



第五讲 应用安全

白杨 alicepub@163.com



应用安全

网络与系统层之上承载着直接提供服务功能的各种互联网应用。目前，针对应用系统的攻击方法多种多样。





第五讲 应用安全



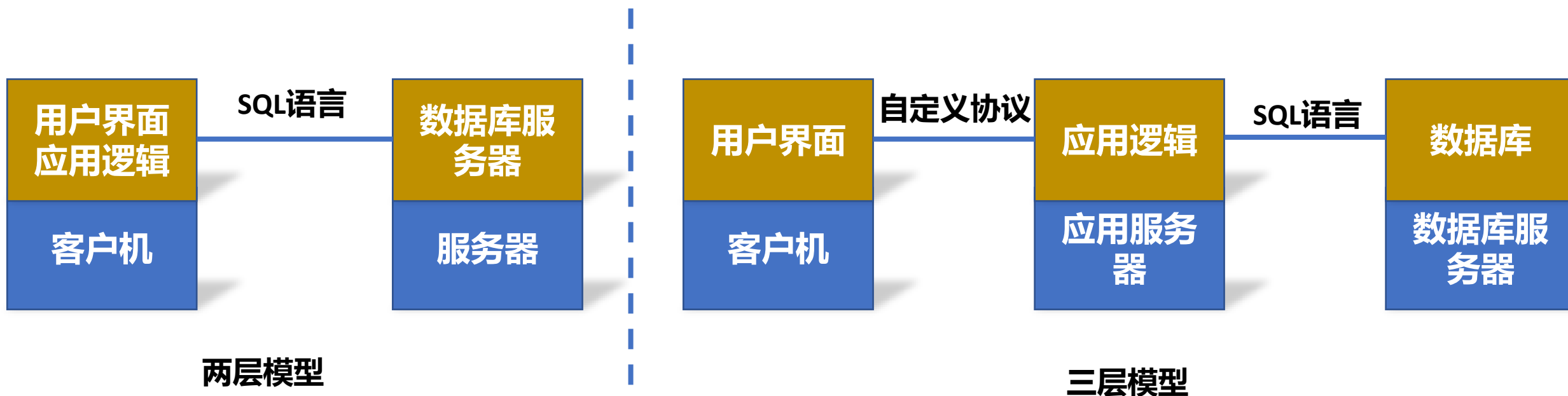
网络空间安全学院
School of Cybersecurity

- ① web应用安全概述
- ② 常见Web应用安全漏洞
- ③ 恶意代码
- ④ 中间件安全
- ⑤ 数据库安全



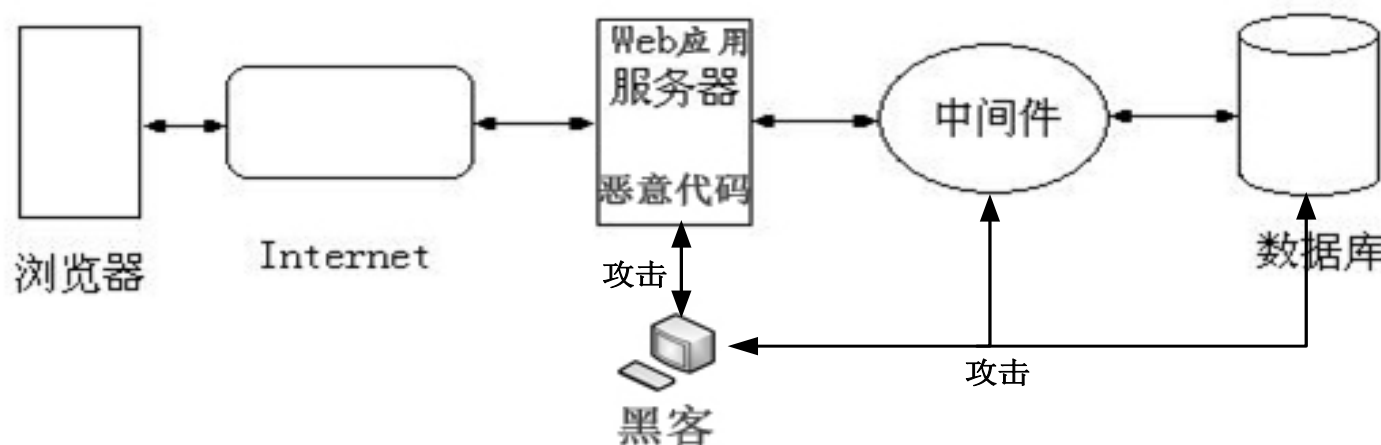
成都信息工程大学
Chengdu University of Information Technology

两层模型和三层模型



应用安全概述

三层结构中，用户使用标准的浏览器（如IE）通过Internet和http协议访问服务方提供的Web应用服务器，Web应用服务器分析用户浏览器提出的请求，如果是页面请求，则直接用http协议向用户返回要浏览的页面。如果有数据库查询操作的请求，则将这个需求传递给服务器和数据库之间的中间件，由中间件再向数据库系统提出操作请求，得到结果后再返回给Web应用服务器，Web应用服务器把数据库操作的结果形成html页面，再返回给浏览器。



应用安全概述

采用三层结构应用系统企业越来越多，应用越来越广泛，随之而来的针对应用安全的威胁日益凸显，攻击者会利用Web应用系统、中间件或数据库漏洞进行攻击：

- 得到Web应用服务器或数据库服务器的控制权限
- 篡改网页内容
- 窃取重要内部数据
- 网页中植入恶意代码，控制应用系统并给访问者带来侵害
-



第五讲 应用安全



网络空间安全学院
School of Cybersecurity

- ① web应用安全概述
- ② 常见Web应用安全漏洞
- ③ 恶意代码
- ④ 中间件安全
- ⑤ 数据库安全



成都信息工程大学
Chengdu University of Information Technology

■ 常见Web应用安全漏洞-文件上传漏洞

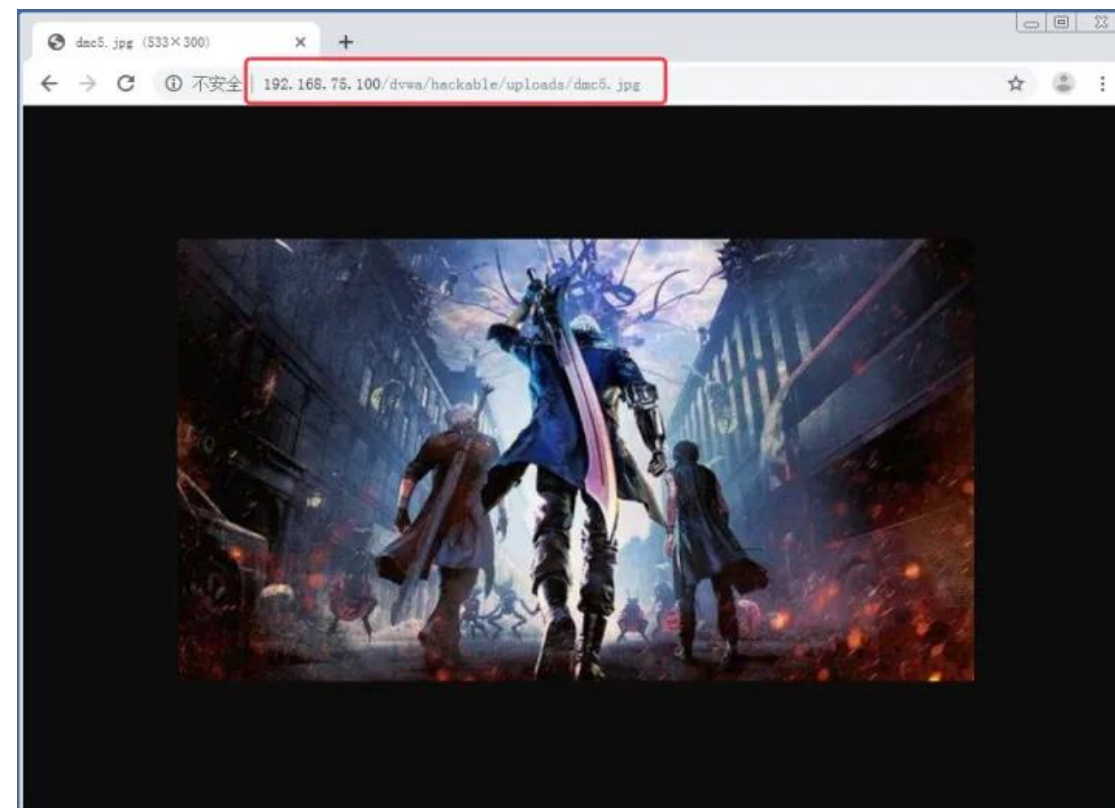
(1) 文件上传漏洞及危害

一些文件上传功能实现代码没有严格限制用户上传的文件后缀以及文件类型，导致允许攻击者向某个可通过Web访问的目录上传任意PHP文件，并能够将这些文件传递给PHP解释器，从而可在远程服务器上执行任意PHP脚本。利用这类漏洞，攻击者可以将病毒、木马、Webshell、其他恶意脚本上传到服务器，为后续攻击提供便利。

