From: Randy Turner <rturner@amalfisystems.com>
Date: August 17, 2008 4:04:58 PM PDT
To: Paul Sangster <Paul_Sangster@symantec.com>
Cc: nea@ietf.org
Subject: Re: [Nea] proposals for attribute categories and attributes, etc.

Hi Paul,

Per your request, I'm forwarding along a proposal we discussed earlier for attribute categories, and corresponding attributes, as well as
a some proposed model/data-type ideas for the WG to consider.

NEA list members:
I tried to format this in as simple an ASCII format as possible (spaces for tabs, etc.). Let me know if there is a problem with readability in certain
email clients...

Thanks!
Randy

This proposal introduces new attribute categories and corresponding attributes, and also suggests a new model for managing software on a particular computing device. The text of this proposal originates from evolving requirements being developed by the Printer Working Group (PWG).

PROPOSED CATEGORIES

==========
"System" Category
==========

The "System" category would serve as a "container" for attributes that are used by core operating system services in the device.
One corollary for this type of category is the "system" group of MIB-2 (RFC 1213).

The following proposed attribute definitions would exist within the "System" category:

"Forwarding Enabled" - a single-bit field or boolean value that indicates whether the system is performing any forwarding of "bits" or any kind of electronic transmissions between interfaces.

"Secure Time Enabled" - a single-bit or boolean value that indicates if the device is configured to acquire the current time in a secure manner. If the device is using something as simple as SNTP, then the device would set this value to "False".

"Time Source" - A variable length field that indicates the source from which the device acquires its' notion of the current date and time. This could be a URL of an SNTP or NTP time source, or could be some fixed identifier that indicates that the time is obtained from an onboard clock/calendar.

"Minimum Cipher Suite" - A variable length string that represents one of the enumerations listed in the IANA "TLS Cipher Suite Registry". An example value would be: "TLS_RSA_WITH_AES_256_CBC_SHA256"

"Configuration State" - attribute is a 32 byte field that uniquely identifies the state of any configuration settings in the device that are included in creation of the attribute. A change to any configuration setting that is included in the creation of the attribute MUST cause a change in this attributes value.
The configuration settings included as part of this attribute SHOULD be administratively configurable. The rationale for this single attribute is to allow device vendors to utilize an industry standard attribute to reflect an arbitrary device configuration, consisting of whatever device-specific information the vendor wishes to include. If for some reason, a vendor did not want to publish these attributes, they can still utilize standards-compliant applications to detect invalid configurations and to potentially remediate the situation. The 32-byte field was chosen to allow the attribute value to be a 256-bit hash over the arbitrary configuration. This field would of course have to be enlarged to support SHA-512 or some other hash that produces a value larger than 256 bits.

==========
"HCD" Category
==========

The "HCD" category would serve as a container for attributes that are specific to "Hard Copy Devices", which could be a very low-end printer, or a very high-end multi-function device (fax, scan, print, etc.).

"PSTN_Fax_Enabled" - a single-bit or boolean value that indicates whether or not the PSTN Fax interface is enabled.

"Admin_PW_Enabled" - a single-bit or boolean value that indicates whether or not the factory default administrator password for the device has been changed to a "site-appropriate" value.

=======
ISSUES:
=======

The Printer Working Group is soliciting the opinion of the NEA working group as to the appropriate SMI location for a potential HCD category SMI sub-tree.

The two alternatives under consideration are

1. The HCD SMI sub-tree would reside within the SMI tree being defined by the NEA working group for the initial "standardized" categories.

or

2. The HCD SMI sub-tree would reside within an existing SMI tree that has been IANA-assigned to the Printer Working Group.

The above attribute proposals also pre-suppose the existence of a "boolean" data type for the wire-encoding/information model for the PA protocol. The PWG is also proposing that this type of attribute data type be supported in the information model (and presumably wire encoding).

Software "Module" Attribute Proposal

The TNC model for trusted software configurations presupposes that all devices are basically PCs, and that the software architectural model is based on a "bios", "operating system", and "application" model.

Since it is reasonable to assume that a network administrator might want to use a single tool for monitoring network device configurations in a topology,
and also assuming that devices other than standard PCs are a part of this topology, this proposal suggests that the idea behind managing software
components should be "moved up" a level of abstraction. Using the right type of abstraction would allow practically any type of device to be supported
in the management of software components, whether these components be "applications", an "operating system", or a "bios".

A software component or "module" instance might be a suitable level of abstraction to allow non-PC devices to be managed in the same way as PC devices.
The software module abstraction would be a complex data type that can be multiply instanced. The complex data type would consist of (at least) 3 pieces of information:

- Module Type (This could be a value indicating OS, Bios, or application, but might not be required at all for remediation.

- Module Vendor - The manufacturer of this particular software module

- Module Name - The name of the module such as "Mac OS X", or "Windows Vista Ultimate"

- Module Version - This could either be a version number, build number, build date and time, or whatever the vendor uses to identify unique versions.

The "module" idea can be either evolved as is, or used to stimulate discussion for an appropriate level of abstraction to represent individual, updateable software components within a device.

The rationale behind supporting non-PC devices is not theoretical, in that there is a "spectrum" of network-connected devices that exist today. The spectrum originating with devices that utilize only a single, monolithic software load module, and end with devices that could have tertiary or that utilize a single, monolithic software load, through devices that support a tertiary structure (like PCs), to devices that utilize a quarternary or even larger number of unique software components.

Using the "module" paradigm would allow all of these architectural permutations to exist simultaneously, and be managed (and remedied) using similar methods.

This is just a first stab at an abstraction that might encompass all classes of devices that wish to utilize the NEA protocols. It is likely that other information may be required of this attribute model.

An example of a monolithic device module would consist of a single-instance
of the module data type:
(Type="System",Vendor="HP", Name="HP Laserjet System",Version="5J2-R2")

A typical PC would consist of a multiply-instanced attribute or attributes:

(Type="BIOS", Vendor="Phoenix", Name="Phoenix DCore BIOS", Version-"7R2")
(Type="OS", Vendor="Microsoft", Name="Windows Vista Ultimate", Version="SP1")
(Type="APP", Vendor="Symantec", Name="AntiVirus", Version="3.2R2")
(Type="APP", Vendor="Microsoft", Name="IE", Version="7.3")
(Type="APP", Vendor="Symantec", Name="Firewall", Version="4.3")
(Type="CFG", Vendor="Symantec", Name="fwrules", Version="16")

Using the "module" abstraction would even allow the creation, and subsequent
management/remediation of individual, updateable components like firewall
rulesets, which could be updated without necessarily updating the application
itself. NOTE: the "CFG" module type as illustrated above.
