



MITIGATING SSRF VULNERABILITIES IN GO

A PRACTICAL GUIDE

B S i d e s A t h e n s 2 0 2 3

Marcin Niemiec

ABOUT ME



- Cloud Security Engineer at Form3
- Blog: xvnpw.github.io
- Twitter: [@xvnpw](https://twitter.com/xvnpw)

FORM3 IS REAL-TIME PAYMENTS PROCESSING PLATFORM



- One single integration to multiple payment schemes
- Multi-cloud platform (AWS, GCP, Azure)
- Go, IaC (Terraform)
- Fully remote

SSRF

Server-Side Request Forgery



VULNERABLE CODE

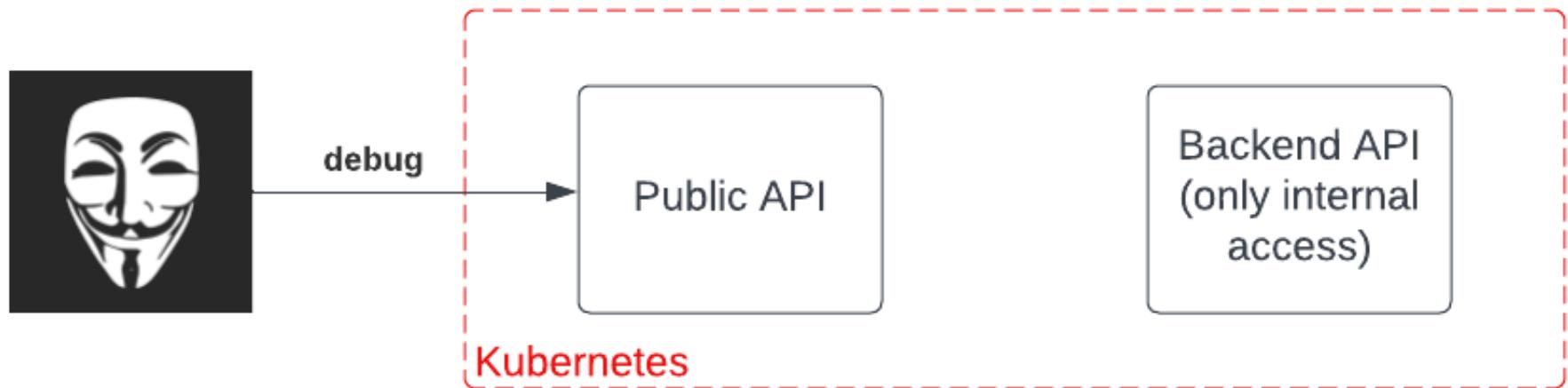
```
router.GET("/debug", func(context *gin.Context) {
    urlFromUser := context.Query("url")
    // no validation yloo
    resp, err := http.Get(urlFromUser)
```

OWASP TOP 10

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL.

