A Scanning Tool for PC Root Public Key Stores

Adil Alsaid and Chris J. Mitchell
{A.Alsaid, C.Mitchell}@rhul.ac.uk

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK.

Abstract: As has recently been demonstrated, a malicious third party could insert a self-issued CA public key into the list of trusted root CA public keys stored on an end user PC. As a consequence, the malicious third party could potentially do severe damage to the end user computing environment. In this paper, we discuss the problem of fake root public keys and suggest a solution that can be used to detect and remove them. We further describe a prototype implementation of this solution.

C. Wolf, S. Lucks, P.-W. Yau (Eds.): WEWoRC 2005, LNI P-74, pp. 45–52, 2005.

1 Introduction

Many internet applications, e.g. online banking and e-commerce, rely on Public Key Infrastructure (PKI) [FB01] functionality to support the security services necessary to ensure the authenticity, integrity, and confidentiality of the communications. The correct functioning of a PKI relies on trusted third parties known as Certification Authorities (CAs) operating correctly. Moreover, users of a PKI must have trusted copies of the public keys of one of more of these CAs in order to be able to verify the public key certificates that the CAs produce. Such CA public keys are usually referred to as root public keys.

The main task of a CA is to issue, i.e. digitally sign, digital certificates (see, for example, [NDJB01]). A typical certificate issuing process involves verifying the identity of the entity requesting the digital certificates. When the entity identity is verified, the CA uses its own private key to digitally sign the public key certificate.

One of the most widely used applications of PKI is web server SSL/TLS [DA99]. The detailed operation of the SSL/TLS protocol is outside the scope of this paper. However, the part of the protocol that is of interest to the discussion here is the web server response to the ClientHello protocol message. When a user requests a secure web page, the web browser sends a ClientHello message to the web server. The web server replies by sending a copy of its certificate, in addition to other protocol data. The browser checks the certificate against the list of trusted root CA public keys installed on the user's PC. If the received

certificate was signed by any of the trusted root CAs whose public keys are installed on the user's PC, the browser uses the appropriate public key to verify the server certificate. If the necessary CA public key is not present, or if the verification fails, the browser may give the user a warning message or abort the communications.

A malicious third party could insert a fake root public key into the list of trusted root public keys, as demonstrated in [AM05]. 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". Moreover, a false public key is undetectable by current attack detection software, whereas a malicious application will often be detected by such software. Detecting and addressing this security threat is an important issue that does not appear to have been previously addressed.

In this paper, a tool to detect the insertion of fake root CA public keys is discussed, and the implementation of a prototype tool is described. The rest of the paper is organised as follows. Section 2 outlines ways in which a root key insertion attack might be conducted. Section 3 discusses possible means to deal with unauthorised insertion of root public keys. Section 4 describes a tool to detect and remove suspicious root CA public keys. A prototype implementation of the tool discussed in Section 4 is described in Section 5.

2 Root Key Insertion Attacks

A malicious third party could insert a self-issued public key [ES00] into the list of trusted root public keys on the end user's PC, as demonstrated in [AM05]. As a consequence, the malicious third party could potentially do severe damage to the end user computing environment. For example, the malicious third party could sign applets, macros, and emails and claim that they originate from a reputable software company or web site. A possible scenario for such an attack is discussed in the following paragraph.

One possible means by which a fake root public key insertion attack could be exploited is through web spoofing [FBDW97]. In such an attack, the malicious third party installs the fake root public key into the victim PC, e.g. using the technique described in [AM05], and then convinces the victim to visit a spoofed secure web site. When the victim's navigates to the spoofed secure web site, the victim's browser will receive an applet apparently signed by a legitimate party. Depending on the security settings, the browser will either run this applet without notifying the user, or will ask the user's permission to execute it whilst providing (false) assurance to the user regarding the provenance of the applet. Detecting such an attack would be difficult for an average user. One possible way to detect the attack is to examine the URL of the visited web site. However, a determined malicious third party could fake the browser bar that displays the URL of the genuine web site, as discussed in [YYS02]. The web spoofing attack scenario shows how dangerous fake root insertion can be. Other attack scenarios exist.

The following paragraphs outline possible means by which a fake root key insertion attack could be launched, as described in [AM05].

First, a malicious third party creates a self issued root public key using freely available

tools, such as Microsoft's makecert.exe [Cor04a]. Second, the self-issued root public key needs to be inserted into the user's root public key repository. Three possible approaches to implementing a root key insertion attack are discussed below.

