

and with the list of resources stored in the second field. The privilege defines an access right of the application program **104** to access each resource in the list of resources.

[0035] An author of the application may create a manifest such as manifest **108** with a trust information section. An application author may also assign a strong name to the application and sign the application's manifest (e.g., with a digital signature or certificate). When an application is installed, the operating system **102** may be configured to check one or more certificate stores to validate the certificate and signature of the application manifest. In one embodiment, only signed driver packages are installed. For example, an enterprise may have its own certificate store. Similarly, a particular system may have a certificate store against which an application manifest may be validated. Once validated, the operating system **102** may be configured to manage trust actions based on the manifest data and pre-configured default policies.

[0036] A manifest such as manifest **108** may be signed in several ways. For example, manifests may be signed using an authenticode process with the certificate kept in a store for verification. Domain administrators may also sign manifests for their particular enterprise or domain. For example, a deployment manifest may be used to specify which applications are signed for a particular installation. Local administrators may also sign manifests. Each individual machine may also be configured with a signing key.

[0037] In one embodiment, the manifest **108** may include both a weak name and a strong name. The weak name may correspond to a traditional application file name, while the strong name may correspond to the file name, version number, culture, and public key. In another embodiment, the strong name may be a hash of the module signed private key. In yet another embodiment, the strong name may be a public key token.

[0038] For example, the following XML may represent one strong name for the manifest **108**.

```
<assemblyIdentity
  version="1.0.0.0"
  processorArchitecture="x86"
  name="SampleApp"
  publicKeyToken="0123456789abcdef"
  type="typeA"/>
```

[0039] The following is a sample trust information section of one embodiment of the manifest **108**.

```
<trustInfo>
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel
        leastPrivileged="true"
        adminPrivileged="true"
        requireDefaultDesktop="false"/>
    </requestedPrivileges>
  </security>
</trustInfo>
```

[0040] For applications that do not have the manifest **108**, the operating system **102** may be configured to generate the

manifest **108** with requested privileges set according to a predefined default. For example, the manifest **108** may be configured to request the least privileged level of user access.

[0041] Alternatively or in addition, the operating system may also observe the actions of an application and customize a manifest to provide only the privileges that the application actually uses. After a number of executions of the application, the manifest is locked and explicit user input or administrator policy is required to extend the privileges granted by the manifest. In some embodiments, the likelihood that a vulnerable application will be compromised soon after installation is relatively low compared to the possibility that the application will be compromised later. If the application is compromised after the manifest is locked, the behavior of the compromised application is limited to the behaviors allowed by the manifest, which were determined by the uncompromised behavior of the application.

[0042] Methods for protecting resources using application identities are next described.

[0043] Providing Access Control

[0044] Referring next to **FIG. 2**, an exemplary flow chart illustrates operation of an access control method. In one embodiment, the invention grants an application program access to a resource on a computing system. The method includes receiving a request from an application program for access to a resource identified in the request at **202**, determining an application identifier for the application program at **204**, identifying a privilege from a manifest (e.g., the operating system manifest and/or the application program manifest) as a function of the determined application identifier and the identified resource at **206**, and granting the application program access to the identified resource according to the identified privilege at **208**. In one embodiment, determining the application identifier for the application program at **204** includes tagging every file, folder, system setting change (e.g., registry key and value) or resource with a unique, consistent, persistent, repeatable identifier.

[0045] In one embodiment, an operating system executes the method illustrated in **FIG. 2**. In another embodiment, an application program or service separate from the operating system executes the method illustrated in **FIG. 2**. One or more computer-readable media have computer-executable instructions for performing the method illustrated in **FIG. 2**.

[0046] Various exemplary privileges or other forms of access are next described with reference to a sample mitigation architecture for protecting resources.

[0047] Exemplary Mitigation Architecture

[0048] Referring next to **FIG. 3**, an exemplary flow chart illustrates a mitigation architecture for protecting various resources. In one embodiment, the method illustrated in **FIG. 3** enforces the resource privileges described in the manifest based on the application identifier of the application program attempting to access the resources. While certain privileges are described herein, various privileges, levels of privilege, or access not described herein are within the scope of the invention. Likewise, while certain resources are described herein, the various resources not described herein are within the scope of the invention.