Technology Brief

Keeping Your Files Secure

CONTENTS

Executive Summary

The FileMaker product family supports industry standards and best practices for secure data management. A mix of technologies and techniques is available to support secure configuration and deployment of FileMaker systems.

As is the case with all modern data management systems, care must be taken to protect the physical security of files containing sensitive data and/or valuable intellectual property. In doing so you minimize exposure to brute force assault or other methods for obtaining forced access to the contents of your files.

FileMaker is notable for the degree of granular and dynamic control it provides over access to the content of database files. This includes the specification of multi-level access to individual scripts, layouts (screens), value lists, data tables, records and fields. In the case of record access, an option for configuring data dependent access to individual records is also available.

Where feasible, the use of external (domain-level) authentication is recommended, and this is fully supported by FileMaker Server. Where this is not an option, additional measures (removal of admin access accounts) should be considered as an adjunct to other security.

Techniques for automation of access control management and for enhancing the user interaction model as it relates to the security architecture of FileMaker solutions are presented. This includes provisions for a conventional login/logout process without having to close your solution files

A number of additional technologies are available for use in conjunction with FileMaker's native security configuration. This ranges from data encryption to biometric authentication. However, careful consideration should be given to methods of implementation to ensure that new vulnerabilities are not introduced.

In addition, this paper includes practical advice for developers regarding configuration options and techniques, and proposes a strategic approach to identification and amelioration of security threats.

A Broad View of Security

FileMaker database files contain at least three kinds of potentially sensitive information, which might be characterised as:

- Structural/Concrete – the code and intellectual property associated with the solution itself.
- Implicit – the business rules, procedures and processes that are embedded in the code and interface.
- Explicit – the data, which may contain private, confidential or otherwise privileged information.

If any of the above is of value or importance to you, or of a sensitive nature, then safeguarding your data matters.

"Safeguarding" is a broad term that acquires meaning once we are able to define the threats, risks or contingencies that must be managed. Among them, are threats of malicious damage, misappropriation of code or information and inadvertent modification or deletion of data.

Keeping files secure is one of many facets to consider when managing potential risks and threats. However, it is not useful to consider it entirely in isolation. There is little value in investing a great deal of effort and expense in security measures while overlooking other potential hazards and threats – eg to data integrity and application stability. Similarly, the benefits of application security are diminished if inadequate attention is given to physical security. A balanced approach is always desirable.

There are some instances when the level of threat to security is moderate and where the consequences of a breach of security would not be far-reaching. Nevertheless, even for solutions that operate within a secure environment or include no overtly sensitive information, there is an investment (both in the data and in the solution itself) to protect. In cases where security is added as an afterthought the levels of risk are needlessly high.

FileMaker's flexibility, from single use solutions to corporate systems, is supported by a powerful security capability that can be configured to provide a high degree of granularity over access to data. The security model can, in turn, be leveraged to provide a number of benefits to the developer(s) and the owner(s) of the data.

Along with other measures for the protection of critical data, security serves a number of purposes. Security measures are principally concerned with:

- Who has access?

Protection of the confidentiality of business data – meeting public expectations and statutory obligations regarding privacy – and safeguarding trade secrets, intellectual property and other commercial-in-confidence information.

- How data is accessed/used?

  A means for guiding and limiting the scope of action and interaction that individuals have with the application and the data it contains. This can provide one of the keys to maintaining the integrity and validity both of data and of the systems that store and support the data.

- The life-cycle of systems and data?

  Providing one of the principle mechanisms by which we are able to determine how data will come into existence and how and when it will depart (eg by deletion or archiving). This adds an important dimension to the quality of the data while also ensuring its survival to meet ongoing needs.

This is the "who, how, when and where" of data and system use. When you are able to control and manage these factors, the fate of your data – and the business that depends on it – will be greatly improved.

Threats and Risks

There are many different ways that data and files can come under threat.

- A careless user
- A poorly written and tested script
- An unscrupulous business competitor
- An underworld hacker

Threats can range from a simple action to a real or imagined threat. In fact, the dramatic images of the secret operative stealing company information or a hacker intercepting corporate secrets remotely can distract attention from the fact that most risks and threats are considerably more mundane.

If data or proprietary code is going to leave your premises without your permission, it is as likely to do so as a result of a petty break and enter or an innocent human error as by more insidious means. Network encryption and 24-hour security watch will not help at all if there are passwords written on sticky notes throughout the office. Moreover, system failures are caused by a hot cup of coffee being spilled into the database server just as surely as they are caused by cyber vandals or corporate saboteurs. It is worth considering the range of possible hazards, their magnitude and their likely consequences, in order to respond with a mix of measures which will add up to a reasoned response to the assessed risks.

As part of a balanced perspective, keep in mind that by nature security is a matter of degree and not of absolutes. We can take measures to increase the difficulty with which unauthorised access to a system can be gained but we cannot achieve absolute certainty that any measures we set in place will not be breached. Even with an unlimited budget, security cannot be absolutely guaranteed.

One response to this lack of certainty is to consider the motives of those who might breach security measures and to measure the response accordingly. This provides a reasonable framework within which to make a judgement about how much security is enough.

The ever-present motive for poor security is human inertia. A lack of care or vigilance leads to

- Poor choices and errors
- Data being written into the wrong fields
- Printouts left lying around
- Passwords being shared

and so on.

To address the potential issues it is necessary to design systems that make it easier to do things the right way than to do them the wrong way. Staff who are required to use a dozen different passwords will write them all on pieces of paper on their desk. Give them a single sign-on and the problem disappears. If the data entry screens do not match the stationery information comes in on you run the risk of sensitive data ending up in the wrong fields. However, if you provide a good match between screen layouts and forms the problem will be greatly reduced.

There are, of course, a few more motives for breaches of security that are associated with darker human emotions such as greed, envy and, revenge. Most of these motives have a price attached – so the effort and expense likely to be expended defeating security measures is apt to be roughly proportional to the anticipated gains. If you are able to increase the apparent level of difficulty of defeating your security to the point where the cost is greater than any benefits then most will be deterred from even trying.

FileMaker Pro provides versatile and robust security options that can offer levels of security that will withstand any casual intrusion and that provide a level of protection against more purposeful attacks. It is the purpose of this paper to outline the core technologies of FileMaker security and to propose deployment and configuration choices to best suit a range of situations and requirements.

The FileMaker Security Architecture

FileMaker Inc offers a suite of products that can be configured to suit a wide range of needs. Because of this, there are some challenges when it comes to describing optimal security measures. What might be appropriate for a stand-alone single-user solution is not necessarily well suited for a server or web-based deployment – and vice versa.