- 上传文件是病毒或者木马时，主要用于诱骗用户或者管理员下载执行或者直接自动运行；
- 上传文件是WebShell时，攻击者可通过这些网页后门执行命令并控制服务器；
- 上传文件是其他恶意脚本时，攻击者可直接执行脚本进行攻击；
- 上传文件是恶意图片时，图片中可能包含了脚本，加载或者点击这些图片时脚本会悄无声息的执行；
- 上传文件是伪装成正常后缀的恶意脚本时，攻击者可借助本地文件包含漏洞(Local File Include)执行该文件。如将bad.php文件改名为bad.doc上传到服务器，再通过PHP的include，include_once，require，require_once等函数包含执行。

常见Web应用安全漏洞-文件上传漏洞

(2) 文件上传攻击实例

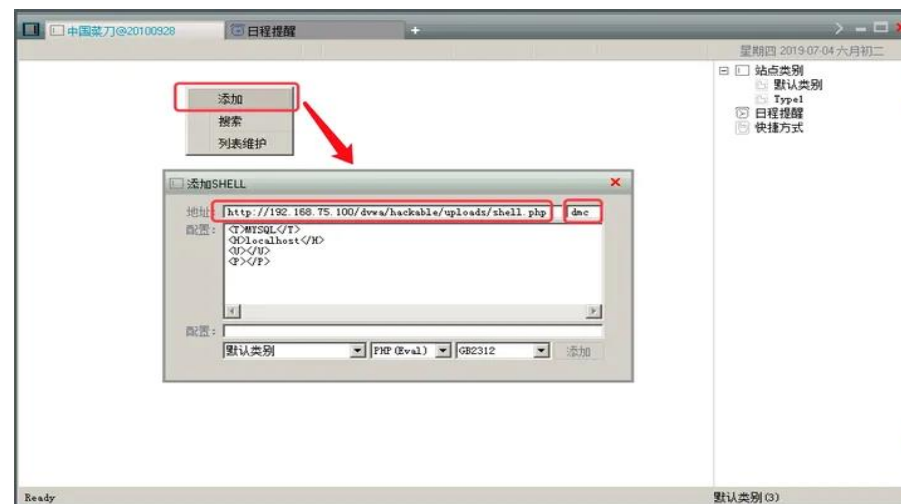


常见Web应用安全漏洞-文件上传漏洞

(2) 文件上传攻击实例

创建一个 PHP 一句话木马，文件后缀为 php，代码如下：

`<?php @eval($_POST[dmc]);?>` //dmc 为变量，作用类似于连接的密码，可以自定义，把一句话木马进行上传，发现可以直接上传成功。

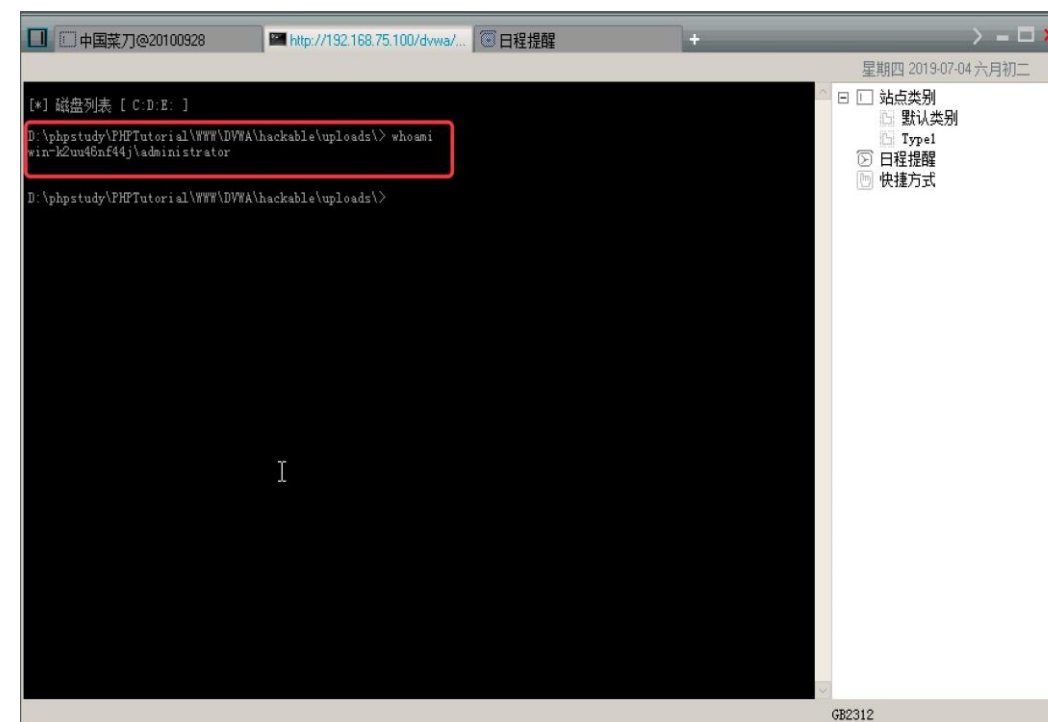
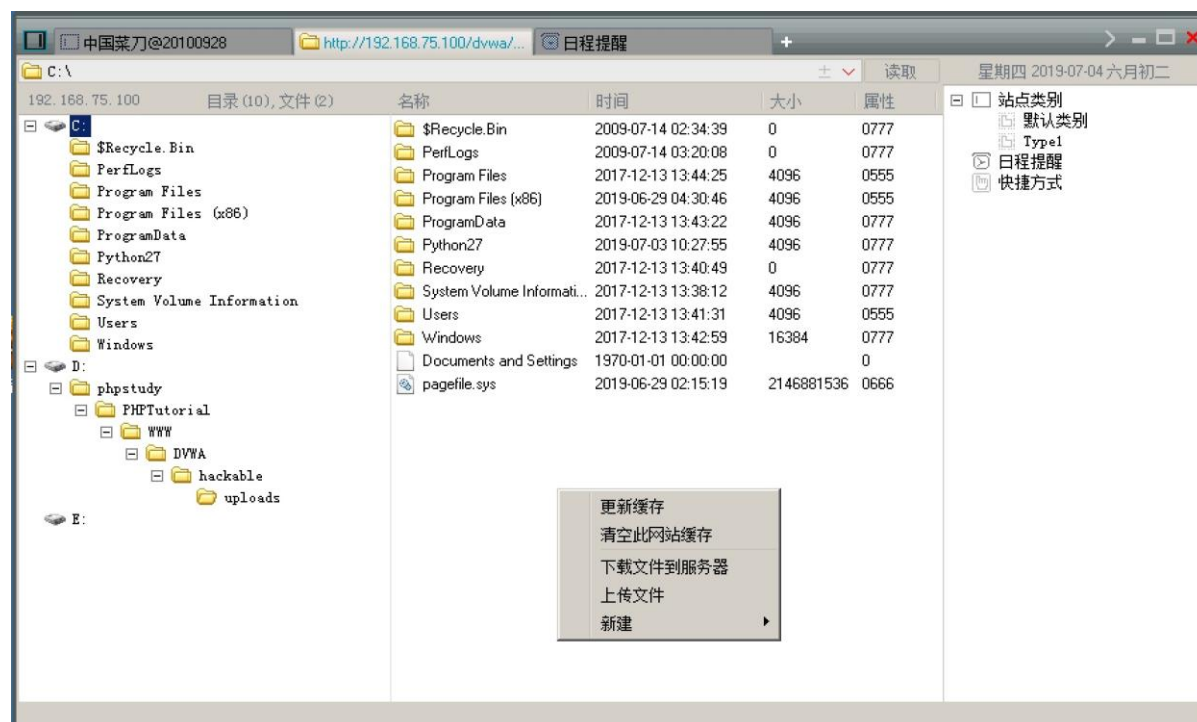


