> 安全工程师的核心竞争力不在于他能拥有多少个0day，掌握多少种安全技术，而在于他对安全理解的深度，以及由此引申的看待安全问题的角度和高度。

一、我的安全世界观

（1）安全问题的本质是信任问题。

（2） 安全3要素(CIA)

1. 机密性(Confidentiality) ~加密
2. 完整性(Integrity) ~签名
3. 可用性(Availability) ~HA

其他：可审计性、不可抵赖性...

（3） 安全评估4个阶段

1. 资产等级划分 ~数据
2. 威胁分析 ~头脑风暴/威胁建模
3. 风险分析
4. 确认解决方案

威胁建模：STRIDE（微软）

威胁	定义	对应的安全属性
Spoofing(伪装)	冒充他人身份	认证
Tampering(篡改)	修改数据或代码	完整性
Repudiation(抵赖)	否认做过的事情	不可抵赖性
InformationDisclosure(信息泄露)	机密信息泄露	机密性
Denial of service(拒绝服务)	拒绝服务	可用性
Elevation of privilege(提升权限)	未经授权获得许可	授权

风险分析，DREAD模型（微软）

等级	高 (3)	中 (2)	低 (1)
Damage Potential	获取完整验证权限；执行管理员权限；非法上传文件	泄露敏感信息	泄露其他信息
Reproducibility	攻击者可以随意攻击	攻击者可重复攻击，但有时间限制	攻击者很难重复攻击过程
Exploitability	初学者在短期内能掌握攻击方法	熟练的攻击者才能完成这次攻击	漏洞利用条件非常苛刻
Affect Users	所有用户，默认配置，关键用户	部分用户，非默认配置	极少用户，匿名用户
Discoverability	漏洞很显眼，攻击条件很容易获得	在私有域，部分人能看到，需深入挖掘	发现该漏洞极其困难

设计安全方案（有针对性）

1. 能有效解决问题
2. 用户体验良好
3. 高性能
4. 低耦合
5. 易于扩展和升级

（4）设计安全方案技巧

技巧			
Secure by default	总则	黑名单、白名单	防火墙、服务器软件、Web...
Defense in deepth	纵深防御	不同层面、不同方面的安全方案相互配合；针对问题根源	Web应用、操作系统、数据库、网络环境安全
数据与代码分离	漏洞成因	注入；缓存区溢出	SQL、XSS、CRLF、X-Path...
不可预测性	克服攻击方法	有效对抗基于篡改、伪造的攻击；加密算法、随机算法、哈希算法...	CSRF~token

安全是一门朴素的学问，也是一门平衡的艺术。

二、浏览器安全

1.同源策略（Same origin policy）

浏览器同源策略：限制来自不同源的"document"或脚本，对当前"document"读取或设置某些属性。

影响"源"的因素：host（域名/IP）、子域名、端口、协议

加载JavaScript页面的域决定JavaScript的origin

```
a.com -> <script src = http://b.com/b.js></script>
b.js的源是a.com
浏览器限制b.js不能读写返回的内容
```

跨域加载资源的标签（可发送第三方cookie）：`<script>`、`<img>`、`<iframe>`、`<link>`...

XMLHttpRequest受同源约束，不能跨域，如果能，则CSRF~token(敏感数据泄露)

W3C XMLHttpRequest跨域标准：通过目标返回的HTTP授权是否允许跨域（前提：JavaScript无法控制该HTTP头）

同源策略限制：DOM、Cookie、XMLHttpRequest ~第三方（各自的）：Flash、Java Applet、Silverlight、Google Gears

如果同源策略被绕过，则基于同源策略的安全方案失效。

2.浏览器沙箱

多进程架构、sandbox

chrome：浏览器进程、渲染进程（sandbox）、插件进程（flash、java、pdf）、拓展进程... | 浏览器进程和插件进程严格隔离

插件安全：浏览器加载的第三方插件不受sandbox管辖

3.恶意网址拦截

网址黑名单（挂马、钓鱼） <- Google、微软、PhishTank...

