목차

UEFI Secure boot in Libvirt

Support for UEFI Secure Boot is one of the features planned for the Wallaby release of the OpenStack Compute project, Nova. Nova has supported UEFI for instances via the libvirt virt driver since the Mitaka release (nova 13.0.0) and it is in fact required to boot AArch64 (ARM64) guests, however, how this has been implemented leaves a lot to be desired. The introduction of Secure Boot functionality has given us the opportunity to clean up some of the tech debt around this feature.

UEFI support in QEMU and libvirt

Naturally, for nova's libvirt virt driver to support UEFI, both libvirt and QEMU need to support it. This has been the case for many years, but recent versions of libvirt and QEMU have made working with UEFI significantly easier than previously. As noted in the [Secure Boot spec][0], libvirt 5.3 introduced support for the firmware auto-selection functionality provided by QEMU since QEMU 2.9. This QEMU feature relies on firmware JSON files that describe what each firmware file is for and how it can be described, as described in the QEMU spec. These files are typically provided by your distro and on a Fedora 33 host can be found at /usr/share/qemu/firmware. Here's one such file, 40-edk2-ovmf-x64-sb-enrolled.json, taken from my host and provided by Fedora's edk2-ovmf package:

```
{
    "description": "OVMF for x86_64, with SB+SMM, SB enabled, MS certs
enrolled",
    "interface-types": [
        "uefi"
    ],
    "mapping": {
        "device": "flash",
        "executable": {
            "filename": "/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd",
            "format": "raw"
        },
        "nvram-template": {
            "filename": "/usr/share/edk2/ovmf/OVMF_VARS.secboot.fd",
            "format": "raw"
        }
    },
    "targets": [
        {
            "architecture": "x86_64",
            "machines": [
                "pc-q35-*"
            ]
        }
    ],
    "features": [
        "acpi-s3",
```

```
        "amd-sev",
        "enrolled-keys",
        "requires-smm",
        "secure-boot",
        "verbose-dynamic"
    ],
    "tags": [

    ]
}
```

Since libvirt 5.3, libvirt has parsed these files and included them in the domain capabilities output, accessible via virsh domcapabilities and equivalent APIs. For example, I can see what firmwares are available to me when using the q35 machine type like so:

```
$ virsh domcapabilities --machine pc-q35-5.1 | xmllint --xpath
'/domainCapabilities/os' -
```

This will yield:

```
<os supported="yes">
  <enum name="firmware">
    <value>efi</value>
  </enum>
  <loader supported="yes">
    <value>/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd</value>
    <value>/usr/share/edk2/ovmf/OVMF_CODE.fd</value>
    <enum name="type">
      <value>rom</value>
      <value>pflash</value>
    </enum>
    <enum name="readonly">
      <value>yes</value>
      <value>no</value>
    </enum>
    <enum name="secure">
      <value>yes</value>
      <value>no</value>
    </enum>
  </loader>
</os>
```

However, this isn’t all this can do. Libvirt can also negotiate the firmware for you when creating a new guest. Previously, to create a UEFI-based guest, one would need to specify something like the following:

```

```

```
<domain type="kvm">
  <!-- ... -->
  <os>
    <type arch="x86_64" machine="pc-q35-5.1">hvm</type>
    <loader readonly="yes" secure="no"
type="pflash">/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd</loader>
    <nvram
template="/usr/share/edk2/ovmf/OVMF_VARS.secboot.fd">/home/stephenfin/.confi
g/libvirt/qemu/nvram/q35-uefi-experiment_VARS.fd</nvram>
    <boot dev="hda"/>
  </os>
  <!-- ... -->
</domain>
```

I won't go into the specifics of what each file means - libvirt has some good documentation on the matter

This is no longer necessary, and libvirt will now do this for us. We can instead specify e.g.:

```
<domain type="kvm">
  <!-- ... -->
  <os firmware="efi">
    <type arch="x86_64" machine="pc-q35-5.1">hvm</type>
    <loader secure="no"/>
    <boot dev="hda"/>
  </os>
  <!-- ... -->
</domain>
```

How helpful! Using this, it's now easier than ever to create guests using UEFI.

