Unintended Route Exposure via DefaultServeMux Fallback #3810

Open #3814

Labels bug

August829 opened 3 weeks ago

Unintended Route Exposure via DefaultServeMux Fallback in go-kratos/kratos

https://github.com/go-kratos/kratos

Summary

Field	Value
Product	go-kratos/kratos
Version	v2.9.2 (and all prior v2.x versions)
Component	`transport/http/server.go`
Vulnerability Type	CWE-441 (Unintentional Proxy or Intermediary)
Severity	High
CVSS 3.1 Score	7.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)
Attack Vector	Network

Affected Code

File: `transport/http/server.go`
Lines: 192-193
Function: `NewServer()`

```
func NewServer(opts ...ServerOption) *Server {
    srv := &Server{
        // ...
    }
    srv.router.NotFoundHandler = http.DefaultServeMux      // line 192
    srv.router.MethodNotAllowedHandler = http.DefaultServeMux // line 193
    // ...
}
```

Description

The Kratos HTTP server (`transport/http/server.go`) sets `http.DefaultServeMux` as the fallback handler for both unmatched routes (`NotFoundHandler`) and disallowed methods (`MethodNotAllowedHandler`).

In Go, `http.DefaultServeMux` is a global, shared ServeMux instance. Numerous standard library and third-party packages automatically register handlers on it during `init()` . The most notable example is `net/http/pprof` , which registers debug profiling endpoints. Kratos itself ships a pprof wrapper package (`transport/http/pprof`) that imports `net/http/pprof` .

When any package in the application's dependency tree performs a side-effect import of `net/http/pprof` (i.e., `import _ "net/http/pprof"`), the following debug endpoints become silently accessible through the Kratos HTTP server — **without any authentication, authorization, or explicit registration by the developer**:

- `/debug/pprof/` — Profile index page
- `/debug/pprof/cmdline` — Process command-line arguments
- `/debug/pprof/profile` — CPU profile
- `/debug/pprof/symbol` — Symbol lookup
- `/debug/pprof/trace` — Execution trace
- `/debug/pprof/heap` — Heap memory dump
- `/debug/pprof/goroutine` — Goroutine stack traces

This is because any HTTP request to a path not registered in the Kratos router falls through to `DefaultServeMux` , which serves these debug handlers.

Root Cause Analysis

```
Incoming HTTP Request (e.g., GET /debug/pprof/)
  |
  v
gorilla/mux Router
  |-- Route matched? --> No
  |
  v
NotFoundHandler = http.DefaultServeMux
  |
  v
DefaultServeMux route table (populated by init() side effects):
  /debug/pprof/        -> pprof.Index
  /debug/pprof/cmdline -> pprof.Cmdline
  /debug/pprof/profile -> pprof.Profile
  /debug/pprof/symbol  -> pprof.Symbol
  /debug/pprof/trace   -> pprof.Trace
  |
  v
Response: 200 OK + sensitive profiling data
```

The expected behavior for an unmatched route is to return `404 Not Found`. Instead, the request is proxied to `DefaultServeMux`, which may serve unintended handlers.

Impact

An unauthenticated remote attacker can access debug profiling endpoints to obtain:

1. **Process Command-Line Arguments** (`/debug/pprof/cmdline`): May contain database passwords, API keys, or other secrets passed as flags.
2. **Full Goroutine Stack Dumps** (`/debug/pprof/goroutine?debug=2`): Exposes absolute source code file paths, internal package structure, dependency versions, and runtime state.
3. **Heap Memory Dumps** (`/debug/pprof/heap`): Contains in-memory data that may include credentials, session tokens, encryption keys, PII, or other sensitive runtime data.
4. **CPU Profiles** (`/debug/pprof/profile`): Reveals hot code paths, business logic structure, and internal function names.
5. **Execution Traces** (`/debug/pprof/trace`): Provides detailed request processing traces, including timing information for all goroutines.

Additionally, the `/debug/pprof/profile` endpoint performs a 30-second CPU profile by default, which can be abused for **denial of service** by triggering multiple concurrent profiling requests.

Proof of Concept

Prerequisites

A standard Kratos helloworld application with `net/http/pprof` imported (extremely common in Go production services for diagnostics):

```
package main

import (
    _ "net/http/pprof"  // commonly added for performance debugging
    // ... other imports
)
```

