- 厦门大学网站开发安全最佳实践
  - 整体建议
    - □ 是否申请了虚拟机而不是使用自有机房部署
    - □ 是否将新闻性质的网站和信息系统分开
    - □ 是否已通过等保测评
    - □ 是否反向代理友好
  - 服务器
    - □ 服务器是否专有用途
    - □ 是否选择了正确的操作系统
    - □ 是否启用了安全更新 ★
    - □ 是否启用了动态磁盘扩展功能
    - □ 是否配置并启用了防火墙 ★
    - □ 是否做了备份 ★
    - □ 是否使用了域名而不是IP部署
    - □ 是否支持IPv6
    - □ 是否支持HTTPS
    - □ 是否支持HTTP2
    - □ 服务器代码和数据等目录是否清晰
    - □ 是否使用了强密码
    - □ 是否使用了监控工具
    - □ 是否部署了软WAF
    - □ 是否不允许Web服务器列出文件
    - □ 是否限制了Web服务器除GET|HEAD|POST以外的方法
    - □ 是否隐藏了Web服务器的Banner信息
    - □ 是否添加了X-Frame-Options不允许网页被嵌入其他网站
    - □ 是否添加了CSP
    - □ 是否有使用漏洞扫描工具扫描服务器和网站
  - 数据库
    - □ 是否应用了独立用户，最小权限原则 ★
    - □ 是否限制了Web数据库管理工具的权限
    - □ 是否没有在源代码版本控制系统里面保存数据库等密码
    - □ 是否正确处理数据库连接串
  - 后台程序
    - □ 是否隐藏了管理员入口
    - □ 是否有使用版本控制系统管理源代码
    - □ 是否有使用框架协助开发
    - □ 是否关闭了调试模式 ★
    - □ 是否有对用户的输入进行检查过滤 ★
    - □ 是否有预防注入漏洞 ★
    - □ 是否有预防跨站脚本攻击 Cross-site scripting (XSS) ★
    - □ 是否有预防跨站域请求伪造 Cross Site Request Forgery (CSRF)
    - □ 是否使用了统一身份认证
    - □ 是否对密码进行了加密保存
    - □ 是否对密码、登录等部分做了防暴力破解检查
    - □ 是否对程序文件所在目录设置只读和不可执行权限设置
    - □ 是否对下载文件进行了全面的保护 ★
    - □ 是否使用UTF8格式保存源代码和展示网页
    - □ 是否考虑了国际化 i18n
    - □ 是否对每个需要登录权限的页面都做了完善的检查 ★

        - 是否 OWASP A9:2017 –使用含有已知漏洞的组件
        - □ 是否正确处理了客户端IP获取方法
        - □ 是否有预防 CWE-918: 服务器端请求伪造（SSRF）
    - 网站前端
        - □ 是否有参考了Yahoo!的Best Practices for Speeding Up Your Web Site
    - 页面设计
        - □ 是否已抛弃使用形象首页
        - □ 是否已经不再使用table布局
        - □ 是否采用一些语义化的标签
        - □ 是否有做适合手机和平板浏览的页面
        - □ 是否进行了SEO优化
    - 网站性能
        - □ 是否在上线前对网站性能进行了评估 ★
    - 移动客户端APP
        - □ 是否有通过互联网平台的安全检查
        - □ 是否对代码进行了清理
        - □ 是否对代码进行了混淆
        - □ 是否进行加壳保护
        - □ 是否预防了反编译和二次打包风险
        - □ 是否正确使用了证书对应用进行签名
        - □ 是否防止了Webview File同源策略绕过漏洞
        - □ 是否对传输进行TLS加密 ★
        - □ 是否只请求了必要的系统权限 ★
        - □ 是否将类似accesstoken等保存到客户端
        - □ 是否正确的保护了App的数据库
        - □ 是否对重要信息界面的输入和截屏攻击进行预防
        - □ 是否定期更新打包的框架和第三方SDK
    - 其他
        - □ 是否所有日志都保留6个月并且远程存储 ★

