Node.js OpenSSL Compatibility

When running `ng build` with **Node.js 22.9.0** and **npm 10.8.3**, you encounter the error:

```
error 0308010C digital envelope routines::unsupported
```

This error typically arises due to a compatibility issue between newer versions of Node.js (like 22.9.0, which uses OpenSSL 3.0 or later) and certain cryptographic algorithms relied upon by tools like Webpack, which Angular uses under the hood. Below is a step-by-step guide to fix this issue.

Solution

Step 1: Enable the OpenSSL Legacy Provider The simplest and most immediate fix is to enable the OpenSSL legacy provider by setting the `NODE_OPTIONS` environment variable. This allows Node.js to use older cryptographic algorithms that may no longer be supported by default in OpenSSL 3.0+.

- **For macOS/Linux**: Open your terminal and run:

  ```
  export NODE_OPTIONS=--openssl-legacy-provider
  ng build
  ```

- **For Windows (Command Prompt)**: Open Command Prompt and run:

  ```
  set NODE_OPTIONS=--openssl-legacy-provider
  ng build
  ```

- **For Windows (PowerShell)**: Open PowerShell and run:

  ```
  $env:NODE_OPTIONS="--openssl-legacy-provider"
  ng build
  ```

After setting the environment variable, run `ng build` again. This should resolve the error in most cases by allowing Node.js to process the unsupported routines.

Step 2: Verify and Update Angular CLI (If Needed) If the error persists after Step 1, your version of the Angular CLI might not be fully compatible with Node.js 22.9.0. Updating it to the latest version can help.

- Check your current Angular CLI version:

  ```
  ng --version
  ```

- Update Angular CLI globally:

  ```
  npm install -g @angular/cli
  ```

- Then, try running `ng build` again.

Step 3: Check and Update Project Dependencies (Optional) If the issue still isn't resolved, outdated dependencies in your project might be causing the problem. To address this:

- Open your `package.json` file and review the versions of your dependencies (e.g., `@angular/core`, `@angular/cli`, etc.).
- Update them carefully to their latest compatible versions:

  ```
  npm install
  ```

  or, if you want to update all dependencies:

  ```
  npm update
  ```

- Run `ng build` again to test.

Note: Be cautious with updates, as they might introduce breaking changes. Test your application thoroughly afterward.

Step 4: Downgrade Node.js (Last Resort) If none of the above solutions work, Node.js 22.9.0 might not yet be fully supported by your project's current configuration. Downgrading to a Long-Term Support (LTS) version of Node.js, such as **20.x** or **18.x**, could resolve the issue.

- Uninstall your current Node.js version.
- Download and install an LTS version from the official Node.js website.
- Verify the installed version:

  ```
  node -v
  ```

- Reinstall your project dependencies:

  ```
  npm install
  ```

- Run `ng build` again.

Recommended Approach

Start with **Step 1** (setting `NODE_OPTIONS`), as it's the quickest fix and allows you to keep using Node.js 22.9.0. If that doesn't work, proceed to **Step 2** and **Step 3**. Use **Step 4** only if necessary, as downgrading Node.js might limit access to newer features.

This should resolve the "error 0308010C digital envelope routines::unsupported"and get your `ng build` command working again!

Thought for 9 seconds

> Okay, so the user is encountering an error after running `ng build` in their Angular project. The error message is `error 0308010C digital envelope routines:: unsupported`. I remember that this error is related to the Node.js crypto module, specifically with the OpenSSL library.
>
> First, I need to figure out why this error is happening. It might be due to a recent update in Node.js or OpenSSL. I recall that Node.js versions beyond 17.x started using the newer OpenSSL 3.x by default, which might have some breaking changes. The error code 0308010C suggests it's an issue with the digital envelope routines, which are part of the crypto module.
>
> I should check if the project is using environment variables to specify the OpenSSL provider. If the project is using `NODE_OPTIONS=--openssl-legacy-provider`, that might be causing the issue. I can suggest removing or modifying that environment variable.
>
> Another thought is whether the project has any dependencies that might be incompatible with the newer OpenSSL version. Maybe some packages were built against an older OpenSSL and are now causing issues. Updating or downgrading those dependencies could help.
>
> If the user is using a specific version of Node.js, they might need to switch back to a version that uses OpenSSL 1.x. Alternatively, they could try using the legacy OpenSSL provider by setting the environment variable before running `ng build`.
>
> I should also consider if there are any known issues with Angular and the newer OpenSSL versions. Perhaps Angular has made adjustments, but maybe the user's project hasn't been updated yet.
>
> Lastly, if none of these solutions work, there might be a need to adjust the project's configuration or even consider building a custom OpenSSL setup, though that's more advanced.
>
> I should present these solutions in a clear, step-by-step manner, starting with the simplest fixes and moving to more involved options. That way, the user can try each until they find what works for them.

The error `error 0308010C digital envelope routines:: unsupported` typically occurs when there is an issue with the OpenSSL library used by Node.js. This error can happen due to a newer version of OpenSSL being used, which might not be fully compatible with your project's requirements. Here's how you can fix it:

---

1. Set the OpenSSL Provider to Legacy

This error often occurs when using Node.js versions 17.x or newer, which switch to using OpenSSL 3.x by default. To resolve this, you can force Node.js to use the legacy OpenSSL provider.

- Add the following environment variable to your system or IDE:

  ```
  export NODE_OPTIONS=--openssl-legacy-provider
  ```

- You can also add this to your project's `package.json` under the `scripts` section:

  ```
  "build": "set NODE_OPTIONS=--openssl-legacy-provider && ng build"
  ```

---

2. Update Your Node.js Version

If you are using an older version of Node.js, try updating to a newer version (>=18.x) or downgrading to a version that works better with your project.

- Uninstall the current Node.js version and install a different version:
  - On **Windows**:

    ```
    # Download and install from https://nodejs.org/
    ```

  - On **macOS/Linux**:

    ```
    # Use a package manager like nvm
    nvm install --lts
    ```

---

3. Check for Project Dependencies

If the error occurs due to a specific dependency, you might need to update or downgrade certain packages.

- Look for dependencies in your `package.json` that might be causing the issue.
- Try updating or downgrading the `crypto` or `tls` related packages.

---

4. Reinstall Node.js and Dependencies

Sometimes, reinstalling Node.js and your project dependencies can resolve the issue.

- Uninstall Node.js and remove the `node_modules` folder:

  ```
  rm -rf node_modules package-lock.json
  ```

- Reinstall Node.js and run:

  ```
  npm install
  ng build
  ```

---

5. Use a Different OpenSSL Version

If the above methods don't work, you might need to explicitly set the OpenSSL provider.

- Add the following to your `.env` file:

  ```
  OPENSSL_PROVIDER=legacy
  ```

---

6. Use a Custom Build of Node.js

If none of the above works, you can try building Node.js with the legacy OpenSSL provider.

- Clone the Node.js repository and build it with the legacy provider flag.

---

7. Verify the Issue in Your Project

If the error persists, check if there are any specific operations in your project that use the `crypto` module. Replace or update any code that might be causing the issue.

By trying these steps, you should be able to resolve the `0308010C` error and successfully run `ng build`. If the issue persists, let me know and we can explore more advanced solutions.