本人主要研究方向不是网络安全，只是常对安全场景进行数据挖掘和算法识别，故仅侧重原理部分，具体poc和解决方案仅供参考。

本文当前覆盖10种常见的web应用安全漏洞。

SQL注入

原理

SQL注入算是日常听到最多的漏洞之一。SQL注入本质就是通过WEB接口向服务端传入一些特殊字符，达到欺骗服务器执行恶意SQL命令的目的。

比如查询一个用户的个人信息，可能会有下面的执行逻辑

```
-- 前端获取用户名
name = "外部输入名称";

-- 服务端拼接SQL执行
sql = "select * from users where name=" + name;
```

正常情况下不会有问题，但是如果恶意用户输入 `admin` 或 `'' or '1'='1'` 时，就会导致恶意查询

```
select * from users where name='admin';
或
select * from users where name='' or '1'='1';
```

同理，该方式除查询外，还可以进行恶意的增删改。

SQL 注入漏洞属于后端的范畴，但前端也可做体验上的优化。

SQL注入根据攻击者获取数据的方式分为回显注入、报错注入以及盲注。

上面这种直接在注入点获取数据方式就是回显注入。如果没有直接过去到数据，通过数据库的报错信息来推断数据库的结构和内容，这就是报错注入。但是更多时候是服务端会把数据库的错误提示隐藏，攻击者只能通过数据库的逻辑和延时函数来判断注入的结果，这就是盲注。

解决方案

- 不信任任何输入
- 每一个输入都进行严格的权限控制和数据校验。

XSS攻击

原理

XSS（Cross-Site Scripting）全称为跨站脚本攻击，简言之就是对用户输入的数据没有进行严格的校验，导致客户端在渲染服务器的返回时，存在预期之外的脚本被执行。比如获取用户的敏感信息如 Cookie、SessionID 等。

XSS 攻击更偏向前端的范畴，但后端在保存数据的时候也需要对数据进行安全过滤。

比如，A公司需要开发一个搜索页面，根据URL参数决定关键词的内容，于是小明开发了如下页面：

```
<input type="text" value="<%= getParameter("keyword") %>">
<button>搜索</button>
<div>
  您搜索的关键词是：<%= getParameter("keyword") %>
</div>
```

一般情况下，这个页面不会有问题，但是如果请求变成 `keyword="><script>alert('XSS');</script>`

```
<input type="text" value=""><script>alert('XSS');</script>">
<button>搜索</button>
<div>
  您搜索的关键词是："><script>alert('XSS');</script>
</div>
```

浏览器无法分辨出 `<script>alert('XSS');</script>` 是恶意代码，因而将其执行。

这里不仅仅 div 的内容被注入了，而且 input 的 value 属性也被注入， alert 会弹出两次。

这就是一个XSS攻击，浏览器把用户的输入当成了脚本进行了执行，当然这里只是示例了特殊字符，另外 `javascript:` 这样的字符也会引发XSS攻击。

根据攻击来源，XSS攻击可以分为三类，分别是存储型、反射型和 DOM 型。

- 存储型XSS：把用户输入的数据 "存储" 在服务器端，当浏览器请求数据时，脚本从服务器上传回并执行。例如攻击者在社区或论坛上写下一篇包含恶意 JavaScript 代码的文章或评论，文章或评论发表后，所有访问该文章或评论的用户，都会在他们的浏览器中执行这段恶意的 JavaScript 代码。
- 反射型XSS：把用户输入的数据 “反射” 给浏览器，这种攻击方式往往需要攻击者诱使用户点击一个恶意链接，或者提交一个表单，或者进入一个恶意网站时，注入脚本进入被攻击者的网站。
- DOM型XSS：通过恶意脚本修改页面的 DOM 结构，是纯粹发生在客户端的攻击。

解决方案

渲染前端页面（不管是客户端渲染还是服务器端渲染）或者动态插入 HTML 片段时，任何数据都不可信任，都要先做 HTML 过滤，然后再渲染。

CSRF攻击

原理

CSRF（Cross-site Request Forgery） 全称跨站请求伪造攻击，简单的说就是攻击者盗用了你的身份，以你的名义发送恶意请求。一个典型的CSRF攻击流程如下：