SSRF is included in OWASP TOP 10 at position 10 in 2021 edition

SETUP



EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

Code

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // no validation yloo  
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \  
2       http://publicapi/debug?url\  
3           http://backendapi/internal  
4 This is internal sensitive endpoint
```

EXPLOIT 1

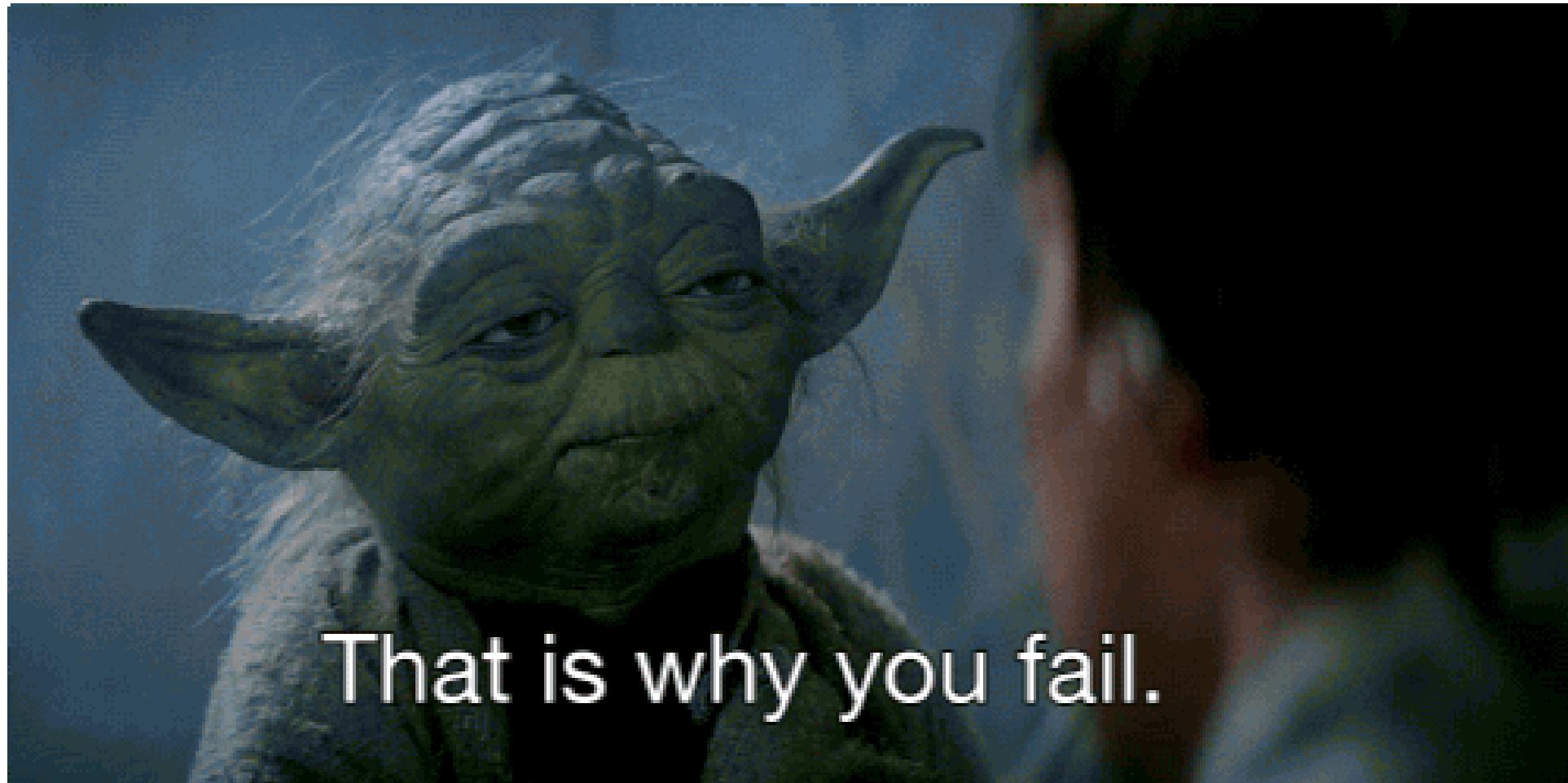
Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // no validation yloo
4     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2       http://publicapi/debug?url\=
3           http://backendapi/internal
4 This is internal sensitive endpoint
```

FAIL...



That is why you fail.

