Reflection Dark Arts

About

- #darkarts on slack (gophers.slack.com)
- breaking Go rules with reflect + unsafe + linker tricks
- lot’s of code snippets

Disclaimer!

Do not use it in production code!

The Go Commandments

The Go Commandments

You shall not use unexported symbols of someone’s else package.

Unexported symbols

```
//go:linkname <symbol> <someone-else-symbol>
```

Unexported symbols - example

```go
//go:linkname isDomainName net.isDomainName
func isDomainName(s string) bool

func main() {
    flag.Parse()
    if !isDomainName(flag.Arg(0)) {
        os.Exit(1)
    }
}
```

The Go Commandments

You shall not read unexported fields of someone’s else struct.

Unexported fields

```go
func main() {
    h := sha1.New()

    fmt.Printf("%#v\n", h)
}
```

Unexported fields

```
$ go run main1.go

&sha1.digest{
    h: [5]uint32{0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0},
    x: [64]uint8{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, ..., 0x0},
    nx: 0,
    len: 0x0,
}
```

Unexported fields

```
$ go run main1.go

&sha1.digest{
    h: [5]uint32{0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0},
    x: [64]uint8{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, ..., 0x0},
    nx: 0,
    len: 0x0,
}
```

Unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()

    for i := 0; i < v.NumField(); i++ {
        fmt.Printf("%[1]T: %[1]v\n", v.Field(i).Interface())
    }
}
```

Unexported fields

```
$ go run main2.go

panic: reflect.Value.Interface: cannot return value obtained from unexported field or method

goroutine 1 [running]:
reflect.valueInterface(0x10b7000, 0xc0000c8000, 0x1b1, 0x1085301, 0x199, 0x19)
	/Users/rjeczalik/go/src/reflect/value.go:989 +0x1be
reflect.Value.Interface(0x10b7000, 0xc0000c8000, 0x1b1, 0x0, 0x10b7000)
	/Users/rjeczalik/go/src/reflect/value.go:978 +0x44
main.main()
	/Users/rjeczalik/src/github.com/rjeczalik/raw/talk/pp/main2.go:17 +0x152
exit status 2
```

Unexported fields

```
$ go run main2.go

panic: reflect.Value.Interface: cannot return value obtained from unexported field or method

goroutine 1 [running]:
reflect.valueInterface(0x10b7000, 0xc0000c8000, 0x1b1, 0x1085301, 0x199, 0x19)
	/Users/rjeczalik/go/src/reflect/value.go:989 +0x1be
reflect.Value.Interface(0x10b7000, 0xc0000c8000, 0x1b1, 0x0, 0x10b7000)
	/Users/rjeczalik/go/src/reflect/value.go:978 +0x44
main.main()
	/Users/rjeczalik/src/github.com/rjeczalik/raw/talk/pp/main2.go:17 +0x152
exit status 2
```

Unexported fields

```
$ go run main2.go

panic: reflect.Value.Interface: cannot return value obtained from unexported field or method

goroutine 1 [running]:
reflect.valueInterface(0x10b7000, 0xc0000c8000, 0x1b1, 0x1085301, 0x199, 0x19)
        /Users/rjeczalik/go/src/reflect/value.go:989 +0x1be
reflect.Value.Interface(0x10b7000, 0xc0000c8000, 0x1b1, 0x0, 0x10b7000)
        /Users/rjeczalik/go/src/reflect/value.go:978 +0x44
main.main()
        /Users/rjeczalik/src/github.com/rjeczalik/raw/talk/pp/main2.go:17 +0x152
exit status 2
```

Unexported fields

reflect/value.go:

```go
// Interface returns v's current value as an interface{}.
func (v Value) Interface() (i interface{}) {
    return valueInterface(v, true)
}
```

Unexported fields

reflect/value.go:

```
// Interface returns v's current value as an interface{}.
func (v Value) Interface() (i interface{}) {
    return valueInterface(v, true)
}
```

Unexported fields

reflect/value.go:

```
func valueInterface(v Value, safe bool) interface{} {
    if v.flag == 0 {
        panic(&ValueError{"reflect.Value.Interface", 0})
    }
    if safe && v.flag&flagRO != 0 {
        // Do not allow access to unexported values via Interface,
        // because they might be pointers that should not be
        // writable or methods or function that should not be callable.
        panic("reflect.Value.Interface: cannot return value obtained from unexported field or method")
    }

    ...
```

Unexported fields

reflect/value.go:

```
func valueInterface(v Value, safe bool) interface{} {
    if v.flag == 0 {
        panic(&ValueError{"reflect.Value.Interface", 0})
    }
    if safe && v.flag&flagRO != 0 {
        // Do not allow access to unexported values via Interface,
        // because they might be pointers that should not be
        // writable or methods or function that should not be callable.
        panic("reflect.Value.Interface: cannot return value obtained from unexported field or method")
    }

    ...
```

Unexported fields

```
v.Field(i).Interface()
```

```
valueInterface(v.Field(i), false)
```

Unexported fields

```go
//go:linkname valueInterface reflect.valueInterface
func valueInterface(v reflect.Value, safe bool) interface{}

func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()

    for i := 0; i < v.NumField(); i++ {
        fmt.Printf("%[1]T: %[1]v\n", valueInterface(v.Field(i), false))
    }
}
```

Unexported fields

```go
//go:linkname valueInterface reflect.valueInterface
func valueInterface(v reflect.Value, safe bool) interface{}

func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()

    for i := 0; i < v.NumField(); i++ {
        fmt.Printf("%[1]T: %[1]v\n", valueInterface(v.Field(i), false))
    }
}
```

Unexported fields

```
$ go run main3.go

[5]uint32: [1732584193 4023233417 2562383102 271733878 3285377520]
[64]uint8: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … ]
int: 0
uint64: 0
```

Unexported fields

```go
func main() {
    h := sha1.New()
    obj := make(map[string]interface{})

    v, t := reflect.ValueOf(h).Elem(), reflect.TypeOf(h).Elem()

    for i := 0; i < v.NumField(); i++ {
        obj[t.Field(i).Name] = valueInterface(v.Field(i), false)
    }

    json.NewEncoder(os.Stdout).Encode(obj)
}
```

Unexported fields

```
$ go run main4.go | jq .

{
    "h": [
        1732584193,
        4023233417,
        2562383102,
        271733878,
        3285377520
    ],
    "len": 0,
    "nx": 0,
    "x": [
        0,
        0,
        0,
        ...
    ]
}
```

The Go Commandments

You shall not write to unexported fields of someone’s else struct.

Unexported fields

```
$ go run main3.go

[5]uint32: [1732584193 4023233417 2562383102 271733878 3285377520]
[64]uint8: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … ]
int: 0
uint64: 0
```

Writing to unexported fields

```
$ go run main3.go

[5]uint32: [1732584193 4023233417 2562383102 271733878 3285377520]
[64]uint8: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … ]
int: 0
uint64: 0
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    f.Set(reflect.ValueOf(42))

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```
$ go run main5.go

panic: reflect: reflect.Value.Set using value obtained using unexported field

goroutine 1 [running]:
reflect.flag.mustBeAssignable(0x1ab)
	/Users/rjeczalik/go/src/reflect/value.go:225 +0x214
reflect.Value.Set(0x10a99c0, 0xc42001e1b0, 0x1ab, 0x10a9240, 0x10dbd18, 0x82)
	/Users/rjeczalik/go/src/reflect/value.go:1351 +0x2f
main.main()
	/Users/rjeczalik/src/github.com/rjeczalik/raw/talk/pp/uu/main5.go:20 +0x150
```

Writing to unexported fields

