Windows Software Publishing Trust Provider

February 1996

Version 0.8

DRAFT

This document defines the behavior of the Software Publishing Trust Provider, a module which supports selected actions under the WinVerifyTrust() API. It is a companion to the Windows Trust Verification Services Specification and should be read after that document. The Software Publishing Trust Provider allows a calling application to determine whether a software component contains digital signatures which identify it as being authentic software released by a publisher trusted on the local user's system.

Contents

Introduction

The Published Software trust provider is a module for handling specified actions under the WinVerifyTrust() API (q.v.). This module supports verification of the trustability of software components, by interpreting local system rules and analyzing cryptographic material associated with those software components, which may either be embedded directly in those modules to be analyzed, or previously installed into a system database for later retrieval.

The initial implementation of the Published Software trust provider described herein provides relatively user-friendly security with weak enforcement, allowing users to install any software desired on their systems, but requiring user interaction prior to installation of software not determined to be trustable by the available cryptographic material. Stronger implementations which support additional actions that can be used to more closely guard the entrance of new software modules into the Trusted Computing Base may be provided at a future date.

This trust provider makes use of X.509 version 3 certificates and PKCS #7 digital signature structures, and the reader is assumed to have working knowledge of these standards. See the References section at the end of this document for additional information about these structures.

Additionally, the trust provider uses specific X.509 version 3 extensions and particular signed attributes within PKCS #7 structures that have been defined for the purpose of signing and authenticating software. These are documented in a separate specification also referred to in the References section below.

Interfaces and Data Structures

The Published Software trust provider is called by using the Win32 API WinVerifyTrust(), documented in the Windows Trust Verification Services specification. Recall that this function has the function prototype:

```
HRESULT
WINAPI
WinVerifyTrust(
      HWND                    hwnd,
      DWORD                   dwTrustProvider,
      DWORD                   dwActionID,
      LPVOID                  ActionData,
      );
```

The Published Software trust provider is identified by the WINBASE.H constant:

```
      #define WIN_TRUST_SOFTWARE_PUBLISHER    (1L)
```

Calls where this value is passed for the dwTrustProvider parameter are handled by the Published Software trust provider. The trust provider supports two actions (passed in the dwActionID parameter) at this time, defined as:

```
      #define WIN_SPUB_ACTION_TRUSTED_PUBLISHER      (1L)[1]
      #define WIN_SPUB_ACTION_PUBLISHED_SOFTWARE     (3L)
```

In the initial implementation, the Published Software trust provider is also the system default provider for these actions. Therefore, if the caller specifies in the dwTrustProvider the constant:

```
      #define WIN_TRUST_PROVIDER_UNKNOWN    (0L)
```

and the action is either of those defined above, then the call will also be handled by this trust provider. This is the usual expected calling method that most applications will employ, since it allows later system configuration options which permit system or network administrators to assign various more or less stringent trust providers to handling these actions without requiring modification to the applications.

ActionData and Subject Structures

For the action WIN_SPUB_ACTION_PUBLISHED_SOFTWARE, the ActionData parameter is expected to be a pointer to a data structure of type WIN_TRUST_ACTDATA_SUBJECT_ONLY.

```
typedef LPVOID WIN_TRUST_SUBJECT

typedef struct _WIN_TRUST_ACTDATA_SUBJECT_ONLY {

    DWORD              dwSubjectType;
    WIN_TRUST_SUBJECT Subject;

} WIN_TRUST_ACTDATA_SUBJECT_ONLY , *LPWIN_TRUST_ACTDATA_SUBJECT_ONLY
```

[1] In the Alpha development release of Sweeper provided at the Internet Professional Developers' Conference in March, 1996, this action will not be supported.

In this structure, the dwSubjectType identifies the format of the file or object being verified, and the Subject is a pointer to a structure which provides information for reading or manipulating the object. Valid values for the subject type include:

```
#define WIN_TRUST_SUBJTYPE_RAW_FILE       (0L)
#define WIN_TRUST_SUBJTYPE_PE_IMAGE       (1L)
```

The Subject structure for either of these types is of the form:

```
typedef struct _WIN_TRUST_SUBJECT_FILE {

    HANDLE  hFile;
    LPSTR   lpPath;

} WIN_TRUST_SUBJECT_FILE, *LPWIN_TRUST_SUBJECT_FILE;
```

and more detail about this structure is provided in the Windows Trust Verification Services specification.

Verification Algorithm

When the WIN_SPUB_ACTION_PUBLISHED_SOFTWARE action is invoked, the subject is inspected to see if it contains a PKCS #7 signed data structure. If it does, the structure is expected to contain a chain of X.509 certificates. In this development release, a root certificate must be present which is a self-signed certificate containing a hard-coded test root public key and signed by the root private key. Additionally, a certificate must be present signed by the root private key and identifying a software publisher's public key. Finally, the PKCS #7 signed data structure must contain an ExternalData signed attribute which contains a digest of the subject being verified.
In the future, each certificate will additionally be inspected to ensure that it contains appropriate X509.3 extensions defining key-usage restrictions and other attributes of the certified parties.
If these conditions are met, the API returns SUCCESS to the calling application.
Otherwise, if an hwnd was provided, a user interface is displayed providing information obtained from any valid certificates present, if available, and containing the path of the subject file in any case. If the user confirms acceptance of the file, the API also returns SUCCESS, and otherwise, it returns a distinguished error code.
Full documentation of error conditions and corresponding return codes will be included in a later version of this document.