FINDING THE PROBLEM WITH GOSEC

gosec is Go security checker

```
$ gosec publicapi/
G107 (CWE-88): Potential HTTP request made with variable url
(Confidence: MEDIUM, Severity: MEDIUM)
    83: urlFromUser := context.Query("url")
    > 84: resp, err := http.Get(urlFromUser)
```

TRY TO FIX

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() != "backendapi") {
8         return true
9     }
10
11    return false
12 }
```

TRY TO FIX

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() != "backendapi") {
8         return true
9     }
10
11    return false
12 }
```

TRY TO FIX

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() != "backendapi") {
8         return true
9     }
10
11    return false
12 }
```

TRY TO FIX

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() != "backendapi") {
8         return true
9     }
10
11    return false
12 }
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2       http://publicapi/debug?url\=
3           http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2       http://publicapi/debug?url\=
3           http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2       http://publicapi/debug?url\=
3           http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\
3     http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\
3     http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

EXPLOIT 2

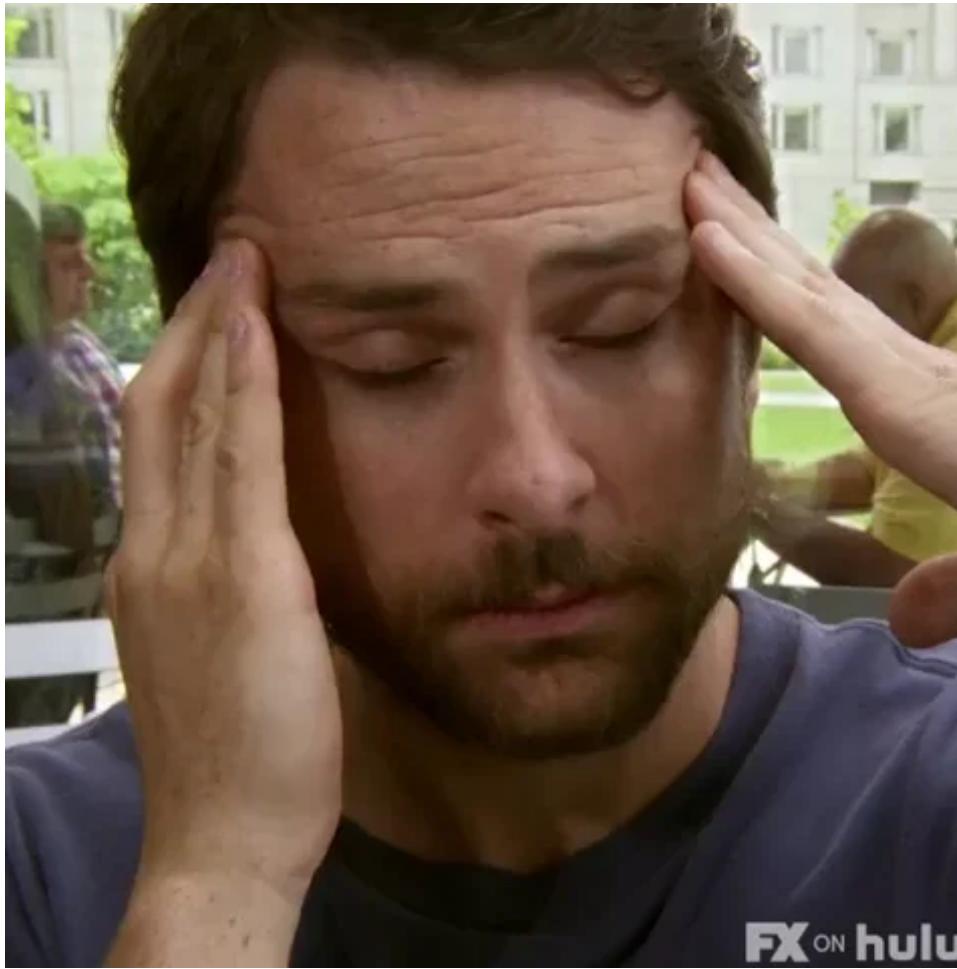
Code

```
1 router.GET("/debug", func(context *gin.Context) {
2     urlFromUser := context.Query("url")
3     // validation because world is full of mean people :(
4     if !validateTargetUrl(urlFromUser) {
5         context.String(http.StatusBadRequest, "Bad url")
6         return
7     }
8     resp, err := http.Get(urlFromUser)
```

Exploit