使用“中国菜刀”连接一句话木马。在菜刀中点击右键，添加，打开添加 SHELL 的对话框。在地址栏输入刚才上传的 PHP 一句话木马地址 `http://192.168.75.100/dvwa/hackable/uploads/shell.php`，后面的文本框中填写刚才设置的变量 `dmc`，点击添加。

常见Web应用安全漏洞-文件上传漏洞

(2) 文件上传攻击实例

双击添加的 Shell 连接，可以管理目标服务器的整个硬盘数据，包括文件上传和下载；右键点击 Shell 连接，点击 虚拟终端，打开命令执行环境，可以在目标服务器上执行命令



■ 常见Web应用安全漏洞-文件上传漏洞

(3) 造成恶意文件上传的原因

1) 文件上传时检查不严

一些应用在文件上传时根本没有进行文件格式检查，导致攻击者可以直接上传恶意文件。一些应用仅仅在客户端进行了检查，而在专业的攻击者眼里几乎所有的客户端检查都等于没有检查，攻击者可以通过NC，Fiddler等断点上传工具轻松绕过客户端的检查。一些应用虽然在服务器端进行了黑名单检查，但是却可能忽略了大小写。

2) 文件上传后修改文件名时处理不当

一些应用在服务器端进行了完整的黑名单和白名单过滤，在修改已上传文件文件名时却百密一疏，允许用户修改文件后缀。

3) 使用第三方插件时引入

好多应用都引用了带有文件上传功能的第三方插件，这些插件的文件上传功能实现上可能有漏洞，攻击者可通过这些漏洞进行文件上传攻击。

常见Web应用安全漏洞-文件上传漏洞

(4) 文件上传漏洞防御

系统开发阶段的防御。系统开发人员应有较强的安全意识，尤其是采用PHP语言开发系统。在系统开发阶段应充分考虑系统的安全性。对文件上传漏洞来说，最好能在客户端和服务端对用户上传的文件名和文件路径等项目分别进行严格的检查。客户端的检查虽然对技术较好的攻击者来说可以借助工具绕过，但是这也可以阻挡一些基本的试探。服务器端的检查最好使用白名单过滤的方法，这样能防止大小写等方式的绕过，同时还需对%00截断符进行检测，对HTTP包头的content-type也和上传文件的大小也需要进行检查。

系统运行阶段的防御。系统上线后运维人员应有较强的安全意思，积极使用多个安全检测工具对系统进行安全扫描，及时发现潜在漏洞并修复。定时查看系统日志，web服务器日志以发现入侵痕迹。定时关注系统所使用到的第三方插件的更新情况，如有新版本发布建议及时更新，如果第三方插件被爆有安全漏洞更应立即进行修补。对于整个网站都是使用的开源代码或者使用网上的框架搭建的网站来说，尤其要注意漏洞的自查和软件版本及补丁的更新，上传功能非必选可以直接删除。除对系统自生的维护外，服务器应进行合理配置，非必选一般的目录都应去掉执行权限，上传目录可配置为只读。

安全设备的防御。文件上传攻击的本质就是将恶意文件或者脚本上传到服务器，专业的安全设备防御此类漏洞主要是通过漏洞的上传利用行为和恶意文件的上传过程进行检测。恶意文件千变万化，隐藏手法也不断推陈出新，对普通的系统管理员来说可以通过部署安全设备来帮助防御。（如华三通信公司发布的SecPath IPS系列产品经过长期的积累，不但可以基于行为对网络中大量文件上传漏洞的利用进行检测，同时还能基于内容对恶意文件进行识别）

■ 常见Web应用安全漏洞-XSS

(1) XSS定义

跨站脚本（Cross-Site Scripting，XSS为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆，故将跨站脚本攻击缩写为XSS）。是一种经常出现在Web应用程序中的计算机安全漏洞，是由于Web应用程序对用户的输入过滤不足而产生的。攻击者利用网站漏洞把恶意的脚本代码（通常包括HTML代码和客户端JavaScript脚本）注入到网页之中，当其他用户浏览这些网页时，就会执行其中的恶意代码，对受害用户可能采取Cookie资料窃取、会话劫持、钓鱼欺骗等各种攻击。

危害：

- 盗取用户cookie
- Xss蠕虫
- 挂马，结合xss蠕虫，危害巨大。

常见Web应用安全漏洞-XSS

(2) 什么是xss攻击?

个人资料信息填写
发表一篇日志
发表一篇留言
发表一篇评论
提出一个问题
回答一个问题
.....

地址栏参数
Dom属性



查看他人资料
查看一篇日志
查看一条留言
查看一个评论
查看一个问题
查看一个答案

点开一个链接
点开一个邮件
.....

注入恶意代码

XSS模型

恶意代码执行

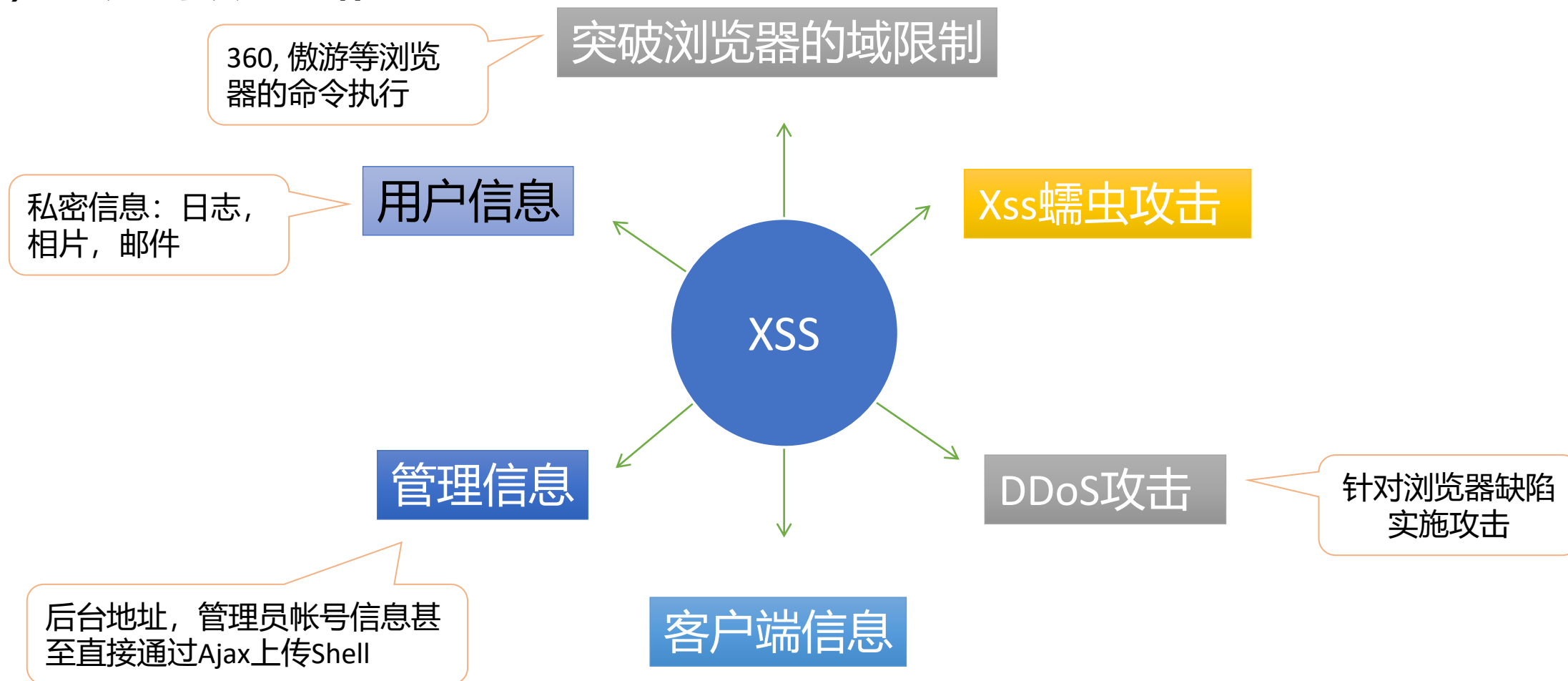
当受害者变为攻击者时，下一轮受害者将更容易被攻击，威力更加明显！

攻击者

受害者

常见Web应用安全漏洞-XSS

(3) xss攻击可以用来做什么?



