



VEROSINT

Defensive API development techniques for Gophers

Bertold Kolics

LASCON '2023, September 27, 2023, Austin, TX USA

My Context

- Question Asker, Bulldog Engineer
- Not a
 - Gatekeeper
 - PEN tester, or even a security tester
- Managing **risk**
- [Verosint](#) - small startup < 20 employees
 - SaaS business
 - Detect & prevent online account fraud
- Past roles
 - IT, pre-sales, QA, developer, manager



VEROSINT



mabl

eso

Healthcare Connected

UnboundID™



Sun®
microsystems



Motivation

- API as a business for many vendors just like Verosint
 - APIs are accessed directly and indirectly
- Rarely any SaaS application is built in isolation
 - i.e. consumers of 3rd party applications exposed via APIs
- Para-functional requirements are implied to deliver customer value:
 - security
 - reliability/availability
 - scalability/performance
- **Malicious actors** may cost business \$\$\$
 - outgages
 - reduced availability
- **Defensive posture** at the application layer needed for a multi-pronged approach

Agenda

- Focus on Go language for building, maintaining and securing HTTP-based APIs
 - code samples, libraries, practices
- Out of scope
 - infrastructure
 - hardware or hosted solutions
 - HTTP/3
 - Non-HTTP APIs
 - GraphQL
- Basic familiarity with Go, HTTP assumed



credit: Pragmatic Programmers



bit.ly/lascon2023

Deployment View

Typical Cloud Deployment

- Often includes
 - Gateways (API, NAT)
 - Load balancers
 - Reverse Proxies
 - External services (e.g. authentication, authorization)
 - **Application**
- Understand what protection, mitigation techniques are available at each layer
- Overlap is OK

API Service Implementation with Go

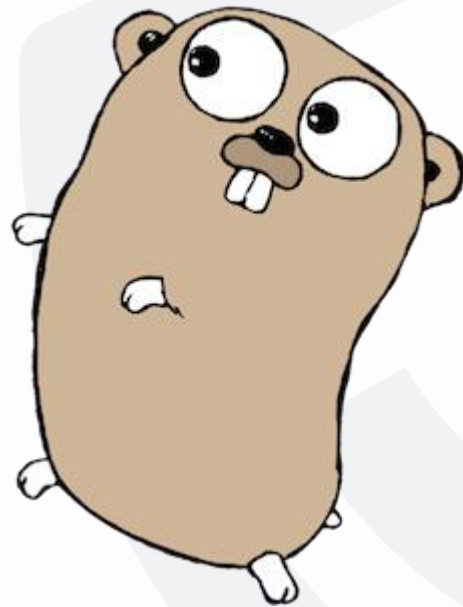
- Deployment options include:
 - Microservice
 - Serverless
- Implementation will need to address
 - authentication
 - authorization
 - request paths to handle
 - HTTP methods to support (and not support)
 - payload (schema) for requests and responses
 - possibly: resource limits

Building HTTP APIs with Go

Go - The Good Side*

- No 3rd-party library required
 - unlike other languages
 - reduced attack surface
 - much reduced risk for supply chain attacks
- Core language supports testing
 - unit, fuzzing, performance

*Is there a bad side? 😬



When You Need 3rd-Party Libraries

- Carefully consider options
 - not just functionality or licenses
- Support is key for both open-source and commercial libraries
- Criteria for evaluating OSS projects
 - age of the project, adoption, responsiveness of maintainers, openness to contributions, commit activity/history, release history/frequency, documentation, automated test coverage, availability of code quality metrics, presence of security tests/scans, number of open issues, rate of closing issues, number of dependencies
- Run your own scanners
- GitHub/GitLab provides plenty of repository information to help assessments

