挖洞经验 | 看我如何发现Paypal内部信息泄露漏洞

转载

普通网友 于 2018-06-24 18:36:34 发布 1029 收藏
分类专栏： java-hack

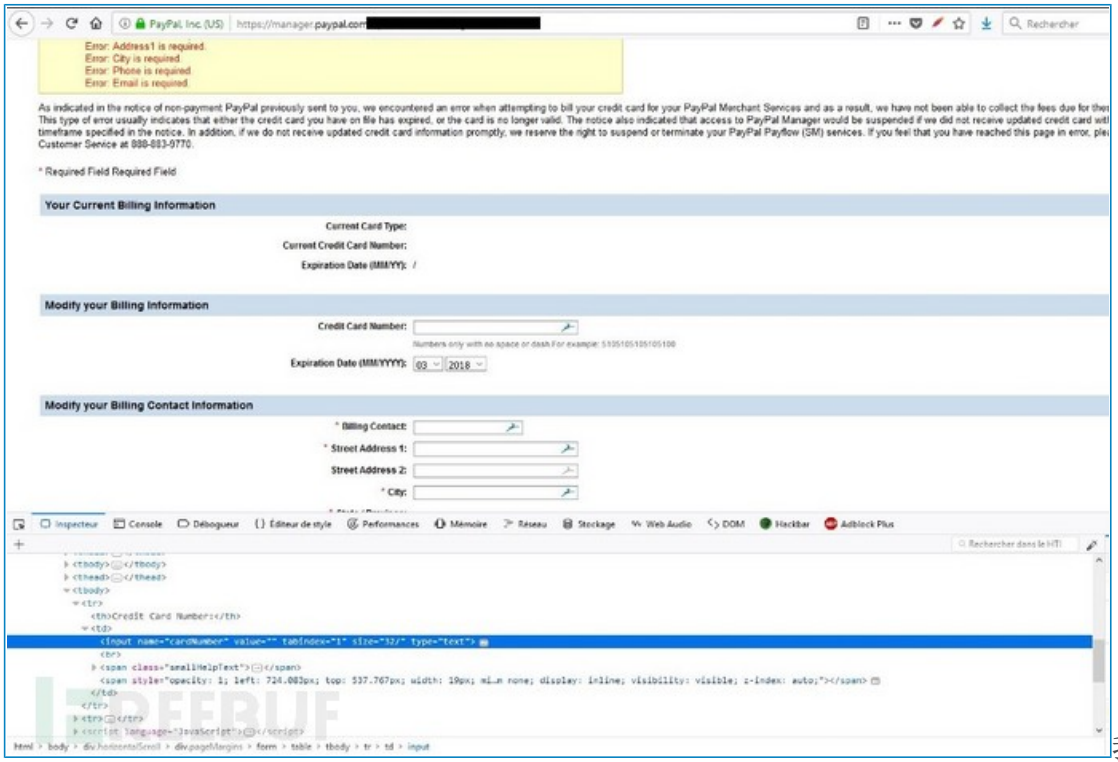

java-hack 专栏收录该内容
68 篇文章 5 订阅
订阅专栏

本文我要分享的是，在Paypal网站manager.paypal.com上的某个页面存在“表达式注入“漏洞（Expression Language Injection），利用该漏洞我可以间接获取到Paypal系统的内部IP、端口和方法类等敏感数据信息。

我从2017年9月开始参与漏洞赏金项目，几乎每天都会花点时间来做一些Web漏洞挖掘测试，我比较喜欢用bountyfactory.io这个漏洞众测平台。在今年2月份，我给自己定了一个目标：发现PayPal的一个漏洞。