```
1 $ curl -s \
2     http://publicapi/debug?url\
3     http://10.96.155.247/internal
4 This is internal sensitive endpoint
```

ARE YOU KIDDING ME?



FX on hulu

MANY WAYS TO WRITE SAME THING

localhost can be:

- 127.0.0.1
- 127.0.0.2
- 2130706433 ([more](#))
- form3.localtest.me

LEARNING

Negative validation is doomed to failure 

TRY TO FIX

Positive validation

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() == "imageapi") &&
8         (u.Port() == "" || u.Port() == "80" || u.Port() == "443") {
9         return true
10    }
11
12    return false
13 }
```

TRY TO FIX

Positive validation

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() == "imageapi") &&
8         (u.Port() == "" || u.Port() == "80" || u.Port() == "443") {
9         return true
10    }
11
12    return false
13 }
```

TRY TO FIX

Positive validation

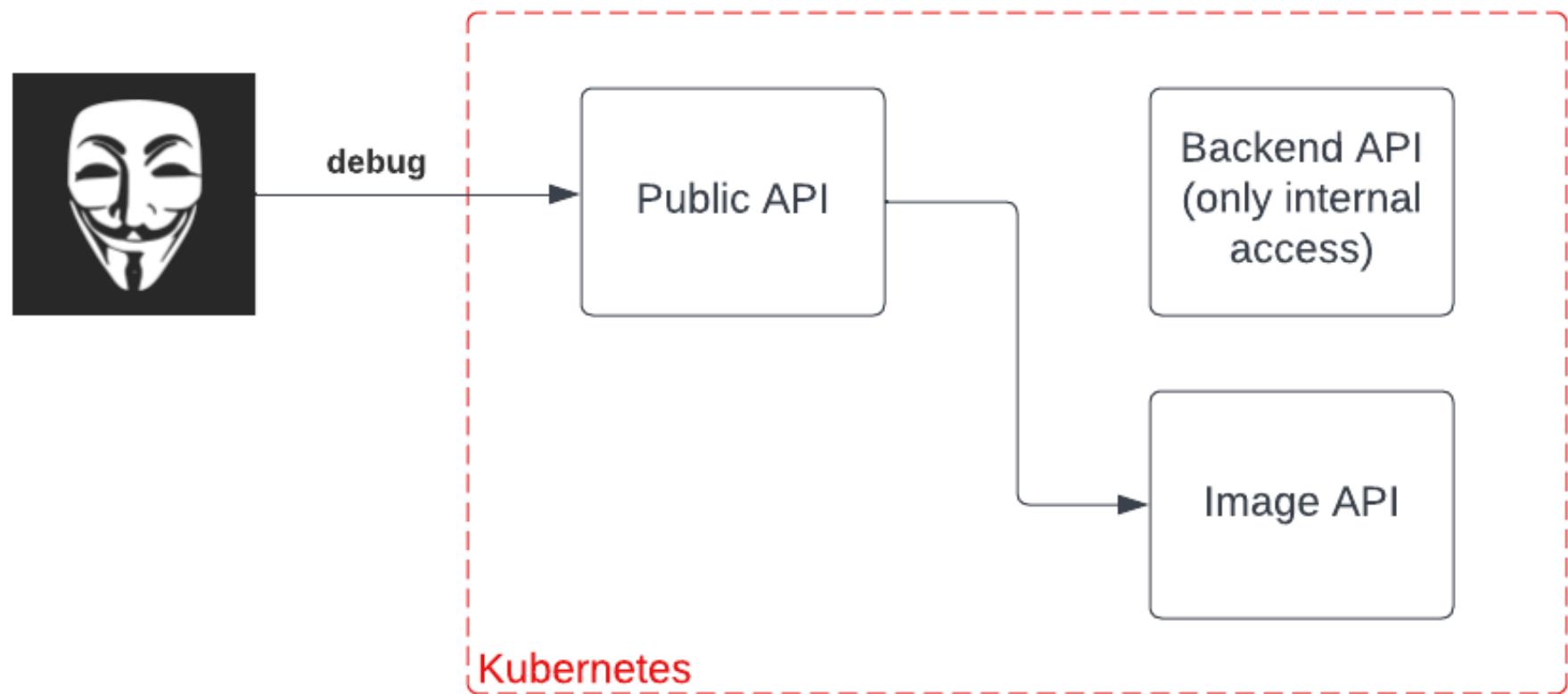
```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() == "imageapi") &&
8         (u.Port() == "" || u.Port() == "80" || u.Port() == "443") {
9         return true
10    }
11
12    return false
13 }
```

TRY TO FIX

Positive validation