There is also the question of the protection of Intellectual Property (IP) rights in the code and interface of the solution. While there is some overlap between the mechanisms that support these twin aspects, the methods of dealing with them are different.

When a FileMaker file is first created, it is essentially, empty. Aside from a default table and a blank layout, a new file contains two user accounts and three default privilege sets. These parts of a file are sufficient to start development of a solution, including developing an appropriate security infrastructure for the file.

From the standpoint of FileMaker solution architecture, several technologies provide the core elements of solution security…

1. The authentication and access control system, with its control of access and privileges at a granular (table, record and field) level within each file. Access control is flexible enough to provide configuration options for both content and structural elements of a file

2. The availability of Secure Socket Layer (SSL) encoding of network data streams when files are made available using FileMaker Server or FileMaker Server Advanced. Network encryption offers a form of protection to access control for solutions that is made available to users of FileMaker Server over LAN or WAN connections. The additional privacy this affords makes remote connections (using the native FileMaker networking capabilities) a reasonable option in some cases depending on the available bandwidth. It is extremely straightforward to implement, requiring only that the relevant option be enabled in the configuration of an installation of FileMaker Server. This can be accomplished within the interface provided by the FileMaker Server Admin tool (SAT), by selecting the configuration icon for "Security" and stepping through the options presented.

   While the interface of the SAT differs between Windows and Macintosh installations, many of the configurable options are the same. In both cases, after selecting to enable (or disable) secure connections, it is necessary to restart FileMaker Server. Once this is done, industry standard SSL encryption will be seamlessly applied to incoming and outgoing communications with the Server.

3. The ability to set up FileMaker access accounts which will be authenticated at the domain level against a server running Active Directory or OpenDirectory. The configuration of this option, and the considerations, benefits and cautions for its use warrant careful consideration and so are discussed in greater detail in the section headed "Physical Security – Data and File Storage Strategies" (below).

4. The capability to remove [Full Access] accounts / access for a file or solution using the Developer Utilities of FileMaker Pro Advanced. This feature is often associated with runtime applications. There are circumstances in which consideration should be given to its use for served or conventionally accessed (i.e. using FileMaker Pro client application) local files. This will be explored in detail below.

While discussing the security architecture of FileMaker, it is appropriate to give some consideration also to the methods used for internal authentication, in particular the storage and use of passwords.

All FileMaker's internal security capabilities are file-based and therefore must be created and maintained within each file.   If passwords (for internal authentication) exist, they must be separately specified and maintained within each file.

FileMaker does not store passwords per se – instead they are stored as a hash. A hash can best be described as a lengthy and sophisticated checksum that has been generated from the original password. When authentication is occurring, a checksum is generated from the supplied password and is compared to the stored password hash value for the relevant account. If they match, it is assumed that the password was correct and access is granted. A key characteristic of this approach is that it is not possible to reconstruct the password from the hash value – its original form is lost during the process that derives the hash.

This system of password authentication has both advantages and disadvantages. Because the password required for file access is not stored – not even in an encrypted state – one avenue for attack against the security system is closed. This may be viewed as a benefit in the sense that the security system is robust – or as a disadvantage if all passwords are lost by their legitimate owners, since there is no direct route to regain access to the files.

It is important to note that techniques and tools exist that are capable of forcing passage past the native security. These tools accomplish this by overwriting the data blocks within a file that contain password verification bits (the hash data) with bogus hash data associated with known password strings. Since these tools are distributed by third parties, their quality may vary and in some cases may cause damage to the files themselves. While such tools might be used to restore access to a file for its legitimate owners, they are also open to abuse and may be employed to gain unauthorized access to data or the IP embedded in a solution. No greater clarity can be given to the truth that absolute security cannot be guaranteed. However, it should be remembered that FileMaker Pro is no different from other contemporary software tools and that equivalent threats to security exist for them as well. It is one of the ever-present realities of information management.

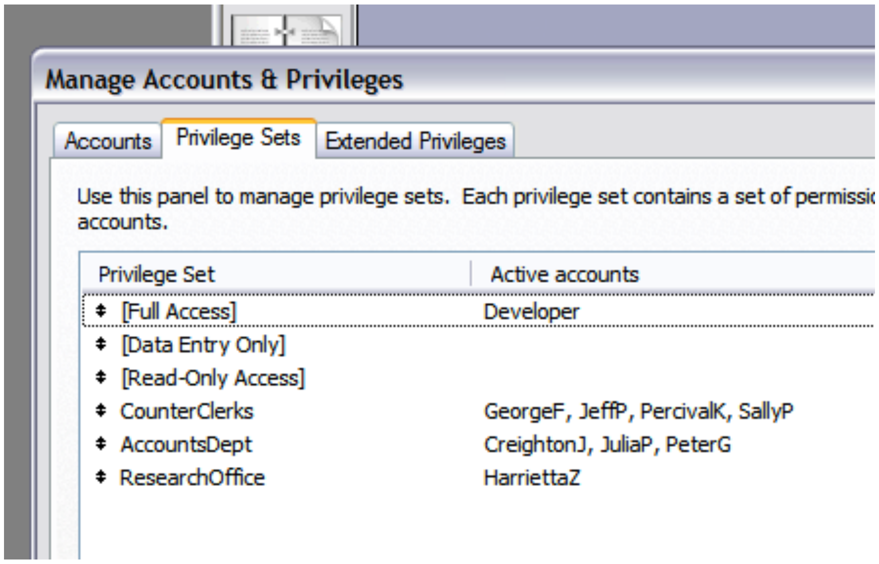
First line of Defence – Accounts and Privileges

FileMaker's model for user access control depends on a versatile structure for user and account management. At the heart of this structure is the concept of the Privilege Set. A Privilege Set represents a comprehensive definition of access that an individual or group of individuals have to data and functionality throughout the database file.

Privilege Sets are defined within the file and are specific to the file. They provide the ability to create granular controls over the behaviour and accessibility of individual elements down to the record and field level within the schema and to the level of individual scripts, value lists and layouts.

Privilege Sets also collect together a set of operational capabilities, and provide them as a group to a number of individuals whose roles and usage requirements will be similar. in other words, privilege Sets provide a mechanism to aggregate users into groups and provide them with similar access. Similarly, Privilege Sets furnish the means to aggregate a variety of controls representing the functional and structural elements of a particular database file.

