Anna-Katharina Wickert | @akwickert

Go Flow Safe: Ensure Data Flow of your App with Taint Analyses

GopherCon Europe 2022

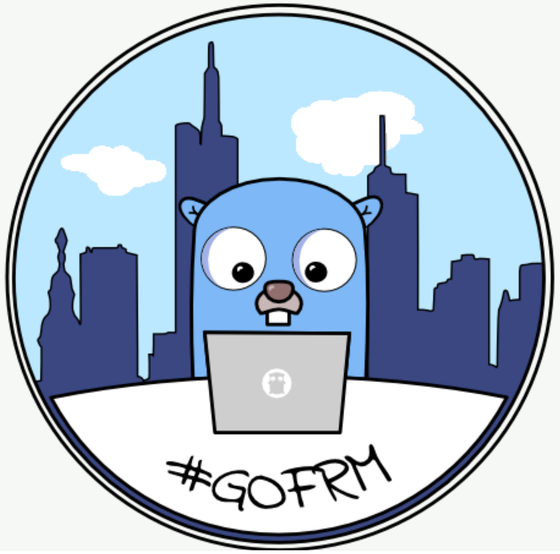
Starstruck

@akwick created a repository that has many stars.

History

Unlocked 26 Mar 2019

akwick/gotcha · ⭐ 16 stars



Disclaimer

Motivation

Vulnerability Details : CVE-2022-29810

The Hashicorp go-getter library before 1.5.11 does not redact an SSH key from a URL query parameter.

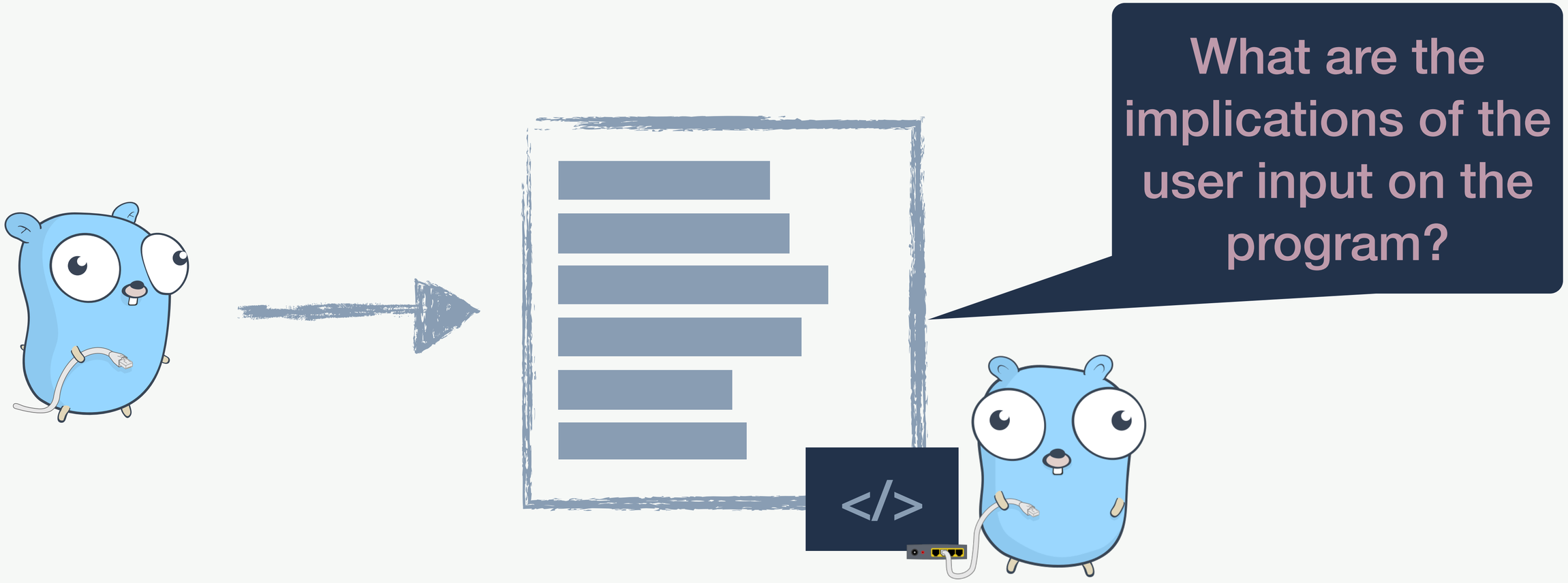
Publish Date : 2022-04-27 Last Update Date : 2022-05-10