```
1 func validateTargetUrl(input string) bool {
2     u, err := url.ParseRequestURI(input)
3     if err != nil {
4         return false
5     }
6     if (u.Scheme == "http" || u.Scheme == "https") &&
7         (u.Hostname() == "imageapi") &&
8         (u.Port() == "" || u.Port() == "80" || u.Port() == "443") {
9         return true
10    }
11
12    return false
13 }
```

SETUP



EXPLOIT 3

SSRF + Open Redirect

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 This is internal sensitive endpoint
```

EXPLOIT 3

SSRF + Open Redirect

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 This is internal sensitive endpoint
```

EXPLOIT 3

SSRF + Open Redirect

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 This is internal sensitive endpoint
```

EXPLOIT 3

SSRF + Open Redirect

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 This is internal sensitive endpoint
```

EXPLOIT 3

SSRF + Open Redirect

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 This is internal sensitive endpoint
```

WHAT???



STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 200 | GET "/debug?url=http://imageapi/redirect?target=
4     http://backendapi/internal"
5
6 # imageapi
7 -> /redirect request: user-agent=Go-http-client/1.1
8 301 | GET "/redirect?target=http://backendapi/internal"
9
10 # backendapi
11 -> /internal request: user-agent=Go-http-client/1.1
12 200 | GET "/internal"
```

STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 200 | GET "/debug?url=http://imageapi/redirect?target=
4     http://backendapi/internal"
5
6 # imageapi
7 -> /redirect request: user-agent=Go-http-client/1.1
8 301 | GET "/redirect?target=http://backendapi/internal"
9
10 # backendapi
11 -> /internal request: user-agent=Go-http-client/1.1
12 200 | GET "/internal"
```

