MITIGATING SSRF VULNERABILITIES IN GO

A PRACTICAL GUIDE

BSides Ljubljana 0x7e7

Marcin Niemiec

FORM3

ABOUT ME

- Cloud Security Engineer at Form3
- Blog: xvnpw.github.io
- Twitter: @xvnpw

FORM3

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

FORM3

SSRF

Server-Side Request Forgery

FORM3

VULNERABLE CODE

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

FORM3

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

FORM3

SETUP


debug
Public API
Backend API
(only internal
access)
Kubernetes

FORM3

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
```

FORM3

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
```

FORM3

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
```

FORMJ

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
```

FORM3

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
```

FORM3

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
```

FORM3

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
```

FORM3

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...

FORM3

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)
```

FORM3

TRY TO FIX

```go
func validateTargetUrl(input string) bool {
  u, err := url.ParseRequestURI(input)
  if err != nil {
    return false
  }
  if (u.Scheme == "http" || u.Scheme == "https") &&
      (u.Hostname() != "backendapi") {
    return true
  }

  return false
}
```

FORM3

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 }
```

FORM3

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 }
```

FORM3

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?

FORM3

MANY WAYS TO WRITE SAME THING

localhost can be:

- 127.0.0.1
- 127.0.0.2
- 2130706433 (more)
- form3.localtest.me

FORM3

LEARNING

Negative validation is doomed to failure 💀

FORM3

TRY TO FIX

Positive validation

```go
func validateTargetUrl(input string) bool {
 u, err := url.ParseRequestURI(input)
 if err != nil {
  return false
 }
 if (u.Scheme == "http" || u.Scheme == "https") &&
  (u.Hostname() == "imageapi") &&
  (u.Port() == "" || u.Port() == "80" || u.Port() == "443")
   return true
 }

 return false
}
```

FORM3

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 }
```

FORM3

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 }
```