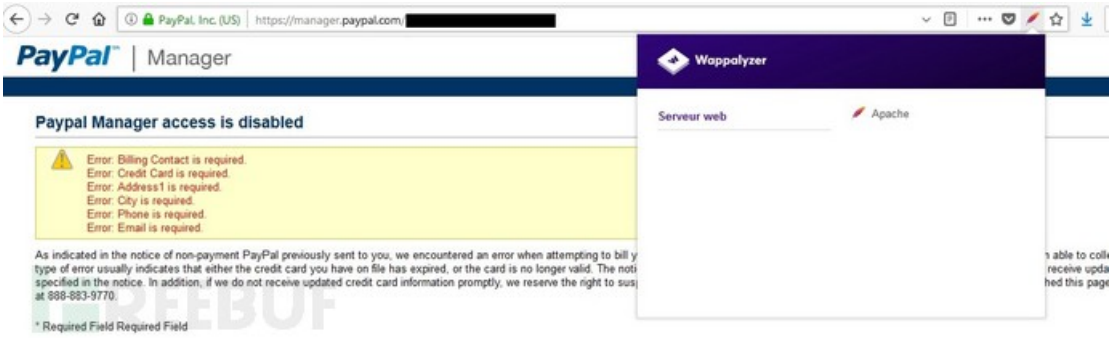
前期发现

PayPal的漏洞众测范围非常广泛，只要是涉及*.paypal.com的网站都算。一开始的套路都是相似的：枚举子域名、枚举可访问文件、枚举目录….。经过几天的折腾研究，我发现了一个有意思的页面：



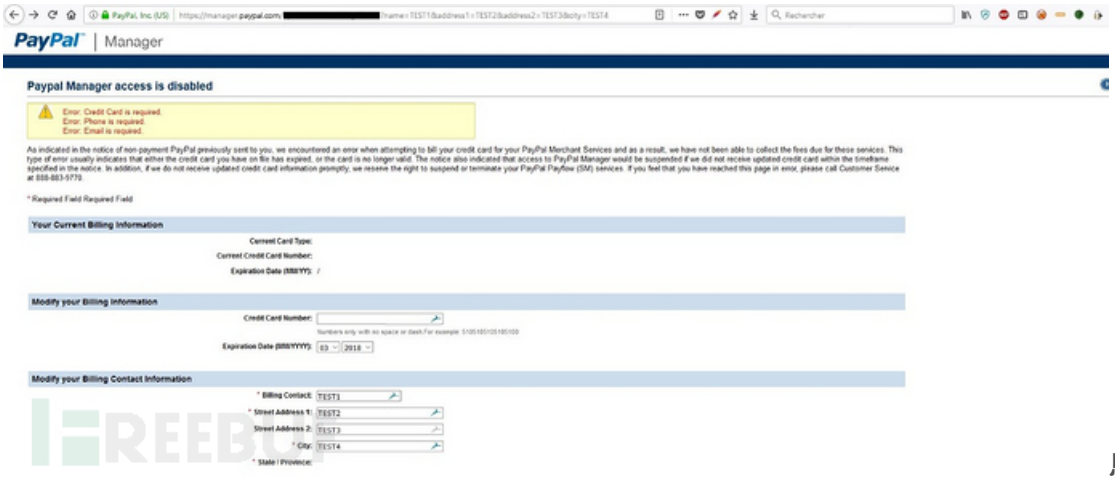
我当时的反应这样的：

 可以看到，这个页面中有好多个可填写区域，但是没提交按钮，和一些新式网页相比，这个页面貌似有些”老旧“，我们尽量通过它来看看能得到些什么东西吧！更奇怪的是，我的网站架构识别插件WappAlyzer竟然好像也失灵了，只识别到了“Apache”，但却没任何版本信息：