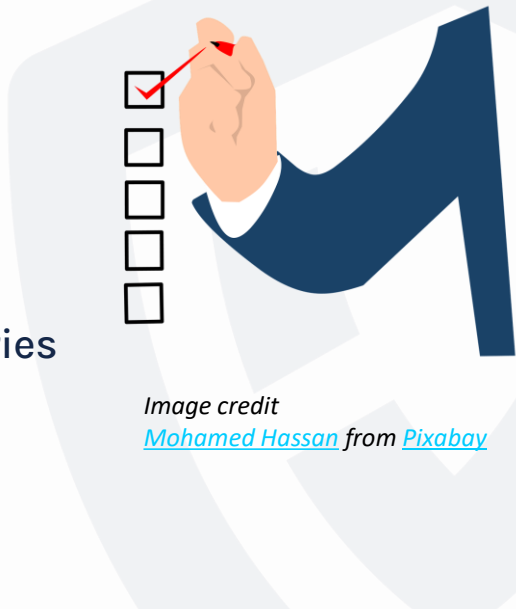


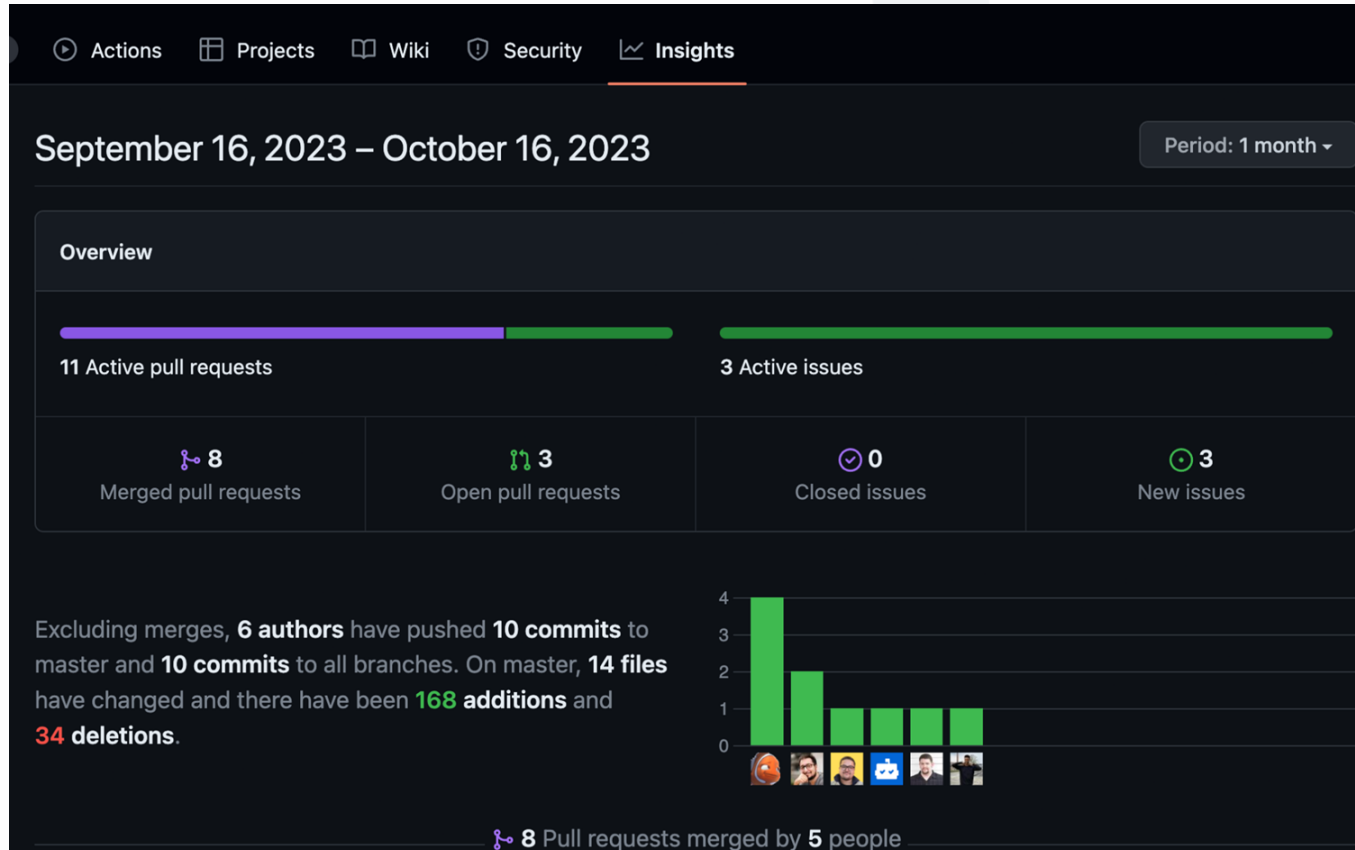


Image credit
[Mohamed Hassan](#) from [Pixabay](#)

