根据燕姐知乎以及面向csdn总结，仅供参考
选择（10*2‘） 填空（20） 问答（60）

HTTPS

（404，200，）

HTTP 请求消息格式：

1.请求行，用来说明请求类型，要访问的资源以及实用的 HTTP 版本

2.请求头部：说明服务器要使用的附附加信息

3.空行

4.请求数据：添加任意的其他数据。

例子：

```
POST / HTTP1.1
Host:www.wrox.com
User-Agent:Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR
3.0.04506.648; .NET CLR 3.5.21022)
Content-Type:application/x-www-form-urlencoded
Content-Length:40
Connection: Keep-Alive

name=Professional%20Ajax&publisher=Wiley
```

请求行：第一行 POST 请求，http 版本 1.1

请求头部：第二到第六行；HOST 将指出请求的目的地.User-Agent,服务器端和客户端脚本都能访问它,它是浏览器类型检测逻辑的重要基础.该信息由你的浏览器来定义,并且在每个请求中自动发送等等

空行：第七行

请求数据：第八行

URI（统一资源标识符）：输数据和建立连接。URL 是一种特殊类型的 URI，包含了用于查找某个资源的足够的信息

URL（统一资源定位符）：互联网上用来标识某一处资源的地址

URL 各字段意义：举一个例子： http://standford.edu:81/class?name=cs155#homework

协议部分： “http”

域名部分：“standford.edu”

路径部分：/class

端口部分： “：80” 当不定义端口时，使用默认端口

锚点部分：#homework 用于定位到页面 id=homework 的地方

参数部分： ？name=css155 z 也叫搜索部分

明文传输的安全隐患：不对数据加密，数据只要在网络上是广播的，就不可避免地被嗅探。

HTTPS 与 HTTP 的区别：

- https 协议需要到 ca 申请证书
- http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。
- http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。
- http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全

SSL 协议：

1.SSL 记录协议：建立在 TCP 之上，为高层协议提供数据封装、压缩、加密等基本功能的支持；

2.SSL 握手协议：建立在 SSL 记录协议上，用于实际的数据传输开始前，通讯双方进行身份验证，协商加密算法、交换加密密钥等。

SSL/TLS 协议提供的服务：

1.认证：认证服务器和用户，确保数据发送到正确的客户机和服务器；

2.保密性：加密数据防止数据传输途中被窃取；

3.完整性：维护数据完整性，确保数据在传输过程中不被改变。

证书生成的过程：生成 CA 的公私密钥对，并生成自签根证书，根证书包含 CA 机构的信息（CA 颁发机构名、签名哈希算法、有效期、公钥等）。 具体步骤：生成 CA 私钥（.key）->生成 CA 证书请求 （.csr）->自签名得到根证书（.crt）。

CA 签发证书以及浏览器和网站身份验证建立安全连接的具体流程：

1.CA 建立，生成 CA 机构的根证书；

2.CA 与浏览器厂商联系，如果浏览器厂商信任这个 CA，就会把这个 CA 证书加入到自己开发的浏览器中。

3.一个想要获得 HTTPS 服务的网站，首先要将自己网站的验证信息（域名，网站信息）和自己的公钥发送给 CA 机构，向 CA 申请自己的服务器证书，CA 在验证这个网站的合法性后，会用自己的私钥对网站的验证信息和公钥打包进行签名，形成这个网站的证书，并将这个签名的证书颁发给网站。 现在网站已经有自己的证书了，于是它可以搭建自己的 https 服务。

4.用户使用浏览器链接到该网站。在加密连接建立之前，网站首先将自己的 CA 签名的证书发给用户浏览器，浏览器结合自带的 CA 证书，对网站证书（被 CA 私钥签名的）进行验证（用 CA 的公钥对网站证书进行解密，再核对网站域名等是否正确），验证成功后，建立安全连接