1. *Inserting the root public key under user control.*

   In this approach, the attacker requires physical access to the victim's PC. The attacker creates the fake root public key in advance, using another PC with the required tools, and transfers it to a removable medium. Using the removable medium, the fake root public key is transferred to the victim's PC. On the victim's PC, the attacker launches an Operating System certificate management program to insert the fake root public key into the root public key repository. The attacker needs to interact with the OS certificate management program to complete the attack.

   The attacker would typically rely on the operating system programs and utilities. For example, in the Microsoft Windows operating system, the attacker could use the 'Microsoft Certificate Import Wizard' program to insert the fake root public key. An obvious limitation of this approach is that the attacker needs physical access to the victim's PC. Physical access is required to transfer the fake root public key to the victim's PC, and to interact with the OS-specific program that handles the root public key insertion process. The next two approaches avoid this limitation.

2. *Writing directly to the root certificate store.*

   When writing directly to the root certificate store, the attacker does not need to have physical access to the victim's PC. The attacker bypasses the OS root certificate handling programs and utilities and interacts directly with the root certificate store. In this approach, the attacker needs to create malicious code to conduct the attack and also needs to find a means to execute this code on the victim device. As discussed in [AM05], so far it has not been possible to implement such an attack because of a security protection mechanism deployed on the certificate store that require special access privileges.

3. *Installing the root public key without user intervention.*

   In this approach, the root public key is inserted into the root public key store using the OS certificate management program to overcome the special access control protection set by the OS, as discussed in the previous paragraph. The attacker creates a special program, e.g. a virus or trojan horse, to interact with the OS certificate management program. The main objective of the attacker's special program is to hide all warning messages or security alerts displayed by the certificate management program. An implementation of this attack has been described in [AM05].

The focus of this paper is on measures to address attacks after they have occurred, rather than on preventative measures. Such preventative measures are a topic for future study. In the next section, possible means to deal with unauthorised insertion of root public keys are discussed.

3 Addressing Root Key Insertion Attacks

It would be very difficult for the vast majority of users to detect the insertion of a false root key without the aid of supporting tools or utilities. However, general strategies can be devised to facilitate the detection of such an attack. The possible strategies are discussed in the following paragraphs.

One possible strategy to detect and possibly eliminate inserted root keys is by using a root public key scanning tool. The scanning tool searches the user's root public key store for fake root public keys. When a fake root public key is found, the scanning tool provides the possibility to delete, view, or backup the fake root public key. This strategy is discussed in more detail in Section 4.

Another possible strategy is the use of integrity check tools. Here, a tool is used to compute an integrity check value (ICV), e.g. a cryptographic hash code (see, for example, [MvOV97, Chapter 9]) on the root public key store. The ICV can be recomputed at any time and compared with the previously computed value. If the two values do not match, the tool could alert the user of the fact that changes have been made to the root public key store. However, it would not be possible for the tool to distinguish between a malicious or an innocent insertion of a root public key. Moreover, such a check will not reveal exactly which root public key is causing the check values to be different.

A third possible strategy is to check the status of the certificate online using a protocol similar to the Online Certificate Status Protocol (OCSP). When adding a new root public key to the root public key repository, its authenticity should be checked online. However, a motivated attacker might set up a rogue server to engage in such a protocol and fake the status of the newly added root public key.

A fourth strategy is to use backup tools. Here a backup tool maintains a separate copy of the root public key store. On demand, the backup tool compares the current root public key store with the backup copy and reports any differences. Such a tool could detect newly inserted root public keys and, if required, delete them. It would also be possible for such a tool to restore the root public key store to a previous state.

4 The Scanning Tool

The main objective of a root public key scanning tool is to detect and remove fake root public keys. The scanning tool requires the following two functionalities in order to achieve its objectives.

1. The tool should have access to the root public key store, which holds the root public keys currently installed on the user's PC. The appropriate access right is required to allow the tool to remove fake root public keys.

2. The tool should have some means of distinguishing between 'genuine' and 'fake' root public keys.

A possible technique for distinguishing between 'genuine' and 'fake' root public keys is to maintain a list of known genuine root public keys. The tool compares the list of genuine root public keys with the set of keys found on the user's PC to detect any mismatch. Once a mismatch is found, the scanning tool has detected a 'suspicious' root public key. This technique is the basis of the prototype discussed in Section 5. The scanning tool cannot guarantee that a detected root public key is actually a fake, because users may add their own root public keys. The scanning tool would need a separate list of known fake root public keys in order to be able to mark any key as certainly 'fake'. The list of genuine root public keys could be obtained in various ways. One possible approach would be to bundle with the tool the list of root public keys supplied by the manufacturer of the browser. This list can be updated to include newly added root public keys.