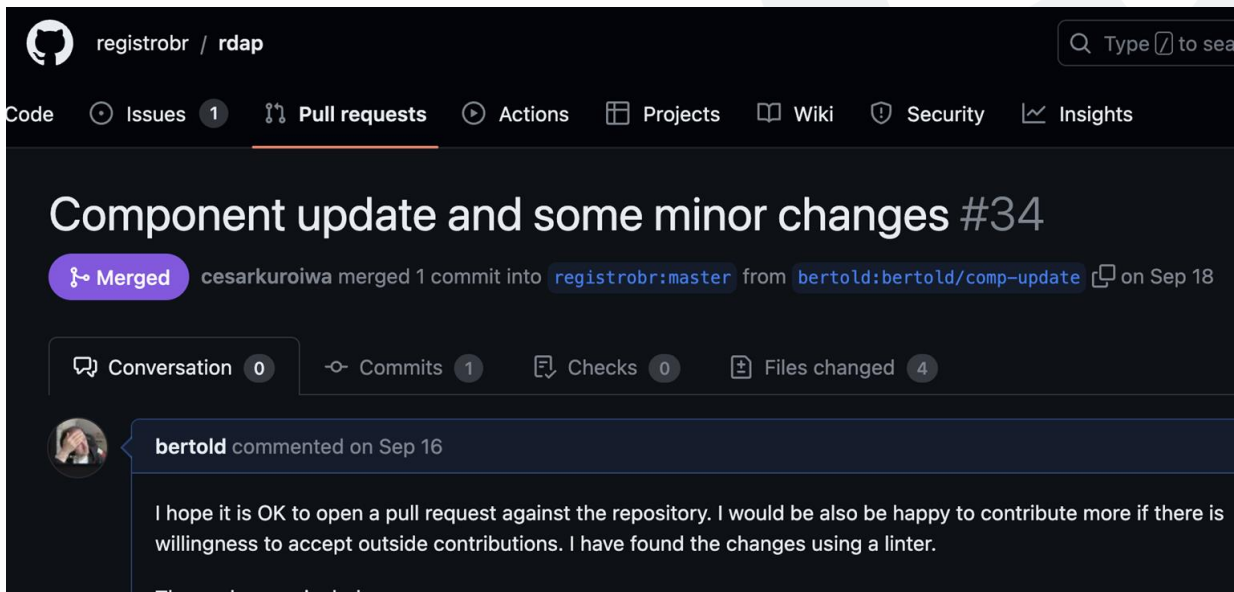
```
chi / go.mod   
  
 pkietyka go.mod: set to go 1.14 directive  
  
Code Blame 3 lines (2 loc) · 41 Bytes  
  
1 module github.com/go-chi/chi/v5  
2  
3 go 1.14
```

GitHub Insights for Assessing 3rd-Party Libraries



3rd-Party Libraries

- Test openness
 - open a pull request
 - open an issue
- And test the time it takes to get a response & the quality of response



registrobr / rdap

Code Issues 1 Pull requests Actions Projects Wiki Security Insights

Component update and some minor changes #34

Merged cesarkuroiwa merged 1 commit into `registrobr:master` from `bertold:bertold/comp-update` on Sep 18

Conversation 0 Commits 1 Checks 0 Files changed 4

bertold commented on Sep 16

I hope it is OK to open a pull request against the repository. I would be also be happy to contribute more if there is willingness to accept outside contributions. I have found the changes using a linter.

