CHAPTER 17

Security In Microsoft Office

Reality dictates that while most programmers use their power and knowledge for good, there are also those who work with a far darker intent. Because of this, security has always been a big concern to educated Office users. Many safeguards have therefore been implemented to both complement and protect against the great amount of power that has been put at the fingertips of developers in the form of VBA.

The benefits of VBA are incredible, as it provides us with tremendous flexibility when working with a user's system, but VBA also affords the same capability to those with nefarious intent. While we can craft routines that automate entire business intelligence applications and span thousands of lines of code, it only takes a single line of correctly crafted code to render a system unusable. Not wanting to completely remove the functionality and benefits of automation, Microsoft has been left the difficult task of balancing the two sides of this coin: giving developers the access they need while protecting users from those with ill intentions. It is a difficult balance to strike, to be sure.

Fortunately, Office 2007 provides several enhancements to the security model that are targeted at both protecting the end user and making the life of the developer easier. Since every dynamic customization that we create requires using a VBA callback, it is imperative that we understand and master the concepts of security in the Office environment. This chapter discusses each of the concepts behind Office security, both old and new. Our goal is that by the end of this chapter, not only will you understand the concepts, you'll also feel comfortable and confident with the protections that they provide.

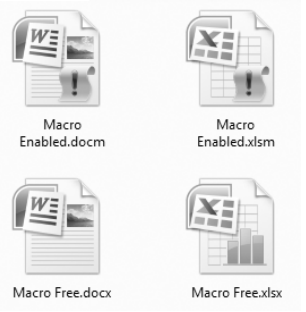

Figure 17-1: Word and Excel file icons

Notice that both of the macro-enabled file formats now bear an exclamation mark, signifying that they (most likely) hold macro code. This distinction provides a very obvious cue to the user that there may be more to the macro-enabled files than meets the eye. A user can take solace in the fact that if the file is saved in a `docx` or `xlsx` file format, it cannot contain any macros, as all VBA code will be stripped from the file when it is saved.

NOTE When working with Word and Excel, keep in mind that macros and VBA are synonymous. Unlike Access macros, all Word and Excel macros are written in VBA.

NOTE Remember that files stored in a binary format, despite the fact that they don't have separate file formats for a macro-enabled and macro-free distinction, can still hold VBA code. The binary file format is used by Office 97–2003 and all Access files.

While it's nice that there is a visual cue to differentiate between files that may hold code and those that don't, we all know that many users will either be ignorant of the symbol and its meaning, or will become so conditioned to cautions that they ignore this warning. This is why the file format should be considered only the first line of defense.

The Trust Center

In addition to the file structure split, there are also other enhancements to the Office 2007 core that affect both end users and developers alike.

Microsoft regrouped all of the security settings and put them into a central place called the "Trust Center." This new collection provides us with a central location for accessing and managing all the security settings related to how Office reacts to files that use potentially dangerous controls. This is important for developers; it is both a

trust locations. Both developers and users will have an instant affinity for the convenience that this setting affords. The interface for establishing Trusted Locations is shown in Figure 17-3.

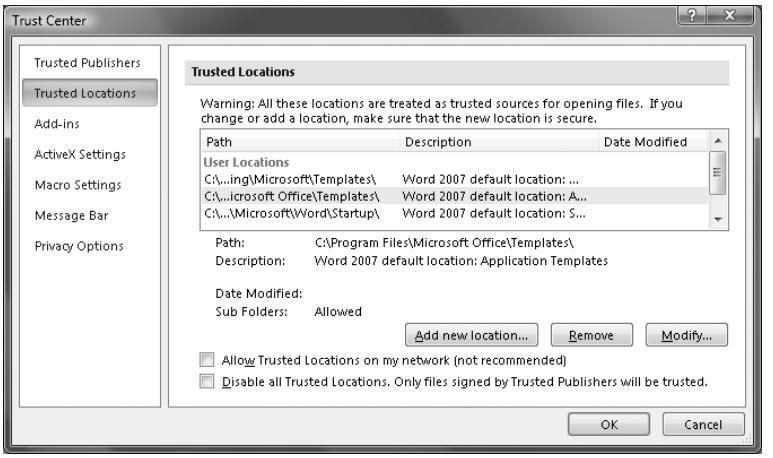

Figure 17-3: The default Trusted Locations tab in Word

To put it simply, the Trusted Locations tab is an interface that enables you to designate certain folders as "safe." This is a fantastic concept and one of the best enhancements to the Office security model for developers.