CVE-2022-24124 Detail

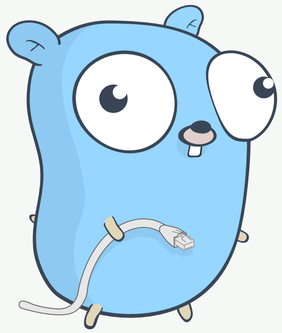
Current Description

The query API in Casdoor before 1.13.1 has a SQL injection vulnerability related to the field and value parameters, as demonstrated by api/get-organizations.

Taint Analysis - Definition



Basic Terminology 1/3

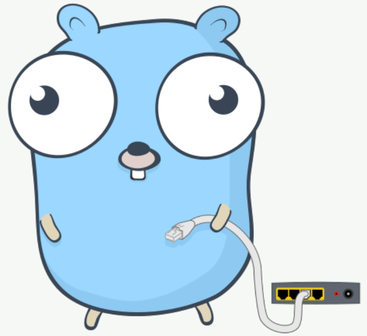


Sources:
Value to be tracked

```
type Authentication struct {
    Username string
    Password string // Source
}
```

```
log.Printf("unable to make authenticated
    request: %v\n", auth) // Sink
```

Sinks:
Instruction that should not be reached



Basic Terminology 2/3

```
func (a Authentication) String() string {
    return fmt.Sprintf("%s:%s", a.Username,
a.Password)
}
```

Propagator:
Instruction process source without relevant modifications

Sanitizer:
Instruction process source with relevant modifications

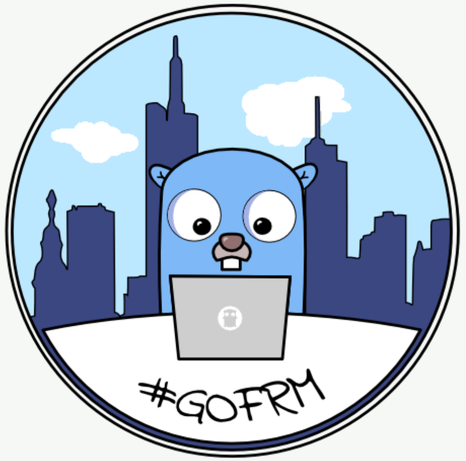
```
func (a Authentication) String()
string {
    return fmt.Sprintf("%s:***",
a.Username)
}
```

Basic Terminology 3/3

- How to argue about the program?
  - Pointer analysis
  - Static single assignment (SSA) analysis

Limitations Static Analyses

What are Go Gophers?

	No Gopher	Gopher
		
True negative		False positive
False negative		
		True positive

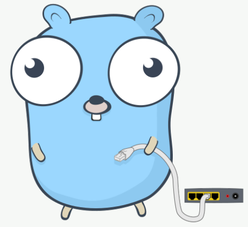
Taint Analyses in Go

SQL Injection

```
func main() {
        db := openDatabase()
        defer db.Close()

        query(source(), db) // query is a sink
}
```

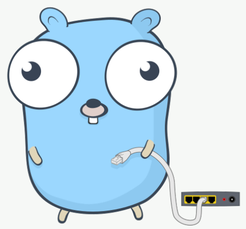
```
func query(q string, db *sql.DB) { // sink
        _, _ = db.Query("SELECT * FROM users WHERE id =" + q)
}
```



Logging Sensitive Information

```
type Authentication struct {
        Username string
        Password string // Source
}
```

```
func authenticate(u User) (*AuhtenticationResponse, error) {
        resp, err := makeAuthenticationRequest(u)
        if err != nil {
                log.Printf("unable to make authenticated request: %v\n",
auth) // Sink
…}
```



Rule of Thumb for Configuration

Sources

1. Identify the Source type(s)
2. Add Source type(s) to the configuration
    A. Describe the type
    B. Use field tags

```
# analyzer_configuration.yaml
Sources:
  - Package: $GoPackageName
    Type: $GoType
    Field: $GoField
```

