Local DataStore with ZeroConfig

Motivation

Currently the options on EVE-OS downloading images for Applications are either from public data-stores such as Amazon S3, Azure and Docker storage, or from private servers hosted in the enterprises. From EVE-OS point of view, they are all IP endpoints outside the EVE device. There is a need for downloading images within the device itself for the applications on the device to be dynamically launched even when the device is not connected to the external network. In such a user-case, the data-store is hosted on one of the App inside the device, and EVE-OS can access that local data-store to download App images to launch other Applications.

Changes

One change in 'downloader' function on EVE-OS is needed since the normal downloading assumes the data-store is external, and it only tries to search for the management ports as the source of the TCP/IP connection. For this local data-store feature, the TCP/IP source is one of the virtual bridges shared by the EVE-OS and the application.

There also needs to have some indication or configuration that relates the local data-store App on the device to the data-store being used for images to launch other applications.

The Proposal

The proposal is to use the ZeroConfig or mDNS protocol functionality to make the automatic connection between the data-store configuration and the local data-store App without statically configuring the IP address to either of them. The example of mDNS protocol is the Bonjour feature on Apple devices to discover dynamically the local services such as printers, video streaming, etc. It has been standardized in RFCs and there are many open source applications on Linux and window OSes to support that.

When configuring the data-store for this feature, the FQDN part needs to have the '.local' domain. For example, if the local data-store App VM's host-name is **ubuntu-4321-http-server**, then the FQDN string will be: http://ubuntu-4321-http-server.local. No other special requirement on data-store configuration for this feature.

The local data-store App on EVE device needs to setup ZeroConfig to support this feature. If Ubuntu is used for this local data-store VM, the 'Avahi' can be used to support the mDNS queries. The local data-store App also needs to run normal Apache server for example to support the downloading of images, but those are the normal http server functionalities.

When the EVE-OS 'downloader' module is instructed to download an image for another App on the device from the data-store, if the App's data-store configuration has the FQDN in the '.local' domain, it knows this data-store is local to the EVE device. The 'downloader' module will send out mDNS query packets on all the internal bridges to multicast address 224.0.0.251 and port 5353 (as specified in [RFC 6762] [RFC 6763]) to ask for service information. Since the local data-store App is on one of those bridges, and the App is running 'Avahi' and is configured to respond to those mDNS queries, it will reply on the bridge to the EVE-OS side with the service running on it's host-name and IP address. The 'downloader' module needs to match the host-name in the reply to the data-store FQDN string, and in this example it is 'ubuntu-4321-http-server', and get the IP address from the reply, e.g. 10.1.0.5, and this IP address will be used for the TCP/IP endpoint of downloading the image. The 'downloader' module will use the URL of 'http:/10.1.0.5' in replacing the FQDN string defined in the data-store configuration and it finds the source IP address on the same internal bridge for the local endpoint of connection. The IP address of the local data-store App can change after reload but this local downloading functionality works the same.

There is no change to the controller side of the code and no API change for this feature.

The github.com/grandcat/zeroconf is imported to support the mDNS query function on the EVE-OS side.

It is possible to use the same local data-store config and the same App on multiple devices to download the same or different sets of images.

Some Details

mDNS and Services

The purpose of ZeroConfig or mDNS is to discover the IP services locally on the network. It extends the DNS protocol to support mDNS features. Unlike normal DNS query is to ask which IP address is for the domain name 'ubuntu-4321-http-sever.local', the mDNS query is to multicast the question on the LAN for services. For our local data-store application usage of mDNS, hopefully the 'ubuntu-4321-http-server' host will reply to the service query and supply the IP address information along. There are many services defined in the IANA registry, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt, and in this feature we will use 'workstation', 'http' and 'https'. At least one of them needs to be configured to advertise in the local data-store VM side. There is no default service enabled even the 'Avahi' is running. The EVE-OS side does not really care about which services will reply, it has the local domain name, and it needs to have a mapping to the IP address it can use for downloading remote endpoint, and any service reply for the mDNS query will have the host-name and IP address.

The 'downloader' will query the service 'workstation' first ('_workstation._tcp.local' service), wait for several seconds, it no reply or if the replies don't match, it will further query 'http' and then 'https' services.

App Side mDNS Setup

If Ubuntu is the OS for the App, and if 'avahi-daemon' is not already running, checkout https://www.howtoinstall.me/ubuntu/18-04/avahi-daemon/.

The configuration is in /etc/avahi/avahi-daemon.conf, and the following changes are recommended (assume again the App host-name is 'ubuntu-4321-http-server'). If there is no other mDNS services already been enabled, this 'publish-workstation=yes' must be configured:

host-name=ubuntu-4321-http-server

use-ipv6=no

publish-workstation=yes

then restart the 'avahi-daemon' if it is already running, by issuing "sudo service avahi-daemon restart". Note that even if the host-name is already defined in the Linux, make this 'host-name' explicit in 'Avahi' to match the data-store FQDN domain name will make sure it will still work even the host-name is changed by other services in the App. The 'use-ipv6=yes' is the default, and people have reported issues when 'Avahi' sometimes only gets the IPv6 address without wait for an IPv4 address and returns the reply, so this 'no' makes the reply have the IPv4 address always. The 'publish-workstation=yes' is to support the service of 'workstation'.

User can optionally setup 'http' or 'https' services, XML files need to be created in /etc/avahi/services directory. For instance, an 'http.service' file may contain:

<?xml version="1.0" standalone='no'?><!---nxml-*-->*

<!DOCTYPE service-group SYSTEM "avahi-service.dtd">

<service-group>

<name replace-wildcards="yes">%h</name>

<service>

<type>_http._tcp</type>

<port>80</port>

</service>

</service-group>

This allows 'avahi-daemon' to advertise http service when receiving query on '_http._tcp.local' service. But if the 'workstation' is already enabled in the above config, this 'http' service is not mandatory for this feature. The same applies to the 'https' service.

At least one of the 'workstation' or 'http' or 'https' services has to be defined in the 'Avahi' for this local datastore feature to work.

Verify the App Side mDNS advertisement

An easy way to see if the App is running 'avahi-daemon' and advertising service/hostname correctly, One can issue "ps aux | grep 'avahi-daemon' " to see if it is running. Or one can issue "sudo systemctl status avahi-daemon" to see the 'avahi' status. Then install the 'avahi-browse' program by "sudo apt install avahi-utils". The 'avahi-browse -a" command output should display the hostname, service, etc info:

```
ubuntu@ubuntu:/etc/avahi$ avahi-browse -a | grep server
+ eth0 IPv4 ubuntu-4321-http-server [00:16:3e:00:01:03] Workstation local
+ lo IPv4 ubuntu-4321-http-server [00:00:00:00:00:00] Workstation local
```

Local DataStore Protocol Scheme

Since the EVE-OS 'downloader' module uses IP address in URL to fetch the images from the local Datastore, and also the datastore is internal to the device and under control by the user/enterprise, the protocol scheme should be **HTTP.** The downloaded images are verified by the datastore specified hash value for integrity of the image. In the case the HTTPS scheme needs to be used for the local datastore, the user needs to make sure the local datastore App interface IP address is statically assigned on the Network Instance configuration (so it will not change after reboot, etc.) and also the certificate's 'subjectAltName' needs to include this interface IP address, and the certificate chain is included in the local datastore configuration for EVE-OS to verify during the downloading operation.