Breaking Type Safety in Go: An empirical study on the use of the unsafe package

Diego Elias Costa, Suhaib Mujahid, Rabe Abdalkareem, Emad Shihab
The Go programming language

• Major programming language
  • Clean syntax
  • C-like performance
  • Modern language features

• Go has a **strong** and **static** type-system
  • Type-safe by design
Go is memory-safe…

…unless you use the **unsafe** package
The unsafe package

• Step around the type-safety of Go programs

```go
import "unsafe"

// Pointer arithmetic - (C-style)
p = unsafe.Pointer(uintptr(p) + offset)

// Convert between two types (without compiler checks)
func Float64bits(f float64) uint64 {
    return *(*uint64)(unsafe.Pointer(&f))
}
```
The unsafe package

Pros
• Avoid compiler checks
• Low-level memory manipulation
• Interface with system calls

Cons
• Avoid compiler checks
• Risk of non-portability
• No guarantees of compatibility
• Easy to write bad code
Beware of the unsafe package!

With the unsafe package there are no guarantees.

— “Go Proverbs” by Rob Pike

**Warning:** Avoid unsafe like the plague; if you can help it.

Never use unsafe. Performance is never that critical. – Volker Nov 27 '15 at 10:31

First of all, unsafe is usually a bad idea. So is reflection, but unsafe is at least an order of magnitude worse.

Here is your example using pure reflection (http://play.golang.org/p/JTJ6Mhg8q9):
Beware of the unsafe package!

With the unsafe package there are no guarantees.  
— “Go Proverbs” by Rob Pike

Do Go developers use the unsafe package?

Never use unsafe. Performance is never that critical.  
— rumpel Nov 27 '17 at 1:09

First of all, unsafe is usually a bad idea. So is reflection, but unsafe is at least an order of magnitude worse.

Here is your example using pure reflection (http://play.golang.org/p/6TJ6Mhq8q9):
Studying breaking type-safety in Go

Prevalence?

Why?

Consequences?
Studied projects

2590 most popular Go software projects

GitHub

Projects logos: Cockroach DB, moby project, kubernetes, Go, Syncthing, influxdb
Do developers use unsafe?

24% of projects use unsafe in their code

- Bindings: 88%
- Blockchain: 62%
- ML Libraries: 60%
- User Interface: 50%
- Database: 47%
- Security: 12%
- Multimedia: 8%

All domains
Why developers use unsafe?

- System Calls
- C code
- Efficient Casting
- Atomic Operations
- Pointer Arithmetic
- Reflection
- Inspecting Object Size
- ...

- System Integration
  - 45%
  - Performance
  - 30%
  - Go + Others
  - 25%

% of Usages (sampled)
Consequences of unsafe

Deployment restriction (20 projects)

"I wanted to use this package within a Google App Engine project, and due to package "unsafe" being used, it is not compatible"
Consequences of unsafe

Deployment restriction (20 projects)
Runtime errors (16 projects)

Prometheus crashes and hangs on fatal error: found bad pointer in Go heap (incorrect use of unsafe or cgo?) #2263

Unsafe use of unsafe that leads to data corruption #3
Consequences of unsafe

Deployment restriction (20 projects)
Runtime errors (16 projects)
Wrong API usage (13 projects)

...and the list goes on
To summarize

Prevalence?

24% of projects use unsafe

Why?

System Integration
Performance Optimization

Consequences?

Higher risk of
- Restrictions
- Runtime errors
- Bugs
- Breakages
Other team members were more optimistic that developers would avoid or could implement their project without using package unsafe.

I think this result will justify spending more time on making package unsafe easier to use.

Matthew Dempsky, maintainer of the GO compiler
Impact on the GO Language

Wrong slice conversion is one of the most common API misuses

New static analysis was released with Go 1.16

Language updates scheduled for Go 1.17
To summarize

Prevalence?

24% of projects use unsafe

Why?

System Integration
Performance Optimization

Consequences?

Higher risk of
- Restrictions
- Runtime errors
- Bugs
- Breakages

Impact on the GO Language

New static analysis was released with Go 1.16

Wrong slice conversion is one of the most common API misuses

Language updates scheduled for Go 1.17

Go 1.17
unsafe: add Slice(ptr *T, len anyIntegerType) [IT #19367]
unsafe: add Add function #40481