Example: go-resty

- Makes interacting with RESTful HTTP APIs more convenient
- But ...
 - maintainer non-responsive for a long time
 - release frequency was poor until last month
- And with a [defect](#) present in 2.7.0
 - sync pool data race condition
 - occurred a few times a day on a production system
 - spent a lot of time chasing the issue
 - only fixed in March without a release tag

v2.8.0

last month d9bcfaa zip tar.gz Notes

v2.7.0

on Nov 4, 2021 3d08b36 zip tar.gz Notes

Example: go-resty

Library used to incorrectly handle buffers across concurrent requests

Event Attributes

Trace (1)

Metrics

Processes

```
{
  ⚙️ body      {"data":
               {"host":"apollo.io","debug_message":"","domain_age_found":true,"domain_registered":
               "yes","domain_creation_date":"2018-01-09","domain_age_in_days":1627,"domain_age_in_
               months":52,"domain_age_in_years":4},"credits_remained":1016494.53,"estimated_querie
               s":"2,032,989","elapsed_time":"0.36","success":true,"error":""}rue,"error":""}
  env         dev
  error       invalid character 'r' after top-level value
```



Keep Go and Dependencies Up-to-Date

- Dependencies - regular updates in all repositories
 - [Renovate](#) bot is a life saver
- Use [govulncheck](#) to test for vulnerable components
- Go - sign up for release announcements
 - 1.21.3 addresses rapid stream reset vulnerability
- Recent entry from Cloudflare [blog](#)
 - HTTP/2 Zero-Day vulnerability results in record-breaking DDoS attacks

```
module github.com/bertold/lascon2023

go 1.21
toolchain go1.21.3
```

Verosint > public > Verosint CLI > Merge requests > 166

Update module github.com/tidwall/gjson to v1.17.0

Merged deploy_bot_v2 requested to merge renovate/all-minor-patch into main Sep 25, 2023, 6:05 AM

Overview 1 Commits 1 Pipelines 2 Changes 2

This MR contains the following updates:

Package	Type	Update	Change
github.com/tidwall/gjson	require	minor	v1.16.0 -> v1.17.0

Release Notes

► tidwall/gjson (github.com/tidwall/gjson)

Common Pattern for Go API Implementations

- **Configure the routes**
 - Associate query paths with handler functions using a multiplexer/router
 - Implicitly configure the HTTP methods to handle
- **Different router packages available: [built-in](#), [chi](#), [gorilla](#)**
- **Implement the handler function**
 - router invokes handler function
 - **parallel** executions should be expected
- **Handler function** ←
 - **validates** request (request parameters, headers, payload)
 - **executes** business logic
 - **sends** response to client

Recovery function

- An unrecoverable issue in the handler might cause unexpected state in the application
 - for example: nil pointer dereference
 - in a go routine: it may crash the app
- Create a recovery function
 - allows graceful recovery
 - and the recovery function can also log the details about the crash for diagnosis

Example Recovery Handler

```
func Recovery(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        defer func() {
            if rvr := recover(); rvr != nil && rvr != http.ErrAbortHandler {
                fmt.Println("I am panicking now!")
                debug.PrintStack()

                w.Header().Add("Content-Type", "application/json")
                w.WriteHeader(http.StatusInternalServerError)
                _, _ = w.Write([]byte(`{"error": "internal server error"}`))
            }
        }()
        next.ServeHTTP(w, r)
    })
}
```

Basic Checks

- Disable methods not used:
 - TRACE, HEAD, OPTIONS (may be needed for CORS)
 - but possibly other unused ones: GET, PUT, POST, DELETE
- Check request headers

Header	Questions
Accept	can the client accept the content you produce?
Content-Type	do you support this content from the client?
Content-Length	is it present, is valid, is it too large?
Content-Encoding	do you really need to accept chunked encoding?

Rate limiting

- Rate limits could be tied to
 - source IP/port (if no authorization is needed),
 - access token,
 - or a combination of rules
- Go has simple built-in rate limiting
 - better to use a library such as [redis-go](#)
 - especially when multiple containers/apps are serving
- Most implementations provide hints to the clients about rate limits using [response headers](#)
 - Ratelimit-Limit, Ratelimit-Remaining, Ratelimit-Reset

Fuzzing

- Fuzzing framework built into Go
 - can be run for a limited time
 - can be pre-seeded with corpus (~ test data)
- Best option: fuzz the business logic
- Alternatively:
 - fuzz the handler
 - fuzz the API over network - don't run it against production(!)

