

New issue

[Jump to bottom](#)

net/http/httputil: ReverseProxy should not forward unparseable query parameters #54663

✓ Closed neild opened this issue on Aug 24 · 11 comments

Assignees



Labels

NeedsFix Security

Milestone

Go1.20

neild commented on Aug 24

Contributor

We generally treat malformed value pairs in URL queries as a soft error, ignoring the invalid pair but accepting others. For example, the following ([playground link](#)) accepts the value for the key `b` even as it rejects the invalid one for `a`:

```
u, _ := url.Parse("http://go.dev/?a=%x&b=ok")
v, err := url.ParseQuery(u.RawQuery)
fmt.Println(v, err)
// map[b:[ok]] invalid URL escape "%x"
```

`ReverseProxy` should not include unparseable query parameters when forwarding a request, since this is a vector for parameter smuggling. In Go 1.17, we changed URL parsing to reject keys containing a semicolon (<https://go.dev/issue/25192>). If a Go 1.17 `ReverseProxy` forwards a request to a backend which treats semicolons as a parameter separator (as Go 1.16 and earlier did), the proxy and backend may disagree on the parameter values of the request.

Thanks to Oxeye for pointing out this issue: <https://www.oxeye.io/blog/golang-parameter-smuggling-attack>



1

seankhliao added Security NeedsFix labels on Aug 24

gopherbot commented on Aug 24

Change <https://go.dev/cl/425417> mentions this issue: `net/http/httputil`: `ReverseProxy` should not forward unparseable query parameters

📌  **seankhliao** added this to the **Go1.20** milestone on Aug 27

chrisguiney commented on Aug 29

Contributor

Is this going to be present in a 1.19.x release, or strictly a 1.20 change?

Unfortunately it's behavior that I'm currently depending on, so I'll need to find a way to maintain the current behavior. For context, I'm reverse proxying to a server that doesn't use `;` as a separator, but does allow them as part of a key or value.

I've accepted that there's apparently no room for discussion on how to handle semicolons. I do hope that there can at least be a way to restore prior behavior. Working around this is going to be difficult given the changeset in <https://go.dev/cl/425417>

neild commented on Aug 29

Contributor

Author

Is this going to be present in a 1.19.x release, or strictly a 1.20 change?

We don't even know what the fix for this is yet.

The basic requirement as I see it is that if a proxy `Director` or `Rewrite` func parses the query parameters, we must not forward unparsed parameters by default. Given that, what change can we make that has the least impact and preserves the most flexibility?

chrisguiney commented on Aug 30

Contributor

Ah sorry, I hadn't looked closely enough at the PR to see it wasn't merged yet.

My ideal solution at this point would be to introduce `url.QueryParser` and `url.QueryEncoder` interfaces, with default implementations calling `url.ParseQuery` and `Values.Encode` respectively (and exposed through package variables `url.DefaultQueryParser` and `url.DefaultQueryEncoder`).

From there, updating `ReverseProxy` with a `url.QueryParser` field that defaults to `url.DefaultQueryParser`. The same could be done for the `net/http` package, and anywhere else urls are parsed in the standard library.

This approach should maintain backwards compatibility, while allowing people with particular needs the ability to supply their own parser/encoder. In the long run, I think this is the way to go -- urls are tricky things that are both ubiquitous and varying interpretations of the RFCs exist throughout the internet.

I also realize that's touching more components than just `httputil.ReverseProxy` -- and would be a much bigger lift. I'd personally be happy with even just a boolean flag, or context variable to restore previous behavior, or maybe a `QueryParser func(s) (Values, error)` field

quick sketch of the interface described above:

```
var (  
    DefaultQueryParser QueryParser = defaultQueryParser{  
    DefaultQueryEncoder QueryEncoder = defaultQueryEncoder{  
)  
  
type QueryParser {  
    Parse(string) (Values, error)  
}  
  
type QueryEncoder {  
    Encode(Values) (string, error)  
}  
  
type defaultQueryParser struct{  
  
func (defaultQueryParser) Parse(s string) (Values, error) {  
    return ParseQuery(s)  
}  
  
func (defaultQueryParser) Encode(v Values) (string, error) {  
    return v.Encode(), nil  
}
```

 **neild** self-assigned this on Sep 12

neild commented on Sep 12

Contributor

Author

Behavior which I think strikes the right balance between security and compatibility:

Clean the outgoing request's `req.URL.RawQuery` to remove unparseable parameters

- If `req.Form != nil` after calling `ReverseProxy.Director`.
- Before calling `ReverseProxy.Rewrite`.

When using the `Director` hook, `RawQuery` is cleaned only if the hook parsed the query parameters. A `Director` func which doesn't examine query parameters will pass garbage through untouched, but that's okay because it (probably) wasn't making decisions based on it.