These twin "aggregating actions" of the Privilege Set provide the basis of what is sometimes referred to as "role based" security. Role based security operates on the idea of customized access configurations that can be made available to categories of users according to the role they perform and the kinds of interaction with the database which will be appropriate to their needs. Thus in a given database, Clerks might require data entry access to certain tables, Accountants might require access to the reports and audit components, while the Department Research Officer may desire read-only access to the system. Such a situation would be served by the creation of three separate Privilege Sets within the databases, each reflecting the needs of one of the groups of users.


Manage Accounts & Privileges

Use this panel to manage privilege sets. Each privilege set contains a set of permissi
accounts.

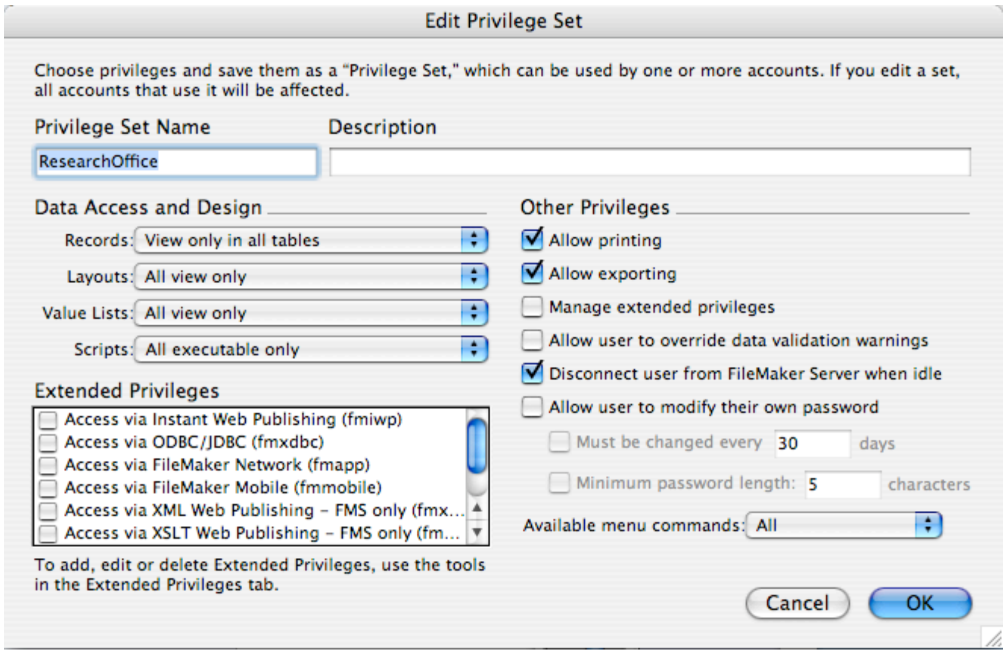
Privilege Set	Active accounts
[Full Access]	Developer
[Data Entry Only]	
[Read-Only Access]	
CounterClerks	GeorgeF, JeffP, PercivalK, SallyP
AccountsDept	CreightonJ, JuliaP, PeterG
ResearchOffice	HarriettaZ

Figure 1 – Privilege Sets defined to reflect user roles.

The creation of multiple Privilege sets is done from the Privilege Sets tab of the Manage Accounts & Privileges dialog — which is accessible from the Manage submenu on the File menu. Figure 1 (above) shows the configuration window displaying an overview of Privilege Sets in line with the example in the previous paragraph.

The definition of access for each privilege set is defined within the "Edit Privilege Set" dialog (which can be accessed by selecting a line within the Privilege Sets tab shown at Figure 1). The Edit Privilege Set dialog provides a high level overview of the access control attributes that have been assigned to the Set. On the right, a number of generic privileges are displayed – these control actions for the whole file, including menu, printing, export and password change controls.

At the upper left of the Edit Privilege Set dialog (see Figure 2), drop-down menus are provided for the specification of access to data (Records), Layouts, Value Lists and Scripts. The simplicity of the interface shown in this dialog hides the granular complexity that is available as we drill down to a further level.



Edit Privilege Set

Choose privileges and save them as a "Privilege Set," which can be used by one or more accounts. If you edit a set, all accounts that use it will be affected.

Privilege Set Name: ResearchOffice

Description:

Data Access and Design

- Records: View only in all tables
- Layouts: All view only
- Value Lists: All view only
- Scripts: All executable only

Extended Privileges

- [ ] Access via Instant Web Publishing (fmiwp)
- [ ] Access via ODBC/JDBC (fmxdbc)
- [ ] Access via FileMaker Network (fmapp)
- [ ] Access via FileMaker Mobile (fmmobile)
- [ ] Access via XML Web Publishing – FMS only (fmx…
- [ ] Access via XSLT Web Publishing – FMS only (fm…

To add, edit or delete Extended Privileges, use the tools in the Extended Privileges tab.

Other Privileges

- [✓] Allow printing
- [✓] Allow exporting
- [ ] Manage extended privileges
- [ ] Allow user to override data validation warnings
- [✓] Disconnect user from FileMaker Server when idle
- [ ] Allow user to modify their own password
  - [ ] Must be changed every 30 days
  - [ ] Minimum password length: 5 characters

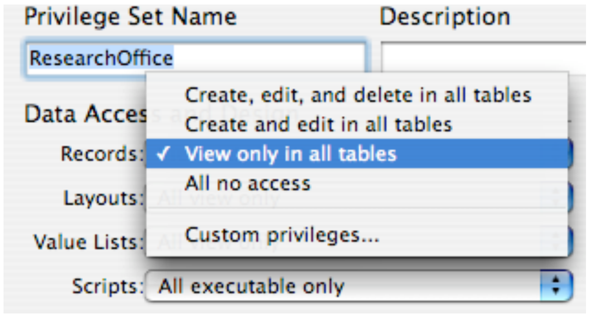
Available menu commands: All

Figure 2 – The Overview of access controls for a Privilege Set.

For each of the four categories of access represented by the drop-down menus, three (or in the case of Records, four) generic options are provided. For two of the four access categories the generic options are:

- All modifiable
- All view only
- All no access

And for the Records access control, the "All modifiable" option is further sub-divided, as shown at Figure 3, to provide editing capabilities either with or without the ability to delete records. The wording on this menu is extended to make it clear that an option selected at this level will apply equally to all tables within the file. As such, the generic options will be suitable in some situations. However if the generic options do not meet your requirements or needs, it may be necessary to select the separated option at the bottom of the list, which is labelled "Custom privileges…".