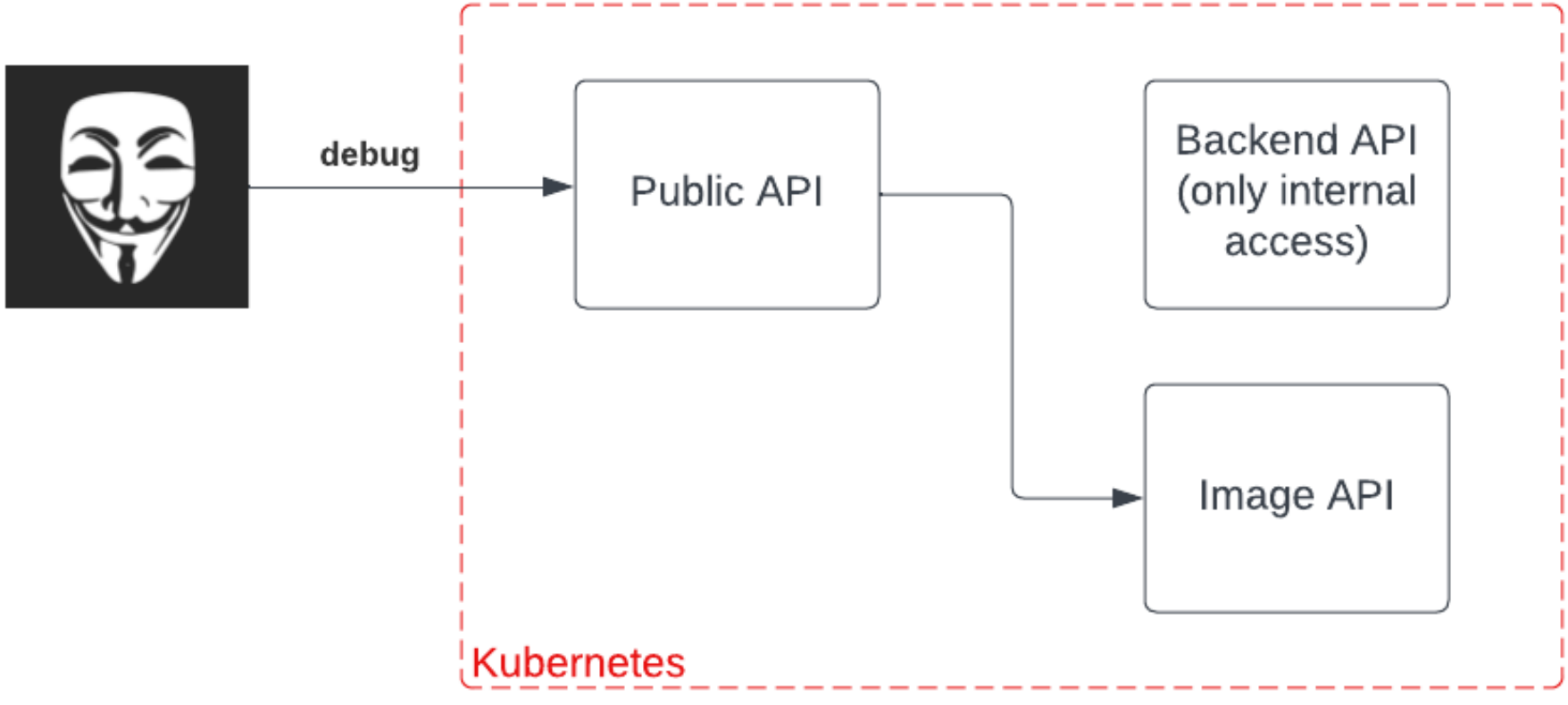
TRY TO FIX

Positive validation

```go
func validateTargetUrl(input string) bool {
 u, err := url.ParseRequestURI(input)
 if err != nil {
  return false
 }
 if (u.Scheme == "http" || u.Scheme == "https") &&
  (u.Hostname() == "imageapi") &&
  (u.Port() == "" || u.Port() == "80" || u.Port() == "443")
   return true
 }

 return false
}
```

SETUP


debug
Public API
Backend API
(only internal
access)
Image API
Kubernetes

FORM3

EXPLOIT 3

SSRF + Open Redirect

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

FORM3

EXPLOIT 3

SSRF + Open Redirect

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

FORM3

EXPLOIT 3

SSRF + Open Redirect

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

FORM3

EXPLOIT 3

SSRF + Open Redirect

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

FORM3

EXPLOIT 3

SSRF + Open Redirect

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

WHAT???

FORM3

STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
200 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"

# backendapi
-> /internal request: user-agent=Go-http-client/1.1
200 | GET "/internal"
```

FORM3

STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
200 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"

# backendapi
-> /internal request: user-agent=Go-http-client/1.1
200 | GET "/internal"
```

FORM3

STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
200 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"

# backendapi
-> /internal request: user-agent=Go-http-client/1.1
200 | GET "/internal"
```

FORM3

STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
200 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"

# backendapi
-> /internal request: user-agent=Go-http-client/1.1
200 | GET "/internal"
```

