第一章:Web应用程序安全与风险

Web攻击基础知识

1. 什么是Web应用攻击

Web攻击的本质，就是通过HTTP协议篡改应用程序
可利用点：请求方法、请求头、数据体
利用过程：认证、会话、授权
利用途径：Web、客户端、html、其他协议

2. Web攻击的特点

广泛性
匿名性
利用难度低（对比主机漏洞、木马病毒）

3. 常见的Web安全薄弱点

中间件：针对http底层服务器软件的攻击，如IIS、Apache、nginx、tomcat
Web应用：授权、认证、输入验证、程序逻辑、钓鱼等
数据库：SQL注入
Web客户端：客户端软件漏洞、浏览器、APP、小程序等
传输：中间人攻击、窃听等
可用性：DDOS、CC攻击

Web渗透测试学习框架（涵盖哪些方面？遵循什么流程？）

1.信息收集

```
挖淘宝的漏洞：https://www.taobao.com
登录网站，挖漏洞：
（1）端口
（2）子站  abc.taobao.com; a.b.taobao.com
（3）新业务
（4）手机业务
（5）微信小程序
（6）友情链接
（7）旁站
```

2.漏洞攻防

```
SQL注入
XSS
CSRF
SSRF
命令注入
目录穿越
暴力破解
文件读取   ../../../etc/passwd
文件上传   .php
文件包含   include(filename)
XXE        代码解析xml文件时解析外部实体引发漏洞
逻辑漏洞 / 业务漏洞
中间件     IIS、Tomcat、Nginx、Weblogic...
```

3.编程语言

```
PHP
Python
Java
JavaScript
```

4.代码审计

5.应急响应

第二章:Web应用程序技术

1. HTTP协议

HTTP （HyperText Transfer Protocol）即超文本传输协议，是一种详细规定了浏览器和万维网服务器之间互相通信的规则，它是万维网交换信息的基础，它允许将HTML （超文本标记语言）文档从Web服务器传送到Web浏览器。

如何发起一个HTTP请求？这个问题似乎很简单，当在浏览器地址栏中输入一个URL，按下回车键后就发起了这个HTTP请求，很快就会看到这个请求的返回结果。

URL （统一的资源定位符）也被称为网页地址，是互联网标准的地址。URL的标准格式如下：

```
协议://服务器ip [:端口]/路径/[?查询]
```

例如，http://www.hacker.com/post/test.html 就是一个标准的URL。

借助浏览器可以快速发起一次HTTP请求，如果不借助浏览器应该怎样发起HTTP请求呢？

其实可以借助很多工具来发起HTTP请求，例如，在Linux系统中的curl命令。严格地说，浏览器也属于HTTP工具的一种。

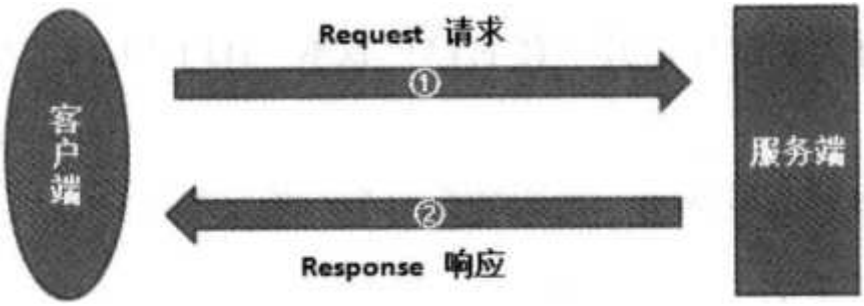
在Windows中，也可以用 curl.exe 工具来发起请求，通过curl + URL命令就可以简单地发起一个HTTP请求，非常方便。

```
Get 请求
curl https://www.baidu.com
Post 请求
curl -d "user=admin&password=admin" https://www.baidu.com
```

2. HTTP协议详解

HTTP是一种无状态的协议。无状态是指Web浏览器与Web服务器之间不需要建立持久的连接，这意味着当一个客户端向服务器端发出请求（Request），然后Web服务器返回响应（Response），连接就被关闭了，在服务器端不保留连接的有关信息。

也就是说，HTTP请求只能由客户端发起，而服务端不能主动向客户端发送数据。



HTT协议P遵循请求（Request）/响应（Response）模型，Web浏览器向Web服务器发送请求时，Web服务器处理请求并返回适当的响应。下面通过实例来观察HTTP的请求与响应。