Privilege Set Name: ResearchOffice

Description:

Records:

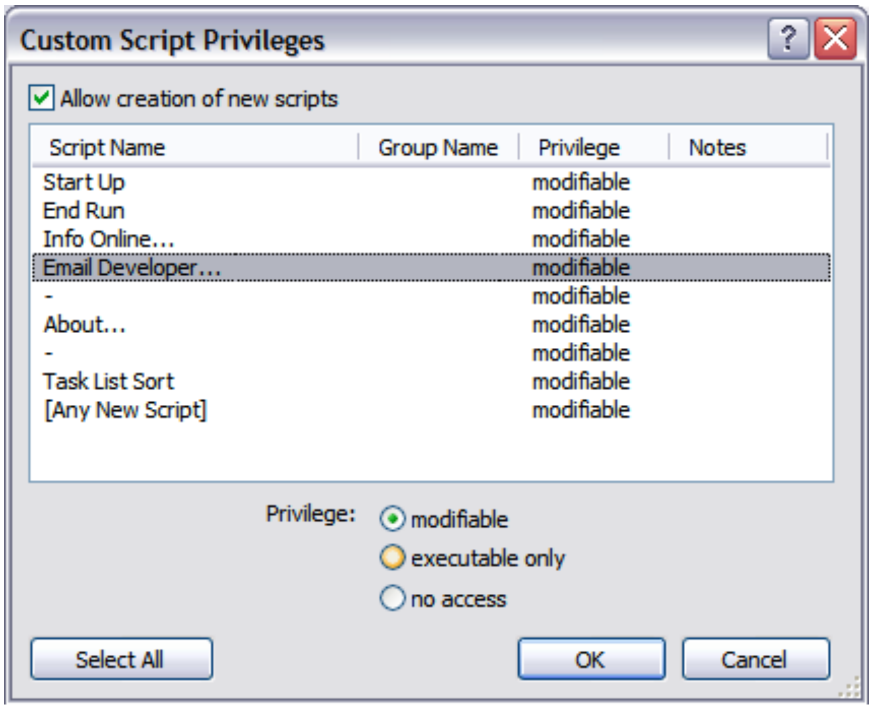
- Create, edit, and delete in all tables
- Create and edit in all tables
- ✓ View only in all tables
- All no access
- Custom privileges…

Scripts: All executable only

Figure 3 – Generic options for data access.

The custom privileges options on each of the four drop-down menus provide access to associated elements (tables, lists, scripts, layouts) where a range of access attributes can be applied. Again, three of the four lists (Layouts, Value Lists and

Scripts) are simpler than the Records list and provide, essentially, one further layer of detail.


Custom Script Privileges

[✓] Allow creation of new scripts

Script Name	Group Name	Privilege	Notes
Start Up		modifiable	
End Run		modifiable	
Info Online…		modifiable	
Email Developer…		modifiable	
-		modifiable	
About…		modifiable	
-		modifiable	
Task List Sort		modifiable	
[Any New Script]		modifiable	

Privilege: [✓ modifiable] executable only no access

Figure 4 – Drill-down specification of Custom Script Privileges.

The Custom Script Privileges dialog, shown above, is provided to separately apply the modifiable/executable/no access privilege setting for each individual entity within the relevant category. Although this dialog pertains to Scripts access, the features available and the way you interact with this dialog Layouts and Value Lists are similar --- with the notable exception that the Custom Layout Privileges allows the setting of two separate privilege bits for each layout. One option allows you to control access to the layout (to view or make changes to it in layout mode) and the other to access to data via the layout. Nevertheless, the principle is the same and 'modifiable', 'view only' and 'no access' options may be specified for both modes (layout mode and browse mode) for each layout in the file.

Defining custom record privileges is more complex than the other three because it provides a means to build controls that operate simultaneously and separately at the table, record and field level. To achieve this, the dialog first presents a list of tables defined within the file.

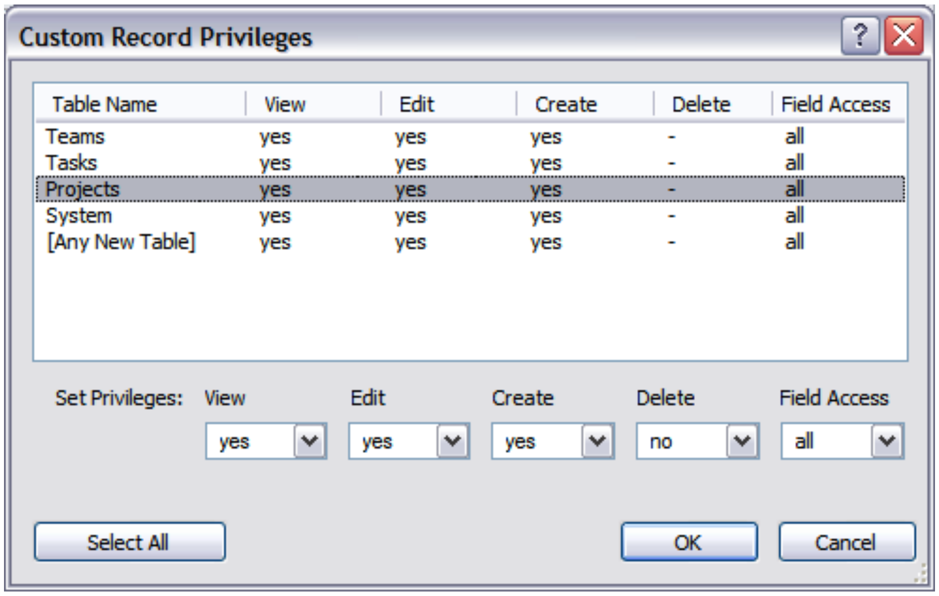

Custom Record Privileges

Table Name	View	Edit	Create	Delete	Field Access
Teams	yes	yes	yes	-	all
Tasks	yes	yes	yes	-	all
Projects	yes	yes	yes	-	all
System	yes	yes	yes	-	all
[Any New Table]	yes	yes	yes	-	all

Set Privileges: View: yes; Edit: yes; Create: yes; Delete: no; Field Access: all

Figure 5 – Drill-down specification of Custom Record Privileges.

After selecting a table (as shown at Figure 5 above), you can apply granular access controls within five categories. The drop-down menus below the list panel provide the control options for each of the five privileges and four of the five provide three options.

The controls for View, Edit and Delete each provide the options of "yes", "limited" and "no as shown in Figure 6. The yes and no options are self-explanatory, however in each case the "limited…" option invokes a calculation dialog that allows you to construct an expression that will determine at runtime whether access will be granted.

