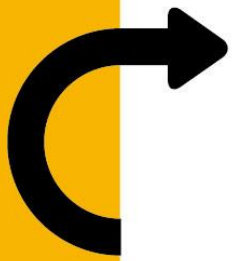
WEB安全测试基础

（PDF测试岗位课程）

（适用于L1，L2）

苏宁易购

suning.com



第一部分 WEB安全测试概念

第二部分 WEB安全测试技术

第三部分 业务逻辑安全测试技术

WEB安全测试概念

安全领域主要指的是信息安全，信息安全的定义是：为数据处理系统建立和采取的技术和管理的安全保护。保护计算机硬件、软件、数据不因偶然的或恶意的原因而受到破坏、更改、泄露【ISO27000】。

软件安全领域有有很多分支，且相互交叉不好界定，大致可分为企业安全、网络安全、数据安全、WEB（应用）安全等方面，我们主要研究的是**WEB安全测试技术。**

WEB安全测试，指的是通过运用一些恰当的测试方法，去针对WEB应用中的物理部署、逻辑部署、容器服务器、应用服务器、数据库、客户端与服务端的交互，以及产品的流程和功能逻辑等进行基于安全考虑的测试。

目前，越来越多的企业开始重视安全测试，然而安全测试因普及度不足，面临着一些困境：

1.传统的软件测试理论很难适用于安全测试领域；

2.安全测试基础理论薄弱,当前测试方法缺少理论指导，也缺乏规范、技术、产品和工具。

安全测试与功能测试的区别

安全测试与功能测试区别：

- 目标不同：功能测试以发现BUG为目标，安全测试以发现安全隐患为目标。
- 假设条件不同：功能测试假设导致问题的数据是用户不小心造成的，接口一般只考虑用户界面。安全测试假设导致问题的数据是攻击者处心积虑构造的，需要考虑所有可能的攻击途径。
- 思考域不同：功能测试以系统所具有的功能为思考域。安全测试的思考域不但包括系统的功能，还有系统的机制、外部环境、应用与数据自身安全风险与安全属性等。
- 问题发现模式不同：功能测试以违反功能定义为判断依据。安全测试以违反权限与能力的约束为判断依据。

安全测试与渗透测试的区别

安全测试与渗透测试区别，安全测试只关注漏洞的可利用性分析，但不关注漏洞如何被真实利用的技术：

- 出发点差异：渗透测试是以成功入侵系统，证明系统存在安全问题为出发点；而安全测试则是以发现系统所有可能的安全隐患为出发点。
- 视角差异：渗透测试是以攻击者的角度来看待和思考问题，安全测试则是站在防护者角度思考问题，尽量发现所有可能被攻击者利用的安全隐患，并指导其进行修复。
- 覆盖性差异：渗透测试只选取几个点作为测试的目标，而安全测试是在分析系统架构并找出系统所有可能的攻击界面后进行的具有完备性的测试。
- 成本差异：安全测试需要对系统的功能、系统所采用的技术以及系统的架构等进行分析，所以较渗透测试需要投入更多的时间和人力。
- 解决方案差异：渗透测试无法提供有针对性的解决方案；而安全测试会站在开发者的角度分析问题的成因，提供更有效的解决方案。

