NV40-B Setup Procedure

The following document will go through the steps required to set up an NV40-B verifier from scratch.

Requirements

- an HPE DL360 Gen 10 server
- a Silicom FPGA card
    - it must be installed in the server in the right side PCIe slot (as viewed from the front of the server)
    - it must be encripted with the EXFO private key
- a network for the iLO port of the HPE server
- a network for one of the built in ethernet ports of the HPE server (can be the same network as iLO)
- a bootable USB key of Kubuntu Desktop x64
- a way to get the files found under http://10.16.128.104/lastmile/ on the server. The easiest way is on the bootable USB key.

Step 1: BIOS settings

Start by accessing the server via the iLO web interface and use the HTML5 console to set up the BIOS.

1. Power up or reboot the server as necessary, then enter the BIOS settings by pressing `F9` when prompted during boot up.
2. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Workload Profile` to `Custom`.
3. Click `Save`.
4. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Server Security` -> `Secure Boot Settings` -> `Attempt Secure Boot` to `Disabled`.
5. Go up one level, then click `Save and Exit` and accept to reboot.
6. Enter the BIOS once again.
7. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Boot Options` -> `Boot Mode` to `Legacy BIOS Mode`.
8. Click `Save and Exit` and accept to reboot.
9. Enter the BIOS once again.
10. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `System Options` -> `Serial Port Options`:
    - `Embedded Serial Port` to `COM1`
    - `Virtual Serial Port` to `COM2`
    - `BIOS Serial Console and EMS`:
        - `BIOS Serial Console Port` to `Auto`
        - `BIOS Serial Console Emulation Mode` to `VT100+`
        - `BIOS Serial Console Baud Rate` to `115200`
        - `EMS Console` to `Disabled`
11. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Processor Options` -> `Processor x2APIC Support` to `Disabled`.
12. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Virtualization Options`:
    - `Intel(R) Virtualization Technology (Intel VT)` to `Disabled`
    - `Intel(R) VT-d` to `Disabled`
    - `SR-IOV` to `Disabled`
13. Set `System Utilities` -> `System Configuration` -> `BIOS/Platform Configuration (RBSU)` -> `Advanced Options` -> `Fan and Thermal Options` -> `Thermal Configuration` to `Increased Cooling`.
14. Set `System Utilities` -> `System Configuration` -> `Embedded RAID 1 : HPE Smart Array P408i-a SR Gen10` -> `Set Bootable Device(s) for Legacy Boot Mode` -> `Select Bootable Logical Drive` -> `Logical Drive 1 (Logical Drive 1) Array A` -> `Set as Primary Bootable Device`.
15. Click `Save and Exit` and accept to reboot.

Step 2: Boot the Kubuntu ISO

Plug in the bootable USB key and power up or reboot the server. When given the choice, pick the option to try Kubuntu.

Step 3: Flash the FPGA card

1. Set up the required tools under Kubuntu:

    1. Get these files on the server if they are not on the USB key:

        ```
        http://10.16.128.104/lastmile/lastmile.tar.gz
        http://10.16.128.104/lastmile/racer_0x1000_0x2306_0x0006.bit
        http://10.16.128.104/lastmile/racer_0x1002_0x2306_0x0001.bit
        ```

        The racer files are the FPGA loads to write on the flash memory of the PCIe card. The 0x1000 file is for the primary flash, and the 0x1002 file is for the failsafe flash.
    2. Open a terminal window (the application is called `Konsole`) and go to the location of the required files above.
    3. Install the build-essential package:

        ```
        sudo apt install -y build-essential
        ```

    4. Extract the lastmile archive:

        ```
        tar -xzf lastmile.tar.gz
        ```

    5. Build and load the lastmile driver:

        ```
        cd lastmile/driver
        make
        sudo ./load_driver.sh
        ```

    6. Build the rawcardtool utility:

        ```
        cd ../rawcardtool
        make
        ```

    7. Verify that everything was successful so far:

        ```
        ./rawcardtool --status
        ```

        You should see similar output to this:

        ```
        rawcardtool v1.3.0
        FPGA version: 1.0.3.0 Type: 0xA5 Model: 0x17 Date: n/a
        MAC addresses:
          00:21:B2:26:1B:40
          00:21:B2:26:1B:41
          00:21:B2:26:1B:42
          00:21:B2:26:1B:43
          00:21:B2:26:1B:44
          00:21:B2:26:1B:45
          00:21:B2:26:1B:46
          00:21:B2:26:1B:47
        Serial number: FB2CGHH@KU15P-2.0436
        PCB version: 2
        Flash Manufacturer Id:  Micron NOR (0x20)
        Flash Memory Type:      N25Q 1V8 (0xBB - 2)
        ```

```
Flash Memory Capacity:  512 Mbit (0x20)
FPGA Temperature 45.44C

RAM Calibration:
RAM Bank 0 calibration passed: OK
RAM Bank 1 calibration passed: OK

Link status:
Port 0: No module present
Port 1: No module present
```