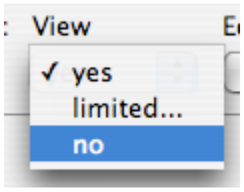


Figure 6 – View control options.

Together these three controls provide what is commonly referred to as Record Level Access (RLA). These expressions are evaluated independently on each record to determine whether the current user should have the designated privilege. These controls can each be separately configured for every table in the file. Thus, it is possible to create a Privilege Set that allows users to (for example)…

- Delete Project records only if they were created more than two years ago
- Edit a Project record only if the project name field contains the words "music" or "soundtrack"
- View all Project records that have "NY" in the State field

Similarly, access to each table in the file can be set individually, providing support for highly granular access control.

The Custom Record Privilege options for the Create setting for each table in the file are simpler than the three controls described above. Since an option to determine privileges via a calculation is less useful here, the only choices available for this control are "yes" and "no".

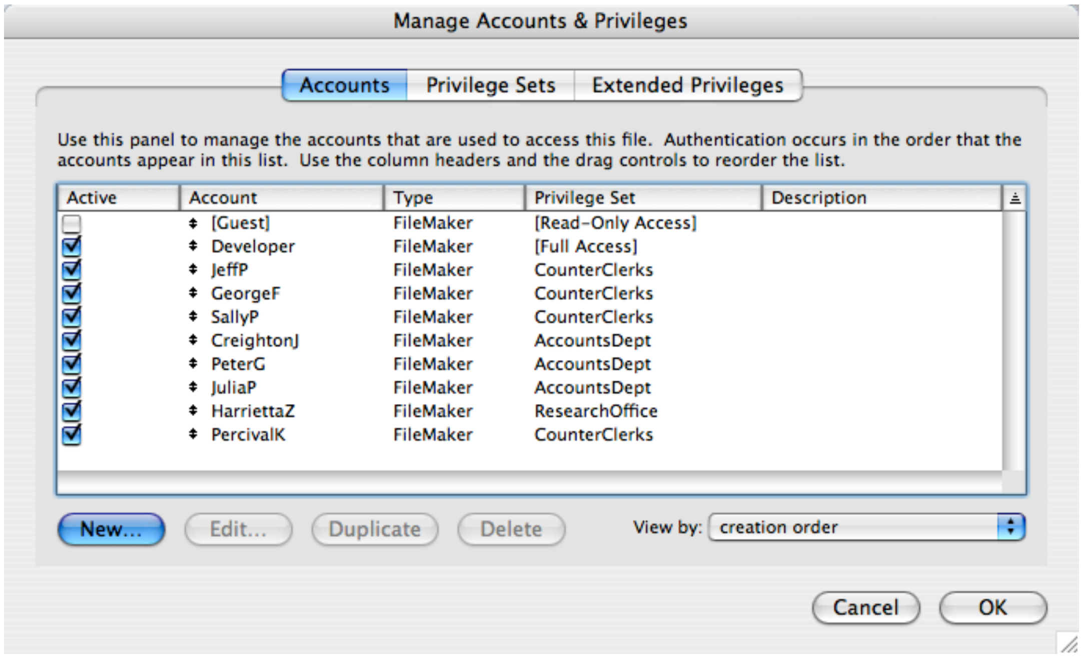
The fifth control takes you into a deeper level of granularity --- allowing you to specify accessibility for each field within the selected table. Here, you can set whether a field can be modifiable, read only or no access for the current Privilege Set. This option completes the drill-down of schema-based access controls, spanning tables, records and fields and tying the Privilege Set intricately to the individual code and structural elements within the file.

The remaining section of the Manage Accounts & Privileges dialog (as shown in Figure 2) provides the capability for extended privilege controls. This serves as a vehicle for enabling or disabling by privilege set, a range of functions and features including several key FileMaker options such as networking and web access. However, it is possible to create custom privilege flags here, which can be selectively assigned to privilege sets within the file.

When custom extended privileges have been created and assigned, it is then possible to retrieve a list of the extended privileges that are active for the current user by using the Get(ExtendedPrivileges) function. When using this function a list of privilege keywords, as defined in the access controls for the current privilege set, is returned. This powerful and extensible feature provides the basis for open-ended control of the behaviour of elements of the file.

Having built one or more privilege sets to provide appropriate constraints and rights for groups of users of a file, the next step is to set up the means by which users will access the file and how will they be associated with one of the privilege sets defined in the file. This is achieved via the Accounts tab of the Manage Accounts & Privileges dialog, as shown at Figure 7 (below).

Using the Accounts control, you simply need to name accounts (names must be unique), assign a password (if the account is to be internally authenticated) and associate an account with one of the privilege sets created in the file. At first it is likely you will want to use this interface to set up a basic set of accounts (and assign them to appropriate privilege sets) within each database file.


Manage Accounts & Privileges

Use this panel to manage the accounts that are used to access this file. Authentication occurs in the order that the accounts appear in this list. Use the column headers and the drag controls to reorder the list.

Active	Account	Type	Privilege Set	Description
☐	[Guest]	FileMaker	[Read-Only Access]	
✓	Developer	FileMaker	[Full Access]	
✓	JeffP	FileMaker	CounterClerks	
✓	GeorgeF	FileMaker	CounterClerks	
✓	SallyP	FileMaker	CounterClerks	
✓	CreightonJ	FileMaker	AccountsDept	
✓	PeterG	FileMaker	AccountsDept	
✓	JuliaP	FileMaker	AccountsDept	
✓	HarriettaZ	FileMaker	ResearchOffice	
✓	PercivalK	FileMaker	CounterClerks	

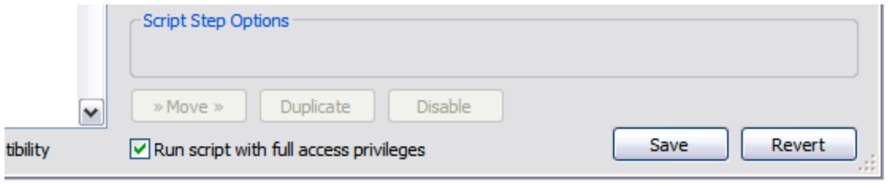
View by: creation order

Figure 7 – Creation and Management of User Accounts.

In the ongoing use of a solution, it is probable that you will wish to consider providing a control framework so that the security configuration can be maintained and modified via scripts within the file (or across the file set, if applicable). This is discussed in detail in the following section.