STEP BY STEP

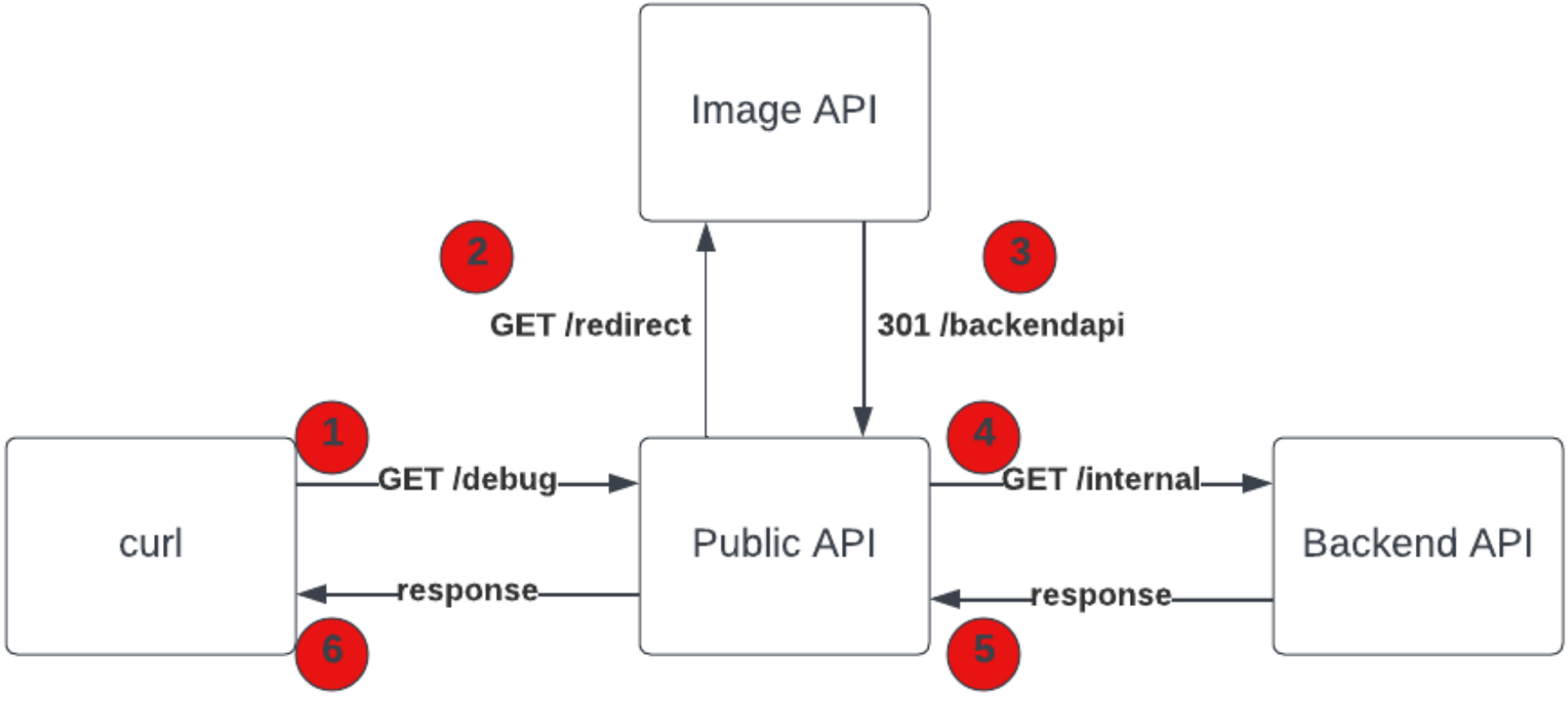


Image API
2
GET /redirect
3
301 /backendapi
1
GET /debug
4
GET /internal
curl
Public API
Backend API
response
response
6
5

FORM3

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...

FORM3

SAFEURL

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

doyensec/safeurl

FORM3

FIX WITH SAFEURL

Positive validation enhanced

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

FORM3

FIX WITH SAFEURL

Positive validation enhanced

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

FORM3

FIX WITH SAFEURL

Positive validation enhanced

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

FORM3

FIX WITH SAFEURL

Positive validation enhanced

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

FORM3

FIX WITH SAFEURL

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

FORM3

FIX WITH SAFEURL

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

FORM3

FIX WITH SAFEURL

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

FORM3

FIX WITH SAFEURL

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

FORM3

FIX WITH SAFEURL

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

FORM3

SAFEURL STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
400 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"
ip: 10.96.45.24 not found in allowlist

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"
```

FORM3

SAFEURL STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
400 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"
ip: 10.96.45.24 not found in allowlist

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"
```

FORM3

SAFEURL STEP BY STEP

```
# publicapi
-> /debug request: user-agent=curl/7.86.0
400 | GET "/debug?url=http://imageapi/redirect?target=
  http://backendapi/internal"
ip: 10.96.45.24 not found in allowlist

# imageapi
-> /redirect request: user-agent=Go-http-client/1.1
301 | GET "/redirect?target=http://backendapi/internal"
```

FORMJ

WHAT ELSE IT CAN DO?