Use the keys `PackageRE`, `TypeRE`, and `FieldRE` for regular experssion

Rule of Thumb for Configuration

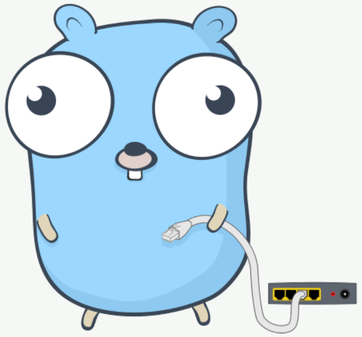
Sources

1. Identify the Source type(s)
2. Add Source type(s) to the configuration
    A. Describe the type
    B. Use field tags

```
# analyzer_configuration.yaml
FieldTags:
  - Key: $TagKey
    Value: $TagValue
```

Rule of Thumb for Configuration

Sinks and Sanitizers



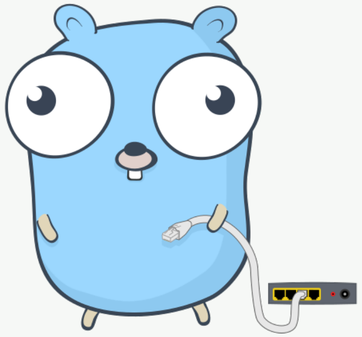
1. Identify the Sink function(s)
2. Add Sink function(s) to the configuration
    A. Describe the function(s)

```yaml
# analyzer_configuration.yaml
Sinks:
  - Package: $GoPackageName
    Receiver: $GoReceiver
    Method: $GoMethod
```

Use the keys `PackageRE`, `ReceiverRE`, and `MethodRE` for regular expressions

Rule of Thumb for Configuration

Sinks and Sanitizers



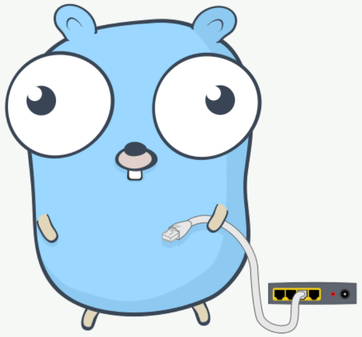
1. Identify the Sink function(s)
2. Add Sink function(s) to the configuration
    A. Describe the function(s)

```
# analyzer_configuration.yaml
Sinks:
  - Package: $GoPackageName
    Receiver: $GoReceiver
    Method: $GoMethod
```

Match any package if missing `Package` or `PackageRE`

Rule of Thumb for Configuration

Sinks and Sanitizers



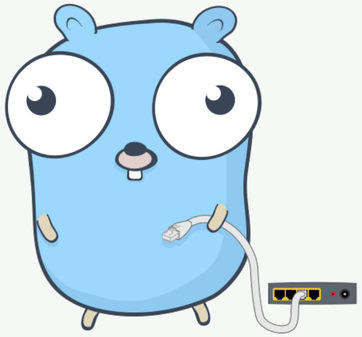
1. Identify the Sink function(s)
2. Add Sink function(s) to the configuration
    A. Describe the function(s)

```
# analyzer_configuration.yaml
Sinks:
  - Package: $GoPackageName
    Receiver: $GoReceiver
    Method: $GoMethod
```

Optional, if not defined matches any/no receiver

Rule of Thumb for Configuration

Sinks and Sanitizers



1. Identify the Sink function(s)
2. Add Sink function(s) to the configuration
    A. Describe the function(s)

```yaml
# analyzer_configuration.yaml
Sinks:
  - Package: $GoPackageName
    Receiver: $GoReceiver
    Method: $GoMethod
```

Replace `Sinks` with `Sanitizers` to define sanitizers

Configuration SQL Injection

```
Sources:
  - Type: Authentication
    Field: Password
Sinks:
  - Package: database/sql
    Method: Query
```

Matches all packages with a `Authentication` type

Configuration SQL Injection

```
Sources:
  - Type: Authentication
    Field: Password
Sinks:
  - Package: database/sql
    Method: Query
```