■ 常见Web应用安全漏洞-XSS



(4) XSS跨站脚本攻击

- <?php
- \$username = \$_GET["name"];
- echo "<p>欢迎您, ".\$username."</p>";
- ?>



(4) XSS跨站脚本攻击

- 我们继续提交name的值为<script>alert(/我的名字是张三/)</script>, 即完整的URL地址为
`http://localhost/test.php?name=<script>alert(/我的名字是张三/)</script>`



欢迎您,



常见Web应用安全漏洞-XSS

(5) XSS跨站脚本攻击的分类

1) 反射型XSS跨站脚本攻击

反射型XSS脚本攻击即如我们上面所
据直接或未经过完善的安全过滤就在浏览
据。由于此种类型的跨站代码存在于URL
代码的链接发给用户，只有用户点击以后



2) 存储型XSS跨站脚本攻击

存储型XSS脚本攻击是指Web应用程序会将用户输入的数据信息保存在服务端的数据库或其他文件形式中，网页进行数据查询展示时，会从数据库中获取数据内容，并将数据内容在网页中进行输出展示，因此存储型XSS具有较强的稳定性。

常见Web应用安全漏洞-XSS

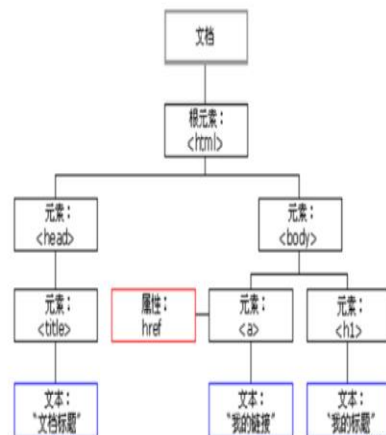
(5) XSS跨站脚本攻击的分类

3) 基于DOM的XSS跨站脚本攻击

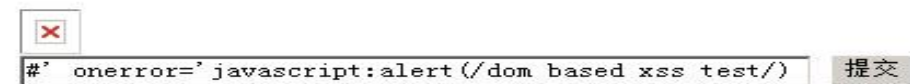
基于DOM的XSS跨站脚本攻击是**通过修改页面DOM节点数据信息而形成的XSS跨站脚本攻击**。不同于反射型XSS和存储型XSS，**基于DOM的XSS跨站脚本攻击往往需要针对具体的javascript DOM代码进行分析**，并根据实际情况进行XSS跨站脚本攻击的利用。

```
<html>
<head>
  <title>文档标题</title>
</head>
<body>
  <h1>我的标题</h1>
  <a href="#">我的链接</a>
</body>
</html>
```

浏览器加载



CSDN @碳烤小肥羊。。。



(5) XSS漏洞常见防护手段

过滤特殊字符。过滤特殊字符的方法又称做XSS Filter，其作用就是过滤客户端提交的有害信息，从而防范XSS攻击。如果跨站代码要想执行，必须得用到一些脚本中的关键函数或者标签，如果能写一个较为严密的过滤函数，将输入信息中的关键字过滤掉，那跨站脚本就不能被浏览器识别和执行了。

使用实体化编码。在测试和使用的跨站代码中几乎都会使用到一些常见的特殊符号。有了这些特殊符号，攻击者就可以肆意地进行闭合标签、篡改页面、提交请求等行为。在输出内容之前，如果能够对特殊字符进行编码和转义，让浏览器能区分这些字符是被用作文字显示而不是代码执行，就会导致攻击者没有办法让代码被浏览器执行。编码的方式有很多种，每种都适应于不同的环境。下面介绍两种常见的安全编码。

Html编码。这是对于Html中特殊字符的重新编码，称为HtmlEncode。为了对抗XSS，需要将一些特殊符号进行Html实体化编码，当输出中需要这些特殊符号时，在HTML源码中会显示为编码后的字符，由浏览器将它们翻译成特殊字符在用户页面上输出。由此，浏览器便直到这些特殊字符只能作为文本显示，不能当作代码执行，从而规避了XSS风险。

JavaScript编码。类似上述情况，用户的输入信息有时候被嵌入到JavaScript代码块中，作为动态内容输出。出于同样的安全角度，考虑到JS语言的特性，需要对输出信息中的特殊字符进行转义。