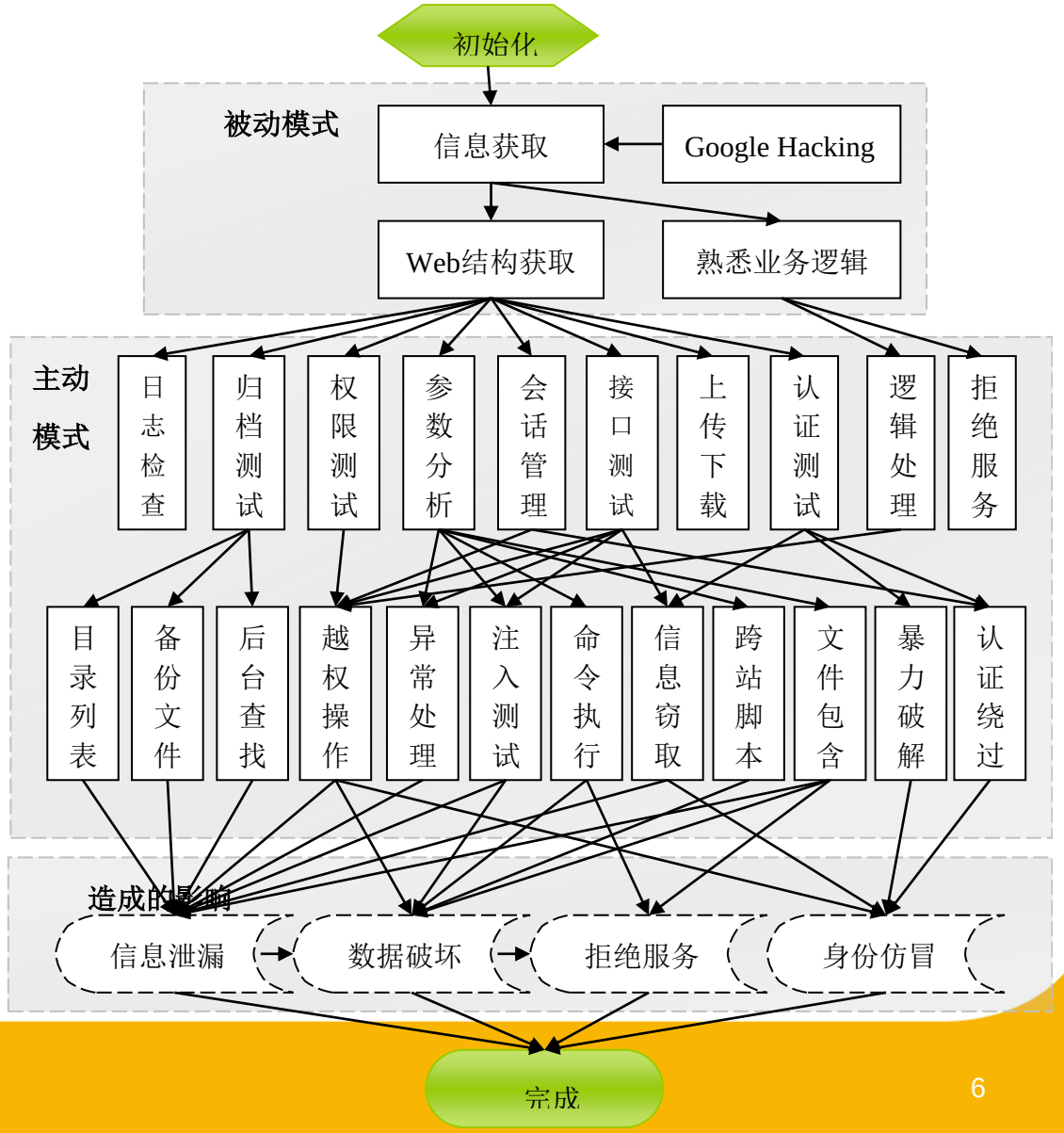
注意事项

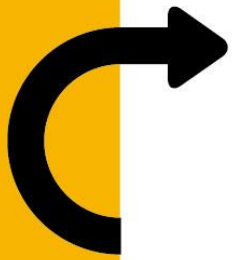
结合我司的产品特点，安全测试需遵守以下准则

- Web安全测试的执行，对于被测系统，或多或少都会存在一些影响（比如性能、垃圾数据），建议只在测试环境中进行；如果一定要在生产环境中执行，那么务必配置专门的测试数据，测试执行是应该非常慎重，只能修改或删除这些测试数据，禁止修改、删除其他，特别是用户的数据。
- 安全测试的目的是发现安全弱点，至于发现一个弱点以后更深一步的渗透测试禁止涉及。例如针对暴力破解测试，我们只给出验证存在暴力破解漏洞的可能，但不能提供暴力破解的方法。
- 测试工具的申请和使用，请遵循公司信息安全相关规定。
- 扫描工具，特别是一些侵入式的工具，例如AWVS等，禁止扫描后台，同时在大促期间严禁扫描生产环境。