EV SSL证书（Extend Validation SSL Certification）

4.高速发展的浏览器安全

XSS Filter
CSP（Content Security Policy）
扩展和插件的权限高于页面的JavaScript权限

三、跨站脚本攻击（XSS）

1.XSS简介

网页中插入恶意脚本，在用户浏览网页时，控制其浏览器

针对不同场景产生的XSS，要区分情景对待

1. 反射型XSS（非持久型XSS） 需诱使用户"点击"链接
2. 存储型XSS（持久型XSS） "存储"在服务端
3. DOM Based XSS 通过修改页面的DOM节点形成XSS

2.XSS攻击进阶

（1）XSS payload

JavaScript脚本、flash、其他富客户端脚本
读取浏览器cookie（加密保存用户登录凭证...）->cookie劫持

（2）强大的XSS payload

1. 构造GET与POST请求
2. XSS钓鱼
3. 识别用户浏览器（UA/浏览器不同版本差异）
4. 识别用户安装的软件/浏览器扩展、插件
   IE：ActiveX classid
   Flash： system.capabilities
   Firefox：
   插件（plugins）：navigator.plugins
   扩展（Extantion）：检测扩展图标（chrome://协议）
5. CSS history hack，style~visited属性，link访问过，颜色改变（文档中超链接类似）
6. 获取用户真实IP

（3）XSS攻击平台

AttackAPI、BeEF、XSS-proxy

（4）XSS worm

samy worm ~ mysapce
百度空间worm

（5）调试JavaScript

Firebug、IE Developer tools、Fiddler、HttpWatch

（7）XSS构造技巧

1. 利用字符编码
   GBK/GB2312 -> Unicode...
2. 绕过长度限制

事件（Event）、location.hash、远程JavaScript、注释...

3. 使用`<base>`标签
   `<base>`：定义页面上所有使用"相对路径"标签的hosting地址，可出现在页面任何地方，并作用于位于该标签后的所有标签
   通过`<base>`劫持当前页面中所有使用"相对路径"的标签
4. windows.name的妙用
   window对象是浏览器窗体，并非document对象，不受同源策略限制，实现跨域、跨页面传递数据

其他

Apache Expect Header XSS
Anehta的回旋镖
Flash XSS
JavaScript开发框架（Dojo、YUI、jQuery(html())）

（8）XSS防御

1.HttpOnly

禁止页面的JavaScript访问带有httponly属性的cookie

2.输入检查

格式：用户名、电话、邮箱、生日...（白名单）
输入检查逻辑必须放在服务器端代码实现
如果放在客户端，则容易被绕过，但可以阻挡大部分误操作的正常用户，节约服务器资源

XSS Filter（对语境理解不完整）：`< > ' " <scrip> <javascript>...`，过滤/编码

3.输出检查

除富文本输出外，变量输出到HTML，编码、转义
安全编码函数：
HTML->HtmlEncode(OWASP ESAPI)
JavaScript->JavascriptEncode(OWASP ESAPI)
XMLEncode
JSONEncode

HtmlEncode将字符转换成HTMLEntities

```
& --> &amp
< --> &lt
> --> &gt
" --> &quot
' --> &#x27
/ --> &#x2F
```

正确的地方使用正确的编码方式

4.正确防御XSS

XSS本质："HTML"注入，用户的数据被当成HTML代码的一部分执行，从而混淆原本语义，产生新语义

变量

1. 在HTML标签中输出：HTMLEncode
2. 在HTML属性中输出：HTMLEncode
3. 在`<script>`标签中输出：JavascriptEncode
4. 在事件中输出：JavascriptEncode
5. 在CSS中输出：
   尽可能禁止用户可控制的变量在"`<style>`标签"、"HTML标签的style属性"及"CSS文件"中输出
   如果需要输出，使用OWASP ESAPI的encodeForCSS()函数
6. 在地址中输出：URLEncode（OWASP ESAPI URLEncode）
   如果变量是整个URL，检查其是否以"http"开头（不是则自动添加），避免伪协议类 XSS

5.处理富文本（Anti-Samy / HTMLPurity）

禁止："事件"、`<iframe>`、`<script>`、`<base>`、`<form>`，用户自定义CSS与style（如果需要，过滤CSS）
标签使用白名单：`<a>`、`<img>`、`<div>`

6.防御DOM Based XSS

从JavaScript输出到HTML页面，相当于一次XSS的输出过程，需分语境使用不同的编码函数
HTML -> 事件/脚本：JavascriptEncode
HTML -> 内容/属性：HtmlEncode

四、跨站点请求伪造（CSRF）

1.CSRF简介

Cross Site Request Forgery
XSS利用站点内的信任用户
CSRF通过伪装来自受信任用户的请求来利用受信任的网站

2.CSRF进阶

（1）浏览器的Cookie策略

浏览器Cookie：
Session Cookie(临时Cookie)：浏览器关闭，Cookie失效
Third-party Cookie(本地Cookie)：服务器在Set-Cookie制定Expire时间

拦截Third-party Cookie的浏览器：IE6、IE7、IE8、safari
不拦截Third-party Cookie：Firefox3、Firefox3、Opera、Chrome、Android...