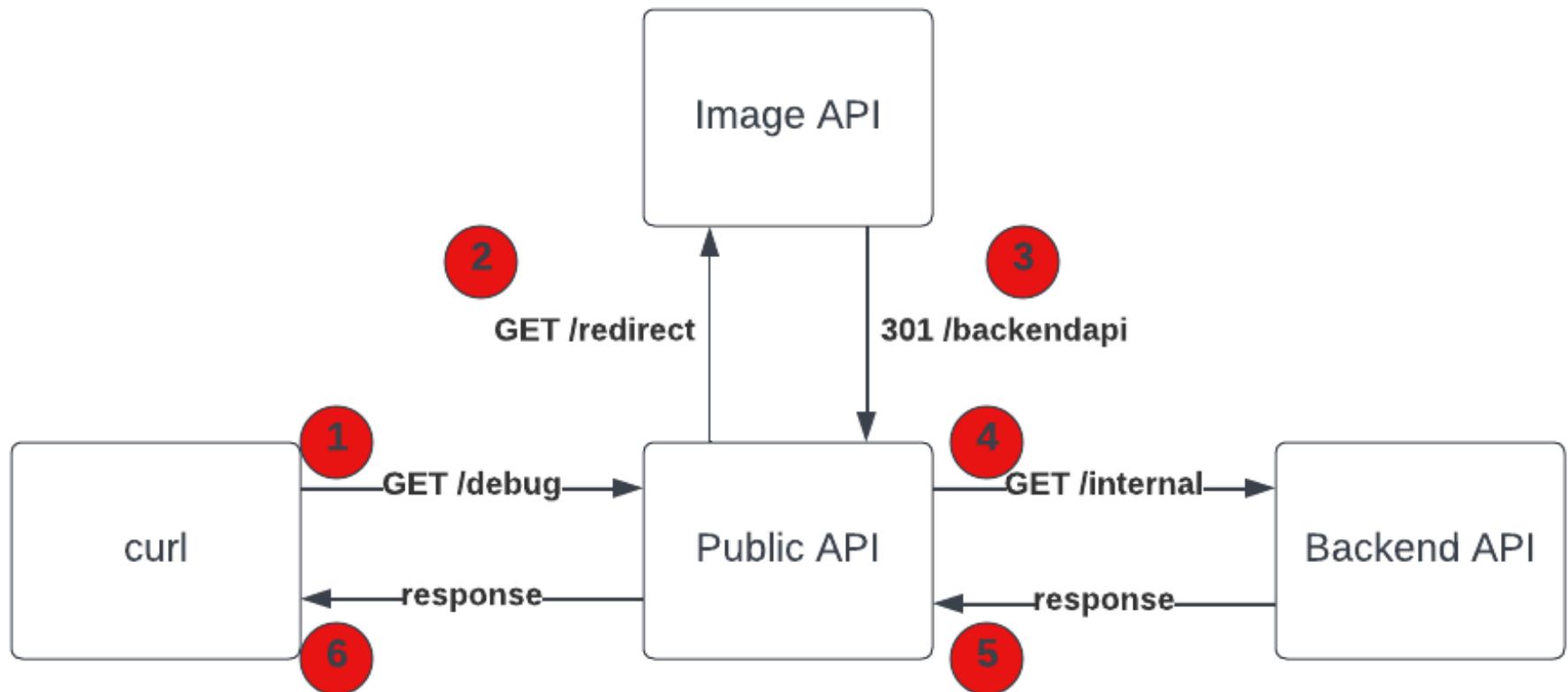
STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 200 | GET "/debug?url=http://imageapi/redirect?target=
4     http://backendapi/internal"
5
6 # imageapi
7 -> /redirect request: user-agent=Go-http-client/1.1
8 301 | GET "/redirect?target=http://backendapi/internal"
9
10 # backendapi
11 -> /internal request: user-agent=Go-http-client/1.1
12 200 | GET "/internal"
```

STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 200 | GET "/debug?url=http://imageapi/redirect?target=
4     http://backendapi/internal"
5
6 # imageapi
7 -> /redirect request: user-agent=Go-http-client/1.1
8 301 | GET "/redirect?target=http://backendapi/internal"
9
10 # backendapi
11 -> /internal request: user-agent=Go-http-client/1.1
12 200 | GET "/internal"
```

STEP BY STEP



VALIDATION PROBLEM

`http.Get` knows nothing about validation

```
1 router.GET("/debug", func(context *gin.Context) {  
2     urlFromUser := context.Query("url")  
3     // validation because world is full of mean people :(  
4     if !validateTargetUrl(urlFromUser) {  
5         context.String(http.StatusBadRequest, "Bad url")  
6         return  
7     }  
8     resp, err := http.Get(urlFromUser)
```

Validation is here

This call knows
nothing about
validation

REDIRECT ME ONE MORE TIME...