厦门大学网站开发安全最佳实践

本网站开发安全最佳实践整合自多个兄弟高校和互联网的相关文档，涉及服务器、网站开发、美工、移动客户端App等内容。如果需要委托其他软件厂商开发，请把本文档发给对方参考。如果有任何意见或者建议，请发邮件到 vhost # xmu.edu.cn 。也可直接在 https://github.com/haishanzheng/SecurityBestPractices 提交PR。本最佳实践只聚焦于高校内，较为初级的安全问题。

整体建议

目前学校提供托管服务器、托管虚拟机、托管虚拟主机、网站群等服务。

□ 是否申请了虚拟机而不是使用自有机房部署

学校机房有空调、UPS（有消防灭火装置）等保障，虚拟机有备份，建议使用学校的虚拟机纳入统一安全管控并自行定期再做好备份。

□ 是否将新闻性质的网站和信息系统分开

应当将新闻性质的网站和信息系统分开，二者采用不同的安全级别保护。新闻性质应当面向公众访问，应当生成静态提高访问速度。信息系统应当区分不同访问角色，根据访问角色来源限制特定IP、院系内、校内、校外多访问策略。

新闻性质网站应当纳入网站群系统，安全性较高，通过三级等保测评，已实现IPv6、HTTPS、HTTP2等功能。网站群还需要关注管理员强密码和离职编辑人员清理问题。新闻发布之前均有专人审核，禁止公示导致个人信息泄露。

□ 是否已通过等保测评

信息安全等级保护，是对信息和信息载体按照重要性等级分级别进行保护的一种工作，在中国、美国等很多国家都存在的一种信息安全领域的工作。

系统应当根据定级指南要求制定等保三级或二级，并按文件要求做好测评工作。

□ 是否反向代理友好

系统应当可被反向代理而不会出现访问错误，系统应当正确处理反向代理传递的IP信息，处理绝对路径和相对路径，应当根据接入反向代理规范要求做调整。

服务器

□ 服务器是否专有用途

服务器应当只做网页和数据库用途，请不要为了传输文件方便在服务器上安装QQ，Ftp等客户端软件，服务器越简洁，遭到黑客攻击才越容易看出端倪。

□ 是否选择了正确的操作系统

不应当选择已经超过安全更新周期EOL外的操作系统。

PHP+MySQL应部署在Linux服务器下，尽量不要部署在Windows服务器上。Java建议部署在Linux服务器上。ASP和ASP.NET部署在Windows服务器。Linux服务器尽量不要做过多的自定义，软件使用包管理器安装。更高级用户可使用诸如Puppet、Chef、Ansible等配置管理工具配置服务器。安装的操作系统和数据库程序的来源应可信，如果是第三方下载的ISO文件应检验官方的MD5值。

□ 是否启用了安全更新 ★

服务器应保持更新到最新，Windows把自动更新打开。Linux把安全包自动安装打开。Windows建议安装Windows 2008 64位及以上。Linux诸如Ubuntu应安装LTS 64位版本。

□ 是否启用了动态磁盘扩展功能

服务器应启用动态磁盘功能，方便后期根据需求扩展磁盘大小。Windows建议2008以上，使用“动态磁盘”。Linux建议使用LVM。把var和根目录分开。

□ 是否配置并启用了防火墙 ★

服务器应配置并启用防火墙。如果临时排查错误需要暂时关闭防火墙，应在排查完重新打开防火墙。防火墙规则根据需求，一般网站只开放80端口。开放部分IP地址的3389和22端口。应当禁止

MySQL和SQLServer的远程访问功能，关闭他们对应的端口。

- Memcached未授权限制
- Redis未授权限制
- NTP反射攻击

□ 是否做了备份 ★

如果你的数据没有备份，那你可以认为这个数据是不存在的。应当至少有一个备份，这个备份不应当在同一台机器，如果有条件，应把备份做在异地。开源备份工具可以考虑BackupPC、Amanda、Bacula等。