2. Write the primary flash:

   1.
      ```
      ./rawcardtool --flash ../../racer_0x1000_0x2306_0x0006.bit
      ```

   2. At this point, the server needs to be rebooted in order for the FPGA card to work. Unfortunately, that means redoing steps 1 to 6 under server setup above. This is required because the factory PCI BAR size is 128KB but the new FPGA load has a BAR size of 250MB. Reboot the server by writing into the terminal:

      ```
      sudo reboot
      ```

   3. Redo steps 1 to 6 under server setup above in order to load the driver once again.
   4. Verify that the FPGA write was successful:

      ```
      cd rawcardtool
      ./rawcardtool --status
      ```

      You should see similar output to this:

      ```
      Last Mile FPGA Type: 0x1000 (10G) FeatureSet: 2306 revision: 6
      MAC addresses:
      ...
      ```

      > Note that the FPGA Type, FeatureSet and revision correspond to the name of the flash file. If you see the same output as before writing the flash, **STOP**! That means that the card is not properly encrypted, and it loaded the failsafe instead of the primary flash because of that. To continue, the card must be properly encrypted and the rawcardtool status must show the output above. Once that is done, `rawcardtool --status` will output the correct text above.

3. Write the failsafe flash **ONLY IF** the primary flash was successfully verified in the previous step:

   1.
      ```
      ./rawcardtool --flashsafe ../../racer_0x1002_0x2306_0x0001.bit
      ```

   2. Reload the FPGA from the flash we just wrote:

      ```
      cd ..
      ./reload_from_flash.sh failsafe
      ```

   3. Verify that it was successful:

```
cd rawcardtool
./rawcardtool --status
```

You should see similar output to this:

```
Last Mile FPGA Type: 0x1002 (failsafe) FeatureSet: 2306 revision: 1
MAC addresses:
...
```

Step 4: Write the verifier image to disk

1. Get the NV40-B disk image the server if it is not already on the USB key:

```
http://10.16.128.104/lastmile/monolith.raw
```

2. Open a terminal window and go to the location of the disk image file.
3. Write the image on the disk:

```
sudo dd if=monolith.raw of=/dev/sda status=progress
```

Step 5: Reboot into the verifier OS

At this point, the FPGA card is ready, and the disk contains the Verifier OS and software.

1. Shut down the server. Powering off is a necessary step. From the terminal:

```
sudo shutdown now
```

2. Remove the USB key.
3. Power up the server and let it boot normally as an NV40-B.

Step 6: Get a VKR file

All verifiers require an appropriate serial number, as well as 2 unique keys called U-Key and NL-Key. The serial number for the NV40-B consists of `71001` followed by the EXFO device serial number provided by SAP. If the SAP serial number is `101`, then the full serial number must be `710010000101` as the full serial number must be 12 characters. To get a corresponding VKR file:

1. Go to https://stmovkportal.exfo.com/ and log in using your EXFO user name and password. Only authorized users can log in. To request the right to log in, contact martin.johnson@exfo.com
2. Once logged in, generate keys for your verifier using the button on the top right called `(+) Generate keys for a new verifier`
3. Enter the full serial number. If you see the text `The serial number belongs to an existing verifier`, then you must cancel and download the existing keys. This should normally never happen, as SAP should not generate overlapping serial numbers.
4. After you generate new keys, your browser will download a zip file. Open it, and open the `.vkr` file. It is a text file with the information you need for the next step.

Step 7: Re-key the Verifier to assign a unique serial number

The most convenient way to access the NV40-B at this stage is via the Virtual Serial Port. Use your favorite SSH client and connect to the NV40-B using the iLO user, password and IP. If you have issues, it might be because the client is refusing the server's accepted key algorythms as insecure. I add the option `-oHostKeyAlgorithms=+ssh-rsa` to my ssh client, you might need to find the equivalent to yours.

1. Once connected to iLO via ssh, access the VSP with the command: `vsp`
2. At this point you are connected to the NV40-B via serial terminal, and if you press enter when the server has finished booting, you should see a login prompt:

```
login:
```

3. Log in as `admin`/`admin`, then type:

```
enable
```

4. Enter the bench configuration with the command:

```
bench-config
```

5. Re-key the verifier. When prompted, use the full serial number (12 characters starting with `71001`), U-Key and NL-Key:

```
rekey-verifier
```

Congratulations, the verifier is now ready to be tested and delivered.