SAFEURL

A Server Side Request Forgery
(SSRF) protection library. Made with
♥ by Doyensec LLC.

doyensec/safeurl

FIX WITH SAFEURL

Positive validation enhanced

```
1 config := safeurl.GetConfigBuilder().
2     SetAllowedHosts("imageapi").
3     Build()
4 router.GET("/debug", func(context *gin.Context) {
5     urlFromUser := context.Query("url")
6     client := safeurl.Client(config)
7     resp, err := client.Get(urlFromUser)
```

FIX WITH SAFEURL

Positive validation enhanced

```
1 config := safeurl.GetConfigBuilder().
2     SetAllowedHosts("imageapi").
3     Build()
4 router.GET("/debug", func(context *gin.Context) {
5     urlFromUser := context.Query("url")
6     client := safeurl.Client(config)
7     resp, err := client.Get(urlFromUser)
```

FIX WITH SAFEURL

Positive validation enhanced

```
1 config := safeurl.GetConfigBuilder().
2     SetAllowedHosts("imageapi").
3     Build()
4 router.GET("/debug", func(context *gin.Context) {
5     urlFromUser := context.Query("url")
6     client := safeurl.Client(config)
7     resp, err := client.Get(urlFromUser)
```

FIX WITH SAFEURL

Positive validation enhanced

```
1 config := safeurl.GetConfigBuilder().
2     SetAllowedHosts("imageapi").
3     Build()
4 router.GET("/debug", func(context *gin.Context) {
5     urlFromUser := context.Query("url")
6     client := safeurl.Client(config)
7     resp, err := client.Get(urlFromUser)
```

FIX WITH SAFEURL

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 Get "http://imageapi/redirect?target=http://backendapi/
6 internal": dial tcp 10.96.45.24:80: ip: 10.96.45.24 not
7 found in allowlist
```

FIX WITH SAFEURL

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 Get "http://imageapi/redirect?target=http://backendapi/
6 internal": dial tcp 10.96.45.24:80: ip: 10.96.45.24 not
7 found in allowlist
```

FIX WITH SAFEURL

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 Get "http://imageapi/redirect?target=http://backendapi/
6 internal": dial tcp 10.96.45.24:80: ip: 10.96.45.24 not
7 found in allowlist
```

FIX WITH SAFEURL

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 Get "http://imageapi/redirect?target=http://backendapi/
6 internal": dial tcp 10.96.45.24:80: ip: 10.96.45.24 not
7 found in allowlist
```

FIX WITH SAFEURL

```
1 $ curl -s \
2     http://publicapi/debug?url\=
3         http://imageapi/redirect?target\=
4             http://backendapi/internal
5 Get "http://imageapi/redirect?target=http://backendapi/
6 internal": dial tcp 10.96.45.24:80: ip: 10.96.45.24 not
7 found in allowlist
```

SAFEURL STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 400 | GET "/debug?url=http://imageapi/redirect?target=
4   http://backendapi/internal"
5 ip: 10.96.45.24 not found in allowlist
6
7 # imageapi
8 -> /redirect request: user-agent=Go-http-client/1.1
9 301 | GET "/redirect?target=http://backendapi/internal"
```

SAFEURL STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 400 | GET "/debug?url=http://imageapi/redirect?target=
4   http://backendapi/internal"
5 ip: 10.96.45.24 not found in allowlist
6
7 # imageapi
8 -> /redirect request: user-agent=Go-http-client/1.1
9 301 | GET "/redirect?target=http://backendapi/internal"
```

SAFEURL STEP BY STEP

```
1 # publicapi
2 -> /debug request: user-agent=curl/7.86.0
3 400 | GET "/debug?url=http://imageapi/redirect?target=
4   http://backendapi/internal"
5 ip: 10.96.45.24 not found in allowlist
6
7 # imageapi
8 -> /redirect request: user-agent=Go-http-client/1.1
9 301 | GET "/redirect?target=http://backendapi/internal"
```

WHAT ELSE IT CAN DO?