若CSRF不需要Cookie，则可忽略浏览器的Cookie策略

（2）P3P头的副作用

The Platform for Privacy Preference
网站返回给浏览器的HTTP头中包含有P3P头，则允许浏览器发送第三方Cookie

P3P头主要用于类似广告等需要跨域的页面，但P3P头设置后，对于Cookie的影响将扩大到整个域中的所有页面

P3P策略：~W3C标准

CSRF防御不能依赖于浏览器对第三方Cookie的拦截策略

（3）GET、POST

CSRF不仅能由GET请求发起，也能由POST请求发起

（4）Flash CSRF

POST、URLRequest、getURL、loadVars

（5）CSRF worm

百度用户中心 CSRF worm

3.CSRF的防御

（1）验证码

（2）Referer Check

用于检查请求是否来自合法的"源"

（3）Anti CSRF Token（保密性、随机性）

1.CSRF的本质

重要操作的所有参数都是可以被攻击者猜测到的

增加token参数
保存位置：

- 用户Session
- 浏览器Cookie：考虑生成多个有效的token，解决多页面共存场景

2.Token的使用原则

Token生成足够随机
在一个用户有效生命周期，Token消耗前使用同一个，已经消耗重新生成
尽量把Token放在表单中，敏感操作由GET改为POST，以form表单（或AJAX）形式提交，避免Token泄露
其他泄露途径：XSS/跨域漏洞

XSRF
当网站同时存在XSS，攻击者可以请求页面后，读取页面内容里的Token，再构造一个合法请求，成XSRF，此时CSRF的Token方案无效

五、点击劫持（ClickJacking）

1.点击劫持

视觉上的欺骗手段，使用一个透明、不可见的iframe，覆盖在一个网页上，诱使用户在该网页上操作，用户点击iframe，通过调整iframe页面的位置，诱使用户恰好点在iframe页面的一些功能性按钮上

2.Flash点击劫持

Flash游戏

3.图片覆盖攻击

Cross Site Image Overlaying XSIO
利用图片的style或控制CSS，图片覆盖在页面上任意位置

4.拖曳劫持与数据窃取

浏览器支持Drag & Drop 的API
浏览器的拖曳对象可以是一个链接，一段文字，可以从一个窗口到另一个窗口，不受同源策略的限制
诱使用户从隐藏不可见的iframe中"拖曳"出想要的数据，然后放在能控制的另一个页面，从而窃取数据

5.ClickJacking 3.0：触屏劫持

TapJacking

6.防御ClickJacking

禁止跨域的iframe

（1）frame busting（可被绕过）

写一段JavaScript，禁止ifr嵌套

限制iframe页面中的JavaScript执行(绕过)：
HTML5中iframe的sandbox属性
IE中iframe的security属性

（2）X-Frame-Options

3个可选值：DENY、SAMEORIGIN、ALLOW-FROM origin

六、HTML 5安全

1.HTML 5新标签

1. XSS Filter中如果不包括HTML 5的新标签，则可能XSS
2. iframe的sandbox：
   allow-same-origin：允许同源访问
   allow-top-navigation：允许访问顶层窗口
   allow-forms：允许提交表单
   allow-scripts：允许执行脚本
3. Link Types：noreferer
   `<a>、<area>`
4. Canvas
   `<canvas>`让JavaScript可以在页面中直接操作图片验证码，也可以操作像素，构造图片区域 ~验证码识别

2.其他安全问题

1. Cross-Origin Resource Sharing
   Origin Header，可以防范CSRF
2. postMessage——跨窗口传递消息（不受同源策略限制）
   允许每一个window（包括当前窗口、弹出窗口、iframe等）对象往其他的窗口发送文本消息，从而实现跨窗口的消息传递
   两个安全问题：

- 必要时，在接收窗口验证Domain，甚至URL，以防止来自非法页面的消息
- 如果接收的消息写入innerHTML，甚至scrip中，则可能导致DOM Based XSS

3. Web Storage（Key-value对，类似非关系数据库）
   分为：

- Session Storage：关闭浏览器失效
- Local Storage：一直存在
  受同源策略约束，每个域所拥有的信息只会保存在自己的域下

七、注入攻击

注入的本质：把用户的数据当代码执行
两个关键条件：

1. 用户能够控制输入
2. 原本程序要执行的代码，拼接了用户输入的数据

1.SQL注入

（1）盲注（Blind Injection）

构造简单条件语句，根据返回页面是否变化，判断SQL语句是否得到执行 and 1 = 1 , and 1 = 2

（2）时序攻击（Timing Attack）

边信道攻击的一种
例子里：利用BENCHMARK()函数，让同一个函数执行若干次，使结果返回的时间比平时要长，通过时间长短，判断注入语句是否执行

MySQL	BENCHMARK(1000000,md5(1)) or SLEEP(5)
PostgreSQL	PG_SLEEP(5) or GENERATE_SERIES(1,10000000)
MS SQL Server	WAITFOR DELAY '0:0:5'