- 受害者登录a.com，并保留了登录凭证（Cookie）
- 攻击者引诱受害者访问了b.com
- b.com 向 a.com 发送了一个请求：a.com/act=xx。浏览器会默认携带a.com的Cookie
- a.com接收到请求后，对请求进行验证，并确认是受害者的凭证，误以为是受害者自己发送的请求
- a.com以受害者的名义执行了act=xx

- 攻击完成，攻击者在受害者不知情的情况下，冒充受害者，让a.com执行了自己定义的操作

举个例子，比如 a 网站上对创作者的关注功能存在CSRF漏洞，当小明访问 a 后并保留了cookie，但是并没有关注创作者。攻击者本身有一个网站 b ，当小明访问 b 时候，b 会向 a 发送关注创作者的请求，此时浏览器也会默认携带小明在a上的Cookie去执行 a 的关注请求，然后小明就在不知情的情况下关注了创作者。

注：JSON劫持也属于CSRF攻击的范畴，一些 Web 应用会把一些敏感数据以 json 的形式返回到前端，如果仅仅通过 Cookie 来判断请求是否合法，那么就可以利用类似 CSRF 的手段，向目标服务器发送请求，以获得敏感数据。

解决方案

CSRF攻击的整个过程攻击者并不能获取到受害者的登录凭证，仅仅是“冒用”。攻击一般发起在第三方网站，而不是被攻击的网站。被攻击的网站无法防止攻击发生，只能通过增强自己网站针对CSRF的防护能力来提升安全性。常用的策略为：

- 阻止不明外域的访问：同源检测，Samesite Cookie
- 提交时要求附加本域才能获取的信息：CSRF Token，双重Cookie验证
- 保证页面的幂等性，后端接口不要在GET页面中做用户操作

SSRF攻击

原理

SSRF（Server-Side Request Forgery）为服务器端请求伪造漏洞，是一种由攻击者构造形成并由服务端发起恶意请求的一个安全漏洞。主要是由于服务器对用户提供的可控URL过于信任，或服务端提供了从其他服务器应用获取数据的功能而没有对目标地址做过滤和限制，没有对攻击者提供的RUL进行足够的检测，导致攻击者可以以此为跳板攻击内网或其他服务器。

正是因为恶意请求由服务端发起，而服务端能够请求到与自身相连而与外网隔绝的内部网络系统，所以一般情况下，SSRF的攻击目标是攻击者无法直接访问的内网系统。所以SSRF漏洞可能导致运行在内网或服务器本地的其他应用程序，如redis、mysql被控制，企业内部的资产信息被盗用，利用file协议读取服务器本地/内网文件，扫描内网端口，探测内网主机存活，跳板攻击等。

另外比如在社交分享、转码服务、在线翻译、图片下载、图文收藏、邮件系统、从远程服务器请求资源等也很容易出现SSRF攻击。

SSRF漏洞相关函数和类如（主要是利用file、http/s和dict协议）：

- file_get_contents()：将整个文件或一个url所指向的文件读入一个字符串中
- readfile()：输出一个文件的内容
- fsockopen()：打开一个网络连接或者一个Unix 套接字连接
- curl_exec()：初始化一个新的会话，返回一个cURL句柄，供curl_setopt()，curl_exec()和curl_close() 函数使用
- fopen()：打开一个文件文件或者 URL

解决方案

- 限制协议为HTTP、HTTPS
- 不用限制302重定向
- 设置URL白名单或者限制内网IP

越权漏洞

原理

越权漏洞（Broken Access Control）是Web应用程序中一种常见的漏洞。主要是因为应用在检查授权时存在纰漏，使得攻击者在获得低权限用户账户后，利用一些方式绕过权限检查，访问或者操作其他用户或者更高权限。

通常情况下，一个程序功能流程是登录 - 提交请求 - 验证权限 - 数据库查询 - 返回结果。如果验证权限不足，便会导致越权。常见的程序都会认为通过登录后即可验证用户的身份，不会做下一步验证，最后导致越权。