When using the `Rewrite` hook (new in 1.20, intended to be the recommended replacement for `Director` for all uses), `RawQuery` gets cleaned by default. (We can't easily conditionalize this on whether the hook parses the form or not, because a design goal of `Rewrite` is that we give it the outgoing request we intend to send - we don't want to modify that request further after `Rewrite` returns.) To disable this, you can just restore the old value in `Rewrite`:

```
Rewriter: func(r *ProxyRequest) {  
    r.Out.URL = r.In.URL // use uncleaned query parameters  
}
```

gopherbot commented on Sep 22

Change <https://go.dev/cl/432976> mentions this issue: `net/http/httputil`: avoid query parameter smuggling

neild commented on Sep 23

Contributor

Author

This is [CVE-2022-2880](#).

neild commented on Sep 23

Contributor

Author

@**gopherbot** please open backport issues

 This was referenced on Sep 23

net/http/httputil: ReverseProxy should not forward unparseable query parameters [1.18 backport] #55842

 Closed

net/http/httputil: ReverseProxy should not forward unparseable query parameters [1.19 backport] #55843

 Closed

gopherbot commented on Sep 23

Backport issue(s) opened: [#55842](#) (for 1.18), [#55843](#) (for 1.19).

Remember to create the cherry-pick CL(s) as soon as the patch is submitted to master, according to <https://go.dev/wiki/MinorReleases>.




gopherbot closed this as completed in [7c84234](#) on Sep 23

gopherbot commented on Sep 23

Change <https://go.dev/cl/433695> mentions this issue: [release-branch.go1.18] net/http/httputil: avoid query parameter smuggling

gopherbot commented on Sep 23

Change <https://go.dev/cl/433735> mentions this issue: [release-branch.go1.19] net/http/httputil: avoid query parameter smuggling

 **gopherbot** pushed a commit that referenced this issue on Sep 28



[release-branch.go1.19] net/http/httputil: avoid query parameter smug... ...

f6d8445

 **gopherbot** pushed a commit that referenced this issue on Sep 28



[release-branch.go1.18] net/http/httputil: avoid query parameter smug... ...

9d2c73a

 **bradfitz** pushed a commit to tailscale/go that referenced this issue on Oct 5



[release-branch.go1.19] net/http/httputil: avoid query parameter smug... ...

89f9aec

 **bradfitz** pushed a commit to tailscale/go that referenced this issue on Oct 5



[release-branch.go1.19] net/http/httputil: avoid query parameter smug... ...

fa8477c

 **bradfitz** pushed a commit to tailscale/go that referenced this issue on Oct 5



[release-branch.go1.19] net/http/httputil: avoid query parameter smug... ...

7599bf9

 **rcrozean** pushed a commit to rcrozean/go that referenced this issue on Oct 14





[release-branch.go1.18] net/http/httputil: avoid query parameter smug... ...

be3e286



rcrozean pushed a commit to rcrozean/go that referenced this issue on Oct 14



[release-branch.go1.18] net/http/httputil: avoid query parameter smug... ...

1ff22d7



chrisguiney mentioned this issue on Oct 18

proposal: net/url: add QueryParser and QueryEncoder interfaces #56300

🕒 Open



wormi4ok added a commit to wormi4ok/packages that referenced this issue on Oct 20



golang: update to v1.19.2 ...

4959bb2



wormi4ok mentioned this issue on Oct 20

golang: update to v1.19.2 openwrt/packages#19652

🔗 Merged



rcrozean pushed a commit to rcrozean/go that referenced this issue on Oct 24



[release-branch.go1.18] net/http/httputil: avoid query parameter smug... ...

34f51dd



wormi4ok added a commit to wormi4ok/packages that referenced this issue 12 days ago



golang: update to v1.19.2 ...

0ad7a2f



1715173329 pushed a commit to immortalwrt/packages that referenced this issue 12 days ago



golang: update to v1.19.2 ...

d69bed8



1582130940 pushed a commit to 1582130940/OpenWrt-Packages that referenced this issue 11 days ago



golang: update to v1.19.2 ...

f969d94




1582130940 pushed a commit to 1582130940/OpenWrt-Packages that referenced this issue 11 days ago



golang: update to v1.19.2 ...

2282f70



 **1582130940** pushed a commit to 1582130940/OpenWrt-Packages that referenced this issue 11 days ago

 golang: update to v1.19.2 ...


923c998

 **1582130940** pushed a commit to 1582130940/OpenWrt-Packages that referenced this issue 11 days ago

 golang: update to v1.19.2 ...

d2e428b

Assignees

 neild

Labels

NeedsFix Security

Projects

None yet

Milestone

Go1.20

Development

No branches or pull requests

4 participants