2.数据库攻击技巧

（1）导出Webshell

MySQL（当前数据库用户具有读写相应系统文件或目录权限）

- LOAD_FILE() 读取系统文件
- INTO DUMPFILE 写入本地文件（适用于二进制，将目标文件写入同一行）
- INTO OUTFILE（适用于文本文件）

（2）命令执行

利用用户自定义函数UDF（User-Defined Function）执行命令
lib_mysqludf_sys.so 上传到数据库能访问到的路径下，创建UDF

- sys_eval，执行任意命令，并将输出返回
- sys_exec，执行任意命令，并将退出码返回
- sys_get，获取一个环境变量
- sys_set，创建或修改一个环境变量

（3）攻击存储过程（存储过程本身可能存在注入漏洞）

存储过程使用CALL、EXECUTE执行

MS SQL Server
xp_cmdshell：

- 2000 默认开启，如果禁止可使用sp_addextendproc开启
- 2005/2008 默认禁止，sysadmin权限，使用sp_configure开启

操作注册表：
xp_regread
xp_regaddmultistring
xp_regdeletekey
xp_regnumkeys
xp_regremovemultistring
xp_regwrite

xp_servicecontrol 允许用户启动，停止服务
xp_availablemedia 显示机器上有用的驱动器
xp_dirtree 允许获得一个目录树
xp_enumdsn 列举服务器上的ODBC数据源
xp_loginconfig 获取服务器安全信息
xp_makecab 允许用户在服务器上创建一个压缩文件
xp_ntsec_enumdomains 列举服务器可以进入的域
xp_terminate_process 提供进程的ID，终止此进程

（4）编码问题

“基于字符集”注入...
统一数据库、操作系统、Web应用字符集 ~UTF-8

（5）SQL Column Truncation

MySQL的sql-mode设置为default，即没有开启STRICT_ALL_TABLES，MySQL对用户插入的超长值只会提示warming，而不是error（error插入不成功），导致发生“截断”问题

3.防御SQL注入

找到所有的SQL注入->修补之（废话）

（1）使用预编译语句，绑定变量

（2）使用存储过程

需先将SQL语句定义数据库中
避免存储过程使用动态SQL语句（如无法避免，应严格输入过滤、使用编码函数）

（3）检查数据类型

检查输入数据的数据类型
数据格式/类型检查：邮箱、时间、日期...

（4）使用安全函数

编码函数
OWASP ESAPI encodeForSQL

（5）数据库自身角度

最小权限原则，避免Web应用直接使用root、dba
同一数据库不同应用，每个应用分配不同账户
Web应用使用的数据库账户，不应有创建自定义函数、操作本地文件的权限

4.其他注入攻击

1. XML注入，与HTML类似
2. 代码注入

- eval() PHP，JSP的动态include（文件包含漏洞）导致代码执行
- 命令注入 system()

3. CRLF注入（HTTP头可以看做kv对，value中编码所有 \r\n）
   \r\n -> 0x0d，0x0a
   凡是使用CRLF作为分隔符的地方都可能存在
   log注入，HTTP头注入（Http Response Spliting）

八、文件上传漏洞

1.概述

用户上传一个可执行的脚本文件，通过此脚本文件获得执行服务器端命令的能力 （解析+访问）
文件上传后导致的安全问题：

1. 代码执行：上传Web脚本语言，服务器的Web容器解释执行
2. 控制策略文件：flash的crossdomain.xml，控制flash在该域下的行为...
3. 病毒、木马
4. 钓鱼欺诈
5. 溢出服务器后台处理程序，如图片解析模块
6. 文件包含：文件包含PHP脚本，利用LFI执行

（1）FCKEditor

黑名单限制上传文件类型

（2）绕过文件上传检查功能

- 文件名：%00、0x00截断 ~x.php[\0].jpg
- 文件头：伪造一个合法的文件头，将PHP附在文件头后（需要解析）

2.Apache、IIS、PHP CGI

1. Apache 1.x/2.x，对于文件名从后往前，文件类型定义在mime.types中
2. IIS 6
   ; 截断 ~a.asp;b.jpg
   ../x.asp/下的所有文件都当做asp文件解析

- Web DAV，IIS PUT
  目录可写，开启WebDAV，PUT，MOVE
  OPTIONS -> PUT -> MOVE

3. PHP CGI
   PHP 5.1.2, PHP 5.3.1
   cgi.fix_pathinfo = 1
   http://www.test.com/path/tets.jpg/xxx.php (xxx.php不存在)
   解析test.jpg

