a generic handler method which must be in your library exactly as it is in the demo application. You can copy it from there. Be sure it is correctly named and the variables correctly specified. Handledbs is called by the Security Guard libraries to set up the user database variables from the ones Security Guard has set. Any failures will be written to the error log. You can expect a failure for each project alias.

The sgsetsessionpws(strLibName String, strPassedName string) logical method:

This method is used by setpws and not by you directly. It only needs to exist in your uses statement as shown.

The method removeDBs(strDbVal string) method and the removeSession(strSesVarName string) method:

These methods exist in the demo application and should also be copied to your own application libraries. You call these in that order (db's first, sessions second) to kill database variables and session variables which were created for user authorization. See the "Log Out" choice in the sgdemochoice method to see how to call them.

Security Guard Management Console for the Internet:
If you have the LAN based Security Guard then you can continue to use that Management Console or the Management Console for the Internet which runs from the sgmcweb.ldl library. When you open your browser to the SGSERVER.FSL the first time it will present you with the login screen (from the pages directory you must place below the directory the server itself is in). Yes will be selected. Press Submit. When you are presented with the user and password challenge type Newuser for both the user name and the password. Press Submit. Select Security Guard and begin adding yourself, users, systems etc. You can create Systems and Users separately from each other. Both must exist before you try to link them. When you link users to systems or systems to users be sure you link them to the correct rights level. You can also add custom rights levels if you chose.