5.SSL 握手：1.服务器向客户端发送被 CA 签名的证书和服务器的其他相关信息；
2.浏览器用 CA 的公钥对证书解签，比较证书里的信息是否和服务器传回的信息一致；
3.客户端验证成功后，将自己所支持的加密方式以及自己的公钥发给服务器
4.服务器根据客户端发送的支持的加密方式，选择一种加密程度最高的方案，用客户端公钥加密，发送给浏览器。
5.客户端通过自己的私钥解密后，选择一个会话密钥用服务器的公钥加密发送给服务器
6.接下来的数据传输都使用该对称密钥进行加密。

https 的局限性：1.https 比 http 耗费更多的服务器资源；SSL 证书需要钱。。；
2. SSL 可以允许多种密钥交换算法，而有些算法，如 DH，没有证书的概念，这样 A 便无法验证 B 的公钥和身份的真实性，从而 C 可以轻易的冒充，用自己的密钥与双方通信，从而窃听到别人谈话的内容。
3. A 与 B 网站通信，B 有了证书以后，如果 C 用自己的证书替换掉原有的证书之后，A 的浏览器会弹出一个警告框进行警告，但又有多少人会注意这个警告呢？

Web 相关（HTML，CSS，JS，PHP）

html ： <img> <a> <form>
CSS 的优点：1.实现了内容与表现分离；
2.极大的提高了工作效率，通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观。
3. 由于允许同时控制多重页面的样式和布局，能够为每个 HTML 元素定义样式，并将之应用于你希望的任意多的页面中；如需进行全局的更新，只需简单地改变样式，然后网站中的所有元素均会自动地更新。
CSS position 属性：1.static(静态定位)：元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。
2.relative（相对定位）：元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。
3.absolute（绝对定位）：元素框从文档流完全删除，并相对于其包含块定位。
4.fixed（固定定位）：相对于浏览器窗口来对元素进行定位。
CSS 框模型（Box Model）
border（边框）、padding（内边距）：上右下左、margin（外边距）
样式层叠次序优先级（从小到大）：
浏览器缺省设置、外部样式表、内部样式表（位于 <head> 标签内部）、内联样式（在 HTML 元素内部）

css 历史攻击：原理是利用了伪类，伪类向选择器添加特殊效果（比如链接访问果后会变成紫色）

JS 正则表达式：^表示字符串的开头，但它还有另外一个含义。当在一组方括号里使用^是，它表示"非"或"排除"的意思，常常用来剔除某个字符。

例：^[a-z] 匹配小写字母字符串

[^a-z] 匹配除了小写字母以外的所有字符

[^\\\/\^\"\'] 匹配除了"\"、 "/"、 "^ "、 " 、' 以外的所有字符。

JS 对 DOM 的支持：见知乎 https://zhuanlan.zhihu.com/p/51022879

GET 和 POST 的区别：

GET 方法的查询字符串实在 GET 请求的 URL 中发送的

如：

```
/test/demo_form.asp?name1=value1&name2=value2
```

POST 方法的查询字符串是在 POST 请求的 HTTP 消息主题中发送的

如：

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

其他区别如下：

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。

PHP form 数据，数据库操作见知乎 https://zhuanlan.zhihu.com/p/51710475

攻击（CSRF、XSS、SQL 注入、点击劫持、DOS 攻击）

一、CSRF

1.Cookie 的作用：HTTP 是一个 Stateless，无状态协议；用户刷新一个页面对于服务器而言，就是两次完全没有关系的访问。Cookie 的作用就是帮助服务器记忆。一般来说，在用户首次登陆网站的时候，网站会给用户设置一个 Cookie，这个 cookie 保存在用户浏览器端，一般记录下用户的用户名信息等，然后每次用户访问该网站的网页，浏览器会自动的将 cookie 带上。服务器接收到 cookie 后进行解析，判断是哪个用户。

2.CSRF 原理：利用浏览器每次访问网站都会主动的携带上自己保存的相应网站的 cookie；只要用户当前浏览器中嗨保留着目标网站如 Zoobar.com 的 cookie，然后又浏览了恶意攻击网站，而攻击网站又主动发出了对目标网站的请求，浏览器就会携带上目标网站的 cookie。即盗用受害者身份，以受害者的名义向第三方网站发送恶意请求。

3.Referer 防御： referer 是 http 请求 header 中的一个字段，当浏览器向 web 服务器发送请求的时候，一般会带上 referer，告诉服务器该请求是从哪个网站发出去的。Referer 防御的原理就是通过检验 referer 是否匹配合法请求的网站。
又由于浏览器可以控制修改请求中是否带有 Referer 字段（about：config 中搜 referer 可对 referer 参数进行控制）。
但是 又有哪个用户去这样操作（禁止发送 referer 字段）呢？

