The language used in the provided code is **Go (Golang)**.

This code is related to Linux capabilities. Linux capabilities are a feature of the Linux kernel that breaks the privileges traditionally associated with superuser into distinct units, which can be independently enabled or disabled. The function `DefaultCapabilities` returns a slice of strings representing a set of default capabilities.

Potential security issues with this code include:

1. **Broad Set of Capabilities**:

   - The default set of capabilities provided is broad. Each capability grants certain privileges. For example, `CAP_SYS_CHROOT` allows changing the root directory, `CAP_SETUID` allows changing user IDs, and `CAP_NET_RAW` allows raw socket usage (which can be used for crafting custom packets). Providing a broad set of capabilities can be risky, especially in containerized environments where least privilege is a best practice.

2. **Potential for Privilege Escalation**:

   - If an attacker can exploit a vulnerability in a process inside a container that has these capabilities, they can potentially escalate their privileges and perform actions they shouldn't be able to.

3. **Lack of Context**:

   - Without knowing the exact context in which `DefaultCapabilities` is used, it's challenging to gauge its appropriateness. If it's used as a default set for all containers, regardless of their purpose, then it's potentially insecure. Different containers and applications have different requirements, and not all of them need all these capabilities.

4. **Static Defaults**:

   - Hardcoding a static list of default capabilities can be problematic for maintenance and updates. As Linux evolves and new capabilities are added or old ones are deprecated or changed, this list might become outdated. It's a good practice to regularly review and update such lists and to have a mechanism for easy updates without hardcoding.

To enhance security:

- Review the list and ensure that only the necessary capabilities are granted by default.

- Ensure that capabilities are granted based on the principle of least privilege. Only give a container or process the capabilities it absolutely needs.

- Regularly review and update the list of capabilities as the Linux kernel evolves and as the requirements of the applications change.

- Consider dynamic configurations instead of static hardcoded lists.

Containers, especially in Docker and similar technologies, should be run with the least amount of privileges necessary to perform their intended tasks. This minimizes the potential damage if an attacker manages to compromise a container.