```
$ go run main5.go

panic: reflect: reflect.Value.Set using value obtained using unexported field

goroutine 1 [running]:
reflect.flag.mustBeAssignable(0x1ab)
        /Users/rjeczalik/go/src/reflect/value.go:225 +0x214
reflect.Value.Set(0x10a99c0, 0xc42001e1b0, 0x1ab, 0x10a9240, 0x10dbd18, 0x82)
        /Users/rjeczalik/go/src/reflect/value.go:1351 +0x2f
main.main()
        /Users/rjeczalik/src/github.com/rjeczalik/raw/talk/pp/uu/main5.go:20 +0x150
```

Writing to unexported fields

reflect/value.go:

```go
// Set assigns x to the value v.
// It panics if CanSet returns false.
// As in Go, x's value must be assignable to v's type.
func (v Value) Set(x Value) {
    v.mustBeAssignable()
    x.mustBeExported()

    ...

    *(*unsafe.Pointer)(v.ptr) = x.ptr
}
```

Writing to unexported fields

reflect/value.go:

```
// Set assigns x to the value v.
// It panics if CanSet returns false.
// As in Go, x's value must be assignable to v's type.
func (v Value) Set(x Value) {
    v.mustBeAssignable()
    x.mustBeExported()

    ...

    *(*unsafe.Pointer)(v.ptr) = x.ptr
}
```

Writing to unexported fields

```
f.Set(reflect.ValueOf(42))
```

```
p := &f.value
*p = 42
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*uint64)(unsafe.Pointer(p)) = 42

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*uint64)(unsafe.Pointer(p)) = 42

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```
$ go run main6.go

uint64: 42
```

Writing to unexported fields

fixed memory layout

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*uint64)(unsafe.Pointer(p)) = 42

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*string)(unsafe.Pointer(p)) = "foo"

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*string)(unsafe.Pointer(p)) = "foo"

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*string)(unsafe.Pointer(p)) = "foo"

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New()

    v := reflect.ValueOf(h).Elem()
    f := v.Field(3)

    p := f.UnsafeAddr()
    *(*string)(unsafe.Pointer(p)) = "foo"

    fmt.Printf("%[1]T: %[1]v\n", valueInterface(f, false))
}
```

```
$ go run main7.go

uint64: 17604203
```

Writing to unexported fields

```go
// digest has the same memory layout
// as sha1.digest struct
type digest struct {
    h   [5]uint32
    x   [64]uint8
    nx  int
    len uint64
}
```

Writing to unexported fields

```go
// digest has the same memory layout
// as sha1.digest struct
type digest struct {
    h   [5]uint32
    x   [64]uint8
    nx  int
    len uint64
}
```

```go
func main() {
    h := sha1.New() // *sha1.digest

    p := reflect.ValueOf(h).Elem().UnsafeAddr()

    *(*digest)(unsafe.Pointer(p)) = digest{
        nx:  123,
        len: 321,
    }

    fmt.Printf("%#v\n", h)
}
```

Writing to unexported fields

```
$ go run main8.go

&sha1.digest{
    h: [5]uint32{0, 0, 0, 0, 0},
    x: [64]uint8{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, ..., 0x0},
    nx: 123,
    len: 321,
}
```

Writing to unexported fields

```
$ go run main8.go

&sha1.digest{
    h: [5]uint32{0, 0, 0, 0, 0},
    x: [64]uint8{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, ..., 0x0},
    nx: 123,
    len: 321,
}
```

Writing to unexported fields

dynamic memory layout

Writing to unexported fields

```
*(*digest)(unsafe.Pointer(p)) = digest{
    nx:  123,
    len: 321,
}
```

Writing to unexported fields

```
*(*digest)(unsafe.Pointer(p)) = digest{
    nx:  123,
    len: 321,
}

 n := unsafe.Sizeof(digest{}) // 100
```

Writing to unexported fields

```go
*(*digest)(unsafe.Pointer(p)) = digest{
    nx:  123,
    len: 321,
}

 n := unsafe.Sizeof(digest{}) // 100

 raw := *(*[100]byte)(p)
```

Writing to unexported fields

```
*(*digest)(unsafe.Pointer(p)) = digest{
    nx:  123,
    len: 321,
}

 n := unsafe.Sizeof(digest{}) // 100

 raw := *(*[100]byte)(p)
```

```
copy(*(*[100]byte)(p), *(*[100]byte)(modified))
```