Matches all packages with a `Authentication` type

Configuration Logging Sensitive Information

```
---

FieldTags:
  - Key: datapolicy
    Value: password

Sinks:
  - Package: log
    Method: Printf
```

```
type Authentication struct {
        Username string
        Password string `datapolicy:"password"`
}
```

Update source code

Configuration Logging Sensitive Information

```
---

FieldTags:
  - Key: datapolicy
    Value: password
```

```
type Authentication struct {
        Username string
        Password string `datapolicy:"password"`
}
```

Update source code

```
Sinks:
  - Package: log
    Method: Printf
```

Rule of Thumb for Configuration

False Positives

```
// levee.DoNotReport

log.Printf(“my maybe secret value: %v",
safeValue)
```

- Verify that the sink is a false positive for all paths
- Let others know why it’s a false positive
- Re-check your assumptions from time to time

Why Don’t Software Developers Use Static Analysis Tools to Find Bugs?

Brittany Johnson, Yoonki Song, and Emerson Murphy-Hill
North Carolina State University
Raleigh, NC, U.S.A.
bijohnso,ysong2@ncsu.edu,emerson@csc.ncsu.edu

Robert Bowdidge
Google
Mountain View, CA, U.S.A.
bowdidge@google.com

***Abstract*—Using static analysis tools for automating code inspections can be beneficial for software engineers. Such tools can make finding bugs, or software defects, faster and cheaper than manual inspections. Despite the benefits of using static analysis tools to find bugs, research suggests that these tools are underused. In this paper, we investigate why developers are not widely using static analysis tools and how current tools could potentially be improved. We conducted interviews with 20 developers and found that although all of our participants felt that use is beneficial, false positives and the way in which the warnings are presented, among other things, are barriers to use.**

There are many situations where a developer may consider using a static analysis tool to find defects in their code. Let us consider a developer, Susie. Susie is a software developer at a small company. She wants to make sure that she is following the company’s standards while maintaining quality code. She needs a way of checking her code in her IDE, before submitting it to the general code repository, without worrying about any outside dependencies that she has no control over. Susie decides that her best bet is to install a static analysis tool. She decides to install FindBugs because she likes the

Execute Go Flow Levee

```
1) Install Go Flow Leeve
$ go install github.com/google/go-flow-levee/cmd/levee@latest

2) $ go vet -vettool=$(which levee) -config=./
analyzer_configuration.yaml ./...



Output: # gopherconeu22/taint/injection

./authentication.go:12:17: a source has reached a sink

 source: ./authentication.go:10:22
```

Execute Go Flow Levee

```
1) Install Go Flow Leeve
$ go install github.com/google/go-flow-levee/cmd/levee@latest

2) $ go vet -vettool=$(which levee) -config=./
analyzer_configuration.yaml ./...


Output: # gopherconeu22/taint/injection

./authentication.go:12:17: a source has reached a sink

 source: ./authentication.go:10:22
```

Limitations Go Flow Levee

- A source is identified by package, type, and field name & tags are only allowed for fields
  -> no method possible, may require adaption of code
- Many uncaught exceptions that “poison” the output
- No information if a source was found -> debugging is not trivial
- Some flows are not modelled, e.g., `query(a.Password, ..)`

Some thoughts by Damian Gryski:
https://twitter.com/dgryski/status/1451562750354739205

Comparison: Taint with Semgrep

	Go Flow Levee	Semgrep
Multiple Languages	❌	✅
Multiple Sink Definitions	❌	✅
Functions as source	❌	✅
Flow over multiple files	✅	❌