□ 是否使用了域名而不是IP部署

服务器有条件应尽量使用域名访问而不是IP访问。服务器应可支持IPv6。Web应用应只开在80端口。如果一个服务器需要开多个Web应用，应采用基于域名的虚拟主机技术或者采用开虚拟目录的模式而不是采用不同端口的模式。

□ 是否支持IPv6

最新的操作系统已默认支持IPv6，应给服务器配置固定的IPv6地址。程序代码也应支持IPv6，保存访问者的IP地址的字段应能够容纳IPv6地址长度。分析IP地址的代码应能支持IPv6格式。

□ 是否支持HTTPS

HTTPS可以确保网站的真实性，确保内容传输不会被第三方篡改，被第三方窃听。系统应当部署HTTPS，并且不再支持不安全的SSLv2等协议。应当可通过 https://www.ssllabs.com/ssltest/ 测试。

□ 是否支持HTTP2

HTTP2为下一代HTTP协议，有条件应当支持HTTP2。

□ 服务器代码和数据等目录是否清晰

Web应用，应当只分为下面5个角色：

- 程序代码。程序代码应考虑放到诸如SVN，Git等源代码版本控制工具内管理，在服务器备份时，程序代码无需备份，重新部署时只需重新下载程序代码即可。
- 第三方程序。第三方程序包括操作系统光盘，数据库安装文件，应用框架代码，报表控件等安装文件，这些为可下载到的文件，保留是为了部署方便，可不用备份。
- 数据库。
- 数据或者上载文件。上载文件目录建议整个Web应用只有一个，如果可能，请放到Web应用外面，设置为不可执行，使用杀毒工具杀毒。
- 日志。

在目录结构上，建议以上5个角色目录结构清晰。Windows桌面应干净，像处女座一样。可变的数据库和数据或者上载文件应跟操作系统分区分开，以防在操作系统崩溃后该分区还可读。日志文件由于会不断增加，如果没有logrotate机制，应把日志所在的分区独立分出，以免日志空间膨胀占用所有磁盘空间影响整个系统无法操作。

□ 是否使用了强密码

□ 是否使用了监控工具

可使用Nagios、Zabbix、Icinga2、Ganglia、Grafana等。

□ 是否部署了软WAF

Web应用防护系统（也称为：网站应用级入侵防御系统。英文：Web Application Firewall，简称： WAF），免费的有安全狗，360IIS版等。

□ 是否不允许Web服务器列出文件

□ 是否限制了Web服务器除GET|HEAD|POST以外的方法

□ 是否隐藏了Web服务器的Banner信息

□ 是否添加了X-Frame-Options不允许网页被嵌入其他网站

杜绝clickjacking，点击劫持。

□ 是否添加了CSP

□ 是否有使用漏洞扫描工具扫描服务器和网站

可使用学校提供的漏洞扫描工具或者互联网服务对服务器进行扫描。扫描过程中需要对扫描IP开放白名单，并在扫描结束后关闭。

数据库

□ 是否应用了独立用户，最小权限原则 ★

Secure By Default的另一层含义就是"最小权限原则"。最小权限原则也是安全设计的基本原则之一。最小权限原则要求系统只授予主体必要的权限，而不要过度授权，这样能有效地减少系统、网络、应用、数据库出错的机会。比如在Linux系统中，一种良好的操作习惯是使用普通账户登录，在执行需要root权限的操作时，再通过sudo命令完成。这样能最大化地降低一些误操作导致的风险；同时普通账户被盗用后，与root帐户被盗用所导致的后果是完全不同的。
http://book.51cto.com/art/201204/330066.htm

□ 是否限制了Web数据库管理工具的权限

不应当在Web部署服务器上部署数据库管理工具，比如PHPMyAdmin，如果要部署，应当更改默认路径，限制IP访问，设置强密码，剔除无关账户。

□ 是否没有在源代码版本控制系统里面保存数据库等密码

不应当在版本控制系统诸如Git，SVN内保存诸如数据库密码和发送邮件账户的密码等，应当把这些写入配置文件，配置文件不放入版本控制系统，只放置模板配置文件供开发修改使用。