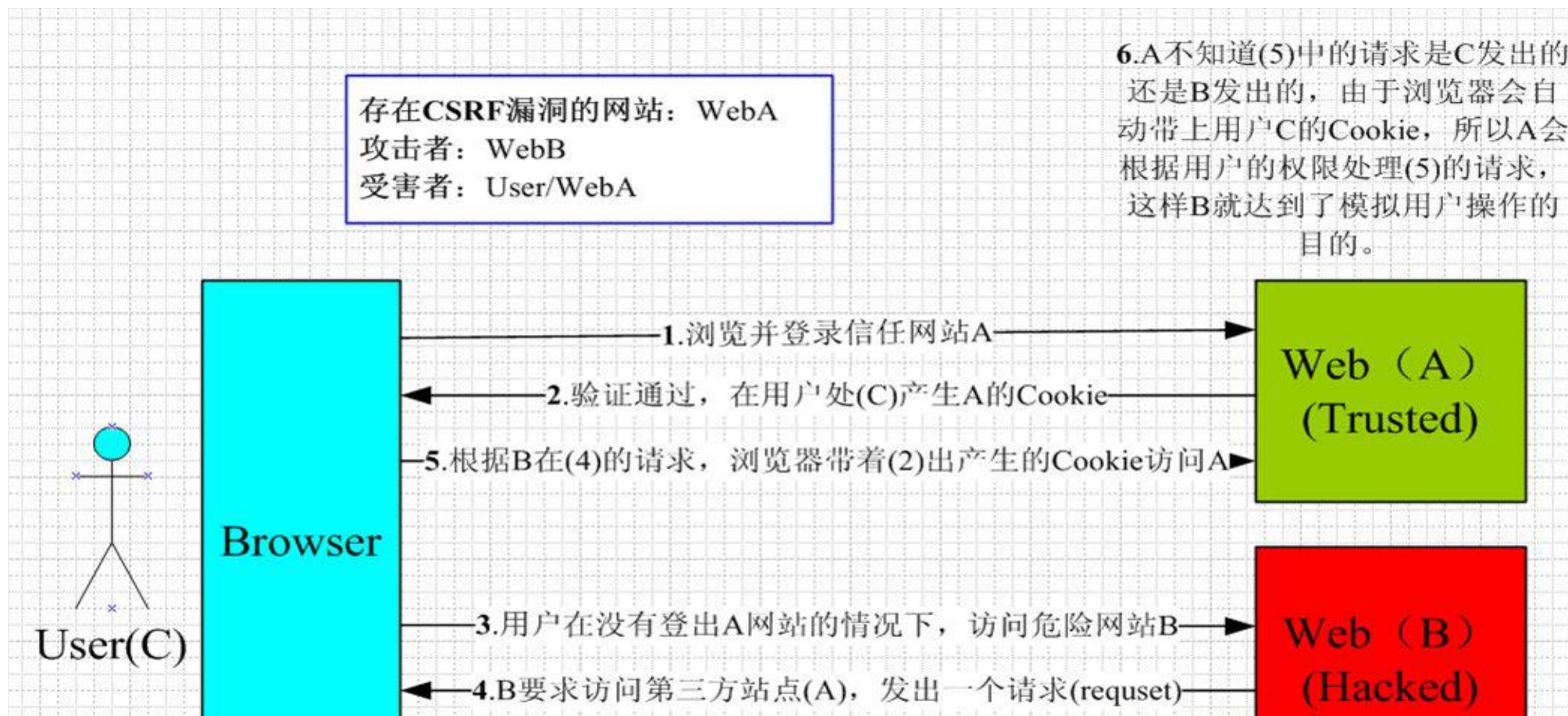
(1) CSRF定义

- CSRF，全称Cross-Site Request Forgery，**跨站请求伪造**，是指利用受害者**尚未失效的身份认证信息**（cookie、会话等），诱骗其点击恶意链接或者访问包含攻击代码的页面，在受害人不知情的情况下**以受害者的身份**向（身份认证信息所对应的）服务器发送请求，从而完成非法操作（如转账、改密等）。
- CSRF与XSS最大的区别就在于，CSRF并没有盗取cookie而是直接利用。
- 在2017年发布的新版OWASP Top 10中，CSRF排名第8。

常见Web应用安全漏洞-CSRF



(2) CSRF的原理



(2) CSRF的简介

- CSRF的整个攻击过程示例（银行转账）：

假如一家银行用以运行转账操作的URL地址如下

`http://www.examplebank.com/withdraw?account=AccoutName&amount=1000&for=PayeeName`

- 1、受害者登录银行网站，通过身份验证，在本机生成Cookie
- 2、受害者单击了含恶意代码的链接，或者直接访问了第三方网站evil.com，并浏览带有下面HTML代码的网页：
- ``

(2) CSRF的简介

- 3、恶意代码利用受害者的身份发送一个请求，执行CSRF
- 4、由于受害者Alice的用户之前刚访问过银行不久，登录信息尚未过期，又访问了恶意站点，而她那么她就会损失1000资金。

透过例子能够看出，攻击者并不能通过CSRF攻击来直接获取用户的账户控制权，也不能直接窃取用户的任何信息。他们能做到的，是欺骗用户浏览器，让其以用户的名义运行操作。

在整个攻击过程中，交易是以受害者的身份发起的。

(3) CSRF与XSS的区别

	CSRF	XSS
名字	跨站请求伪造	跨站脚本
脚本	不是必须，如GET的CSRF	需要借助JavaScript等脚本
产生原因	采用了隐式的验证方式	对用户输入没有正确过滤
防御技巧	验证来源referer，使用验证码、token等	输入过滤、输出编码等
关系	1、如果一个网站存在XSS，很有可能也存在CSRF 2、均利用用户的会话执行某些操作 3、CSRF的恶意代码可能位于第三方站点，过滤用户输入可以防御XSS，但不能防御CSRF	

(5) CSRF漏洞常见防护手段

1、添加验证码

由于攻击者只能仿冒用户发起请求，并不能接收服务器给用户返回的申请。因此可在关键点添加验证码措施，当关键业务的请求发起后，还需输入验证码进行校验。这样可有效避免攻击者伪造请求的情况。

2、验证referer

由于CSRF请求发起方为攻击者，这样在referer处会与当前用户所处的界面完全不同。可验证referer值是否合法，即可通过验证请求来源方式得出此次请求是否正常。但是在部分情况下referer并不一定完全可用，因此推荐利用referer用以监控CSRF行为，如果用于防御，效果并不一定可达到良好。

3、利用token

针对CSRF漏洞，Web系统在建设时主流方式是利用token开展当前用户身份真实性的识别。Token在当前用户第一次访问某项功能页面时生成，且token需具有一次性，并在生成完毕后由服务器端发送给客户端。用户端接收到token之后，会在下一步业务时提交token并由服务器进行有效性验证。由于攻击者在CSRF利用时无法获得当前用户的token，导致就算链接发送成功也无法实现功能的正常执行。

(1) 远程代码执行漏洞

远程代码执行漏洞是指攻击者可以随意执行系统命令。**它属于高危漏洞之一，也属于代码执行的范畴。**远程代码执行漏洞不仅存在于B/S架构中，在C/S架构中也常常遇到。

部分Web应用程序提供了一些命令执行的操作，例如，如果想测试一个网站是否可以正常连接，那么Web应用程序底层就很可能去调用系统操作命令，如果此处没有过滤好用户输入的数据，就很有可能形成系统代码执行漏洞。

(2) 远程代码执行漏洞执行原理

- 我们通过一个案例来说明远程代码执行漏洞的原理，比如在php里存在一种 `preg_replace` 函数，该函数的作用是执行一个正则表达式的搜索和替换，它的参数如下：
- `mixed preg_replace (mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count]])` 搜索subject中匹配pattern的部分，以replacement进行替换。`preg_replace`函数常用语对传入的参数进行正则匹配过滤，实现对参数的输入的有效过滤。因此广泛用于各类系统功能中。

■ 常见Web应用安全漏洞-远程代码执行漏洞



(2) 远程代码执行漏洞执行原理

该函数主要问题在于，当参数\$pattern处存在一个”/e”修饰符时，\$replacement的值会被当成php代码来执行。参考下例：

```
eval.php - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?php
echo preg_replace("/test/e", $_GET['h'], "just test!");
?>
```




■ 常见Web应用安全漏洞-远程代码执行漏洞



(2) 远程代码执行漏洞执行原理

然后远程打开页面并传入参数h=phpinfo()时，看到效果为：

PHP Version 5.5.9-1ubuntu4.9



System	Linux pcc 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64
Build Date	Apr 17 2015 11:43:17
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/fpm
Loaded Configuration File	/etc/php5/fpm/php.ini
Scan this dir for additional .ini files	/etc/php5/fpm/conf.d
Additional .ini files parsed	/etc/php5/fpm/conf.d/05-opcache.ini, /etc/php5/fpm/conf.d/10-pdo.ini, /etc/php5/fpm/conf.d/20-json.ini, /etc/php5/fpm/conf.d/20-mysql.ini, /etc/php5/fpm/conf.d/20-mysqli.ini, /etc/php5/fpm/conf.d/20-pdo_mysql.ini
PHP API	20121113
PHP Extension	20121212



(3) 远程代码执行漏洞防范

禁用高危系统函数。

很多高危函数在真实应用中并没有太多使用。那么针对不用的高危函数可直接禁用，这样就可以从根本上避免程序存在命令执行类漏洞。

严格过滤关键字符

在进行命令执行漏洞利用时都会利用特殊字符进行实现。因此如果将其中的特殊字符进行过滤，那么就可保证攻击失败。

严格限制允许的参数类型

命令执行功能的初衷都是希望用户输入特定的参数实现更丰富的应用开展，如本地命令执行环境，业务系统希望用户输入IP地址来实现ping功能。因此如果能对用户输入参数进行有效的合法性判断，那么既可以避免在原有命令后面拼接多余命令，实现远程命令执行攻击。



第五讲 应用安全



网络空间安全学院
School of Cybersecurity

- ① web应用安全概述
- ② 常见Web应用安全漏洞
- ③ 恶意代码
- ④ 中间件安全
- ⑤ 数据库安全



成都信息工程大学
Chengdu University of Information Technology

恶意代码(malicious code)又称为恶意软件，是能够在计算机系统中进行非授权操作的代码。恶意代码是一种程序，它通过把代码在不被察觉的情况下镶嵌到另一段程序中，从而达到破坏被感染电脑数据、运行具有入侵性或破坏性的程序、破坏被感染电脑数据的安全性、完整性和可用性，盗取应用系统和用户重要数据的目的。