越权分为水平越权漏洞和垂直越权漏洞。

- 水平越权（权限类型不变，权限ID改变）：水平越权访问是一种 “基于数据的访问控制” 设计缺陷引起的漏洞。攻击者尝试访问与他拥有相同权限的用户资源。如，用户A和用户B属于同一角色，拥有相同的权限等级，他们能获取自己的私有数据（数据A和数据B），但如果系统只验证了能访问数据的角色，而没有对数据做细分或者校验，导致用户A能访问到用户B的数据（数据B），那么用户A访问数据B的这种行为就叫做水平越权访问。
- 垂直越权（权限ID不变，权限类型改变）：垂直越权是一种 “基于URL的访问控制” 设计缺陷引起的漏洞，又叫做权限提升攻击。由于后台应用没有做权限控制，或仅仅在菜单、按钮上做了权限控制，导致恶意用户只要猜测其他管理页面的URL或者敏感的参数信息，就可以访问或控制其他角色拥有的数据或页面，达到权限提升的目的。

解决方案

- 前后端同时对用户输入信息进行校验，双重验证机制
- 执行关键操作前必须验证用户身份，验证用户是否具备操作数据的权限
- 特别敏感操作可以让用户再次输入密码或其他的验证信息。
- 从用户的加密认证 cookie 中获取当前用户 id，防止攻击者对其修改。或在 session、cookie 中加入不可预测、不可猜解的 user 信息。
- 直接对象引用的加密资源ID，防止攻击者枚举ID，敏感数据特殊化处理

DDoS攻击

原理

DoS（Distributed Denial of Service） 为分布式拒绝服务攻击，是DoS 攻击（Denial of Service）的升级版。这是一个偏暴力的攻击方式，DoS是攻击者不断地提出服务请求，让合法用户的请求无法及时处理。而DDoS就是利用分布式集群进行DoS攻击。

当网络数据包的数量达到或者超过上限的时候，会出现网络拥堵、响应缓慢的情况。DDoS就是利用这个原理，发送大量网络数据包，占满被攻击目标的全部带宽，从而造成正常请求失效，达到拒绝服务的目的。

DDoS按照攻击方式可以分为反射型、流量放大型、脉冲波型、链路泛洪和混合型等。

- 反射型：攻击者并不直接攻击目标服务的 IP，而是利用互联网的某些特殊服务开放的服务器，通过伪造被攻击者的 IP 地址向有开放服务的服务器发送构造的请求报文，该服务器会将数倍于请求报文的回复数据发送到被攻击 IP，从而对后者间接形成 DDoS 攻击。
- 流量放大型：通过递归等手法将攻击流量放大的攻击类型，比如攻击者将 Search type 设置为 ALL。搜索所有可用的设备和服务，这种递归效果产生的放大倍数是非常大的，攻击者只需要以较小的伪造源地址的查询流量就可以制造出几十甚至上百倍的应答流量发送至目标。

- 脉冲波型：之所以将这种新技术命名为脉冲波，是由于其攻击流量展现出来的图形看起来很像不连贯的重复的脉冲状。这类攻击通常呈现一个有上有下的斜三角形的形状，这个过程就是攻击者正在慢慢组装机器人并将目标对准待攻击的目标。一次新的脉冲波攻击从零开始，在很短的时间跨度内达到最大值，然后归零，再回到最大值，如此循环重复，中间的时间间隔很短。脉冲波型 DDoS 相对难以防御，因为其攻击方式避开了触发自动化的防御机制。
- 链路泛洪：不直接攻击目标而是以堵塞目标网络的上一级链路为目的。对于使用了 IP Anycast 的企业网络来说，常规的 DDoS 攻击流量会被分摊到不同地址的基础设施，这样能有效缓解大流量攻击。所以攻击者发明了一种新方法，攻击至目标网络 traceroute 的倒数第二跳，即上联路由，致使链路拥塞。
- 混合型：比如 TCP 和 UDP、网络层和应用层攻击同时进行，这样的攻击既具备了海量的流量，又利用了协议、系统的缺陷，尽其所能地展开攻势。对于被攻击目标来说，需要面对不同协议、不同资源的分布式的攻击，分析、响应和处理的成本就会大大增加。