测试发现

好吧，那我来尝试看看网站会不会发生什么有意思的响应，所以，我在几个区域中填写了字段，并试图向Paypal发起请求：



思路一：尝试发现反射型XSS

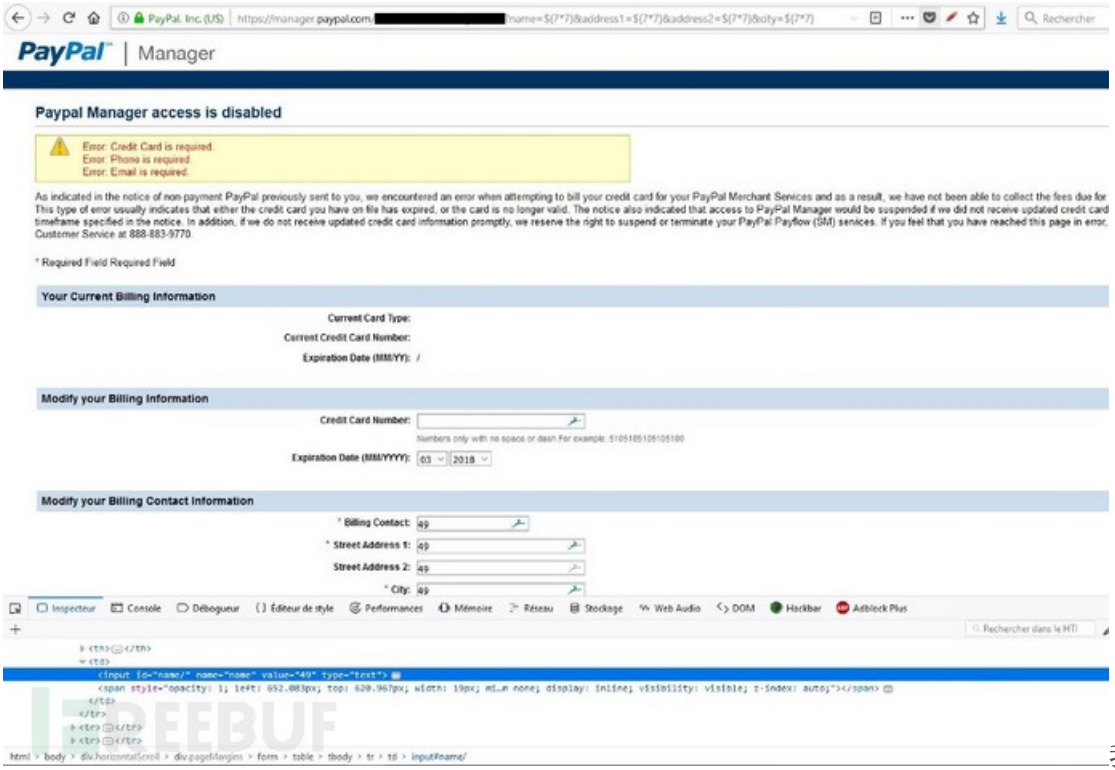
只可惜这招非常不灵，无疑 PayPal 的WAF非常管用，我没能发现任何可以绕过它的方法，当然也有可能是我太菜了。

思路二：尝试注入漏洞

由于反射型XSS的测试无效，我决定测试一些我之前在其它众测项目中学到的注入型Payload，即：

```
{7*7}

{{7*7}}

${7*7}
```

没想到，最后这种`${7*7}` `Payload` 竟然成功了，能有效返回 `7*7` 表达式的结果`49`：



我就乐了


既然能让Paypal服务器来执行一些这种数学操作，这就有意思了，但是还不能说明问题，于是，我决定再来研究研究目标服务器上的Java服务，以及这种"${param}"参数的运行机制。

"JSTL, JSTL everywhere"

在我朋友 Nico 的帮助下，最终我发现，在这个页面中存在一个JSTL标记库，并且可以执行多种有意思的表达式语法命令。

> JSTL：在JSP页面中，使用标签库代替传统的Java片段语言来实现页面的显示逻辑中，由于自定义标签很容易造成重复定义和非标准的实现，因此，JSP 标准标记库（JSP Standard Tag Library，JSTL）就出现了，JSTL 是一个实现 Web 应用程序中常见的通用功能的定制标记库集，这些功能包括迭代和条件判断、数据管理格式化、XML 操作以及数据库访问。