Payload Validation Using JSON Schema

- JSON payload in HTTP requests may have malicious content
- JSON schema has powerful ways to validate content
 - libraries such as [gojsonschema](#) makes eliminates the need for writing additional code
- Examples of rules:
 - setting minimum / maximum length for strings
 - leveraging built-in types (e.g. IPv4 address, UUID)
 - **limiting** possible property values with a regular expression
 - setting minimum, maximum size for arrays, mandating unique values
 - **disabling additional properties** to prevent actors using undefined properties
 - allow only a list of fixed values (enumerations)
 - making properties **mandatory**
- Relevant specifications: [OpenAPI](#), [JSON Schema](#)

Payload Validation Using JSON Schema

Example from Verosint API docs at
<https://docs.verosint.com>

ip **required**

IPv4 address or IPv6 global unicast address

IPV4

IPV6

```
"IdentifierIP": {  
  "oneOf": [  
    {  
      "$ref": "#/components/schemas/IPv4"  
    },  
    {  
      "$ref": "#/components/schemas/IPv6"  
    }  
  ],  
  "example": "192.0.2.1",  
  "description": "IPv4 address or IPv6 global unicast address"  
},
```

```
"IPv4": {  
  "description": "IPv4 address",  
  "type": "string",  
  "format": "ipv4",  
  "example": "172.56.89.52"  
},  
"IPv6": {  
  "description": "IPv6 address",  
  "type": "string",  
  "format": "ipv6",  
  "example": "2607:fb91:128d:af60:d5f2:2a2e:b89e:b411"  
},
```

Payload Validation Using JSON Schema

timestamp date-time	<input type="text" value="2023-02-28T15:4"/>
accountId string Account identifier.	<p>type: string format: date-time</p>
ip required IPv4 address or IPv6 global unicast address	<p>Use example value 2023-02-28T15:49:33.121-06:00</p>
type string Event type	<p>✓ LOGIN_SUCCESS LOGIN_FAILED MFA_SUCCESS MFA_FAILED VERIFICATION_SUCCESS</p>

ADD OBJECT

ESPONSES

```
"timestamp": {  
  "type": "string",  
  "format": "date-time",  
  "example": "2023-02-28T15:49:33.121-06:00"  
},
```

```
"type": {  
  "type": "string",  
  "description": "Event type",  
  "enum": [  
    "LOGIN_SUCCESS",  
    "LOGIN_FAILED",  
    "MFA_SUCCESS",  
    "MFA_FAILED",  
    "VERIFICATION_SUCCESS"  
  ]  
}
```


There is so much more to cover ...

That we did not talk about.

- HTTP Server configuration options
 - timeouts (read, header read, idle time out)
 - connection management
 - maximum size of header
 - TLS configuration
- Rate limiting [headers](#)
- Authentication/authorization
- Nuances of each HTTP method
 - e.g. GET - URL escaping, leakage of information in logs
- Preventing caching of responses

Recap

- Understand the deployment of your application
 - what protections are available at what layer
- Building secure APIs require secure toolchain
 - including Go runtime and 3rd-party dependencies
 - keep them up to date
 - be selective about dependencies - less is more
- Make your APIs resilient
 - protect the application from crashes
 - rate limit clients
- Inspect incoming requests
 - headers, payload length, format
 - reduce manual coding using JSON schema validation
 - emit logs that can trigger automated defensive actions
- Test your APIs, business logic with fuzzing

Additional Resources

- [OWASP Top 10 API Security Risks](#)
- [Open Source Security Foundation](#)
 - [scorecard](#) app
- [Getting started with Fuzzing](#)
- [How to Parse a JSON Request Body in Go](#)
- [Make resilient Go net/http servers using timeouts, deadlines and context cancellation](#)
- [Tool selection](#) from ISTQB Certified Tester Advanced Level Test Manager Syllabus



bit.ly/lascon2023

Thank you

See you at

<https://bit.ly/bertold>
<https://www.verosint.com>