Credit to @hex0punk for his input in the #static-analysis channel on Slack

Vulnerability Details : CVE-2022-29810

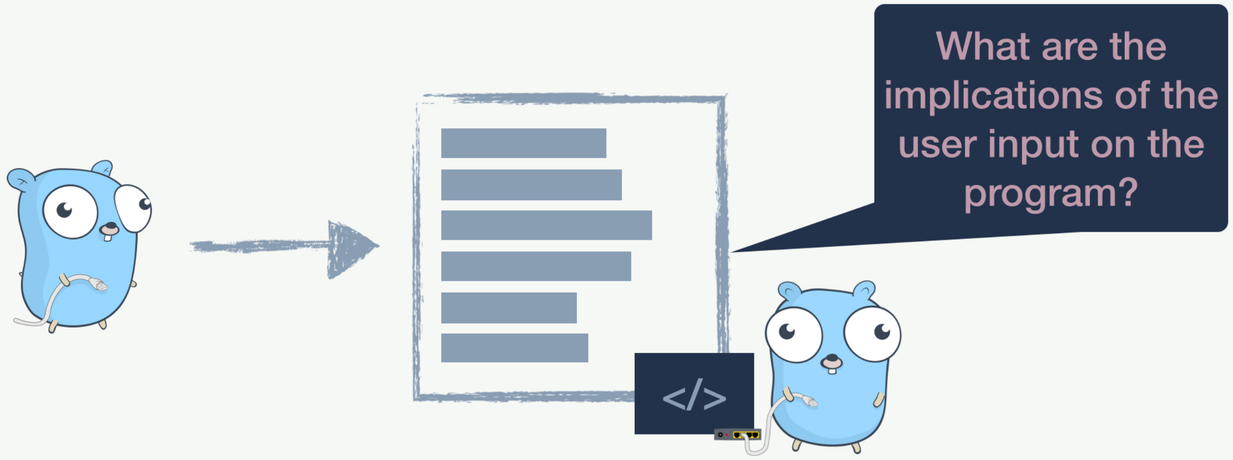
The Hashicorp go-getter library before 1.5.11 does not redact an SSH key from a URL query parameter.

Publish Date : 2022-04-27 Last Update Date : 2022-05-10

CVE-2022-24124 Detail

Current Description

The query API in Casdoor before 1.13.1 has a SQL injection vulnerability related to the field and value parameters, as demonstrated by api/get-organizations.




```
---
FieldTags:
  - Key: datapolicy
    Value: password

Sinks:
  - Package: log
    Method: Printf
```

```
type Authentication struct {
        Username string
        Password string `datapolicy:"password"`
}
```

Update source code

@akwickert

```
1) Install Go Flow Leeve
$ go install github.com/google/go-flow-levee/cmd/levee@latest

2) $ go vet -vettool=$(which levee) -config=./
analyzer_configuration.yaml ./...


Output: # gopherconeu22/taint/injection
./authentication.go:12:17: a source has reached a sink
 source: ./authentication.go:10:22
```

Credits

- Image title slide: Photo by Nicholas Selman on Unsplash
- Disclaimer image: Photo by Tingey Injury Law Firm on Unsplash
- Networ gopher: Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers/blob/master/vector/projects/network-side.svg> and <https://github.com/egonelbre/gophers/blob/master/vector/projects/network.svg>
- GRPC-web gopher: Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers/blob/master/vector/projects/go-grpc-web.svg>
- Protected gopher: Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers/blob/master/vector/projects/go-fuzz.svg>
- Magic gopher: Artwork by Egon Elbre licensed under CC0 found on GitHub <https://github.com/egonelbre/gophers/blob/master/vector/fairy-tale/witch-too-much-candy.svg>
- TP, TN, FP, FN slide: credit at respective events/locations & gopher image by Lukáš Vaňátko on Unsplash

Backup

Configuration Example

Sources

```
---
Sources:
  - TypeRE: "^Secret$|Token"
  - PackageRE: "k8s.io/client-go/tools/clientcmd/api(?:/v1)?"
    TypeRE: "^(?:Named)?AuthInfo$"
    FieldRE: ""
  - PackageRE: "^k8s.io/client-go/rest$"
    TypeRE: "^TLSClientConfig$"
    FieldRE: "Password|BearerToken$|"
  - PackageRE: "^k8s.io/client-go/rest$"
    TypeRE: "^Config$"
    FieldRE: "Password|BearerToken$|"
```

https://github.com/google/go-flow-levee/blob/master/configuration/example-config.yaml

Configuration Example

Sinks and more :)

```
Sinks:
  - PackageRE: "log"
    ReceiverRE: ""
    MethodRE: "Info|Warning|Error|Fatal|Exit"

Sanitizers: []

Exclude:
  - PackageRE: "^k8s.io/kubernetes/test"
```

https://github.com/google/go-flow-levee/blob/master/configuration/example-config.yaml