□ 是否正确处理数据库连接串

数据库连接本地应当使用localhost，如果必须远程，应当直接使用IP地址，使用域名有可能因为反向代理原因导致域名和IP地址无法正确对应。

后台程序

□ 是否隐藏了管理员入口

管理员入口不应当在首页显示，应当限制IP访问。

□ 是否有使用版本控制系统管理源代码

应当使用Git等版本控制系统管理源代码，所有密码均不能写到源代码内。

□ 是否有使用框架协助开发

应选择一个框架以减轻开发工作量。框架的选择应是选择社区活跃度最高，更新较频繁，安全的框架。PHP可选择CakePHP，Symfony，Zend。Python可用Django，轻量级可用web.py、Flask。ASP.NET使用微软自带的MVC或者WebForms。JavaScript框架可选择MVC框架包括AngularJs、Ember.js、Backbone.js、Knockout等，展示框架可选择JQuery UI、Bootstrap等。

□ 是否关闭了调试模式 ★

生产环境应当关闭调试信息。

□ 是否有对用户的输入进行检查过滤 ★

代码应不信任用户的任何输入，URL链接、Header、Cookie、表单数据都是可以精心伪造的。

□ 是否有预防注入漏洞 ★

将不受信任的数据作为命令或查询的一部分发送到解析器时，会产生诸如SQL注入、NoSQL注入、OS注入和LDAP注入的注入缺陷。攻击者的恶意数据可以诱使解析器在没有适当授权的情况下执行非预期命令或访问数据。

□ 是否有预防跨站脚本攻击 Cross-site scripting (XSS) ★

应当对所有输出执行HTML编码，或使用内容安全策略（Content Security Policy，CSP）定义。

□ 是否有预防跨站域请求伪造 Cross Site Request Forgery (CSRF)

增加CSRF token。

□ 是否使用了统一身份认证

为了方便师生员工登录系统，应当使用统一身份认证，密码只在统一身份认证网站输入。统一身份认证账户的网站内置的密码应当关闭。

□ 是否对密码进行了加密保存

加密应当加盐再摘要保存。

□ 是否对密码、登录等部分做了防暴力破解检查

应限制密码输入错误的次数，限制单个IP对某个敏感资源请求的次数。

□ 是否对程序文件所在目录设置只读和不可执行权限设置

程序文件所在目录应当设置只读，更新代码后重新设置权限。上载目录设置可写不可执行，以防被上载木马。

□ 是否对下载文件进行了全面的保护 ★

下载文件应放在Web目录以外。下载首先判断权限，再去系统读取需要下载的文件，通过头部输出`content-type`和`content-disposition attachment;filename=`等提供文件名。不应在一个页面判断完权限后直接重定向到真实的文件下载，这样有可能被直接下载或者被迅雷等工具记录文件地址而被盗链。ASP.NET可使用Response.TransmitFile加速下载，Nginx和Apache2可使用x_sendfile。

是否对上载文件实现先审后下。网站用户上传的图片等文件应当在没有审核状态下无法被下载和查看，比如用户更新头像等。

□ 是否使用UTF8格式保存源代码和展示网页

□ 是否考虑了国际化 i18n

□ 是否对每个需要登录权限的页面都做了完善的检查 ★

杜绝平行权限漏洞，越权，提权。

应对每个需要登录的页面在程序最开端检查用户的权限。不应漏掉提交的页面，因为用户有可能直接构造数据提交。

是否 OWASP A9:2017 –使用含有已知漏洞的组件

组件（例如：库、框架和其他软件模块）拥有和应用程序相同的权限。如果应用程序中含有已知漏洞的组件被攻击者利用，可能会造成严重的数据丢失或服务器被接管。

应当经常更新使用的组件。

□ 是否正确处理了客户端IP获取方法

客户端真实IP地址应当正确处理X-Forwarded-For和IPv6。X-Forwarded-For的反向代理必须加白名单。

□ 是否有预防 CWE-918: 服务器端请求伪造（SSRF）

SSRF，Server-Side Request Forgery，服务端请求伪造，是一种由攻击者构造形成由服务器端发起请求的一个漏洞。一般情况下，SSRF 攻击的目标是从外网无法访问的内部系统。漏洞形成的原因大多是因为服务端提供了从其他服务器应用获取数据的功能且没有对目标地址作过滤和限制。

