Cheatsheet: 8 security best practices for Go developers

snyk

1. Use Go Modules

The Go Modules system is Go's official dependency management system and allows for dependency version pinning, including transitive modules, while providing assurance against unexpected module mutation.

2. Scan dependencies for CVEs

- Use tools like Snyk to test your application's dependencies for known vulnerabilities.
- Use Snyk's suggestions to upgrade and fix any issues found.
- Continuously monitor your projects to catch new issues as they are identified in future.

3. Use Go standard crypto packages

- Prefer Go standard crypto packages over third-party ones.

4. Use `html/template` to help avoid XSS attacks

Rendering HTML pages using the **`html/template`** package is a simple way to help protect users from cross-site scripting (XSS) attacks by automatically encoding web content rather than simply outputting plain text, as the **`text/template`** would do.

5. Subshelling

- Sanitize all input data passed in or out the Subshell.

Subshells give direct shell access to your system and can easily be compromised.

6. Avoid `unsafe` and `cgo`

Where possible, avoid the **`Cgo`** and **`unsafe`** standard packages, as they permit developers to break out of Go's type-safety restrictions and their use could potentially enable attackers to break Go's memory safety.

7. Use reflection sparingly

Reflection can be a powerful tool, but with Go's typing and interface system, it should be rarely used as it can easily cause unexpected problems.

8. Minimizing container attack surface

In containers, utilize Go's static binding capabilities in conjunction with minimal base images (like **`scratch`**) to reduce the effective filesystem to a minimum.

Eric Smalling
@ericsmalling
Sr. Dev. Advocate at Snyk

Gerred Dillon
@devgerred
Co-creator of KUDO