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

Read more

FORM3

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

FORM3

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

FORM3

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

FORM3

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

FORM3

BONUS - EGRESS FILTERING

Smokescreen is a HTTP CONNECT proxy. It proxies most traffic from Stripe to the external world (e.g., webhooks).

```
func BuildProxy(config *Config) *goproxy.ProxyHttpServer {
  proxy := goproxy.NewProxyHttpServer()
  proxy.Verbose = false
  configureTransport(proxy.Tr, config)

  proxy.Tr.DialContext = dialContext
```

smokescreen

FORM3

WHAT IF YOU CAN'T USE SAFEURL?

or want to have more protections

FORM3

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

FORM3

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>.
```

FORM3

IS THAT ALL?

THERE IS MORE...

FORMJ

~10K$ BOUNTY FROM GOOGLE

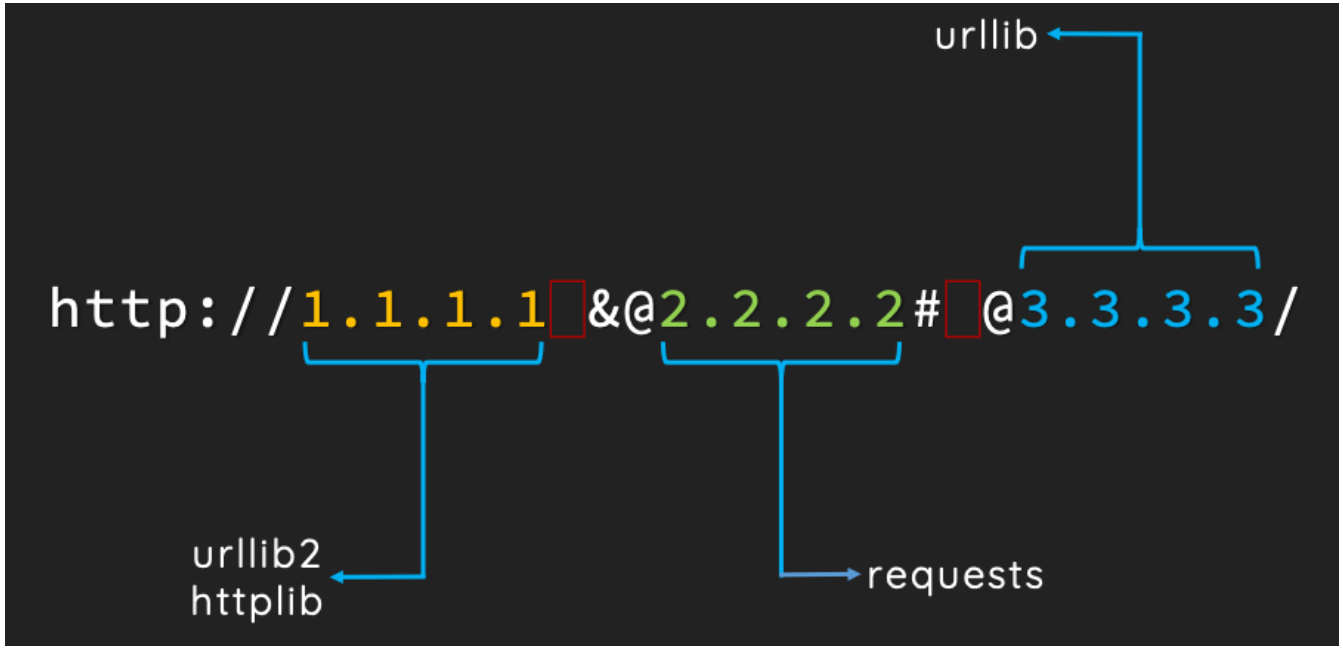
\@ trick to bypass whitelist

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

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

HTTP LIBS/CLIENTS ISSUES

Python example:



Orange Tsai - A New Era of SSRF

FORM3

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

FORM3

REAL LIFE SSRF ATTACKS

Typical places where SSRF can occur:

- Webhooks
- File imports from urls
- PDF generators

FORM3

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

FORM3

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

FORM3

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

FORM3

THANK YOU FOR LISTENING!

QUESTIONS?