Supporting Considerations –Design for Security

The scripting capabilities of FileMaker Pro include a number of commands that provide access control to a file and for maintenance of the access of the file. Foremost among these are the Add Account[ ] and Delete Account[ ] script steps. These commands are available as button commands and as script steps within ScriptMaker. These, and other account management script commands, are constrained by the current level of access privileges. Modifications to accounts generally requires full access to the file, so it will generally be necessary to use these commands within scripts that are defined to run with full access privileges. This option can be enabled on a script-by-script basis by selecting the checkbox that appears at the bottom of the Edit Script… dialog (see Figure 8, below).



✓ Run script with full access privileges

Figure 8 – The option to run a script with fill access privileges.

When this option is selected, all access constraints are lifted for the duration of the script's execution. It should be remembered that commands and functions which interrogate the login status (eg Get(PrivilegeSetName) or Get(Account)) will not reflect the login status of the current user while the script is being executed. If it were necessary to obtain a valid contextual result for a function such as one of these, one way to do that would be to call a subscript (one which is not defined to run with fill access privileges) and have it return the required value as a script result.

The Add Account[ ] command requires three parameters –

- Account name
- Password
- Privilege set

Furthermore, the Add Account [ ] script step includes the option to require that the user change the password on next login.

Since account names are required to be unique, the Add Account[ ] command will fail (returning error 12) if an account with the same name already exists. With a script that first calls the Delete Account[ ] command to remove an existing account and subsequently calls the Add Account[ ] command to add the same account assigned to a different privilege set, it is possible to reassign accounts between privilege sets. Using this technique, you can create an interface where users can perform functions such as this by calling scripts provided within the solution.

Only internally authenticated accounts can be added in this manner – although externally authenticated accounts/groups can be deleted using the Delete Account[ ] command. A limitation with the process is that user passwords are not stored internally and therefore it is not possible to create a new account with the same password as an account that has been (or is being) deleted. If the account in question belongs to the current user, it is possible to set up the script to post a dialog requesting the user password, then use the password for creation of the new account. However, this is only appropriate for procedures that will be managed by the users themselves. In all other cases, it is necessary to reset the password and advise the user of the change. Care should be taken to ensure that the new password is transmitted as securely as possible. The addition of the requirement that the password be changed on next login reduces the prospect of a breach of security should the communication of the new password is seen by others.

Also available among the script steps and button options are the commands Reset Account Password[ ] and Change Password[ ], which can be used for automatic or user-interactive updates of passwords by account. These commands represent the most secure means to automate the management of passwords for internally authenticated user accounts. However, it should be noted that these commands might be unsuitable in cases where it is desirable that the accounts and passwords be maintained in sync between multiple files in a solution. In that case, it may be preferable to script a procedure which collects the necessary information (both account name and password) and then, after processing them internally within the current file, passes them (eg as script parameters) to other scripts in other files in the solution, where the same authentication update can be processed.

There is a concern with procedures of this kind that, albeit for a relatively short time, it is necessary to hold and transmit the password as data. While this concern is real, it is possible to structure the supporting code in such a way as to avoid (or at least greatly minimize) any practical risk. This is done within the controlling script by…

1. Locking the solution interface
2. Disallowing user abort

3. Presenting a custom dialog to request the new password
4. Retrieving the password and deleting it from the global field which was part of the custom dialog

Within this process, the custom dialog should be configured to conceal the password by the use of the bullet character.

In this sequence, the window of vulnerability (during which the password exists as data) is confined to the period between when the user accepts the dialog and when the temporary, global field is cleared immediately afterwards. Since the values of global fields are specific to the client workstation and the client workstation is in use by the client (and operating under interface and abort locks) during the millisecond or so of the duration when the password value exists as data, the risk of interception of the password by a third party is negligible.

Having retrieved the password and cleared the value from the global field used as the pointer for the dialog input, the password now exists within memory. If the password has been transferred to memory as clear text, then a further brief window of vulnerability exists. If an unauthorized person were able to take control of the workstation at this instant to halt or pause the running process and to gain access to the contents of memory; data that includes the yet-to-be-activated password would become available to them. Since this is a small fraction of a second, the risk is more theoretical than real. If such a risk is considered unacceptable, then the password should be encoded or encrypted before being passed into memory as a variable or declared as a script parameter for transmission to other files.

Completing the suite of scripting commands for management of accounts and security are the Enable Account[ ] command and the Re-Login[ ] command.

The Enable Account[ ] command is deceptive in that it provides two options, neither of which corresponds to its name. It can be used to either Activate or to Deactivate a designated account. This command is a useful feature in a scripted account management strategy. Security is increased by scripting automated processes that activate specific accounts on an as-needed basis. For example, a solution might be configured to activate an account at start-up only if it is running on a particular workstation and to deactivate the same account again on closure. With such a mechanism in place, the account will not be activated unless the solution is opened in a specific known environment.

Finally, the Relogin[ ] command provides a mechanism by which automated and interactive logins can be processed. For instance, the command can be used to invoke a login dialog that prompts the user for account name and password in the conventional way (with or without default input values for either or both fields). Alternatively, it can be used in the background to change the login status of the file without reference to the user.

It is worth noting that the security scheme for FileMaker does not include a logged-out state as distinct from when the file/s are closed. In other words, when open, a solution requires that a valid account be the current user account. For situations where it is desirable that users be able to end their login session without exiting from the solution, it is helpful to create a constrained privilege set that simulates a logged out state.

To achieve this, create a privilege set which has no access to data in any tables, and has access to one "blank" layout (no fields, but sporting perhaps a system logo and a single login button and an exit button) and one script (the login script). Attach a single "Logout" account to this privilege set and specify a fixed password that cannot be changed by the user. Then create a "Log out" script that takes the user to the blank layout, closes extraneous windows, locks down the interface and performs a re-login to activate the Logout account.

When the user runs the Log out script, the file is placed into a secure state where the only actions available to the user are to close the file or log in with a different account. Where multiple users may have cause to access a database on a single workstation, this provides an efficient way for them to secure the database without inconveniencing other users by closing the files. It also provides enhanced security by enabling users to log out while on a break or absent from their office for a period, thus reducing opportunities for unauthorized access to their account.

Physical Security – Data and File Storage Strategies

There are a number of benefits from a security perspective of server-based deployment. As noted previously, one of these benefits is the built-in capability for two-way SSL encryption of network data. This provides a significant improvement of the security of shared data and is especially important if database access is required in situations where firewall protection is not in place (eg on a wide area network).