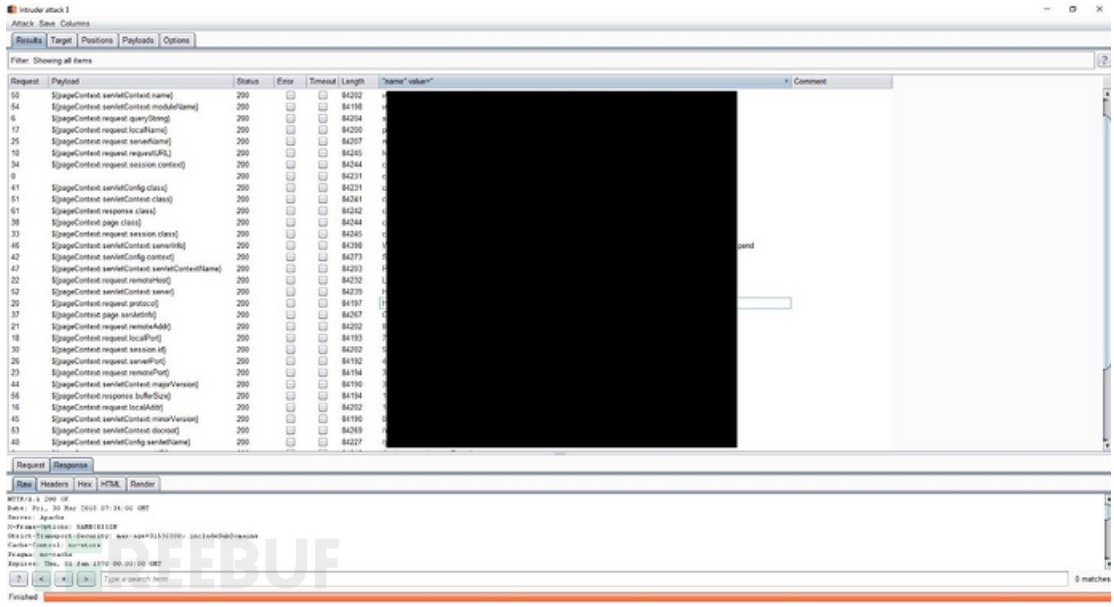
该漏洞的有趣部分是关于JSP页面和Servlet属性的。现在我们已经了解了如何访问请求参数、标记头、初始化参数、cookies和作用域变量，JSTL隐式对象还有一种功能可以测试利用：也就是访问servlet和JSP属性，例如请求的协议或服务器端口，或者容器支持的servlet API的主版本和次版本。通过pageContext隐式对象，你可以发现更多此类数据信息，通过这些发现的数据信息，可以实现对请求、响应、会话和应用程序等servlet context接口信息的访问。ServletContext,是一个全局的储存信息的空间，被Servlet 程序用来与Web 容器通信。

由于pageContext对象是JSP中很重要的一个内置对象，其pageContext隐式对象包含了多种属性：

```
pageContext.request

pageContext.response

pageContext.properties

pageContext.page

pageContext.servletConfig

pageContext.servletContext
```

这些属性我就不一一介绍了，但它们都是用于开发者和Web页面的交互，以及servlet 和 server服务器之间的通信。于是，我就通过一些公开文档和fuzzing方法，尝试来发现一些可用参数，我把以上属性涉及到的相关请求Payload全部添加到BurpSuite的Intruder功能中去：

很好，这样，虽然可能性有限，但是不是就可以用形如SSRF的方式来提取出PayPal目标服务器中的信息了呢？以下是最终的测试效果图：



利用 fuzzing 我发现我之前没发现的一个参数：${pageContext.servletContext.docRoot}，利用它可以显示服务器上编译过的WAR文件相对路径。

可以实现RCE漏洞吗？

有了以上的发现之后，我尝试利用表达式注入来实现RCE漏洞，但最终还是不行。参考了很多JSTL表达式注入相关的writeup和PoC，这里的JSTL EL注入还是无效，我猜想可能由于PayPal服务器上的WAF阻拦了我的注入测试请求，所以最终响应的总是一些301跳转页面。

我也尝试用最基本的Payload方式来测试：${T(java.lang.System).getenv()}，但还是不行，服务器响应总是在我注入时发生停止。当然，也有可能还是我技术不行，没找对方法实现RCE。最终，我只好如此这样把这些发现提交给了PayPal ，希望漏洞有效。

漏洞报送进程

[2018-03-06] 发现漏洞

[2018-03-07] 报送给 PayPal

[2018-03-08] 得到PayPal响应 ： 安全风险不明，无条件获得赏金，这….

[2018-03-29] 再次得到PayPal响应 ： 漏洞有效

[2018-04-03] 漏洞修复

[2018-04-11] 获得不菲赏金和2018年6月名人堂

*参考来源：medium，clouds 编译，转载请注明来自FreeBuf.COM