恶意代码特点：

- (1) 传染性 最重要的判别条件。
- (2) 破坏性 良性，恶性
- (3) 潜伏性 潜伏期不容易发现，触发机制。
- (4) 可执行性
- (5) 可触发性 时间，日期，文件类型，特定数据
- (6) 隐蔽性 病毒短小精悍，感染病毒后的程序不宜区别。

(1) 破坏数据

很多恶意代码发作时直接破坏计算机的重要数据，所利用的手段有格式化硬盘、改写文件分配表和目录区、删除重要文件或者用无意义的的数据覆盖文件等。

(2) 占用磁盘存储空间

引导型病毒的侵占方式通常是病毒程序本身占据磁盘引导扇区，被覆盖的扇区的数据将永久性丢失、无法恢复。文件型的病毒利用一些DOS功能进行传染，检测出未用空间把病毒的传染部分写进去，所以一般不会破坏原数据，但会非法侵占磁盘空间，文件会不同程度的加长。

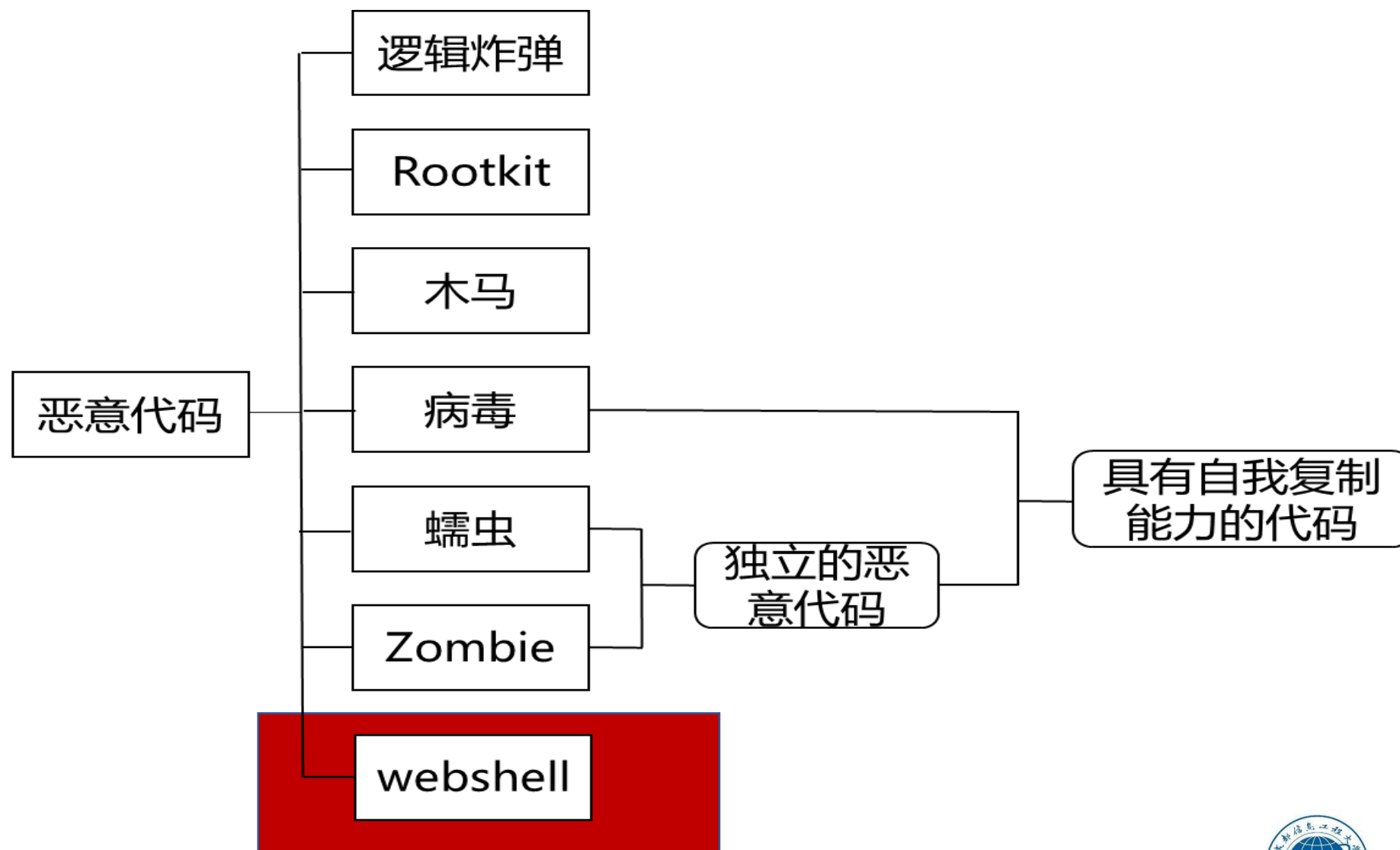
(3) 抢占系统资源

大部分恶意代码在动态下都是常驻内存的，必然抢占一部分系统资源，致使一部分软件不能运行。恶意代码总是修改一些有关的中断地址，在正常中断过程中加入病毒体，干扰系统运行。

(4) 影响计算机运行速度

恶意代码不仅占用系统资源覆盖存储空间，还会影响计算机运行速度。比如，恶意代码会监视计算机的工作状态，伺机传染激发；还有些恶意代码会为了保护自己，对磁盘上的恶意代码进行加密，CPU要多执行解密和加密过程，额外执行了上万条指令。

恶意代码-分类



- 服务器安全设置
- ①加强对脚本文件的代码审计，对出现FSO、Shell对象等操作的页面进行重点分析。借助漏洞扫描工具，及时发现系统漏洞，安装相关补丁；经常关注微软官方网站，及时安装操作系统及相关软件的补丁，并对WEB服务器进行安全设置，关闭不必要的端口，停止不必要的服务，禁止建立空链接；建立本地安全策略和审核策略。
- ②Web服务器通过正则表达式、限制用户输入信息长度等方法对用户提交信息的合法性进行必要的验证、过滤、可以有效防范SQL注入攻击和跨站脚本攻击；尽量使用参数化的SQL查询来代替动态拼接的SQL注入语句，尽量完善操作日志记录和日志记录，也是防范SQL注入攻击的有效手段。
- ③数据库是Web应用系统的重要组成部分，使用数据库系统自身的安全性设置访问数据库权限。如果数据库允许匿名访问，建议创建个别具有高权限的用户，并以此用户执行数据库的操作。

应用安全防护

- ①Web软件开发的安全
 - 程序开发时**防止文件上载的漏洞**，防sql注入、防暴库、防COOKIES欺骗、防跨站脚本攻击。
- ②ftp文件上载安全
 - 设置好ftp服务器，防止攻击者直接使用ftp上传木马程序文件到web程序的目录中。
- ③文件系统的存储权限
 - 设置好web程序目录及系统其它目录的权限，相关目录的写权限只赋予给超级用户，部分目录写权限赋予给系统用户。
 - 将web应用和上传的任何文件（包括）分开，保持web应用的纯净，而文件的读取可以采用分静态文件解析服务器和web服务器两种服务器分别读取（Apache/Nginx加tomcat等web服务器），或者图片的读取，有程序直接读文件，以流的形式返回到客户端。
- ④不要使用超级用户运行web服务
 - 对于apache、tomcat等web服务器，安装后要以系统用户或指定权限的用户运行，如果系统中被植入了asp、php、jsp等木马程序文件，以超级用户身份运行，webshe11提权后获得超级用户的权限进而控制整个系统和计算机。



第五讲 应用安全



网络空间安全学院
School of Cybersecurity

- ① web应用安全概述
- ② 常见Web应用安全漏洞
- ③ 恶意代码
- ④ 中间件安全
- ⑤ 数据库安全



成都信息工程大学
Chengdu University of Information Technology

定义：中间件是一种独立的系统软件或服务程序，分布式应用程序借助这种软件在不同的技术之间共享资源。中间件位于客户机/ 服务器的操作系统之上，管理计算资源和网络通讯。

中间件作用：

中间件屏蔽了底层操作系统的复杂性，使程序开发人员面对一个简单而统一的开发环境，减少程序设计的复杂性，将注意力集中在自己的业务上，不必再为程序在不同系统软件上的移植而重复工作，从而大大减少了技术上的负担。