是否提供向别的URL获取信息的功能，导致内网信息泄露。

网站前端

□ 是否有参考了Yahoo!的Best Practices for Speeding Up Your Web Site

http://developer.yahoo.com/performance/rules.html

在HTTP2下面，有些规则已经不再适用，HTTP2本身已有相应加速方法。

页面设计

□ 是否已抛弃使用形象首页

形象首页也叫扉页，不应在网页使用扉页。有些网站有中英文版，不同运营商网络选择，或者访问首页首先是进入一段动画或者Flash，点击后才进入真正的首页。这个页面导致用户访问需要多点击一次，对于网站的SEO也较为不利，应直接进入首页，中英文版通过右上角切换。

□ 是否已经不再使用table布局

table应当只做大量数据展示使用，不应当使用table来布局。应使用Div布局。即使在页面没有CSS的情况下整个网页也可轻松访问。

□ 是否采用一些语义化的标签

可以使用一些语义化的标签。也可给DIV等起上通用的名字，比如新闻标题定义class或者id为news_title，新闻内容定义class或者id为news_content。方便机器抓取。

□ 是否有做适合手机和平板浏览的页面

可使用响应式布局。

□ 是否进行了SEO优化

不应使用flash。站点应当有站点地图，站点地图目录结构清晰。URL美观。不应使用中文文件名。源代码尽量减少非语义化的内容。可在META头部里加入关键字等信息。

网站性能

□ 是否在上线前对网站性能进行了评估 ★

网站性能评估可使用Apache基金会的ab进行，也可采用LoadRunner等商业工具。PHP可用xdebug + webgrind定位性能瓶颈。PHP可安装诸如APC等加速工具。可采用Nginx或者Squid等工具加速网站。

移动客户端APP

移动客户端由于源代码暴露在互联网上，应当对源代码等进行保护，移动客户端调用远程Web服务器的相关漏洞请参考上面，以下不再详细说明，只针对移动客户端特定相关。

□ 是否有通过互联网平台的安全检查

移动App开发完毕应当提交给互联网平台做好病毒扫描等检测，做好安全整体测评。

□ 是否对代码进行了清理

清理包括但不止于：调试信息，开发者个人信息，测试证书等等。

□ 是否对代码进行了混淆

□ 是否进行加壳保护

□ 是否预防了反编译和二次打包风险

□ 是否正确使用了证书对应用进行签名

不应当使用调试证书。应当使用正确的开发者证书或者企业证书。

□ 是否防止了Webview File同源策略绕过漏洞

□ 是否对传输进行TLS加密 ★

应当对整个传输进行TLS加密，并且在App判断远程服务器证书，对服务器身份进行完整性检测，以杜绝中间人攻击。对于远程主机下发的更新请求应当检测完整性。

□ 是否只请求了必要的系统权限 ★

不应当请求App不使用的系统权限。不应当违规记录用户的行为信息。

□ 是否将类似accesstoken等保存到客户端

第三方OAuth类似的accesstoken应当保存在服务器，不应当保存在客户端数据库内。

□ 是否正确的保护了App的数据库

应当防止数据库被恶意备份，任意读写。

□ 是否对重要信息界面的输入和截屏攻击进行预防

□ 是否定期更新打包的框架和第三方SDK

应当定期更新，杜绝CVE漏洞，定期重新发布版本。

其他

□ 是否所有日志都保留6个月并且远程存储 ★

根据最新2016年《中华人民共和国网络安全法》，采取监测、记录网络运行状态、网络安全事件的技术措施，并按照规定留存相关的网络日志不少于六个月；