3.设计安全的文件上传功能（结合业务需求）

1. 文件上传目录设置为不可执行
2. 判断文件类型
   MIME，后缀，白名单
   图片，压缩函数/resize函数
3. 使用随机函数改写文件名或路径
4. 单独设置文件服务器的域名

九、认证与会话管理

1.Who am I

认证 Authentication，识别用户身份
授权 Authorization，决定用户权限

2.密码

1. 对抗暴力破解

- 长度：6+（普通）/8+（重要），且双因素认证
- 复杂度：区分大小写，大小写、数字、字符两种以上组合，避免连续、重复字符

2. 防止包含用户隐私

- 不要使用用户的公开数据
- 不要使用与个人隐私相关的数据（QQ、身份证、昵称、电话（手机）、生日、英文名、公司名...）

3. 保存

- 加密
- 散列

3.多因素认证

支付宝：支付密码，手机动态口令，数字证书...

4.Session与认证（会话劫持，Session hijacking）

密码与证书等用于登录认证，认证成功后，用SessionID替换
SessionID保存

- Cookie 加密保存
- URL（不靠谱）

Session劫持：SessionID在生命周期内被窃取，使用该SessionID登进目标账户，如果SessionID保存在Cookie中，则可以称为Cookie劫持

Session泄露途径：XSS、网络嗅探、本地木马窃取

会话劫持防御：

- Web层

1. 更改Session名称
2. 关闭透明化SessionID
3. Httponly
4. 关闭所有phpinfo类的dump request信息页面
5. 使用User-Agent检测请求的一致性
6. 加入Token校验

- 网络层

1. telnet , rlogin -> openSSH or SSH
2. FTP -> SFTP
3. HTTP -> SSL
4. IP -> IPSec
5. 任何远程连接 -> VPN
6. Hub -> Switch

5.会话固定（Session fixation）

（1）概述

用户登录网站过程中，登录前后SessionID没有改变，则称为会话固定，诱骗用户使用攻击者指定的SessionID

（2）步骤

1. 攻击者重置目标用户的SessionID，监听会话状态
2. 目标用户携带攻击者设定的SessionID登录
3. 攻击者通过SessionID获得合法会话

（3）重置SessionID的方法

1. 使用客户端脚本来设置Cookie到浏览器 ~Httponly
2. 使用HTML的 `<META>` 标签加Set-Cookie属性
3. 使用Set-Cookie的HTTP响应头设置Cookie

（4）防御

1. 用户登录时生成新的SessionID
2. Httponly，关闭透明化SessionID，UA验证,Token校验

6.Session保持攻击

（1）Session生命周期：

- 用户长时间未活动
- 用户点击退出
  服务器将销毁Session

（2）方法

刷新页面/修改Cookie中的Expire时间，以保持Session不过期，使Session变成Third-party Cookie

（3）对抗

1. 规定有效期，过期强制销毁 ~影响用户体验
2. 当用户客户端发生变化，要求重新登录（IP、UA...）
3. 每个用户只允许拥有一个Session

7.单点登录（SSO）

Single Sign On，登录一次，访问所有系统
风险集中化，考虑双因素认证，OpenID

十、访问控制

设计方案应满足“最小权限原则”

1.What can I do

权限/访问控制：
某个主体（Subject）对某个客体（Object）需要实施某种操作（Operation），而系统对这种操作的限制就是权限控制

ACL，访问控制列表，主体、客体、操作之间的关系构成ACL

- 网络中：防火墙ACL
- 操作系统中：文件ACL
- Web应用中：
  基于URL的访问控制
  基于方法（method）的访问控制
  基于数据的访问控制（水平）

2.垂直权限管理

基于角色的访问控制（Role-Based Access Control） RBAC

RBAC定义不同角色，不同角色拥有不同权限，一个用户可拥有多个角色，角色有高低之分（权限高低）

Spring security（认证、授权...）
基于URL、方法的访问控制
基于表达式的访问控制

PHP-Zend Framew

高权限角色访问低权限角色的资源，允许
如果低能访问高，则发生“越权访问”

3.水平权限管理

基于数据的访问控制

表现：同一角色的用户，A可以访问B的数据

解决方案参考：用户组（Group）、规则引擎
没有特别完美的解决方案

4.OAuth简介

OpenID解决认证
OAuth解决授权
三个角色:

消费方	服务提供方	用户

1.0 consumer	server provider	user
client	server	resource owner

十一、加密算法与随机数

1.加密算法

（1）分组加密算法

DES、3-DES、Blowfish、IDEA、AES...