4.验证码防御： 强制用户在执行重要操作之前必须进行交互，但是为了考虑用户体验友好，不能给所有的操作都加上验证码。

5.Token 防御：

- 用户登录网站，服务端生成一个Token，放在用户的Session
- 在页面表单附带上Token参数，为了不影响用户，可以设置type=hidden
- 用户提交请求时，表单中的这一参数会自动提交， 服务端验证表单中的Token是否与用户Session中的Token一致，一致为合法请求，不是则非法请求
- 每次提交，token值可以更新

因为这个Token值是每个用户不同，并且开发人员可以设置粒度，用户每次登录不同，还是每次提交不同，这样彻底使得攻击者无法伪造请求。

6.Session 和 Cookie 的区别：

http 是一个无状态协议，一旦数据交换完毕，客户端和服务端的连接就会关闭，再次交换数据就需要建立新的连接，意味着服务器无法从连接上 跟踪会话（会话指用户登陆网站后一系列操作）。

常用的会话跟踪技术是 Cookie 和 Session；

区别：

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗
考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能
考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

具体见：https://www.cnblogs.com/8023-CHD/p/11067141.html

二、XSS 攻击

1.XSS 的原理： xss 是跨站脚本攻击；XSS 通过在用户端注入恶意的可运行脚本，若服务器端对用户输入不进行处理，直接将用户输入输出到浏览器，则浏览器将会执行用户注入的脚本，就发生了 XSS 攻击。

xss 存在的三个条件：网站应用程序必须接受用户的输入信息；用户的输入会被网站用来创建动态内容；对用户的输入验证不充分。

2.XSS 的分类：

1.反射型 XSS：通过 GET 和 POST 方法，向服务器端输入数据。用户输入的数据通常被放置在 URL 中，或者是 form 数据中。如果服务器端对输入的数据不进行过滤，验证或编码，就直接将用户输入的信息直接呈现，则可能会造成反射型 XSS。

2.存储型 XSS:。通常是因为服务器端将用户输入的恶意脚本没有通过验证就直接存储在数据库，并且每次通过调用数据库的方式，将数据呈现在浏览器上。则该 XSS 攻击将一直存在。若其他用户访问该页面，则恶意脚本就会被触发。

3. XSS 的防御：

1.输入过滤：设置 disallowed 标签过滤非法字符（有缺陷，用编码绕过防御）

2.输出转义： 用 htmlspecialchars（$profile） 对输入的 profile 转义。

3.httponly cookie：用来保护 cookie 的特殊措施

- 大多数浏览器都支持httponly cookie
- 仅在传输HTTP（或HTTPS）请求时才使用httpOnly会话cookie
- 防止通过XSS窃取会话cookie

4. Ajax 的优点：

1. **局部更新。仅仅更新数据。**
2. **异步更新，在请求受阻，其他部分的代码可以继续执行。**
3. **接收回复并处理。**

5.Ajax 的属性：XMLHttpRequest 是 Ajax 的基础；

1）生成 XMLHttpRequest 对象：

```
xmlhttp=new XMLHttpRequest();
```

2）open 函数

open(*method*,*url*,*async*)规定请求的类型、URL 以及是否异步处理请求。

- *method*：请求的类型；GET 或 POST
- *url*：文件在服务器上的位置
- *async*：true（异步）或 false（同步）

3）send 函数

send（message）；默认为空

4）responseText 属性 返回字符串形式的响应

5）readyState：

存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。

- 0: 请求未初始化
- 1: 服务器连接已建立
- 2: 请求已接收
- 3: 请求处理中
- 4: 请求已完成，且响应已就绪

SOP 同源策略：　同源是指域名、协议、端口相同。不同源的客户端将本在没有明确授权情况下，不能读写对方的资源。

CORS Http 访问控制：可以设置 http 响应头 使得跨域成功。

通过在HTTP Header中加入扩展字段，服务器在相应网页头部加入字段表示允许访问的domain和HTTP method，客户端检查自己的域是否在允许列表中，决定是否处理响应。
实现的基础是JavaScript不能够操作HTTP Header。某些浏览器插件实际上是具有这个能力的。