Exploitation

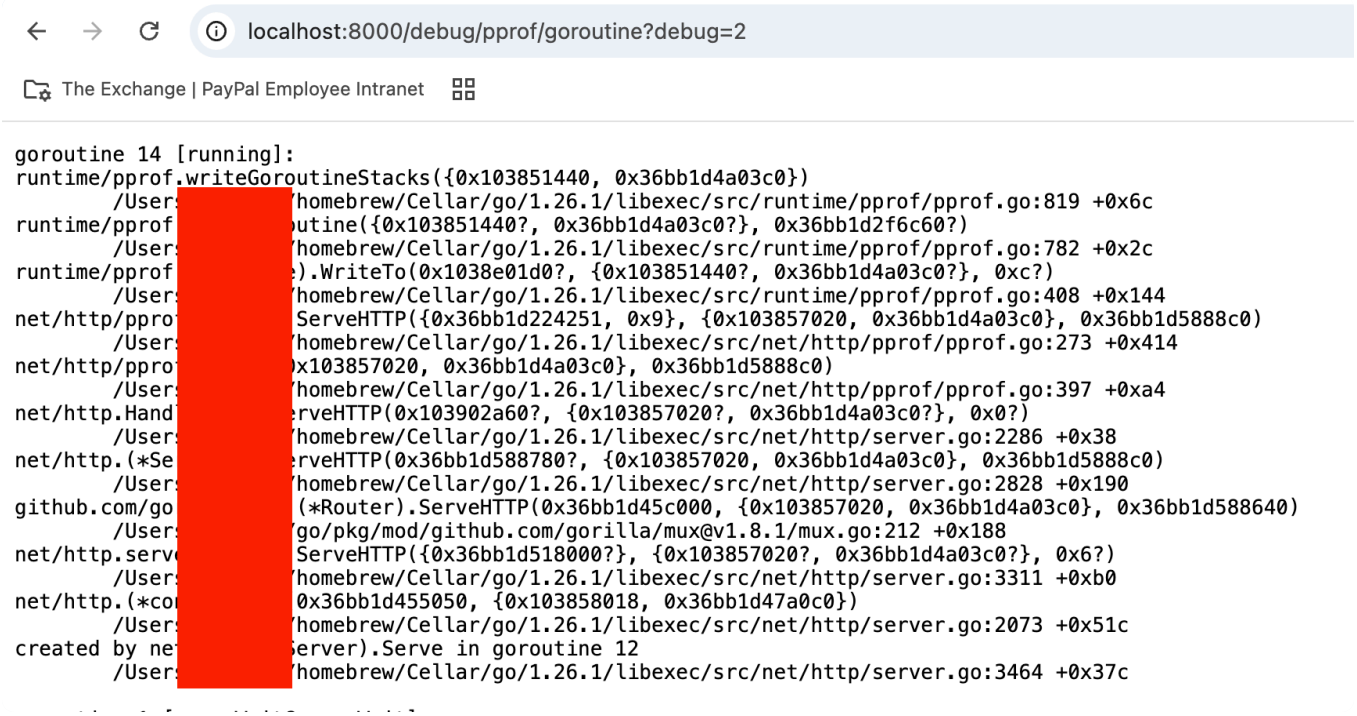
```
# 1. Access pprof index — lists all available profiles
curl http://target:8000/debug/pprof/

# 2. Retrieve process command-line arguments (may contain secrets)
curl http://target:8000/debug/pprof/cmdline

# 3. Dump all goroutine stacks with full source paths
curl "http://target:8000/debug/pprof/goroutine?debug=2"

# 4. Download heap memory dump and extract strings
curl -s http://target:8000/debug/pprof/heap > heap.out
strings heap.out | grep -iE "password|secret|token|key"

# 5. Capture execution trace
curl -o trace.out "http://target:8000/debug/pprof/trace?seconds=5"
go tool trace trace.out
```



Verified Results

Tested against Kratos v2.9.2 helloworld application on `localhost:8000`:

Endpoint	HTTP Status	Data Exposed
`GET /debug/pprof/`	**200**	HTML index listing all profile types (allocs, block, goroutine, heap, mutex, etc.)
`GET /debug/pprof/cmdline`	**200**	`./bin/helloworld -conf ./configs`
`GET /debug/pprof/goroutine?debug=2`	**200**	Full goroutine stacks with absolute paths: `/Users/*/kratos/v2@v2.9.2/app.go:143`, dependency versions, framework internals
`GET /debug/pprof/heap`	**200**	1571 bytes binary heap profile
`GET /debug/pprof/trace?seconds=1`	**200**	Binary execution trace data
`GET /debug/pprof/symbol`	**200**	Symbol resolution endpoint

All endpoints returned **200 OK** with sensitive data. No authentication was required.

Suggested Fix

Replace `http.DefaultServeMux` with dedicated handlers that return proper HTTP error responses:

```
func NewServer(opts ...ServerOption) *Server {
    srv := &Server{
        // ...
    }
    // FIXED: Return proper 404/405 instead of falling through to DefaultServeMux
    srv.router.NotFoundHandler = http.HandlerFunc(func(w http.ResponseWriter, r *http.Request
        w.WriteHeader(http.StatusNotFound)
    })
    srv.router.MethodNotAllowedHandler = http.HandlerFunc(func(w http.ResponseWriter, r *http
        w.WriteHeader(http.StatusMethodNotAllowed)
    })
    // ...
}
```

Mitigation (for current users)

Until an official patch is released, users can override the default handlers when creating the server:

```
import "net/http"

srv := http.NewServer(
    http.NotFoundHandler(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        w.WriteHeader(http.StatusNotFound)
    })),
    http.MethodNotAllowedHandler(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request
        w.WriteHeader(http.StatusMethodNotAllowed)
    })),
    // ... other options
)
```

References

- Vulnerable source code: https://github.com/go-kratos/kratos/blob/v2.9.2/transport/http/server.go#L192-L193
- Go `net/http/pprof` auto-registration: https://pkg.go.dev/net/http/pprof
- Kratos security policy: https://github.com/go-kratos/kratos/blob/main/SECURITY.md
- CWE-441: https://cwe.mitre.org/data/definitions/441.html

August829 added **bug** 3 weeks ago

1911860538 2 weeks ago Contributor

Thanks for reporting this security issue. I think you are correct, this is a valid vulnerability. Can you submit a pull request to fix this issue?

Yanhu007 added a commit that references this issue 2 weeks ago

`fix: replace http.DefaultServeMux fallback handlers with safe default` ··· `0284a5b`

Yanhu007 linked a pull request that will close this issue 2 weeks ago

fix: replace http.DefaultServeMux fallback with safe defaults #3814

Metadata

Assignees

No one assigned

Labels

bug

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

fix: replace http.DefaultServeMux fallback with safe defaults
go-kratos/kratos

Participants