中间件安全-中间件分类

1、应用服务类中间件

为应用系统提供一个综合的计算环境和支撑平台，包括对象请求代理（ORB）中间件、事务监控交易中间件、JAVA应用服务器中间件等。

2、应用集成类中间件

应用集成类中间件是提供各种不同网络应用系统之间的消息通信、服务集成和数据集成的功能，包括常见的消息中间件、企业集成EAI、企业服务总线以及相配套的适配器等。

3、业务架构类中间件

业务架构类中间件包括业务流程、业务管理和业务交互等几个业务领域的中间件。



(1) Apache的解析漏洞

Apache 解析文件的规则是从右到左开始判断解析,如果后缀名为不可识别文件解析,就再往左判断,直到碰到合法后缀才进行解析,和IIS的相似。(index.php.xxx, 用来绕过黑名单限制)

(2) Tomcat的任意上传文件漏洞

Tomcat在Windows系统下,启用了put请求方法,我们可以构造请求数据包上传脚本木马。

(3) IIS6.0的解析漏洞

这个漏洞的原理就是IIS没有对文件后缀进行安全检测,误认为一些个非法文件是可执行脚本。原理就是绕过检查上传了一个表面合法文件,但是执行的时候而是当成脚本文件处理了。比如一个原始文件是test.asp,为了绕过将这个文件命名成 test.asp.jpg,将这个文件上传成功后,然后IIS将它当成asp文件进行解析,这个是IIS的解析漏洞。

.....



中间件安全-中间件安全案例

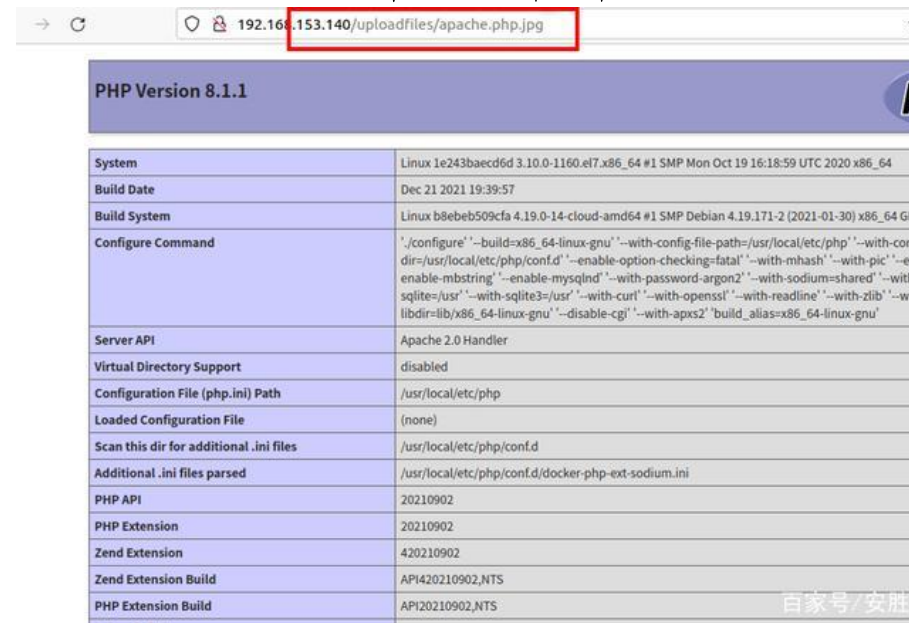
Apache安全事件

运维人员在 Apache 的配置文件 “httpd.conf” 中添加了对 .php 后缀增加了处理器：“AddHandler application/x-httpd-php.php”。在有多个后缀的情况下，只要一个文件含有 .php 后缀（.php 不需要是最后的后缀）的文件即将被识别成 PHP 文件。利用这个特性，将会造成一个可以绕过上传白名单的解析漏洞。

```
AddHandler application/x-httpd-php .php
```

```
DirectoryIndex disabled  
DirectoryIndex index.php index.html
```

```
<Directory /var/www/>  
Options -Indexes  
AllowOverride All  
</Directory>
```



PHP Version 8.1.1	
System	Linux 1e243baecd6d 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020 x86_64
Build Date	Dec 21 2021 19:39:57
Build System	Linux b8eb509cfa 4.19.0-14-cloud-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64 GNU
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pgsql=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-openssl' '--with-readline' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS



第五讲 应用安全



网络空间安全学院
School of Cybersecurity

- ① web应用安全概述
- ② 常见Web应用安全漏洞
- ③ 恶意代码
- ④ 中间件安全
- ⑤ 数据库安全



成都信息工程大学
Chengdu University of Information Technology

(1) 数据库概述

数据库是指存储数据的“仓库”，是长期存放在计算机内、有组织、可共享的大量数据的集合。数据库中的数据按照一定数据模型组织、描述和存储，具有较小的冗余度，较高的独立性和易扩展性，并为各种用户共享。

数据库技术是计算机处理与存储数据的最有效、最成功的技术，这里的典型代表就是关系型数据库，有着非常广泛的应用。关系数据库早期是设计为基于主机/终端方式的大型机上的应用，其应用范围较为有限，随着客户机/服务器方式的流行和应用向客户机方向的分解，关系数据库又经历了客户机/服务器时代，并获得了极大的发展。

SQL是英文（Structured Query Language）的缩写，意思为结构化查询语言，是用于对存放在计算机数据库中的数据进行组织、管理和检索的一种工具。

(2) 数据库基础概念

数据(Data)

图像、语音、文字等

- 在计算机系统中, 各种字母、数字符号的组合、语音、图形、图像等统称为数据。

数据库(Database)

Access、MSSQL、Oracle、SQLITE、MySQL等

- 数据库是按照数据结构来组织、存储和管理数据的“仓库”。

数据库管理系统(DBMS)

Access、MSSQL、Oracle、SQLITE、MySQL等

- 数据库管理系统(database management system)是一种操纵和管理数据库的软件, 用于建立、使用和维护数据库。它对数据库进行统一的管理和控制, 以保证数据库的安全性和完整性。

结构化查询语言(SQL)