Trusting your development folders means that you can avoid all security messages as you load and unload files, without having to go through the extra steps of adding a digital certificate to each project. For those of us who create test files on a regular basis, this can truly save some time. It also means that we only actually need to worry about handling the files that we will distribute to others at the end of the development process.

The basic theory behind the Trusted Locations model is, of course, that you will only store in these trusted folders files that you are absolutely certain contain safe code. The instant that you violate that idea, you expose your system to any malicious code that resides in the file, and may as well be running without macro security.

CAUTION **Any file in a trusted folder has full rights to run on the user's system, including VBA macros, data connections, and ActiveX controls. Make sure that you only trust folders that you have control over and know are safe! It is strongly recommended that you *not* trust a location that would be a target of automatic downloads, such as your Documents folder or desktop.**

NOTE **As a means of protecting users from themselves and from malicious script, Microsoft has prevented certain locations from becoming a Trusted Location. The folders include the root of the C:\ drive, as well as the Temporary Internet Files folder.**

Trusting Network Locations

There are two other settings on the Trusted Locations tab, the first of which is Allow Trusted Locations on my network (not recommended). As you'd expect, checking this option allows the user to set trusted locations on a network drive.

CAUTION **Chances are fairly good that you do not have control over what is or is not placed in folders elsewhere on the network, so trusting network locations can present a serious security risk!**

NOTE **You can trust locations on a USB thumb drive, but not at the root level. Instead, you must trust a folder on the USB drive. While this can be very handy if you like to carry code from one location to another, it also exposes you to the risk that someone else will have the same folder hierarchy that you have trusted. If that happens, their folder will also be trusted and they will be able to run code unguarded on your system.**

Disabling Trusted Locations

If you are a systems administrator or a user who is concerned about malicious code, then this setting is for you. You can check the box to "Disable all Trusted Locations. Only files signed by Trusted Publishers will be trusted" and stop any project dead in its tracks unless it has a digital signature. (Experiences will vary depending on the settings on the Macro Settings and ActiveX tabs.)

Add-ins

The security settings on the Add-ins tab are specifically designed for treatment of the Add-in file formats that you learned to build in Chapter 16 (including Word's global templates). Unlike the Office 2003 model, which required you to check a box in the security settings in order to trust all installed add-ins and templates, in Office 2007 these files are trusted by default. After all, it typically takes an intentional act to incorporate an add-in. However, there are a few options that allow you to override this setting, as shown in Figure 17-5.

In looking at this pane, it appears that you only have three options: all add-ins will run (the default), only add-ins signed by a trusted publisher will run, or no add-ins will run. However, as you'll soon learn, there are additional settings that provide more flexibility, particularly with the first option.

Figure 17-6: Security settings triggered by allowing only signed add-ins

Disabling All Add-ins

The "Disable all Application Add-ins" setting rightfully carries a warning that you may lose functionality. This setting should only be used in the tightest of security environments.

NOTE **As a very interesting point, note that while setting this checkbox will disable all the VBA code in an add-in from running, the add-in still seems to load. This is demonstrated by the fact that disabling all add-ins does *not* block any RibbonX customizations that exist in those files. In other words, your add-in will still load and create new tabs and groups, and it will still move icons around. In addition, if all of your customizations were based on built-in controls alone, they would still function as designed.**

ActiveX Settings

This section of the Trust Center (not available in Access) is also new in Office 2007. It establishes how ActiveX controls will be treated from a security standpoint. As ActiveX controls are outside the scope of this book, these settings are not explored. However, it might be reassuring to know that the settings contain similar notations and guidance about the options and their effects.

NOTE **For those who have never used ActiveX controls, these are the controls that are added to documents and workbooks. They are found on the Developer tab.**

Of course, "Enable all macros" is the most wide open setting, as it allows everything and anything to run. This setting is rarely recommended. A much better approach is to place applicable files in a trusted location, rather than effectively using a blanket switch to turn security off.

Trusting VBA Project Access

The last setting on this pane enables you to trust access to the VBA Project object model. In short, this setting allows you to successfully run code that can manipulate the VBA project components and structure, instead of just targeting the layer of the file that is seen in the user interface. This type of access allows code that can actually write, modify, or delete code, meaning that the code itself could even add or remove entire code modules.

Based on this simple explanation alone, you can see the power that someone would have if you allow and trust access to the VBA project. As a general rule, you would most likely never want to set this on a user's workstation. This is a global setting and it affects files both inside and outside the trusted location folders.