（2）加密模式

ECB、CBC、CFB、OFB、CTR

（3）流加密算法

RC4、ORXY、SEAL

（4）针对加密算法的攻击

1. 唯密文攻击
2. 已知明文攻击
3. 选择明文攻击
4. 选择密文攻击（CBC模式的Padding Oracle Attack）

2.针对流加密攻击

1. Reuse Key Attack（使用同一个秘钥进行多次加/解密）
2. Bit-fipping attack ~加MAC（消息验证码）
3. 弱随机IV问题

3.WEP破解

WEP使用RC4加密：

- 初始向量IV
- CRC-32检验

Aircrack

4.秘钥管理

避免秘钥硬编码
Web常见做法：
将秘钥（包括密码）保存在配置文件或数据库中

5.伪随机问题

避免弱伪随机算法
时间函数不等于随机数
Seed

6.实践与建议

（1）最佳实践

1. 不要使用ECB模式
2. 不要使用流密码（如RC4）
3. 使用HMAC-SHA1替代MD5（甚至是SHA1）
4. 不要使用相同的key做不同的事
5. salts和IV需要随机产生
6. 不要自己实现加密算法，尽量使用安全专家实现好的库
7. 不要依赖系统的保密性

（2）建议

1. 使用CBC模式的AES256用于加密
2. 使用HMAC-SHA512用于完整性检查
3. 使用带salt的SHA-256或SHA-512用于hashing

十二、Web框架安全

1.MVC框架安全

MVC（Model View Controller）

- View 用户视图，页面展示
- Controller 应用的逻辑实现，接受View层传入的用户请求，并转发给对应的Model做处理
- Model 实现模型，数据处理

MVC可以解决的安全威胁（不涉及业务逻辑）
XSS、CSRF、SQL注入、访问控制、认证、URL跳转

2.模本引擎与XSS防御

XSS发生在view层，输出编码，针对不同上下文的XSS，使用不同的编码方式
Django templates使用filter中的escape作为HTMLENcode的方法
Velocity

3.Web框架与CSRF防御

Web应用开发中区分“读操作”和“写操作”（写用POST）
Security token的私密性（不可预测性原则），是防御CSRF的基础
对于Web框架，自动在涉及POST的代码中添加token，包括：所有的form表单，所有的Ajax POST请求
Rails , Django

防御CSRF，Web框架改动

1. 在Session中绑定token，如果不能保存到服务端session中，可以替代为保存到cookie里
2. 在form表单中自动填入token字段
3. 在Ajax请求中自动添加token
4. 在服务器端对比POST提交参数的token与session中绑定的token是否一致，以验证CSRF攻击

4.HTTP Headers管理

管理跳转目的地址：

1. 如果web框架提供统一的跳转函数，跳转函数内使用白名单
2. 控制HTTP的Location字段，~白名单

5.数据持久层与SQL注入

使用对象关系映射ORM（Object/Relational Mapping）框架对抗SQL注入

6.Web框架自身安全

Struts 2（CVE-2010-1870）
Spring MVC（CVE-2010-1622）
Django 0.95

十三、拒绝服务攻击

CC（Challenge Collapasar）、ReDoS（正则表达式DoS）

1.分类

（1）攻击网络宽带资源

1. 直接

- ICMP/IGMP洪水攻击
- UDP洪水攻击

2. 反射和放大

- ACK反射攻击
- DNS放大攻击
- NTP放大攻击
- SNMP放大攻击

3. 攻击链路

- Coremelt攻击

（2）攻击系统资源

1. 攻击TCP连接

- TCP连接洪水攻击
- SYN洪水攻击
- PSH+ACK洪水攻击
- RST洪水攻击
- Sockstress攻击

2. 攻击SSL连接

- THC SSL DoS攻击
- SSL洪水攻击

（3）攻击应用资源

1. 攻击DNS服务

- DNS QUERY洪水攻击
- DNS NXDOMAIN洪水攻击

2. 攻击Web服务

- HTTP洪水攻击
- Slowloris攻击
- 慢速POST请求攻击
- 数据处理过程攻击

（4）混合攻击

攻击分类	洪水	慢速
网络层攻击	ICMP/IGMP	
传输层攻击	UDP、TCP连接、SYN、PSH+ACK、ACK反射攻击、RST、SSL	Sockstress、THC SSL DoS
应用层攻击	DNS QUERY、 DNS NXDOMAIN、DNS放大攻击、HTTP、SNMP放大攻击、NTP放大攻击	Slowloris、慢速POST请求、数据处理过程

（5）一些防御措施

1. 应用代码做好性能优化 ~memcache
2. 网络架构优化（CDN、负载均衡、反向代理、集群、缓存、HA）
3. 限制每个IP的请求频率
4. 验证码
5. Yahoo（Detecting system abuse）

十四、PHP安全

1.文件包含漏洞（File Inclusion）

（1）导致文件包含的函数

1. PHP
   include()、include_once()、require()、require_once()、fopen()、readfile()...
2. JSP/Servlet
   ava.io.File()、java.io.FileReader()...
3. ASP
   include file、include virtual...

