GO 1.8

PLUGINS

PLUGINS

- Sistema para extender el funcionamiento de una aplicación dada.
- Tipos de plugin:
  - Interpretados (Interprete embebido)
    - Lua, JS…
  - Compilados (Nativos al sistema en el que se ejecutan)
    - Propietarios, .dll, .so…

ANTECEDENTES

ANTECEDENTES: VARIOS SISTEMAS DE PLUGINS {IN DA WILD}™

- Tecnologías:
  - goRPC
  - HTTP
  - Raw sockets
- La mayoría son PoCs
- Solamente 2 que se puedan considerar validos para producción

ANTECEDENTES: GOPIE (NATEFINCH)

- Sistema de plugins sobre JSON-RPC
- Plugins como *providers*
  - El plugin ofrece una API
- Plugins como *consumers*
  - El plugin consume de una API
- Es muy facil de usar
- El desarrollo está estancado, pero es funcional

ANTECEDENTES: GO-PLUGIN (HASHICORP)

- Sistema de plugins sobre go-RPC.
- Implementado mediante interfaces.
- Soporta comunicación bidireccional.
- Pueden correr como demonios.
  - El *launcher* permite *deattaching* y *reattaching*.
- La documentación es muy insuficiente dada la complejidad de la librería.

ANTECEDENTES: GO-PLUGIN (HASHICORP)

- Soporta autenticación del plugin!
- Pendiente el soporte para el firmado de plugins.
- La implementación en el *launcher* es un dolor de...

ES LA MEJOR ALTERNATIVA COMO SISTEMA DE PLUGINS!

PLUGINS NATIVOS

GO 1.8

PLUGINS NATIVOS: ¿COMO FUNCIONAN?

- https://~~tip.~~golang.org/pkg/plugin/
  - Solo en Linux
  - .so (Shared Object)
  - `go build -buildmode=plugin`
  - Métodos `Open()` y `Lookup()`
  - Type `Symbol` es una `interface{}`
  - Muy fácil de usar

PLUGINS NATIVOS: ¿DÓNDE ESTA EL CÓDIGO?

- `$GOROOT/src/plugin/`
  - `plugin.go`
    - Métodos expuestos (`Open`, `plugin.Lookup`)
  - `plugin_dlopen.go`
    - {Where the magic happens}™
  - `plugin_stubs.go`
    - Equivalente a plugins, pero para sistemas no soportados

PLUGINS NATIVOS: UNDER THE HOOD

- Definido en `plugin_dlopen.go`, magia en C (CGO)

```
 9  /*
10  #cgo linux LDFLAGS: -ldl
11  #include <dlfcn.h>
12  #include <limits.h>
13  #include <stdlib.h>
14  #include <stdint.h>
15
16  #include <stdio.h>
17
18  static uintptr_t pluginOpen(const char* path, char** err) {
19          void* h = dlopen(path, RTLD_NOW|RTLD_GLOBAL);
20          if (h == NULL) {
21                  *err = (char*)dlerror();
22          }
23          return (uintptr_t)h;
24  }
25
26  static void* pluginLookup(uintptr_t h, const char* name, char** err) {
27          void* r = dlsym((void*)h, name);
28          if (r == NULL) {
29                  *err = (char*)dlerror();
30          }
31          return r;
32  }
33  */
34  import "C"
```

IMPLEMENTACIÓN

IMPLEMENTACIÓN: EL PLUGIN

```go
package main

import "C"

var A = "I'm a plugin"
```

IMPLEMENTACIÓN: EL LOADER

```go
package main

import (
    "log"
    "plugin"
)

func main() {
    p, err := plugin.Open("test.so")
    checkErr(err)

    s, err := p.Lookup("A")
    checkErr(err)

    foo := *s.(*string)

    log.Println(foo)
}

func checkErr(err error) {
    if err != nil {
        log.Fatal(err)
    }
}
```

IMPLEMENTACIÓN: CASTING DE FUNCIONES

Plugin	Launcher
`func foo() {}`	`s.(func())()`
`func foo(a int) {}`	`s.(func(int))()`
`func foo(a int) error {}`	`s.(func(int) error)()`

IMPLEMENTACIÓN: COMPILACIÓN Y EJECUCIÓN

- Plugin

```
> go build -buildmode="plugin" -o ../test.so main.go
```

- Launcher

```
> go run main.go
2017/02/17 13:31:39 I'm a plugin
>
```

DONE!

ALGO MÁS AVANZADO

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

- Leemos el directorio de plugins
- Abrimos
- Lookup
- Win

```go
func loadPlugins() map[string]func() {
    var bucket = make(map[string]func())

    folderInfo, err := ioutil.ReadDir("./")
    checkErr(err)

    for _, p := range folderInfo {
        fileName := p.Name()

        if !strings.Contains(fileName, ".so") {
            continue
        }

        fmt.Println(fileName)

        p, err := plugin.Open("./" + fileName)
        checkErr(err)
        run, err := p.Lookup("Run")
        checkErr(err)
        name, err := p.Lookup("Name")
        checkErr(err)

        bucket[*name.(*string)] = run.(func())
    }

    return bucket
}
```

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

- Invocamos con `main()`

```
func main() {
    bucket := loadPlugins()

    for name := range bucket {
        bucket[name]()
    }
}
```