（1）HTTP请求
HTTP请求包括三部分，分别是请求行（请求方法）、请求头（消息报头）和请求正文。
下面是HTTP请求的一个例子。

```
POST /login.php HTTP/1.1    //请求行
HOST: www.baidu.com //请求头
User-Agent: Mozilla/5.0 （Windows NT 6.1; rv:15.0） Gecko/20100101 Firefox/15.0
//空白行，代表请求头结束
Username=admin&password=admin   //请求正文
```

格式解析：
第一行：请求行，由三部分组成。第一部分说明了该请求方法是POST请求；第二部分是一个斜杠（/login.php），用来说明请求的是该域名根目录下的 login.php文件；第三部分即为http协议的版本。
第二行至空白行：这部分称为HTTP中的请求头（也被称为消息头）。
其中，HOST代表请求的主机地址，User-Agent代表浏览器的标识。请求头**由客户端自行设定**。关于消息头的内容，在后面章节中将会详细介绍。
最后一行：请求正文，请求正文是可选的，它最常出现在POST请求方法中，而GET请求的内容在请求行的路径中。

（2）HTTP响应
与HTTP请求对应的是HTTP响应，HTTP响应也由三部分内容组成，分别是响应行、响应头（消息报头）和响应正文（消息主体）。
下面是一个经典的HTTP响应。

```
HTTP/1.1 200 OK                          //响应行
Date: Thu, 28 Feb 2013 07:36:47 GMT //响应头
Server: BWS/1.0 //服务器信息
Content-Length: 4199
Content-Type: text/html;charset=utf-8 //提问：告诉谁的？
Cache-Control: private
Expires: Thu, 28 Feb 2021 07:36:47 GMT
Content-Encoding: gzip
Set-Cookie: H_PS_PSSID=2022_1438_1944_1788; path=/; domain=.baidu.com
Connection: Keep-Alive
//空白行，代表响应头结束
<html>  //响应正文或者叫消息主体
<headxtitle> Index.html </title></head>
```

格式解析：
第一行：响应行，其中有状态码（200）以及消息 “OK”。
第二行至空白行：响应头，由服务器向客户端发送。
消息报头之后是响应正文，是服务器向客户端发送的HTML数据，也有可能是json数据。

3. HTTP请求方法

HTTP请求的方法非常多，其中GET、POST最常见。下面是HTTP请求方法的详细介绍。

（1） GET

GET方法用于获取请求页面的指定信息（以实体的格式）。

如果请求资源为动态脚本（非HTML），那么返回文本是Web容器解析后的HTML源代码，而不是源文件。例如请求index.jsp，返回的不是index.jsp的源文件，而是经过解析后的HTML代码。

如下HTTP请求：

```
GET /index.php?id=l HTTP/1.1
HOST: www.baidu.com
```

使用GET请求index.php，并且id参数为1，在服务器端脚本语言中可以选择性地接收这些参数，比如

```
id=1&name=admin
```

一般都是由开发者内定好的参数才会接收，比如开发者只接收id参数，若加了其他参数项，如：

```
Index.php?id=1&username=admin //多个参数项以分隔
```

服务器端脚本不会理会额外加入的内容，依然只会接收id参数，并且去查询数据，最终向服务器端发送解析过的HTML数据，不会因为额外的干扰而乱套。

（2）HEAD

HEAD方法除了服务器不能在响应里返回消息主体外，其他都与GET方法相同。此方法经常被用来测试超文本链接的有效性、可访问性和最近的改变。

攻击者编写扫描工具时，就常用此方法，因为只测试资源是否存在，而不用返回消息主体，所以速度一定是最快的。

一个经典的HTTP HEAD请求如下：

```
HEAD /index.php HTTP/1.1
HOST: www.xxser.com
```

（3）POST

POST方法也与GET方法相似，但最大的区别在于，GET方法没有请求内容，而POST方法是有请求内容的。

POST请求多用于向服务器发送大量的数据。GET请求虽然也能发送数据，但是有大小(长度)的限制，并且GET请求会将发送的数据显示在浏览器端，而POST请求则不会，所以安全性相对来说高一点。

例如，上传文件、提交留言等，只要是向服务器传输大量的数据，通常都会使用POST请求。一个经典的HTTP POST请求如下：

```
POST /login.php HTTP/1.1
Host: www.xxser.com
Content-Length: 26
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Origin: http://home.2cto.com
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17 (KHTML, like Gecko)
Chrome/24.0.1312.57 Safari/537.17 SE 2.X MetaSr 1.0
Content-Type: application/x-www-form-urlencoded
Accept-Language: zh-CN,zh;q=0.8
Accept-Charset: GBK,utf-8;q=0.7,*;q=0.3

user=admin&pw=123456789
```