Writing to unexported fields

```
*(*digest)(unsafe.Pointer(p)) = digest{
    nx:  123,
    len: 321,
}
```

=

```
copy(*(*[100]byte)(p), *(*[100]byte)(modified))
```

```
n := unsafe.Sizeof(digest{}) // 100

raw := *(*[100]byte)(p)
```

Writing to unexported fields

```
modified := ?
```

Writing to unexported fields

```
modified := ?
```

```
func main() {
    h := sha1.New() // *sha1.digest

    p := reflect.ValueOf(h).Elem().UnsafeAddr()

    copy(*(*[100]byte)(p), *(*[100]byte)(modified))

    fmt.Printf("%#v\n", h)
}
```

Writing to unexported fields

```go
func main() {
    h := sha1.New() // *sha1.digest

    p := reflect.ValueOf(h).Elem().UnsafeAddr()

    copy(*(*[100]byte)(p), *(*[100]byte)(modified))

    fmt.Printf("%#v\n", h)
}
```

```go
modified := ?

var obj = map[string]interface{}{
    "h":   [5]uint32{},
    "x":   [64]uint8{},
    "nx":  int(123),
    "len": uint64(321),
}

typ := makeStructType(h) // reflect.StructOf

val := makeStructValue(obj, typ) // reflect.New

modified := val.UnsafeAddr()
```

Writing to unexported fields

```go
func main() {
    h := sha1.New() // *sha1.digest

    p := reflect.ValueOf(h).Elem().UnsafeAddr()

    copy(*(*[100]byte)(p), *(*[100]byte)(modified))

    fmt.Printf("%#v\n", h)
}
```

```go
modified := ?

var obj = map[string]interface{}{
    "h":   [5]uint32{},
    "x":   [64]uint8{},
    "nx":  int(123),
    "len": uint64(321),
}

typ := makeStructType(h) // reflect.StructOf

val := makeStructValue(obj, typ) // reflect.New

modified := val.UnsafeAddr()
```

(github.com/rjeczalik/raw).Copy

raw.Copy

```go
func main() {
    h := sha1.New()
    obj := make(map[string]interface{})

    _ = raw.Copy(h, &obj)

    json.NewEncoder(os.Stdout).Encode(obj)
}
```

THANKS!

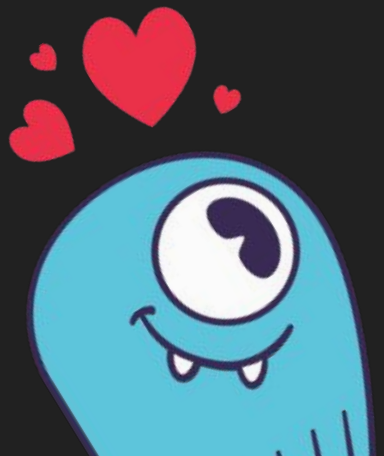
Thanks! Questions?

Rafał Jęczalik

Software Engineer @ScyllaDB

https://github.com/rjeczalik

https://www.scylladb.com (JOIN US!)



makeStructType

```go
func makeStructType(t reflect.Type) reflect.Type {
    var fields []reflect.StructField

    for i := 0; i < t.NumField(); i++ {
        f := t.Field(i)
        f.Name = strings.ToUpper(f.Name)
        f.PkgPath = ""
        fields = append(fields, f)
    }

    return reflect.StructOf(fields)
}
```

makeStructValue

```go
func makeStructValue(obj map[string]interface{}, typ reflect.Type) reflect.Value {
    vv := reflect.New(typ).Elem()
    for name, v := range obj {
        vv.FieldByName(strings.ToUpper(name)).Set(reflect.ValueOf(v))
    }
    return vv
}
```

(todo)

Explain few quirks on how to use it:

- Obligatory _ “unsafe” import
- Force CGO with empty .s file