- Compilamos varios plugins

```
go build -buildmode="plugin" -o ../libworker.1.so plg.go
go build -buildmode="plugin" -o ../libworker.2.so plg.go
```

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

```
plugin: plugin plugin/unnamed-d7e84da40251aba20b17da6cec8391715f1c7acb already loaded
fatal error: plugin: plugin already loaded

goroutine 1 [running]:
runtime.throw(0x55489c, 0x1d)
        /usr/local/go/src/runtime/panic.go:596 +0x95 fp=0xc420045a10 sp=0xc4200459f0
plugin.lastmoduleinit(0x217a020, 0xc420073ed8, 0x7f47cb5bf1a0, 0x2b, 0x7e8040)
        /usr/local/go/src/runtime/plugin.go:25 +0xc1a fp=0xc420045b40 sp=0xc420045a10
plugin.open(0xc420045e38, 0x10, 0x0, 0x0, 0x0)
        /usr/local/go/src/plugin/plugin_dlopen.go:72 +0x25d fp=0xc420045d80 sp=0xc420045b40
plugin.Open(0xc420045e38, 0x10, 0x2, 0xc420074063, 0xe)
        /usr/local/go/src/plugin/plugin.go:30 +0x35 fp=0xc420045db8 sp=0xc420045d80
main.loadPlugins(0x0)
        /go/src/noRepo/pluginstealth/loader.go:34 +0x229 fp=0xc420045ee8 sp=0xc420045db8
main.main()
        /go/src/noRepo/pluginstealth/loader.go:12 +0x34 fp=0xc420045f88 sp=0xc420045ee8
runtime.main()
        /usr/local/go/src/runtime/proc.go:185 +0x20a fp=0xc420045fe0 sp=0xc420045f88
runtime.goexit()
        /usr/local/go/src/runtime/asm_amd64.s:2197 +0x1 fp=0xc420045fe8 sp=0xc420045fe0

goroutine 17 [syscall, locked to thread]:
runtime.goexit()
        /usr/local/go/src/runtime/asm_amd64.s:2197 +0x1
exit status 2
```

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

-LDFLAGS "-PLUGINPATH=XXX"

ALGO MÁS AVANZADO: CARGA AUTOMÁTICA DE PLUGINS

- Bingo!
- Llamada de plugins:
  - `map[string]func()`

```
pluginstealth $ go run loader.go
Loaded: libworker.1.so
Loaded: libworker.2.so
Loaded: libworker.3.so
Plugin 3 running!
Plugin 1 running!
Plugin 2 running!
```

- Siguiente nivel: Watcher
  - Ticker
  - iNotify

ALGO MÁS AVANZADO: RESOLUCIÓN AUTOMÁTICA DE SIMBOLOS

```
$ file libworker.so

libworker.so: ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked, not stripped
```

- https://golang.org/pkg/debug/elf/
  - Utilidades interesantes para trabajar con este formato

ALGO MÁS AVANZADO: RESOLUCIÓN AUTOMÁTICA DE SIMBOLOS

- Sym32 y Sym64

```
type Sym64 struct {
        Name  uint32 /* String table index of name. */
        Info  uint8  /* Type and binding information. */
        Other uint8  /* Reserved (not used). */
        Shndx uint16 /* Section index of symbol. */
        Value uint64 /* Symbol value. */
        Size  uint64 /* Size of associated object. */
}
```

ALGO MÁS AVANZADO: RESOLUCIÓN AUTOMÁTICA DE SIMBOLOS

```go
func extractSyms() {
    f, err := elf.Open("./libworker.1.so")
    checkErr(err)

    syms, err := f.DynamicSymbols()
    checkErr(err)

    for _, s := range syms {
        fmt.Println(s.Name)
    }
}
```

- Los simbolos del plugin son facilmente localizables
  - `plugin/unnamed-d7e84da40251aba20b17da6cec8391715f1c7acb.Run`

¿EN PRODUCCIÓN?

¿EN PRODUCCIÓN? DESDE LA 1.8-RC1

- En un proyecto interno.
- ¿Problemas?
  - Comportamiento extraño con `io.ReaderWriter`
  - Fixed en 1.8-RC2.
- Funcionan a las mil maravillas.

EVIL STUFF 

EVIL STUFF: ¿GO PARA COSAS MALAS?

- Cada día hay más bichos hechos en Go
  - Fácil desarrollo
  - Portable
  - X-Compilable...

- Si, nuestro gopher se usa par cosas malas...

EVIL STUFF: MALWARE EXTENSIBLE

- Los plugins nos permiten crear un pequeño *dropper*

```
3.1M Feb 17 21:11 dropper
```

- Packed:

```
792.2K Feb 17 21:11 dropper
```

- Malware mutable
- *Dropper* descarga .so, almacena en disco y lo carga.

EVIL STUFF: FILE HOLLOWING

- Técnica de infección ancestral
- El payload “no toca el disco”
- Proceso:
  - Buscamos un documento de más tamaño que el .so.
  - Lo *nulleamos* y escribimos nuestro buffer en el fichero.
  - El offset *nulleado* no será tenido en cuenta.

¿PREGUNTAS? ¿TOMATES?

¿DINERO?

MUCHAS GRACIAS!

- Available 4 hire ❤️