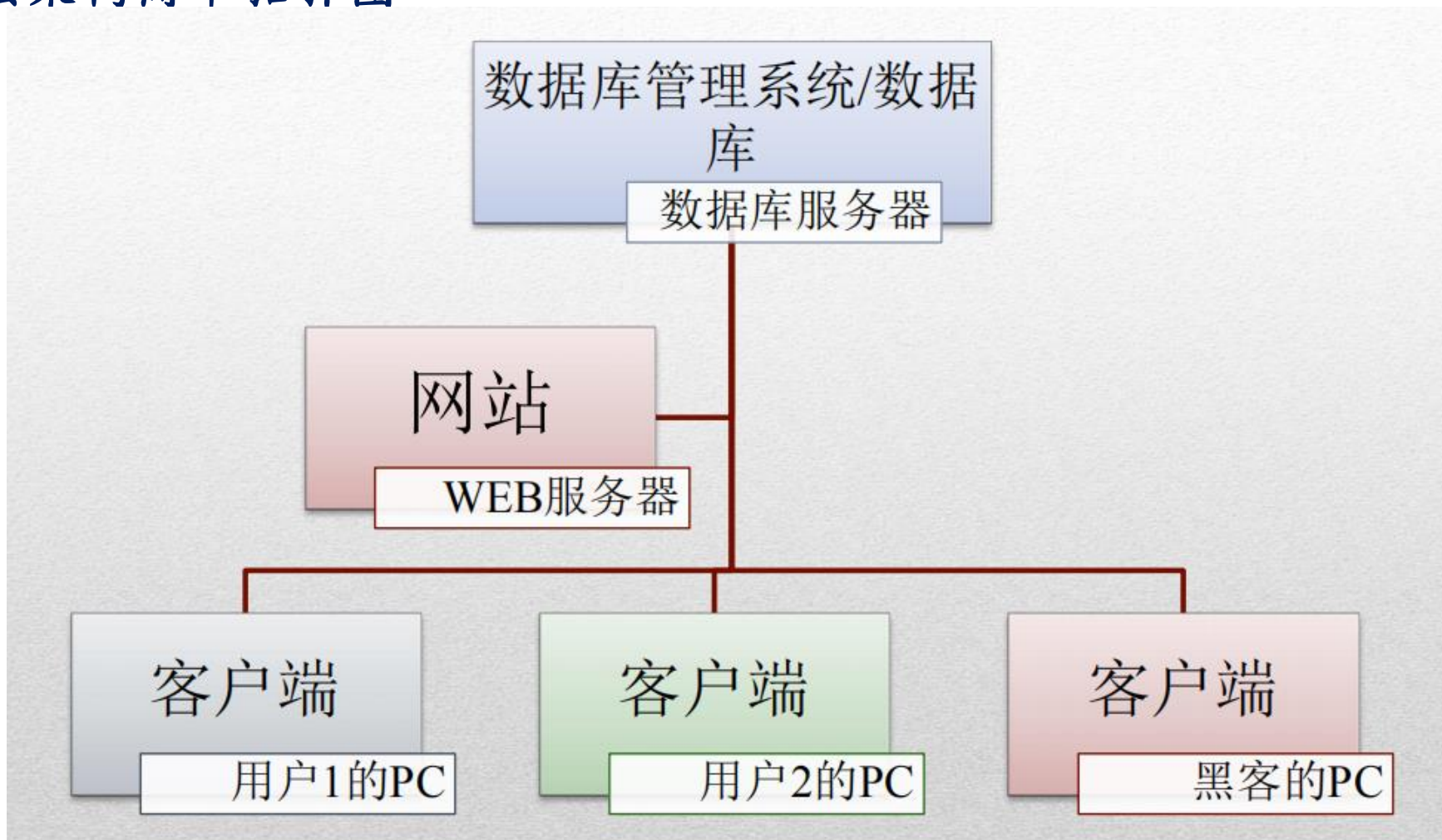
DQL、DDL、DML、TCL、DCL

- 结构化查询语言(Structured Query Language)简称SQL, 结构化查询语言是一种数据库查询和程序设计语言, 用于存取数据以及查询、更新和管理关系数据库系统。

(3) 为什么要使用数据库

- 静态网页：
html或者htm，是一种静态的页面格式，不需要服务器解析其中的脚本。由浏览器如(IE、Chrome等)解析。
 - 1.不依赖数据库
 - 2.灵活性差，制作、更新、维护麻烦
 - 3.交互性交差，在功能方面有较大的限制
 - 4.安全，不存在SQL注入漏洞
- 动态网页：
asp、aspx、php、jsp等，由相应的脚本引擎来解释执行，根据指令生成静态网页。
 - 1.依赖数据库
 - 2.灵活性好，维护简便
 - 3.交互性好，功能强大
 - 4.存在安全风险，可能存在SQL注入漏洞

(4) 三层架构简单拓扑图



数据库安全-SQL注入漏洞

(5) SQL注入原理及危害

就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。通过递交参数构造巧妙的SQL语句，从而成功获取想要的数据库。

分为字符型注入和数字型的注入，由于编程语言不同，所存在的注入类型也不同。

危害：

- 非法查询其他数据库资源，如管理员帐号。
- 执行系统命令
- 获取服务器root权限

(5) SQL注入原理及危害

Test.asp文件代码片段:

```
sqlStr = "select count(*) from admin where username= ' " &username&" ' and  
password= ' "&password&" ' ;  
rs = conn.execute(sqlStr) ;
```

- 现在提交账号为admin, 密码为password, 跟踪SQL语句, 发现最终执行的SQL语句为:
- *select count(*) from admin where username='admin' and password='password'*
- 在数据库中, 存在admin用户, 并且密码为password, 所以此时返回结果为“1”。显然, 1大于0, 所以通过验证, 用户可以成功登录

账号:

密码:

数据库安全-SQL注入漏洞

(5) SQL注入原理及危害

- 输入这个特殊用户 “'or1=1--”
- *select count(*) from admin where username=' 账户' or 1=1--' and password= ' '*
- 此时的password根本起不了任何作用，因为它已经被注释了，而且 username=' 账户' or 1=1这条语句永远为真，那么最终执行的SQL语句相当于：
- *select count (*) from admin* //查询admin表所有的数据条数
- 很显然，返回条数大于0，所以可以顺利通过验证，登录成功。

(6) SQL注入漏洞常见防护手段

1 参数类型检测

参数类型检测主要面向纯字符型的参数查询，可以用以下函数实现：

A. `int intval (mixed $var [, int $base = 10])`：通过使用指定的进制 `base` 转换（默认是十进制），返回变量 `var` 的 `integer` 数值。

B. `bool is_numeric (mixed $var)`：检测变量是否为数字或数字字符串，但此函数允许输入为负数和小数。

C. `ctype_digit`：检测字符串中的字符是否都是数字，负数和小数会检测不通过。

在特定情况下使用这三个函数限制用户输入的为数字型。在一些仅允许用户参数为数字的情况下非常适用。

(7) SQL注入漏洞常见防护手段

2 参数长度检测

当攻击者构造sql语句进行注入攻击时，其SQL注入语句一般都会有一定长度，并且成功执行的SQL注入语句的字符数量通常都会非常多，并且远大于正常业务中有效参数的长度。因此，如果某处提交的内容特点会在一定的长度以内（如密码，用户名，邮箱，手机号等），那么严格控制这些提交点的字符长度，大部分注入语句就没办法成功。这样也就可实现很好的防护效果。

3 危险参数过滤

常见的危险参数过滤方法包括关键字，内置函数，敏感字符的过滤，其过滤方法主要有如下三种：

- A. 黑名单过滤：将一些可能用于注入的敏感字符写入黑名单中，如'（单引号）、union、select等，也可使用正则表达式做过滤，但黑名单可能会有疏漏。
- B. 白名单过滤：指接收已记录在案的良好输入操作，比如用数据库中的已知值校对。
- C. GPC过滤：对变量默认进行addslashes（在预定义字符前添加反斜杠）。

数据库安全-SQL注入漏洞

(7) SQL注入漏洞常见防护手段

4 参数化查询

参数化查询是指数据库服务器在数据库完成 SQL 指令的编译后，才套用参数运行，因此就算参数中含有具有损的指令，也不会被数据库所运行，仅认为它是一个参数。在实际开发中，前面提到的入口处的安全检查是必要的，参数化查询一般作为最后一道安全防线。

php中三种较常见的框架：访问mysql数据库的mysqli包，PEAR：：MDB2包，Data Object框架。

但并不是所有数据库都支持参数化查询。目前Access、SQL Server、MySQL、SQLite、Oracle等常用数据库支持参数化查询。

(2) 数据库安全案例 (MSSQLserver)

在数据库中有很多系统存储过程（预先定义好的“SQL语句集合”），有些是数据库内部使用的，其危险性最高的“小脚本”就是扩展存储过程中的“xp_cmdshell脚本”，它可以执行操作系统的任何指令，具体语法是：

xp_cmdshell {'command_string'} [, no_output] (no_output可选参数，表示执行给定的 command_string，但不向客户端返回任何输出)

前提：

必须获得执行xp_cmdshell存储过程的用户权限（可通过websHELL、sql注入等获取到权限）

该组件默认是关闭的，因此需要把它打开...

(除Access数据库外，其他数据库基本都存在数据库提权的可能)

数据库安全-数据库提权

(2) 数据库安全案例 (MSSQLserver)

如果有一个能执行xp_cmdshell的数据库帐号，比如是空口令的sa帐号。那么就可以执行这样的命令：

```
exec xp_cmdshell 'net user refdom 123456 /add'
```

```
exec xp_cmdshell 'net localgroup administrators refdom /add'
```

上面两次调用就在系统的管理员组中添加了一个用户：refdom。获得了数据库的sa管理员帐号后，就应该可以完全控制这个机器了，可见数据库安全的重要性。

不同的数据库，提权使用的命令、方式不一样



THE END