- Protect against DNS rebinding
- Validation and issuing request done by one library

[Read more](#)

HOW IT WAS IMPLEMENTED?

```
1 func buildHttpClient(wc *WrappedClient) *http.Client {  
2     client := &http.Client{  
3         ...  
4         CheckRedirect: wc.config.CheckRedirect,  
5         Transport: &http.Transport{  
6             TLSClientConfig: wc.tlsConfig,  
7             DialContext: (&net.Dialer{  
8                 Resolver: wc.resolver,  
9                 Control: buildRunFunc(wc),  
10            }).DialContext,
```

client.go

HOW IT WAS IMPLEMENTED?

```
1 func buildHttpClient(wc *WrappedClient) *http.Client {  
2     client := &http.Client{  
3         ...  
4         CheckRedirect: wc.config.CheckRedirect,  
5         Transport: &http.Transport{  
6             TLSClientConfig: wc.tlsConfig,  
7             DialContext: (&net.Dialer{  
8                 Resolver: wc.resolver,  
9                 Control: buildRunFunc(wc),  
10            }).DialContext,
```

client.go

HOW IT WAS IMPLEMENTED?

```
1 func buildHttpClient(wc *WrappedClient) *http.Client {  
2     client := &http.Client{  
3         ...  
4         CheckRedirect: wc.config.CheckRedirect,  
5         Transport: &http.Transport{  
6             TLSClientConfig: wc.tlsConfig,  
7             DialContext: (&net.Dialer{  
8                 Resolver: wc.resolver,  
9                 Control: buildRunFunc(wc),  
10            }).DialContext,
```

client.go

HOW IT WAS IMPLEMENTED?

```
1 func buildHttpClient(wc *WrappedClient) *http.Client {  
2     client := &http.Client{  
3         ...  
4         CheckRedirect: wc.config.CheckRedirect,  
5         Transport: &http.Transport{  
6             TLSClientConfig: wc.tlsConfig,  
7             DialContext: (&net.Dialer{  
8                 Resolver: wc.resolver,  
9                 Control: buildRunFunc(wc),  
10            }).DialContext,
```

client.go

WHAT IF YOU CAN'T USE SAFEURL?

or want to have more protections

NETWORK CONTROLS AND AUTHENTICATION

- All outbound network traffic should go through egress proxy
- Internal Kubernetes network should be segmented using Network Policies
- Microservice endpoints should require authentication

TURN OFF REDIRECTS

You can turn off redirects in build-in `http` package:

```
http.DefaultClient.CheckRedirect =
    func(req *http.Request, via []*http.Request) error {
        return http.ErrUseLastResponse
    }
```

Re-try with exploit 3:

```
$ exploit3.sh
<a href="http://backendapi/internal">Moved Permanently</a>.
```

FORM5

IS THAT ALL?

THERE IS MORE...



~10K\$ BOUNTY FROM GOOGLE

\@ trick to bypass whitelist

```
https://cxl-services.appspot.com/proxy  
?url=https://[your_domain]\@jobs.googleapis.com/
```

<https://feed.bugs.xdavidhu.me/bugs/0008>

REAL LIFE SSRF ATTACKS

With this vulnerability attacker can:

- Steal cloud metadata credentials
(`169.254.169.254`)
- Steal sensitive tokens (attached by service to http call)
- Call internal network sensitive resources
- Escalate to RCE via 3rd party apps, e.g. redis, Confluence

REAL LIFE SSRF ATTACKS

Typical places where SSRF can occur:

- Webhooks
- File imports from urls
- PDF generators

SUMMARY

- If possible not consume full URLs from users
(😅 webhooks)
- Apply application level protection -
`doyensec/safeurl` or custom implementation
- Apply **zero trust** architecture protection - network segmentation and authentication

SUMMARY

Of course there is still the simplest solution for my synthetic example

```
router.GET("/debug", func(context *gin.Context) {  
    param1 := context.Query("param1")  
    validateParam1(param1)  
    resp, err := http.Get("http://imageapi/process?param1=" +  
        param1)
```

Hardcoded url with only part controlled by user

MORE TO READ

- Mitigating SSRF in 2023
- A New Era of SSRF
- Web Security Academy - SSRF
- Sending webhooks securely

REPO WITH CODE

<https://github.com/xvnpw/ssrf-in-go>



THANK YOU FOR
LISTENING!
QUESTIONS?