（2）成功利用FI的两个条件

1. Include()等函数通过动态变量的方式引入需要包含的文件
2. 用户能够控制该动态变量

（3）本地文件包含（Local File Inclusion） LFI

1. 字符串截断（%00） ~web中禁用0字节
2. 操作系统对目录最大长度限制（windows：256字节；Linux：4096字节）

- "./."目录遍历（Path Traversal）
- 不同编码方式绕过服务端逻辑
  %2e -> .
  %2f -> /
  %5c -> \
  以上组合 -> ../
- 某些Web容器支持的编码
  ..%c0%af -> ../
  ..%c1%9c -> ..\
- CVE-2008-2938

3. open_basedir（与safe_mode是否开启无关）

- /home: test、test1、test2
  open_basedir = home/test/ -> test
  open_basedir = home/test -> test、test1、test2
- 多个目录

Windows：分号隔开
Linux：冒号隔开

（4）远程文件包含（Remote File Inclusion）RFI

PHP allow_url_include=ON，则include/require函数可以加载远程文件

（5）LFI利用技巧

1. 包含用户上传的文件
2. 包含data://或php://input等伪协议
3. 包含session文件
4. 包含日志文件，如web server的access log ~（Metasploit）
5. 包含/proc/self/environ文件 ~（UA注入PHP）
6. 包含其他应用创建的文件，如数据库文件、缓存文件、应用日志...

2.变量覆盖漏洞

1. 全局变量覆盖
   变量未初始化，且用户能控制，register_globals=ON更严重
   导致XSS、SQL注入、代码执行..
2. extract()变量覆盖
   extract()能将变量从数组导入当前的符号表
   第二个参数
   EXTR_OVERWRITE （覆盖）
   EXTR_SKIP （跳过）
3. 遍历初始化变量
   `$$k`
4. import_request_variables()变量覆盖
5. parse_str()变量覆盖
   解析URL的query string，mb_parse_str()类似
6. 安全建议

- 确保register_globals = OFF，若不能自定义php.ini，则应在代码中控制
- 熟悉可能造成变量覆盖的函数和方法，检查用户是否能控制变量的来源
- 养成初始化变量的习惯

3.代码执行漏洞

（1）“危险函数”执行代码

直接执行系统命令的函数
popen()、system()、passthru()、exec()

PHP 3.4.3.1（CVE-2011-2505）
MyBB 1.4

挖掘漏洞的过程，通常需要先找到危险函数，然后回溯函数的调用过程，最后看在整个调用过程中用户是否有可能控制输入

（2）“文件写入”执行代码

本地文件写入：file_put_contents()、fwrite()、fputs()

（3）其他执行代码方式

1. 执行执行代码的函数
   eval()、assert()、system()、exec()、shell_exec()、passthru()、escapeshellcmd()、pcntl_exec()
2. 本地包含
   include()、include_once()
   require()、require_once()

3. 本地文件写入
   file_put_contents()、fwrite()、fputs()
   审计时，注意组合类：写入文件、文件包含、危险代码执行
4. preg_replace()代码执行
   第一个参数存在/e模式修饰符，允许代码执行
   如果没有/e，若第一个参数包含变量，用户可控，有可能通过注入/e%00，注入/e
5. 动态函数执行
   用户自定义的动态函数
   $GET create_function()
6. Curly Syntax
7. 回调函数执行代码
   ...太多-_-
   ob_start()
8. unserialize()导致代码执行
   两个条件
   unserialize()参数用户可以控制
   存在_destruct()或_wakeup()

4.定制安全的PHP环境

（1）php.ini

register_globals = OFF
open_basedir = /home/web/html/
allow_url_include = Off
aloww_url_fopen = Off
display_errors = Off（错误回显，用于开发模式）
log_errors = On
magic_quotes_gpc = 0
cgi.fix_pathinfo = 0
session.cookie_httponly = 1
session.cookie_secure = 1

（2）safe_mode

- 共享环境：（如APP Engine）开启，与disable_functions配合
- 单独的应用环境：关闭，依赖disable_functions 开启safe_mode影响的函数...（太多 -_-）

开启safe_mode，exec()、system()、passthru()、popen()等并非禁用，而是只能在"safe_mode_exec_dir"指定目录下可执行文件，如果需要允许这些函数，设置safe_mode_exec_dir的值并设置目录不可写