Message Bar

The message bar settings, shown in Figure 17-8, control how the application reacts when macro content has been disabled.

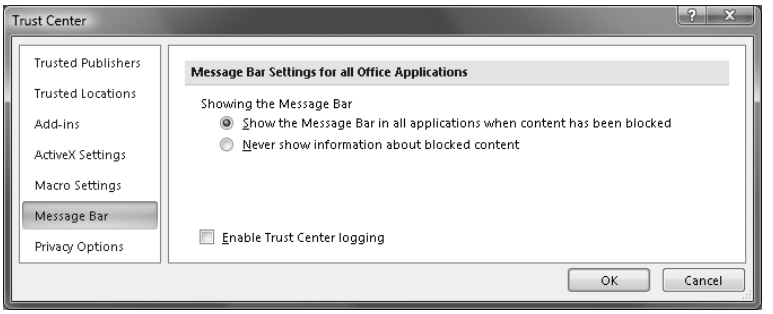

Figure 17-8: Message bar settings

By default, the message bar is shown when content is blocked, and appears as shown in Figure 17-9. Changing the setting to "Never show . . ." will, of course, stop the system from prompting the user when macro content has been disabled.

Removing this message also removes the convenient option that allows the user to enable macros associated with the file.

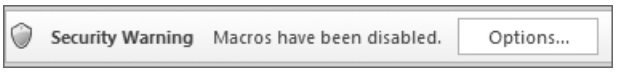

Figure 17-9: Message bar warning of disabled content

Assuming that the user agrees to trust the developer, the system adds his name to the list of trusted developers, uses the key to unlock the code, and puts the key in safekeeping until the next time. Thereafter, any file that is signed with the identical signature will be unlocked with the stored public key and the code will be allowed to execute. Keep in mind that several things had to happen to make this possible, including the user specifically stating that they trust the developer. It also requires that the code is still digitally signed. We're about to discuss some nuances to that.

NOTE While the digital certificate proves who signed the code, it makes absolutely no guarantee that the code within the file is safe. It is completely up to users if they want to trust the issuer of the certificate on their system.

Where the digital certificate proves its worth is in the mandate that the private key must sign the code. Any modifications made to code in a digitally signed file forces a validation of the digital certificate. Failing to get validated, the code in the file will no longer have a digital certificate.

Assume that in the preceding case, one of the developer's clients decided to modify the code. As they begin to make modifications, the lock pops open completely. Unlike most padlocks, however, which can be relocked simply by closing the hasp, this lock requires the private key to relock the file. Since the private key resides on the developer's computer, and not on the client's, the file cannot be relocked and the digital certificate evaporates. Unfortunately for the client, the developer's locksmith will not issue him a copy of the private key, so he is left with an unsigned file. Any attempts to open the file in future will then be treated as unsigned code and reacted to with the security permissions set in the Trust Center's Macro Settings area.

As you can see, the digital certificate offers assurance to both the developer and the client. The client can be assured that the code was delivered as intended by the developer, and the developer can be assured that the client has not changed the code in any way (or, if the code has been changed, it no longer carries the developer's digital certificate). This can be quite useful for the developer from a liability standpoint if any destruction is caused and blamed on his file. If the digital certificate is gone, it indicates someone has tampered with the code.

NOTE While the preceding anecdote explains how a digital certificate works, a far more technical explanation can be found on Microsoft's Technet site at the following URL: `www.microsoft.com/technet/security/guidance/cryptographyetc/certs.mspx`.

Acquiring a Digital Certificate

While digital code signing certificates can be purchased from a third-party vendor, they can also be created without cost using a self-certification program that is installed with Microsoft Office: `SELFCERT.exe`.

At first glance, you might think it is a no-brainer to simply use `SELFCERT.exe` to create a free certificate, rather than pay for a commercial version. As with most things,

NOTE **Despite the warnings shown in Figure 17-10, you can actually trust the public key of a self-signed certificate on another computer. This is of critical importance to developers who want deploy solutions.**

To create your own digital certificate, all you need to do is enter a name and press OK. You will instantly receive notification that your certificate has been created.

Adding a Digital Certificate to a Project

After you have acquired a code signing certificate, either via self-certification or through a commercial certificate authority, you need to assign it to your project. Fortunately, this is also very easy to do.

Open your favorite macro-laden file (all applications use the same process) and enter the VBE. From the Tools menu, choose Digital Signature. You will find yourself at the screen shown in Figure 17-11.