服务器端在HTTP的响应头中加入（页面层次的控制模式）：

```
Access-Control-Allow-Origin: example.com
Access-Control-Request-Method: GET, POST
Access-Control-Allow-Headers: Content-Type, Authorization, Accept, Range, Origin
Access-Control-Expose-Headers: Content-Range
Access-Control-Max-Age: 3600
```

三、点击劫持（依赖 iframe、z-index、opacity）

1.z-index 越大层数越靠近上层

2.三种不同方式的透明化：

opacity:0 ： 实际上是元素透明度为 0，元素仍然存在，不会改变页面布局，绑定的事件依然可以触发；

visibility:hidden ： 元素存在，不改变页面布局，但是绑定事件不可触发。

display:none ： 元素彻底消失，脱离文档流，绑定的事件也无法触发。

3.点击劫持的防御

X-Frame-Options 可以控制页面能否被 iframe

例：header(“X-Frame-Options:DENY”)

四、SQl 注入攻击（推荐看老师的网课，讲的很清楚，网课课后习题）

https://next.xuetangx.com/learn/USTC08091001657/USTC08091001657/1075937/video/643223

主要弄懂 select 语句的执行逻辑，知道各种注入串的效果；

防御方式有：1.过滤特殊符号（一刀切，不友好）；

2.开启 magic_quote_gpc 转义特殊符号（使 单引号‘ 转义成普通字符，不会对上一个 sql 语句的单引号进行匹配。 无法防御“1 or 1”攻击）；

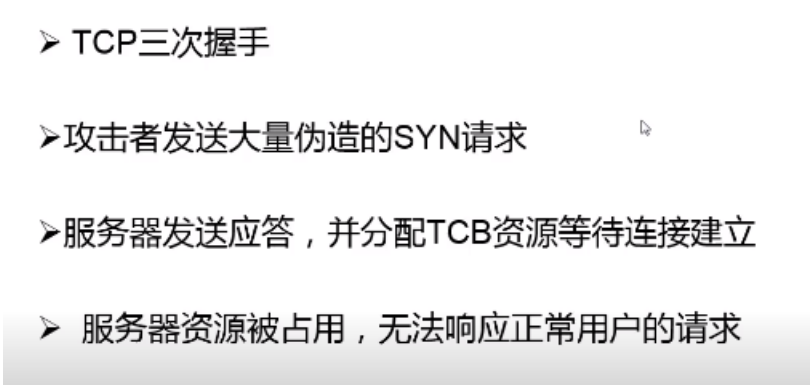
3.显式的区分数据和命令：

```
$stmt = $db->prepare("SELECT * FROM users WHERE
name=? AND age=?");
$stmt->bind_param("si", $user, $age);
```

对输入的信息进行限定 ， 如上例“si”对 user 和 age 进行格式限制，字符串和整型。

五、Dos 攻击

1.SYN Flooding 攻击



2.SYN Flooding 攻击防御

- SYN Cache： 接收到SYN请求时分配少量资源
- SYN Cookie： 接收到ACK时才分配资源
- Linux中SYN Cookie默认开启

3.复杂度 Dos 攻击：

- 什么是基于算法复杂度的DoS攻击？
  - 构造冲突，使哈希操作复杂度增加
  - 使得消耗CPU资源进行哈希查找

4.怎么找到冲突哈希字符串：康网课以及老师知乎

https://next.xuetangx.com/learn/USTC08091001657/USTC08091001657/1075937/video/643230

补充：http 状态字：

200	OK	请求已成功，请求所希望的响应头或数据体将随此响应返回。HTTP/0.9

400	Bad Request	语义有误或请求参数有误，当前请求无法被服务器理解。
401	Unauthorized	当前请求需要用户验证。该响应必须包含一个适用于被请求资源的WWW-Authenticate信息头用以询问用户信息。
402	Payment Required	(该状态码是为了将来可能的需求而预留的。)
403	Forbidden	服务器已经理解请求，但是拒绝执行它。
404	Not Found	请求失败，请求所希望得到的资源未被在服务器上发现。

Ajax 方法 setRequestHeader("content-type","application/x-www-form-urlencoded")的作用是什么?