Another technique for distinguishing between 'genuine' and 'fake' root public keys is to maintain an online repository of fake root public keys. The repository is continuously updated with newly discovered fake root public keys. The scanning tool consults the online repository to check the status of a given root public key, to discover whether it is a known fake. The technique mentioned in the previous paragraph can be combined with this technique to achieve better scanning results.

5 A Prototype Implementation

In this section, a prototype implementation of the root public key scanning tool is discussed and analysed[1]. The tool was implemented on the Microsoft Windows XP operating system and the main user interface for the scanning tools is shown in Figure 1. When executed, the tool performs the following steps.

1. Loads a list of 'genuine' root CA public keys from the tool's database.
2. Loads the list of root CA public keys currently installed on the user's PC.
3. Compares the installed list to the 'genuine' list. When an entry that is not present in the 'genuine' root CA public keys list is found, the tool marks it.

The prototype is implemented using Microsoft Visual Basic .NET and the Microsoft Windows Cryptographic Application Programming Interface (CryptoAPI) [Cor04b]. CryptoAPI contains procedures needed to interact with the root public key repository. The main procedures making up the tool are 'LoadGenuineCAs' and 'LoadAndCheckInstalledCAs'. The following paragraphs discuss these two procedures.

The main task of the LoadGenuineCAs procedure is to load the genuine root CA public keys list from a file. The file is created when the tool is installed and it contains a list of thumbprints of the genuine root CA public keys. The thumbprint is a hash-code computed as a function of the certificate. The list of genuine root CA public keys was generated at the time of tool development by importing the current default root CA public keys on a

[1] The tool can be downloaded from http://www.isg.rhul.ac.uk/~cjm/cascan.zip

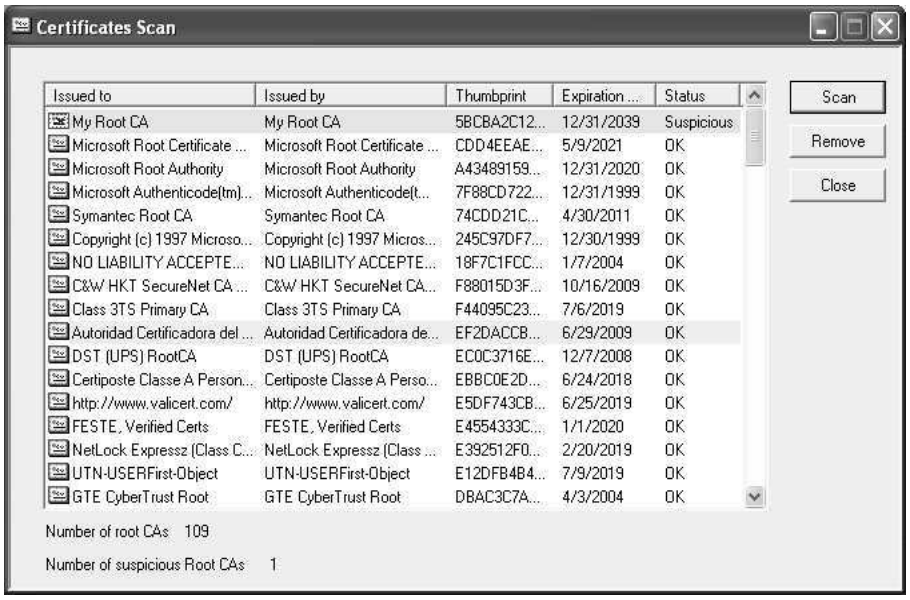

Figure 1: The Scanning Tool main interface

Microsoft Windows platform. Regular updates of the file are required in order to add new genuine CA public keys.

Once the list of genuine root CA public keys is loaded, the LoadAndCheckInstalledCAs procedure is executed and performs the following steps.

1. Open the root public keys store using the 'Open' method of the 'Store' CryptoAPI object, as shown in Figure 2. The 'CertificatesStore' is an instance of the 'Store' object, which is used to obtain the list of installed root public keys on the user PC. Three flags need to be passed to the 'Open' method. The first flag indicates the location of the certificate store. The name of the certificate store is given in the second flag, and the third flag indicates open mode.

2. Once the previous step has been completed, the tool enumerates all installed root CA public keys and search for any root certificate that is not included in the genuine root CA public keys list, as shown in Figure 2. If the tool finds a root certificate that is not in the genuine list, the root certificate is marked as 'suspicious'. The tool uses thumbprints to compare root certificates.