安全测试策略

右图是WEB安全测试的主要策略：





第一部分 WEB安全测试概念

第二部分 WEB安全测试技术

第三部分 业务逻辑安全测试技术

信息搜集

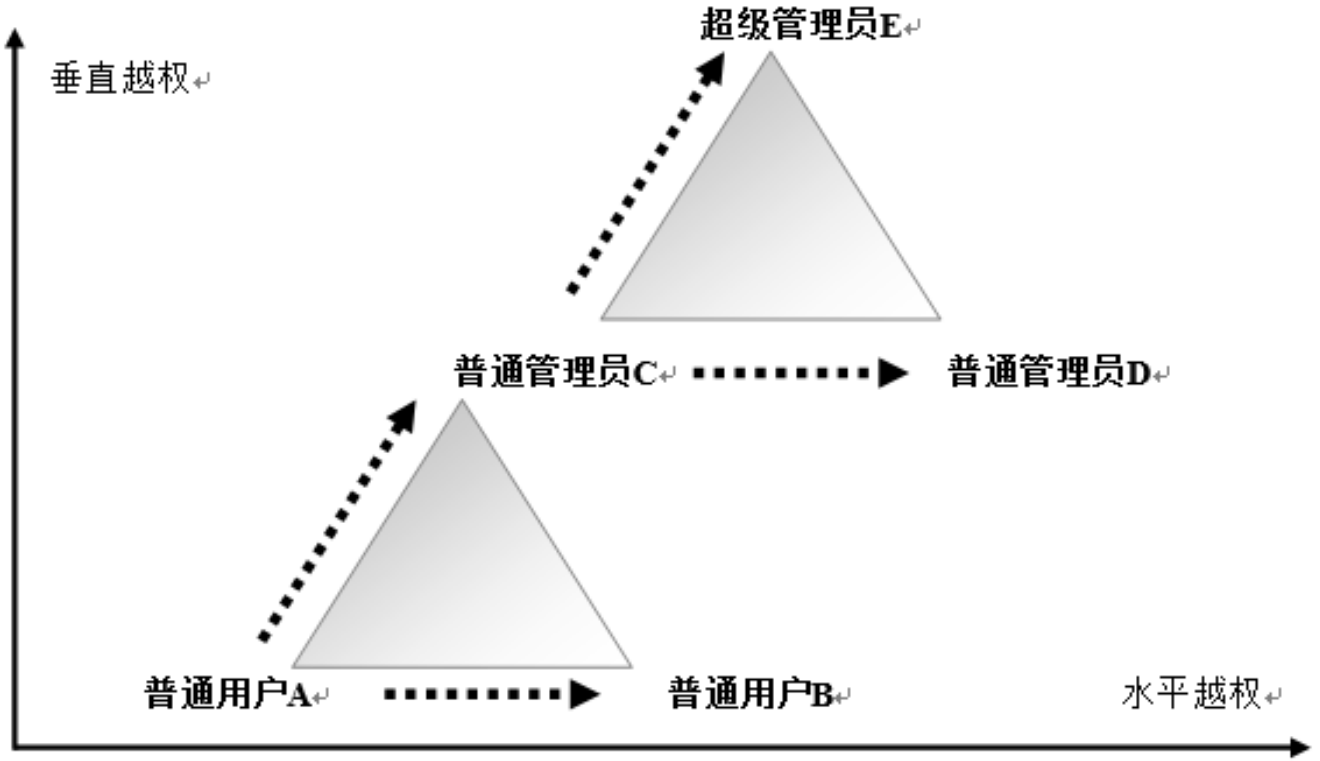
一般情况下，渗透测试的第一步就是信息搜集，信息搜集可为进一步对目标网站攻击提供有价值的信息，搜集的内容包含Whois,真实ip,子域，端口，服务，关联度高的目标，备案，企业资料，历史漏洞，员工邮箱等等信息，而安全测试重点关注是以下三个方面：

