

utility may also pre-define certain parameters and associated parameter values based on information the transform utility has regarding the target platform, such as an operating system specific version. Characters used to identify the start and stop of parameter reference strings may also be defined for a specific implementation. For example, illustrative embodiments in this example use the “\$” characters to introduce the parameter reference strings but these may be altered as needed.

With reference to FIG. 8, a flowchart of a process using the privilege manager of FIG. 3, in accordance with illustrative embodiments is shown. Process 800 is an example of a process using privilege manager 302 of FIG. 3.

Process 800 starts (step 802) and receives transformation request (step 804). The transformation request includes a role of the requester that initiates the transformation. Processing of the transformation request by privilege manager 302 of FIG. 3 only transforms the role-based access control information and is not to be considered a full authorization process. Identification of a role associated with the request (step 806) provides the needed role information. From the role information, a determination is made of the set of privileges defined for the identified role (step 808). The set of privileges defined for the role provides the privileges to be conveyed to the role holder.

The process overview comprises determining for each role applicable to a particular target environment; identify each privilege template referenced by the role. For each identified privilege resolve the parameter references based on the target environment, such as operating system type, system name or subsystem identification, by mapping the defined privilege into the corresponding equivalent definition on the target system. When there is no system equivalent definition, generate an error and end. When the source and target system definitions are equivalent, define the target system role-based on the resolved privileges and end.

An environment in which the conveyed privileges are to operate is established by identification of a target environment of the request (step 810). The environment establishes the operational scope of the privileges. A determination is made as to whether the identified target environment matches the specified environment in the defined set of privileges (step 812). When the target environment matches the defined environment, a “yes” is obtained in step 812. When a target environment does not match the defined environment, a “no” is obtained in step 812. When a “no” is obtained in step 812, an error, raised (step 820). The error is typically the result of an inability to perform an equivalence mapping between source and target representations. Notification of the error is made to the requester (step 822), and process 800 terminates thereafter (step 818). The mismatch of the definitions or incomplete definitions is cause for the operation to stop because a valid transform to the target environment cannot be performed as requested.

When a “yes” is obtained in step 812, a mapping of the parameterized privileges in the privilege templates is made to the target environment (step 814). The mapping transforms the platform or environment independent definitions in the privilege templates and the roles to platform or environment specific forms as needed. The request is then performed in the target environment (step 816), with process 800 terminating thereafter (step 818).

Using the capabilities described in the illustrative embodiments allows for the platform or environment independent definitions of privileges and roles to be more efficiently used. The hierarchical structure of the privilege templates and role definitions allow for the fine-grained specifications to be

made while reducing the occurrence of redundant information. The implementation of the privilege templates provides an intermediate form of data between the specification of the role and the privileges to be conveyed. The intermediate form may then be combined to produce a combination of definitions as needed. The privilege templates also provide a low-level or granular definition for easier and more specific implementation of privileges.

The flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products, according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose, hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the invention. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms “comprises” and/or “comprising,” when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

The corresponding structures, materials, acts, and equivalents of all means or step plus function elements in the claims below are intended to include any structure, material, or act for performing the function in combination with other claimed elements, as specifically claimed. The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the invention. The embodiment was chosen and described in order to best explain the principles of the invention and the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments, with various modifications as are suited to the particular use contemplated.

The invention can take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment containing both hardware and software elements. In a preferred embodiment, the invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

Furthermore, the invention can take the form of a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by, or in connection with, a computer or any instruction execution system. For the purposes of this description, a computer-usable or computer-readable medium can be any tangible