解决方案

设置高性能设备、提高带宽、提升硬件配置、提高web server 的负载能力、清晰异常流量、网站尽可能做成静态页面、分布式集群防御、IP轮询技术、流量预压制、运营商过滤、安全验证。

HTTP报头追踪漏洞

原理

HTTP/1.1（RFC2616）规范定义了 HTTP TRACE 方法，主要是用于客户端通过向 Web 服务器提交 TRACE 请求来进行测试或获得诊断信息。

当 Web 服务器启用 TRACE 时，提交的请求头会在服务器响应的内容（Body）中完整的返回，其中 HTTP 头很可能包括 Session Token、Cookies 或其它认证信息。攻击者可以利用此漏洞来欺骗合法用户并得到他们的私人信息。

解决方案

禁用 HTTP TRACE 方法。

XXE漏洞

原理

XXE 漏洞（XML External Entity）全称 XML 外部实体漏洞，当应用程序解析 XML 输入时，如果没有禁止外部实体的加载，导致可加载恶意外部文件和代码，就会造成任意文件读取、命令执行、内网端口扫描、攻击内网网站等攻击。

这个只在能够接收 xml 格式参数的接口才会出现。所以了解 xml 漏洞前，先大概了解一下xml文档。XML用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。主要由元素（主要构建模块，可包含文本、其他元素或者为空，）、属性（元素的额外信息）、实体（定义普通文本的变量）、PCDATA（被解析器解析的文本）、CDATA（不会被解析器解析的文本）组成。一个完整的示例如下：

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
```

```
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
  <to>George</to>
  <from>John</from>
  <heading>Reminder</heading>
  <body>Don't forget the meeting!</body>
</note>
```

常用的攻击方式由

- 直接通过DTD外部实体声明
- 通过DTD文档引入外部DTD文档，再引入外部实体声明
- 通过DTD外部实体声明引入外部实体声明

解决方案

- 禁用外部实体
- 过滤用户提交的XML数据

敏感数据泄露

原理

由于 Web 服务器或应用程序没有正确处理一些特殊请求，泄露 Web 服务器的一些敏感信息，或没有正确用户保护敏感数据，如信用卡、身份验证凭据等。攻击者可能把这些数据卖给骗子进行诈骗犯罪等。

如姓名、身份证ID、电话号码、银行账户、驾驶证号码、社保卡号、护照号码等个人信息。网站登录的用户名、密码、SSL证书、会话ID、加密使用的密钥。Web服务器的OS类型、版本信息、Web容器信息、数据库类型版本、使用的开源软件版本等。

解决方案

- 严格控制数据安全等级，保证权限的最小必要性
- 对敏感数据必须进行加密存储、使用安全的加密算法
- 应用程序报错时，不对外产生调试信息
- 过滤用户提交的数据与特殊字符
- 保证源代码、服务器配置的安全

文件上传漏洞

原理

文件上传漏洞通常由于网页代码中的文件上传路径变量过滤不严造成，如果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，攻击者可通过 Web 访问的目录上传任意文件，包括网站后门文件（webshell），进而远程控制网站服务器。

解决方案

- 在开发网站及应用程序过程中，需严格限制和校验上传的文件，禁止上传恶意代码的文件。
- 同时限制相关目录的执行权限，防范webshell攻击。

参考文献

[web 应用常见安全漏洞一览]

Web 安全漏洞之 SQL 注入

sql注入的原理与分类

XSS,CSRF,SSRF三种漏洞的区别和共同点

[前端安全系列（一）：如何防止XSS攻击？]

浅说 XSS 和 CSRF

前端安全系列之二：如何防止CSRF攻击？

CTF SSRF 漏洞从0到1

越权漏洞原理及防御方案

越权与逻辑漏洞

DDoS的攻击及防御

浅谈 DDoS 攻击与防御

xxe漏洞的学习与利用总结