Example

The above is all well and good, but a worked example is even better. Let's create a Fedora 33 Workstation guest with UEFI enabled to demonstrate this. This should work with any OS (note: though maybe not right now, as discussed later) but using the Fedora 33 Workstation live image, without any other drives attached, let's us minimise the amount of irrelevant XML present. You can download the Fedora 33 Workstation image from the Fedora website, but before we do that, let's start by ensuring that we have all required packages installed. This should be as simple as installing libvirt, at least on a Fedora 33-based host:

```
$ sudo dnf install libvirt
```

It should also be noted that this guide was only tested on an x86_64 host and results may vary on other platforms.

Next up, let's grab the image. As discussed above, we're going to use the Fedora 33 Workstation image here since it's easy to use for testing purposes:

```
$ cd /tmp
$ wget
https://download.fedoraproject.org/pub/fedora/linux/releases/33/Workstation/
x86_64/iso/Fedora-Workstation-Live-x86_64-33-1.2.iso
```

With the dependencies installed and the image downloaded, we can now create our guest or "domain". Dump the following to e.g. /tmp/fedora-q35-uefi.xml:

```
<domain type="kvm">
  <name>fedora-q35-uefi-experiment</name>
  <metadata>
    <libosinfo:libosinfo
xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://fedoraproject.org/fedora/33"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">4194304</memory>
  <currentMemory unit="KiB">4194304</currentMemory>
  <vcpu>2</vcpu>
  <os firmware="efi">
    <type arch="x86_64" machine="pc-q35-5.1">hvm</type>
    <loader secure="no"/>
    <boot dev="cdrom"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state="off"/>
  </features>
  <cpu mode="host-model" check="partial"/>
  <clock offset="utc">
    <timer name="rtc" tickpolicy="catchup"/>
    <timer name="pit" tickpolicy="delay"/>
    <timer name="hpet" present="no"/>
  </clock>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type="file" device="cdrom">
      <driver name="qemu" type="raw"/>
      <source file="/tmp/Fedora-Workstation-Live-x86_64-33-1.2.iso"/>
      <target dev="sda" bus="sata"/>
      <readonly/>
    </disk>
    <serial type="pty">
      <target type="isa-serial" port="0">
        <model name="isa-serial"/>
```

```
      </target>
    </serial>
    <console type="pty">
      <target type="serial" port="0"/>
    </console>
    <graphics type="vnc" port="-1" tlsPort="-1" autoport="yes">
      <image compression="off"/>
    </graphics>
    <video>
      <model type="virtio"/>
    </video>
  </devices>
</domain>
```

Finally, create the instance:

```
$ virsh create fedora-q35-uefi.xml
```

Since we're using the workstation image, you're going to need a graphical console. As a result, you'll note that we've attached a VNC graphics device to the instance and you can use VNC to connect to the guest. The easiest way to do this is via virt-manager (use the QEMU/KVM User Session connection), but of course you could also use virsh dumpxml fedora-q35-uefi-experiment after creating the guest to view the final XML and get the port that libvirt assigned to the VNC interface, and then use this to connect to the instance from your favourite VNC viewer.

Once you're done, destroy the guest:

```
$ virsh destroy fedora-q35-uefi-experiment
```

Secure boot woes

I'd be remiss if I didn't highlight one issue with this approach, at least using the versions of libvirt (6.6.0) and QEMU (5.1.0) installed on my host at the time of writing. My initial attempts at this didn't use Fedora but rather Alpine Linux, a distro many will be familiar with from Docker containers. In theory this should be a good candidate for playing around with since it's small and well suited to use in guests with limited CPU and memory. However, the Alpine Linux images are not signed and this prevents booting of the guest. To demonstrate the issue and explain why it occurs, it's probably best to go with another example.