用POST方法向服务器请求login.php，并且传递参数

```
user=admins&pw=123456789
```

（4）PUT

PUT方法用于请求服务器把请求中的实体存储在请求资源下，如果请求资源已经在服务器中存在，那么将会用此请求中的数据替换原先的数据，作为指定资源的最新修改版。

如果请求指定的资源不存在，将会创建这个资源，且数据部分位于请求正文中，具体如下：

```
PUT /input.txt
HOST: www.xxser.com
Content-Length: 6

123456
```

这段HTTP PUT请求将会在主机根目录下创建input.txt，内容为123456。

通常情况下，服务器都会关闭PUT方法，因为它会为服务器建立文件，属于危险的方法之一。

（5）DELETE

DELETE方法用于请求源服务器删除请求的指定资源。

服务器一般都会关闭此方法，因为客户端可以进行删除文件操作，属于危险方法之一。

（6）TRACE

TRACE方法被用于激发一个远程的应用层的请求消息回路，也就是说，回显服务器收到的请求。

TRACE方法允许客户端去了解数据被请求链的另一端接收的情况，并且利用那些数据信息去测试或诊断。但此方法非常少见。

（7）OPTIONS

OPTIONS方法是用于请求获得由URI标识的资源在请求/响应的通信过程中可以使用的功能选项。

通过这个方法，客户端可以在采取具体资源请求之前，决定对该资源采取何种必要措施，或者了解服务器的性能。

HTTP OPTIONS请求/响应如下：

```
OPTIONS / HTTP/1.1
HOST: www.xxser.com

HTTP/1.1 200 OK
Allow: OPTIONS, TRACE, GET, HEAD, POST
Server: Microsoft-IIS/7.5
Public: OPTIONS, TRACE, GET, HEAD, POST
X-Powered-By: ASP.NET
Date: Sun, 14 Jul 2013 15:50:58 GMT
Content-Length: 0
```

以上为 HTTP 标准方法。

4. HTTP状态码

当客户端发出HTTP请求，服务器端接收后，会向客户端发送响应信息，其中，HTTP响应中的第一行中，最重要的一点就是HTTP的状态码，内容如下：

```
HTTP/1.1 200 OK
```

此状态码为200。

在HTTP协议中表示请求结果的状态码由三位数字组成，第一位数字定义了响应的类别，且只有以下5种：

```
•1xx：信息提示，表示请求已被成功接收，继续处理。其范围为100∽101。
•2xx：成功，服务器成功地处理了请求。其范围为200∽206。
•3xx：重定向，重定向状态码用于告诉浏览器客户端，它们访问的资源已被移动，并告诉客户端新的资源地址位置。这时，浏览器将重新对新资源发起请求。其范围为300∽305。
•4xx：客户端错误状态码，有时客户端会发送一些服务器无法处理的东西，比如格式错误的请求，或者最常见的是，请求一个不存在的URL。其范围为400∽415。
•5xx：有时候客户端发送了一条有效请求，但web服务器自身却出错了，可能是web服务器运行出错了，或者网站都挂了。5XX就是用来描述服务器内部错误的，其范围为500∽505。
```

http状态码有哪些：

```
100 （继续） 请求者应当继续提出请求。服务器返回此代码表示已收到请求的第一部分，正在等待其余部分。
101 （切换协议） 请求者已要求服务器切换协议，服务器已确认并准备切换。

200 （成功）客户端请求成功，是最常见的状态。
```

201 （已创建） 请求成功并且服务器创建了新的资源。
202 （已接受） 服务器已接受请求，但尚未处理。
203 （非授权信息） 服务器已成功处理了请求，但返回的信息可能来自另一来源。
204 （无内容） 服务器成功处理了请求，但没有返回任何内容。
205 （重置内容） 服务器成功处理了请求，但没有返回任何内容。
206 （部分内容） 服务器成功处理了部分 GET 请求。

300 （多种选择） 针对请求，服务器可执行多种操作。服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。
301 （永久移动） 请求的网页已永久移动到新位置。服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。
302 （临时移动） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。
303 （查看其他位置） 请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。
304 （未修改） 自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容。
305 （使用代理） 请求者只能使用代理访问请求的网页。如果服务器返回此响应，还表示请求者应使用代理。
307 （临时重定向） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