3. The results of the previous steps are displayed to the user, with the suspicious certificates marked. The tool offers the user the possibility to remove a suspicious certificate or display the contents of a certificate.

```
Private Sub LoadAndCheckInstalledCAs()
    Dim CertificatesStore As New CAPICOM.Store

    ......

    CertificatesStore.Open(CAPICOM.CAPICOM_STORE_LOCATION.CAPICOM_CURRENT_USER_STORE,
    CAPICOM.Constants.CAPICOM_ROOT_STORE,
    CAPICOM.CAPICOM_STORE_OPEN_MODE.CAPICOM_STORE_OPEN_READ_WRITE)

    ......

    Dim CertIndex As System.Collections.IEnumerator
    CertIndex = CertificatesStore.Certificates.GetEnumerator()
    While CertIndex.MoveNext()
        If Not (ValidCAs.Contains(Cert.Thumbprint)) Then
            ' the Certificate thumbprint was not found in the
            ' ValidCAs list, mark the certificate as suspicious
        End If
    End While
    ......
End Sub
```

Figure 2: Source code of the Root CA scanning tool

6 Conclusions and Future Work

As discussed and illustrated in this paper, the fake root certificates attack is potentially a serious threat. The single point of trust, i.e. the list of root CA public keys, creates the problem. By default, web browsers trust the list of installed root CA public keys on the user machine without distinguishing between original root CA public keys, i.e. those shipped with the browser, and added root CA public keys. Distinguishing between the two would be useful when the browser is engaged in a secure transaction. When the browser receives a certificate signed by an added root CA, it could alert the user and wait for confirmation before continuing the transaction.

The scanning tool was implemented on the Microsoft Windows operating system and uses Microsoft Windows CryptoAPI services to access the root public key store. It would be possible to enhance the tool to support other browsers and operating systems, e.g. Netscape on Linux.

One limitation of the discussed tool is that, although it can detect fake root public keys, it cannot distinguish between those deliberately added and 'true' fakes. A database of known fake root certificates could be used to help support this functionality. The fake root certificate database could be created by using previously discovered or reported fake root certificates. When a 'suspicious' root certificate is found, the tool would consult the fake root certificate database to search for the 'suspicious' certificate. If it is found in the database, then the tool could guarantee that the root certificate is certainly fake. Another method to overcome this limitation is to create a process to monitor the root public key repository. This process would continuously monitor the root public key repository, and whenever an attempt is made to insert a new root public key, the tool would request user confirmation.

Another limitation of the tool is that it relies on the services provided by the Microsoft

CryptoAPI. Some of the Microsoft CryptoAPI functions require user input to operate. For example, when the user requests the scanning tool to delete a suspicious certificate, the tool makes a call to a CryptoAPI function to delete the certificate. In turn, the CryptoAPI function displays a message box asking the user for confirmation. Implementing a library to interact with the root public key store would be helpful in this situation, and is a topic for further study.

One problem with the scanning tool is that it can be manipulated by an attacker to hide the existence of added root public keys. This problem is shared by all software designed to detect system manipulation, such as antivirus and antispyware packages.

More research is needed on possible means of protecting end users against root key insertion attacks. It may be the case that trusted computing technology [BCP+03] is useful in this context.

References

[AM05] Adil Alsaid and Chris J. Mitchell. Installing Fake Root Keys on a PC. In D. Chadwick and G. Zhao, editors, *EuroPKI 2005*, volume 3545 of *Lecture Notes in Computer Science*, pages 227–239. Springer-Verlag, Berlin, July 2005.

[BCP+03] Boris Balacheff, Liqun Chen, Siani Pearson, David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2003.

[Cor04a] Microsoft Corporation. Certificate creation tool (makecert.exe), May 2004. http://msdn.microsoft.com/.

[Cor04b] Microsoft Corporation. Cryptography, CryptoAPI, and CAPICOM, May 2004. http://msdn.microsoft.com/.

[DA99] Tim Dierks and C. Allen. The TLS Protocol 1.0. RFC 2246, Internet Engineering Task Force, January 1999.

[ES00] Carl Ellison and Bruce Schneier. Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal*, XVI(1):1–7, 2000.

[FB01] Warwick Ford and Michael S. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures & Encryption*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2001.

[FBDW97] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. In *Proceedings of 20th National Information Systems Security Conference*, pages 95–103, October 1997.

[MvOV97] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press, Boca Raton, Florida, 1997.

[NDJB01] Andrew Nash, William Duane, Celia Joseph, and Derek Brink. *PKI: Implementing and Managing E-Security*. Osborne/McGraw-Hill, Berkeley, California, 2001.

[YYS02] E. Ye, Y. Yuan, and S. Smith. Web Spoofing Revisited: SSL and Beyond. Technical Report TR2002-417, Dartmouth College, Computer Science, Hanover, NH, February 2002.