FileMaker Server can be enabled to support authentication against a network domain. However, unlike SSL, this capability also requires that:

1. An appropriate corresponding configuration is in place within the access control settings of the database file(s) that are to be accessed from the server. Specifically, individual accounts must be defined within the security schema of the file(s), to match domain account groups and, where multiple groups have been defined (and especially where it is possible for a user at the domain level to belong to more than one group) an authentication order should also be determined, and

2. A corresponding authentication server for the network domain is operating and is configured correctly. It should have accounts assigned to groups that coincide with one or more groups defined in the database file(s) and that are set up to use external authentication.

In this way, the use of external authentication requires appropriate settings at three levels:

- The database file(s)
- FileMaker Server
- Applicable domain controller.

The set-up of external authentication may therefore be seen as more difficult – however it does provide a number of benefits.

Two key advantages of external authentication, which are worthy of particular note:

1. User and administrator convenience: as noted previously, one of the obstacles to effective security is the reluctance of many users to memorize large numbers of username/password combinations and to suffer frequent interruptions to their work to navigate through a variety of login and authentication checks. External authentication, if configured correctly, can free users from much of this; enabling them to sign on once at the domain level and then have access to all the databases for which their account/group is enabled, without further authentication.

   This, in turn reduces the dependency users may have on sticky notes with passwords, in rows along the bottoms of monitors. Improved productivity (fewer interruptions) reduced staff exasperation (less rote memorization of account details required) and improved security are also gained.

   Similarly, the use of external authentication enables network administrators to manage user accounts in one place, rather than having to expend the additional effort of separately maintaining individual authentication data on the database server.

2. Synchronization across multi-file solutions: one of the limitations of FileMaker's internal authentication structures is that they are file-based. Privilege sets are created within each file and are integrated within the file structure (to tabulate the granular access to specific tables, fields, layouts, lists, scripts etc that reside in the file). However, accounts (username and password) are also specified at the file level.

   While authentication will be passed from file-to-file if the username and password both match, keeping accounts throughout a set of inter-related files synchronized is difficult, problematic and somewhat challenging to fully automate.

   However, these challenges are addressed by the use of external authentication, which provides a mechanism whereby authentication is completed at the domain level, requiring only that generic groups for the respective privilege sets exist within the individual database files. This not only simplifies the process of managing multi-file solutions, but it addresses issues such as the maintenance of accounts and passwords across the multiple files of a database solution.

At this point, it is important to note that while external authentication provides a number of significant benefits, it also carries a notable additional risk. Because any implementation of external authentication removes the security screening from individual database files and entrusts it to the domain controller, there is a risk that if the domain server is compromised, *or* if the database files are exposed to an environment that simulates the group authentication for which they are configured, unauthorized access is possible.

Consequently, the physical security of the database files themselves is considerably more important if externally authenticated groups have been specified, than it would otherwise be. External authentication introduces vulnerability at the file level, which only makes sense if the domain environment and the physical environment in which the files will be used and stored is known and trusted. This applies to all copies of such files (backups, test copies, development copies etc, as well as the 'production' copies on FileMaker Server.

In fact, even in the case of files that have not been "opened up" for external authentication, maintaining physical security of the files is preferable. There are multiple reasons for this…

1. While the code and the data within an .fp7 FileMaker file are not stored as raw text, neither are they stored with any strength of encryption. This provides an avenue through which unauthorized users who have physical access to the file(s) (or to a copy) may attempt to bypass FileMaker access controls by attempting to read and interpret the code and/or the data directly with other applications.
2. The security architecture of individual FileMaker database files is relatively robust. However, a variety of means exist whereby unauthorized access might be gained where the physical file is available. These range from brute force attacks against username/password combinations, to the use of third-party hacker tools to compromise or bypass the internal security data within the file.

Nevertheless there are a variety of circumstances when physical access to database files cannot be withheld. A notable example of this being runtime solutions which are specifically designed for distribution to end users for installation on their own systems. In these cases, other measures should be considered as an aid to security.

The fourth element of FileMaker's native security capabilities, as mentioned at the start of this section, is the ability to permanently remove [Full Access] accounts from the file(s). This is a procedure that should be considered in all cases where the physical security of database files cannot be guaranteed – not just for files that will form part of a runtime solution.

Removal of [Full Access] accounts can be achieved via the set-up options available within the "Developer Utilities…" dialog that is accessible from the Tools menu of FileMaker Pro Advanced. This process should not, of course, be performed on master (developer) copies of the files. A master copy should be retained in a secure environment for reference, backup and further development. However, it is reasonable to consider applying the procedure to all copies of the file(s) that are to be placed into production or made available directly to end-users.

While removal of [Full Access] accounts improves the strength of the file-level security defences, it should not be regarded as a sufficient measure in itself. The procedure provides good protection against direct access to the code (schema, calculation and script definitions etc) within the file. It also provides some benefits in relation to the means that might be employed to obtain indirect access to the code. But it does not necessarily prevent a very knowledgeable individual from reconstructing elements of the internal code by reading the file with other tools. Furthermore, it does not provide a guarantee against unauthorized access to the data within the file (either by accessing the file content with an alternative application or using third-party tools to break or bypass the remaining user-level security account/s).

Although removal of [Full Access] accounts provides a key component of a security strategy, especially where the physical security of database files may not be protected, it is recommended that it be used in conjunction with a mix of other measures which will enhance the security of the interface and perhaps also of the data.

Security in Practice – Ongoing Management

One of the ironies of security is that by its very nature, it presents a challenge to human ingenuity. While we can develop new and stronger methods to protect systems, eventually, our colleagues or competitors will develop new and stronger methods to break past our defences. The time between when a new security technique is publicly discussed and when it becomes clear that a method to defeat it has been devised is frequently as short as a matter of weeks.

One thing you can be sure of – methods of security described in this paper will have several ingenious ways to circumvent them being put into circulation sooner rather than later.

For this reason, best practices rarely remain so and the only true best practice is the habit of remaining alert and keeping one step ahead. As developers, our best defence against flawed security is our own ingenuity. If we are all able to vary our security strategies to take advantage of the technologies that are available and address vulnerabilities when they arise, we will continue to provide rational security frameworks for our solutions.

With that said, here are a couple of techniques and approaches that may be adapted to your needs to provide strengthened security where it is required in your solutions.

Firstly, to guard against the possibility that third party tools will be used to overwrite existing (legitimate) passwords in your solutions with Trojans, consider creating a secured low-access account as the default account for your solutions (see description of a logout account under the Supporting Considerations… topic above) and then have your users run a script to log in so that they can access the solution. Set the default state of user accounts to deactivated so that the login script (which activates the requested account before logging into it) is the only way that users can get access to any account other than the secured ("logout") account.

