CVE Patching in Yocto Using Devtool – Step-by-Step Guide

This document provides a structured approach to **patching a CVE in the Yocto Project** using `devtool`. It includes applying a security patch to the **Yocto Linux kernel**, verifying the fix, and ensuring Yocto recognizes the CVE as patched.

Prerequisites

- **Yocto Build Environment** (Poky or custom setup)
- **Devtool installed** (`oe-init-build-env` initializes it)
- **Internet Access** (to fetch patches from upstream sources)

Step 1: Set Up the Yocto Environment

Before starting, set up the Yocto build environment:

```
source poky/oe-init-build-env
```

Verify that `devtool` is available:

```
devtool --help
```

Ensure CVE checking is enabled in `local.conf`:

```
vim conf/local.conf
```

Add:

```
INHERIT += "cve-check"
```

Step 2: Identify the Kernel Recipe

Find the **kernel recipe name** in your Yocto build:

```
bitbake-layers show-recipes | grep linux
```

For **BeagleBone with Yocto**, the kernel is usually `linux-yocto`. Confirm with:

```
bitbake linux-yocto -e | grep ^SRC_URI
```

Step 3: Modify the Kernel Source Using `devtool`

To modify the kernel source, run:

```
devtool modify linux-yocto
```

This extracts the kernel source into:

```
workspace/sources/linux-yocto
```

Move into the extracted kernel source directory:

```
cd workspace/sources/linux-yocto
```

Step 4: Find and Download the CVE Patch

1. **Locate the patch for your CVE**
   Search on:
   - Kernel.org
   - MITRE CVE Database
   - NVD

Download the patch
Suppose the patch is available at:

```
https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/patch/?id=<PATCH_ID>
```

Download it:

```
wget https://git.kernel.org/.../patch.diff -O CVE-2024-56772.patch
```

Apply the patch inside the extracted kernel source:

```
patch -p1 < CVE-2024-56772.patch
```

Verify that the patch has been applied:

```
git diff
```

Step 5: Update the Recipe with `devtool`

Once the patch is applied, update the recipe:

```
devtool update-recipe linux-yocto
```

This automatically:

- **Generates a patch file**.
- **Adds it to the recipe** in:

```
meta-custom-kernel/recipes-kernel/linux/linux-yocto_6.12.bbappend
```

- Updates the `SRC_URI` entry to include the patch:

```
SRC_URI += "file://CVE-2024-56772.patch"
```

Step 6: Verify the Patch in the Recipe

Check that the patch is added:

```
cat meta-custom-kernel/recipes-kernel/linux/linux-yocto_6.12.bbappend
```

You should see:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
SRC_URI += "file://CVE-2024-56772.patch"
INHERIT += "cve-check"
```

Step 7: Rebuild the Patched Kernel

Now, rebuild the kernel with the applied patch:

```
bitbake linux-yocto
```

If everything is fine, rebuild the full image:

```
bitbake core-image-minimal
```

Step 8: Test the Patched Kernel in QEMU

If you are using **QEMU for BeagleBone**, launch the emulated system:

```
runqemu qemuarm nographic
```

Verify that the patched kernel is running:

```
uname -r
```

To check if the patch is applied:

```
dmesg | grep -i kunit
```

Step 9: Verify the CVE Fix

Run the CVE Check Again

After booting into the patched kernel:

```
bitbake linux-yocto -c cve_check
```

If no **unpatched CVEs** appear, the fix is successful.

Check Kernel Logs for Errors

```
dmesg | tail -n 50
```

If the **CVE exploit no longer triggers a crash**, it is successfully patched.

List Unpatched CVEs

To list only unpatched CVEs:

```
cat tmp/log/cve/cve-summary.json | jq '.[] |
select(.fix-status=="Unpatched")'
```

Or using `grep`:

```
grep -B 4 '"fix-status": "Unpatched"' tmp/log/cve/cve-summary.json
```

Step 10: Clean Up After Testing

Once you confirm that the patch is working, **reset `devtool`**:

```
devtool reset linux-yocto
```

This removes the temporary workspace modifications while keeping the patch applied.

Best Practices for CVE Patching in Yocto

1. Always Check for Official Fixes

- Use **Kernel.org**, **NVD**, and **MITRE CVE** databases.
- If an official patch exists in a **newer kernel version**, consider upgrading instead of manually patching.

2. Follow a Consistent Patch Naming Convention

Name patches based on the **CVE ID**:

```
CVE-YYYY-XXXX.patch
```

Example:

```
CVE-2024-56772.patch
```

3. Maintain a Record of Applied Patches

- Document each applied patch in a **changelog file**.
- Keep track of patches inside:

```
meta-custom-kernel/recipes-kernel/linux/files/
```

4. Run `bitbake linux-yocto -c cve_check` After Every Fix

Always verify that the CVE is marked **"Fixed"** in:

```
cat tmp/log/cve/cve-summary.json | jq '.'
```

5. Keep Yocto's CVE Database Updated

Regularly update CVE records to detect new vulnerabilities:

```
bitbake cve-update-nvd2-native
```

6. Use `devtool` for Clean Patching

- Avoid **directly modifying** kernel sources.
- Always use `devtool modify` and `devtool update-recipe` to ensure patches integrate smoothly.

7. Automate the CVE Patch Testing Process

Consider scripting:

```
bitbake linux-yocto -c cve_check | grep "Unpatched"
```

Use **QEMU testing** before deploying to real hardware.

Summary

Step	Command
Set up Yocto build environment	`source poky/oe-init-build-env`
Modify the kernel source	`devtool modify linux-yocto`
Download & apply the patch	`patch -p1 < CVE-2024-56772.patch`
Update the recipe	`devtool update-recipe linux-yocto`
Build the patched kernel	`bitbake linux-yocto`
Boot and test in QEMU	`runqemu qemuarm nographic`
Verify CVE fix	`bitbake linux-yocto -c cve_check`