400 （错误请求） 服务器不理解请求的语法。
401 （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。
403 （禁止） 服务器拒绝请求。
404 （未找到） 服务器找不到请求的网页。
405 （方法禁用） 禁用请求中指定的方法。
406 （不接受） 无法使用请求的内容特性响应请求的网页。
407 （需要代理授权） 此状态代码与 401（未授权）类似，但指定请求者应当授权使用代理。
408 （请求超时） 服务器等候请求时发生超时。
409 （冲突） 服务器在完成请求时发生冲突。服务器必须在响应中包含有关冲突的信息。
410 （已删除） 如果请求的资源已永久删除，服务器就会返回此响应。
411 （需要有效长度） 服务器不接受不含有效内容长度标头字段的请求。
412 （未满足前提条件） 服务器未满足请求者在请求中设置的其中一个前提条件。
413 （请求实体过大） 服务器无法处理请求，因为请求实体过大，超出服务器的处理能力。
414 （请求的 URI 过长） 请求的 URI（通常为网址）过长，服务器无法处理。
415 （不支持的媒体类型） 请求的格式不受请求页面的支持。

500 （服务器内部错误） 服务器遇到错误，无法完成请求。
501 （尚未实施） 服务器不具备完成请求的功能。例如，服务器无法识别请求方法时可能会返回此代码。
502 （错误网关） 服务器作为网关或代理，从上游服务器收到无效响应。
503 （服务不可用） 服务器目前无法使用（由于超载或停机维护）。通常，这只是暂时状态。
504 （网关超时） 服务器作为网关或代理，但是没有及时从上游服务器收到请求。
505 （HTTP 版本不受支持） 服务器不支持请求中所用的 HTTP 协议版本。

常见的状态码描述如下：

100：客户端继续发送请求，这是临时响应。
200：客户端请求成功，是最常见的状态。
302：重定向。
400：客户端请求有语法错误，不能被服务器所理解。
401： 请求未经授权。
403：服务器收到请求，但是拒绝提供服务。
404：请求资源不存在，是最常见的状态。
500：服务器内部错误，是最常见的状态。
503：服务器当前不能处理客户端的请求，一段时间后可能恢复正常。

5. HTTP消息

HTTP消息又称为HTTP头（HTTP header），有四种：分别是请求头、响应头、普通头和实体头。从名称上看，我们就可以知道它们所处的位置。

（1）请求头

请求头只出现在HTTP请求中，请求报头允许客户端向服务器端传递请求的附加信息和客户端自身的信息。常用的HTTP请求头如下。

①Host
Host请求报头域主要用于指定被请求资源的Internet主机和端口号，例如：

```
HOST: www.baidu.com:8010
```

②User-Agent
User-Agent请求报头域允许客户端将它的操作系统、浏览器和其他属性告诉服务器。登录一些网站时，很多时候都可以见到显示我们的浏览器、系统信息，这些都是此头的作用，如：

```
User-Agent:  My privacy
```

③Referer
Referer包含一个URL，代表当前访问URL的上一个URL，也就是说，用户是从什么地方来到本页面。
如：

```
Referer: www.baidu.com/login.php
```

代表用户从login.php来到当前页面。

④Cookie
Cookie是非常重要的请求头，它是一段文本，常用来表示请求者身份等。

⑤Range
Range可以请求实体的部分内容，多线程下载一定会用到此请求头。
例如：

```
表示头500字节：bytes=0~499
表示第二个500字节：bytes=500~999
表示最后500字节：bytes=-500
表示500字节以后的范围：bytes=500-
```

⑥X-forward-for
X-forward-for即XXF头，它代表请求端的IP，可以有多个，中间以逗号隔开。

⑦ Accept
Accept请求报头域用于指定客户端接收哪些MIME类型的信息，如

```
Accept:  text/html
```

表明客户端希望接收HTML文本。

⑧ Accept-Charset
Accept-Charset请求报头域用于指定客户端接收的字符集。例如：

```
Accept-Charset:utf-8
```

如果在请求消息中没有设置这个域，默认是任何字符集都可以接收。

（2）响应头

响应头是服务器根据请求向客户端返回的HTTP头。

常见的HTTP响应头如下。

①Server
服务器所使用的Web服务器名称，如

```
Server:Apache/1.3.6(Unix)
```

攻击者通过查看此头，可以探测Web服务器名称。所以，建议在服务器端进行修改此头的信息。

②Set-Cookie
向客户端设置Cookie，通过查看此头，可以清楚地看到服务端向客户端发送的 Cookie 信息。

③Last-Modified
服务器通过这个头告诉浏览器，资源的最后修改时间。

④Location
服务器通过这个头告诉浏览器去访问哪个页面，浏览器接收到这个请求之后，通常会立刻访问Location头所指向的页面。
这个头通常配合302状态码使用。