Third party tools can overwrite password verification data with impostor hashes, but do not provide a mechanism to bypass account deactivation. If the only accounts left active in the file are associated with privilege sets that provide no access to data or code, then an attack with tools of this kind (provided [Full Access] accounts have been removed) will not yield access to the solution.

This is but one of a number of methods by which attempts to circumvent the native database security may be thwarted. Here is another approach. Provide a user login script (as outlined above) which, prior to processing the user's authentication request, attempts a pre-coded blind (no dialog) re-login to a low level account using a know predetermined password. Then have the script test for errors. If the password hash data has been tampered with, the blind re-login will fail because the predetermined password will not match the Trojan that has been installed. In the event that this condition is detected, have the login script lock and close the file rather than proceeding with the login.

Of course, neither of the above measures alone will be sufficient, since it is also possible for educated users to interrupt scripts and work around their behaviour. So additional measures will be needed. One such measure could be to have your key security scripts (such as the login script) calculate and store several checksums that capture metrics about its execution and confirm that it ran its course without interruption. Then code strategic calculations and scripts elsewhere in the solution to look for and validate these checksums as tokens on which their scope of action depends. In this way, a solution can be effectively disabled unless conditions you have determined are operable.

The brief outline of several methods provided here is not intended to be exhaustive or to provide the basis of a comprehensive strategy for the protection of your current or future solutions. Rather, it is intended to indicate the ways in which known threats to security can be addressed with a coordinated strategic approach. Rather than seeking an explicit set of instructions for a security design for your solutions, consider adopting a proactive and strategic response.

Two additional elements are key to successful security: current information and current technical resources. Your connections with colleagues and participation in online forums will provide you with an awareness of new developments and techniques – which represent the threats and opportunities regarding the security of your solutions. These will also point you towards new technologies and resources that may provide extensions or enhancements of the security of your solutions.

Third-party products that may provide additional security for your solutions range from…

- Hardware dongles
- Biometric authentication interfaces
- Native and plug-in encryption systems
- Scanners and swipe ID systems

Each of these may be appropriate to particular deployments or implementations. For most purposes, carefully considered application of native security options will provide an appropriate security response. However, be aware that additional options and approaches are available when required.

While retaining an awareness of the range of security enhancements that are available for FileMaker, it is also advisable to approach them with some caution and scepticism. In many cases, alternate technologies address one vulnerability while introducing other vulnerabilities of their own. If you are aware of this, you will be able to make informed decisions and choose approaches to implementation which moderate the impact of any shortcomings.

An example of the way in which an add-on system may create rather than remove a problem is the use of encryption technologies to secure passwords. It is generally understood that the passwords, once encrypted, are far more secure. However, all that is required to retrieve them as plain text is the encryption key. At this point, unless you have considered your implementation carefully, you will have traded one password for another (in this case a key) and the problem of how to securely store the encryption key may be of similar magnitude to the problem you originally set out to solve.

The Human Factor

Bear in mind that the greatest potential strength – and the greatest weakness of any system – is the people who use it. If the system is designed around the needs and habits of the users, it will reinforce their desire to do things the right way and they will recognize that the system is empowering and supporting them – so they will work with the system and flow and synergy will be the result.

Conversely, it is all too easy to design a system that works against the users, imposing obstacles and barriers, displacing problems onto the users rather than solving problems for the users. In part, it is a matter of perspective, but mostly it is a matter of design.

Many architect-designed landscaped environments suffer from what some would characterise as human perversity. Instead of walking on the beautifully made walking paths that wind elegantly across the space between the lawns, people walk across the lawn and create unsightly worn tracks in other places. You know what comes next – signs that say 'keep off the lawn'. Nevertheless, it is not people who are the problem; it is the architects, who placed paths in the wrong places.

Database security systems operate – and fail to operate – in the same ways. Create an obstacle course or a path that is longer than the shortest distance between two points that people are accustomed to taking and the people will work against the system. The solution to this is to find out what it is that people are already doing (where the 'natural' tracks lie) and to build the paths to respect and enhance what is already there. It is a simple but surprisingly effective principle.

Security systems should be conceived and implemented to support users and to make aspects of their work easier – preventing errors, safeguarding data, removing anxieties and facilitating workflow.

Factors such as the choice of passwords and the security of passwords are among the greatest vulnerabilities of any security system. It is therefore important to give thought to system deployment options which will keep the number of passwords users will require to a minimum and will minimize the frequency with which users are required to enter the password.

Beyond passwords, a variety of aspects of the security, privacy and integrity of data are contingent on business process and work practices as well as system design. Our starting premise should be that humans make errors – however there is no shame in that, it's how we are. When systems are designed to be resistant to error and to tolerate and compensate for errors when they occur, the user experience is enhanced.

As an addition to preventive security, it is also important to consider corrective security – systems that provide self-diagnosis and self-correction capabilities. For instance, a truly secure system should include internal provisions for audit of both user sessions and data transactions. Such systems not only provide direct and detailed information about transactions – authorised or otherwise – but can also provide the basis for selective rollback of errors or other data anomalies, whether malicious or accidental.

Conclusions

A successful FileMaker security strategy requires that internal file security configurations be set in place during development and developed as an integrated part of the solution design. FileMaker Pro provides access controls which can be specified to a high level of granularity, which can therefore offer details and highly customised support to authorized users, while significantly reducing the risk of unauthorized access to data and intellectual property.

A comprehensive approach to security is advantageous, recognizing that there is considerable flexibility within the security options available across the FileMaker suite of products. While no one configuration will be appropriate for all environments, a combination of available elements can be set in place to provide an adequate response to most requirements.

Deployment options, including the tools for automation of account management and user interaction, as well as the options for external authentication and role based privilege allocation provide excellent support for developers and database administrators, as well as end users.

FileMaker's use of standards-based technologies such as SSL and domain authentication provide a strong basis for integration within corporate data management environments as well as in traditional workgroup and vertical market applications.

ABOUT THE AUTHOR:

Ray Cologon Ph.D. is Managing Consultant and Principal Developer of NightWing Enterprises <http://www.nightwing.com.au/FileMaker>. Ray has published a range of resources, demos and materials for FileMaker Developers. In 2005, Ray received a FileMaker award for Leadership and Technical Excellence in FileMaker Pro. He is based in Melbourne, Australia and provides custom FileMaker development services, system audits, consulting and developer support to clients in various parts of the world.. Contact Ray at cologon@nightwing.com.au