（3）disable_functions

- 独立web应用，禁用：...（太多）
- 共享环境，参考新浪的SAE：...（太多）
- 禁用类

十五、Web server配置安全

1. Apache

- 减少不必要的module，对于使用的module，检查是否有漏洞
- 单独的用户、用户组
- access log发送到远程服务器

2. Nignx
3. jBoss
   JMX_Console，通过DevelopmentScanner远程加载一个war包
4. Tomcat Tomcat Manager，~war （配置文件中定义Manager的权限）
5. HTTP Parameter Pollution

- HPP通过GET/POST向服务器发送请求时，提交两个相同的参数
- 绕过服务器端逻辑判断

十六、互联网业务安全

1. 安全是产品的一个特性
2. 业务逻辑安全
3. 密码取回
4. 垃圾注册（~机器识别）

- 分析垃圾行为：
  内容：自然语言分析，关键词匹配
  行为：业务逻辑规则
  客户端识别：人机识别，让客户端解析JavaScript

5. 网络钓鱼（金融）

- 网站、邮件
- 识别发件人邮箱
  SPF（Sender Policy Framework）
  Yahoo的DomainKeys
  微软的SenderID
- 钓鱼网站的防控
  控制传播途径
  关停
  用户教育
  自动化识别

6. 用户隐私保护

- 限制数据使用（PCI-DSS）支付卡行业数据与安全标准
- 保护
  用户应有知情权和选择权
  网站应妥善保管收集到的用户数据，不得用于规定范围之外
- Do-Not-Track

十七、安全开发流程（SDL）

1.SDL

Secure Development Lifecycle，安全开发生命周期，微软
Secure at the source

十六个步骤（优化后） ~瀑布法开发

1. 培训

- 对象：开发人员、测试人员、项目经理、产品经理
- 知识：安全设计、威胁建模、安全编码、安全测试、隐私等

2. 安全要求
3. 质量门/bug栏
4. 安全和隐私风险评估

- SRA PRA
- 威胁模式、安全设计评析、渗透测试、模糊测试范围、隐私评级

5. 设计要求
6. 减小攻击面
7. 威胁建模

8. 使用指定工具（编译器、链接器）
9. 弃用不安全函数/API
10. 静态分析
11. 动态程序分析
12. 模糊测试（Fuzzing Test）
13. 威胁模型和攻击面评析
14. 事件响应计划
15. 最终安全评析（FSR）
16. 发布/存档

SDL -> 软件开发商
SAMM -> 自主软件开发者（OWASP ）

2.敏捷SDL

3.SDL实战经验

六条准则：

1. 与项目经理进行充分沟通，排出足够的时间
2. 规范公司的立项流程，确保所有项目都能通知到安全部门
3. 树立安全部门的权威，项目必须由安全部门审核完成后才能发布
4. 将技术方案写入开发、测试的工作手册
5. 给工程师培训安全方案
6. 记录所有的安全bug，激励程序员编写安全的代码

4.需求分析与设计阶段

论证项目的目标、可行性、实现方向等，~checklist

5.开发阶段

1. 提供安全的函数（OWASP ESAPI、微软）
2. 将安全方案写入开发规范中，就真正将安全方案落地
3. 代码审计工具

- 常见：BOON、Bugscam、Bugscan、CodeAssure、CodeSonar、CodeSpy、CovertyPrevent、Cqual、DevPartner SecurityChecker、flawfinder、Fortify Tools、inForce、its4、MOPS、PrexisEngine、Pscan、RATS、smatch、slint

4. 甲方可以根据开发规范来定制代码审计工具，检查开发者是否遵守了开发规范，而不是代码是否安全

6.测试阶段

自动、手动
Web安全扫描器：
Appscan、WVS、w3af、skipfish、openvas、arachni...

十八、安全运营

Find and fix, Defend and defer, Secure at the source.

1. 漏洞修补流程

- 建立类似bugtracker的漏洞跟踪机制，并为漏洞紧急程度选择优先级
- 建立漏洞分析机制，并与程序员一起制定修补方案，同时review补丁的代码实现
- 对曾经出现过漏洞进行归档，并定期统计漏洞修补状况

2. 安全监控
   Nagios....
3. 入侵检测

IDS/IPS，WAF
ModSecurity，PHPIDS

4. 紧急响应

- 报警：邮件、IM、短信
- 紧急响应组：技术负责人、产品负责人、最了解技术架构的资深开发工程师、资深网络工程师、资深运维系统工程师、资深DBA、资深安全专家、监控工程师、公司公关
- 保护安全事件现场（下线机器，分析入侵行为、损失...）
- 以最快速度处理问题