To work through this, let's first grab the Alpine Linux Virtual image, available from the Alpine Linux website:

```
$ wget
https://dl-cdn.alpinelinux.org/alpine/v3.13/releases/x86_64/alpine-virt-3.13
.1-x86_64.iso
```

The domain XML we'll use for booting this image is virtually identical to the one used for Fedora previously. We could of course use the same specs as the Fedora 33-based guest and it would work just fine, but we've chosen to remove the VNC graphics device and display adapter, which are unnecessary for a distro like this, as well as reduce the RAM allocation. Dump the following XML to e.g. /tmp/alpine-q35-uefi.xml:

```
<domain type="kvm">
  <name>alpinelinux-q35-uefi-experiment</name>
  <metadata>
    <libosinfo:libosinfo
xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://alpinelinux.org/alpinelinux/3.13"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">1048576</memory>
  <currentMemory unit="KiB">1048576</currentMemory>
  <vcpu>1</vcpu>
  <os firmware="efi">
    <type arch="x86_64" machine="pc-q35-5.1">hvm</type>
    <loader secure="no"/>
    <boot dev="cdrom"/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state="off"/>
  </features>
  <cpu mode="host-model" check="partial"/>
  <clock offset="utc">
    <timer name="rtc" tickpolicy="catchup"/>
    <timer name="pit" tickpolicy="delay"/>
    <timer name="hpet" present="no"/>
  </clock>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type="file" device="cdrom">
      <driver name="qemu" type="raw"/>
      <source file="/tmp/alpine-virt-3.13.1-x86_64.iso"/>
      <target dev="sda" bus="sata"/>
      <readonly/>
    </disk>
    <serial type="pty">
      <target type="isa-serial" port="0">
        <model name="isa-serial"/>
      </target>
    </serial>
    <console type="pty">
      <target type="serial" port="0"/>
```

```
    </console>
  </devices>
</domain>
```

With the image downloaded and domain XML file created, we can create the guest:

```
$ virsh create alpine-q35-uefi.xml
```

Since this isn't using a graphical device, you can instead connect via the serial console:

```
$ virsh console alpinelinux-q35-uefi-experiment
```

However, upon connecting you'll note that you eventually end up at the bootloader rather than a login prompt and if you connect early enough, you'll probably see something like the following:

```
BdsDxe: loading Boot0001 "UEFI QEMU DVD-ROM QM00001 " from
PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)
BdsDxe: failed to load Boot0001 "UEFI QEMU DVD-ROM QM00001 " from
PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0): Access Denied
BdsDxe: No bootable option or device was found.
BdsDxe: Press any key to enter the Boot Manager Menu.
```

(You can use Ctrl +] to quit the console).

The apparent root cause for this can be seen by investigating the domain XML:

```
$ virsh dumpxml alpinelinux-q35-uefi-experiment | xmllint --xpath '//os'
```

```
<os>
  <type arch='x86_64' machine='pc-q35-5.1'>hvm</type>
  <loader readonly='yes' secure='no'
type='pflash'>/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd</loader>
  <nvram
template='/usr/share/edk2/ovmf/OVMF_VARS.secboot.fd'>/home/stephenfin/.confi
g/libvirt/qemu/nvram/alpinelinux-q35-uefi-experiment_VARS.fd</nvram>
  <boot dev='cdrom'/>
</os>
```

You'll note that we requested secure='no' and libvirt has included this in the final XML, however, UEFI firmware with secure boot support, OVMF_CODE.secboot.fd, has been used, rather than OVMF_CODE.fd. In theory, this firmware should work for non-secure boot instances also but that's not happening. The solution for now is to specify the path to the non-secure boot UEFI firmware when creating the instance, replacing the <os> element included in the XML above with the following:

```
<os>
  <type arch='x86_64' machine='pc-q35-5.1'>hvm</type>
  <loader readonly='yes'
type='pflash'>/usr/share/edk2/ovmf/OVMF_CODE.fd</loader>
  <boot dev='cdrom'/>
</os>
```

Obviously this is less than ideal. The issue has been reported and will hopefully be resolved sooner rather than later but until then, be aware that secure boot with always be enabled when using this auto-configuration of firmware.

참조링크

- https://that.guru/blog/uefi-secure-boot-in-libvirt/