⑤Refresh
服务器通过Refresh头告诉浏览器定时刷新浏览器。

（3）普通头

在普通报头中，有少数报头域用于所有的请求和响应消息，但并不用于被传输的实体，只用于传输的消息。

例如：
Date，表示消息产生的日期和时间。
Connection，允许发送指定连接的选项。指定连接是连续的，或者指定“close”选项，通知服务器在响应完成后关闭连接。
Cache-Control，用于指定缓存指令，缓存指令是单向的，且是独立的。

注意：普通报头作为了解即可。

（4）实体头

请求和响应消息都可以传送一个实体头。实体头定义了关于实体正文和请求所标识的资源的元信息。元信息也就是实体内容的属性，包括实体信息类型、长度、压缩方法、最后一次修改时间等。

常见的实体头如下。

①Content-Type（image、text）
Content-Type实体头用于向接收方指示实体的介质类型。

②Content-Encoding
Content-Encoding头被用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容的编码，因而要获得Content-Type报头域中所引用的媒体类型，必须采用相应的解码机制。

③Content-Length
Content-Length实体报头用于指明实体正文的长度，以字节方式存储的十进制数字来表示。

④Last-Modified
Last-Modified实体报头用于指示资源的最后修改日期和时间。

6. HTTP协议与HTTPS协议的区别

HTTPS 协议的全称为 Hypertext Transfer Protocol over Secure Socket Layer，它是以安全为目标的HTTP通道，其实就是HTTP的“升级”版本，比单纯的HTTP协议更加安全。

HTTPS的安全基础是SSL，即在HTTP下加入SSL层。也就是HTTPS通过安全传输机制进行传送数据，这种机制可保护网络传送的所有数据的隐秘性与完整性，可以降低非侵入性拦截攻击的可能性。

既然是在HTTP的基础上进行构建的HTTPS协议，所以HTTPS请求与响应仍旧是以相同的方式进行工作的。

HTTP协议与HTTPS协议的主要区别如下：
1）HTTP是超文本传输协议，信息是明文传输，HTTPS则是具有安全性的SSL加密传输协议。
2）HTTP与HTTPS协议使用的是完全不同的连接方式，HTTP采用80端口连接，而HTTPS则是443端口。
3）HTTPS协议需要到ca申请证书，一般免费证书很少，需要交费，也有些Web容器提供， 如TOMCAT。而HTTP协议却不需要。
4）HTTP连接相对简单，是无状态的，而HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，相对来说，它要比HTTP协议更安全。

7. Web应用程序编码

（1）URL编码

URL只允许ASCII字符集中的可打印字符（0x20-0x7e），由于其在URL方案或HTTP协议中的特殊含义，这个范围内的一些字符也不可以用于URL中。

URL编码方案主要用于将扩展ASCII字符集中有问题的字符进行编码，使其通过HTTP可以安全传输。任何URL编码的字符都以%为前缀，其后是这个字符的两位十六进制代码。

常见的一些URL编码字符：

```
%3d -> =
%25 -> %
%20 / + -> 空格
%0a -> 新行
%00 -> 空字节
```

（2）Unicode编码

Unicode是一种为支持全世界所使用的各种编写系统而设计的字符编码标准，它采用各种编码方案，其中一些可用于表示web应用程序中的不常见字符。

16位的Unicode编码以%为前缀，其后用16进制表示每个字节。

```
%u2215 -> /
```

（3）HTML编码

HTML编码用于处理问题字符并将其安全并入HTML文档。HTML编码定义了大量html实体来表示特殊的字面量字符。

```
&quot;  -> "

&apos; -> '

&amp; -> &

&lt; -> <

&gt; -> >
```

[html实体编码表]

此外，任何字符都可以使用它的十进制ASCII码进行HTML编码：

```
&#34; -> "
&#39; -> '
```

或者使用十六进制的ASCII码，以x为前缀：

```
&#x22; -> "
&#x27; -> '
```

（4）Base64编码

Base64编码用一个可打印的ASCII字符就可以安全转换任何二进制数据。常用于对电子邮件进行编码，或对基本HTTP验证机制中的用户证书进行编码。

可打印ASCII字符有64个，可以表示6bit数据，因此Base64编码将输入分为每6位一组，每6位用一个字符表示，如果不足6位，用一个或两个等号填充。

（5）16进制编码

许多应用程序在传送二级制数据时直接使用十六进制编码，用ASCII字符表示16进制数据块。

```
daf -> 646166
```