Figure 17-11: Selecting a digital signature

Notice that at this point, there is neither a signature associated with the project, nor a default certificate showing in the Sign As field.

NOTE **After you have signed your first project, the Sign As area will be prefilled with the name of the last certificate you applied. Clicking OK at that point will apply the certificate in the Sign As field to your active project.**

Click the Choose button to be shown a list of the existing digital certificates that are available for signing the code (see Figure 17-12).

As you can see, the certificate that you just created, My Code Certificate, is on the list. With that selected, click OK, and you will be returned to the digital certificate interface. At this point, the project is signed and the certificate name is listed in the Sign As area for easy application to other projects, as shown in Figure 17-13.

Upon clicking OK and saving the file, the project is signed and ready for distribution.

Figure 17-14: Macro security alert details

NOTE Figure 17-14 shows another difference between self-certified code and code signed by a commercial certificate. In Figure 17-6, where the code was signed by a Microsoft certificate, users had the option to trust it immediately; however, a self-signed certificate requires additional steps.

At this point, the user must click the Show Signature Details link to be taken to the details of the digital signature file, as shown in Figure 17-15.

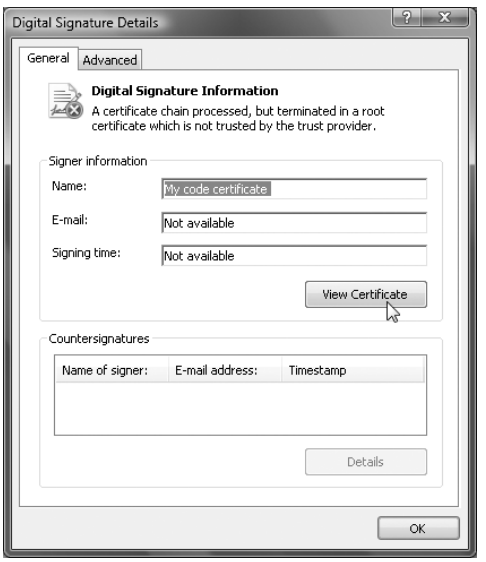

Figure 17-15: Digital Signature Details

Figure 17-17: The digital certificate listed in the Trust Center

One way to remove the certificate and its private key is to locate the certificate file on your computer and delete it. This can be done by going into the VBE and choosing Tools ➪ Digital Signatures ➪ Choose. Select the certificate that you wish to delete and click View Certificate ➪ Details. Scroll down the list of items until you find Thumbprint, as shown in Figure 17-18.

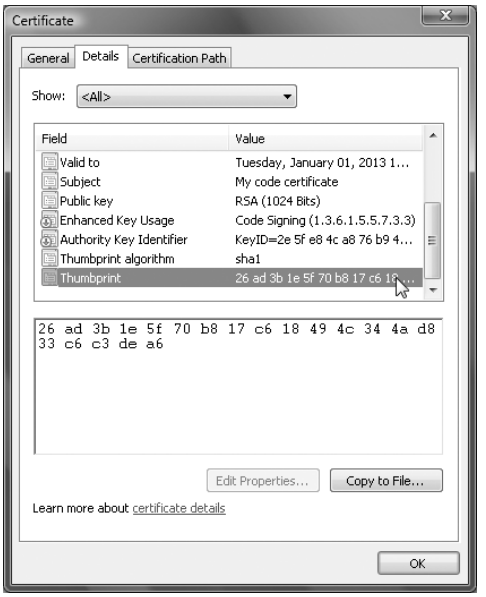

Figure 17-18: Locating a digital signature's thumbprint

Now, based on your version of Windows, browse your computer to locate the following folder:

- Windows Vista:

```
C:\Users\UserName\AppData\Roaming\Microsoft\SystemCertficates\My\
Certificates
```

with. The Trust Center also allows more security configuration options than were offered in prior versions, such as how to manage macro-laden files and ActiveX controls.

The major changes in Office 2007's security model, however, truly benefit developers. In the past, we could only avoid the nagging macro prompts by turning off our macro security settings or by digitally signing every project we started. The creation of the Trusted Folder in Office 2007 makes it easy to eliminate these annoying hassles. With the ability to designate entire directories as safe havens, developers can create and test files locally without needing to worry about the security settings.

After completing the local development and testing process, developers still have the ability to digitally sign the code before deployment. Using digital signatures, even self-signed certificates, is encouraged; and with the proper settings, end users can configure their machines to run trusted code without taking a carte blanche approach and disabling everything.