通 常在 HTTP 协议里，客户端像服务器取得某个网页的时候，必须发送一个 HTTP 协议的头文件，告诉服务器客户端要下载什么信息以及相关的参数。而 XMLHTTP 是通过 HTTP 协议取得网站上的文件数据的，所以也要发送 HTTP 头给服务器。 但是 XMLHTTP 默认的情况下有些参数可能没有说明在 HTTP 头里，这是当我们需要修改或添加这些参数时就用到了 setRequestHeader 方法

x-www-form-urlencoded

当 action 为 get 时候，浏览器用 x-www-form-urlencoded 的编码方式把 form 数据转换成一个字串（name1=value1&name2=value2...），然后把这个字串 append 到 url 后面，用?分割，加载这个新的 url。 当 action 为 post 时候，浏览器把 form 数据封装到 http body 中，然后发送到 server。

请解释 PHP 中 htmlentities()、htmlspecialchars()函数的作用。 **（反馈：已考）**

htmlentities 这个函数转换所有含有对应“html 实体”的特殊字符，比如货币表示符号欧元英镑等、版权符号等，htmlspecialchars 只是把某些特殊的字符转义了， & " ' < >

知乎上的示例问题：

1.CA 的根证书作用是什么？浏览器保存的根证书是 CA 经过私钥加密吗？

答：CA 根证书包含 CA 机构的信息（CA 颁发机构名、签名哈希算法、有效期、公钥等）；预先就存在浏览器中，用于验证被 CA 私钥签名的服务器证书的正确性。

浏览器保存的根证书是 CA 经过私钥签名的，浏览器也保存了 CA 的公钥，用于验证根证书没有被篡改过。

2.浏览器是如何证明网站身份可疑？

答：服务器将自己被 CA 签名的证书以及服务器的相关信息发送给浏览器客户端，客户端通过相应 CA 的公钥对证书解签，验证证书里的内容是否和服务器传回的信息匹配。

3.在公共机房上网，登录了网站处理结束之后，在关闭浏览器之前，你会先退出网站吗？在家上网的时候，会先退出网站还是直接关网页？

答：在公共机房上网，应该先退出网站，因为 Cookie 的机制：。。。

在家上网，也应该先退出网站，防止 CSRF 攻击

4.CSS 历史攻击中，攻击者能不能直接访问用户的浏览器访问历史？

答：不能，此漏洞受到许多限制，其中最重要的是攻击者必须正确生成用户访问过的确切 URL。因此，例如，攻击者可以轻松地检测到您访问了 Bing.com 进行搜索，但是对于他来说，检测到要搜索的内容要困难得多，因为他必须探查所有可能的搜索查询。此外，信息泄漏受到浏览器配置为保留历史记录的持续时间以及浏览器的 nPrivate 状态也会限制。

5. CSRF 攻击中，攻击者能不能获得用户的 cookie?

答：不能，cookie 不能跨域获取。（解释一下 cookie 的机制）

6. 如果网站进行了 CSRF 防御，XSS 攻击进行表单伪造时，还能成功吗?

答：能成功，假如 CSRF 的防御模式是在提交请求里设置 token，且防御粒度是登陆期间；则可以通过 XML.responseTEXT 拿到页面信息，解析获取 token 字段，再通过局部更新加载到伪造的请求表单里。

7. Ajax 进行跨站请求时受到什么限制?

答：受限于 sop（同源策略），

8. 网页使用 JS 方法防御自己被置于 iframe 中，需要注意哪些事项?

答：通过 js 检查页面的顶层是否为本来的页面，若网页被非法页面载入的话，就执行自动跳转到原始页面。但是还要防止被双重 iframe，以及 window.onbeforeunload 等代码的攻击。

9. 在网站进行了 CSRF 有效防御（使用了 token）后，还能怎么样进行跨站攻击?

答：点击劫持

10. 能不能分别举出 DoS 攻击中消耗服务器内存、带宽和 CPU 的例子?

答：消耗内存带宽：synflooding 攻击；消耗 cpu：复杂度 DOS 攻击