- 服务器开放端口
- 服务器开放的HTTP方法
- 服务器指纹信息

测试的方法：
通过工具进行扫描获取（具体请参阅工具的使用介绍PPT）。
提供此类功能的工具有：
AWVS
APPSCAN
Nmap
httprint

越权

越权存在两种类型：垂直（纵向）越权和水平（横向）越权。前者指的是一个低级别攻击者尝试访问高级别用户的资源；而后者指的是攻击者尝试访问与他拥有相同权限的用户的资源。下图能够比较清晰的描绘出它们之间的关系：



越权

垂直越权（Vertical Access Control），也就是权限提升攻击。

导致的原因是由于web应用程序没有做权限控制，或仅仅在菜单上做了权限控制，导致的恶意用户只要猜测其他管理页面的URL，就可以访问或控制其他角色拥有的数据或页面，达到权限提升目的。

这个威胁可能导致普通用户变成管理员权限，进而威胁整个应用安全。

测试的方法：

1、以超级管理员身份登陆Web网站
2、查看源文件，源文件中查找重要的管理菜单（例如用户管理）的URL链接，并拷贝URL链接地址
3、退出登陆
4、以普通用户身份登陆Web网站
5、在浏览器中输入刚才拷贝的URL地址
6、观察结果

越权

水平越权（Horizontal Access Control），也就是水平权限安全攻击。

引发原因是Web应用程序接收到用户请求，修改某条数据时，没有判断数据的所属人，或判断数据所属人时，从用户提交的request参数（用户可控数据）中，获取了数据所属人id，导致恶意攻击者可以通过变换数据ID，或变换所属人id，修改不属于自己的数据。

这个威胁可能导致恶意用户可以删除或修改其他人数据。

测试的方法：

1、使用userA的身份登陆到Web应用
2、进入userA专属页面（例如修改用户信息），修改后提交数据
3、通过代理篡改其中的参数为userB的值，再提交服务器
4、观察结果

越权

防范措施：

- 任何页面和接口地址在业务需要时均加入权限判断；
- 客户端提交的资源标志与已登陆的用户身份进行匹配比对；
- 用户登陆后，服务器端不应再以客户端提交的用户身份信息为依据，而应以会话中保存的已登陆的用户身份信息为准；
- 必须在服务器端对每个请求URL进行鉴权，而不能仅仅通过客户端的菜单屏蔽或者按钮Disable来限制。
- 建议使用成熟的安全框架来处理权限问题，比如spring security

SQL注入

SQL注入，即SQL Injection。就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将恶意的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入恶意SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。

SQL注入的危害主要有：

- 数据库信息泄漏：数据库中存放的用户的隐私信息的泄露。
- 网页篡改：通过操作数据库对特定网页进行篡改。
- 网站被挂马，传播恶意软件：修改数据库一些字段的值，嵌入网马链接，进行挂马攻击。
- 数据库恶意操作：数据库的系统管理员帐户被窜改。
- 服务器被远程控制，被安装后门：经由数据库服务器提供的操作系统支持，让攻击者得以修改或控制操作系统。
- 破坏硬盘数据：瘫痪全系统。

SQL注入

测试方法：
1、工具测试
　　借助工具，例如SQLmap进行辅助测试，详情参阅工具介绍PPT

2、手工测试
2.1、观察参数的值value是否为是否存在风险点，注意区分值的数据类型
2.2、尝试通过盲注进行测试，例如将被测参数后加上测试语句“and 1=1”
2.3、尝试猜数据库、表、字段等方法进行多条件或换条件测试
2.4、尝试进行增删改查等SQL操作测试

SQL注入

防范措施：

- 检查变量数据类型和格式

  只要是有固定格式的变量，在SQL语句执行前，应该严格按照固定格式去检查，确保变量是我们预想的格式，这样很大程度上可以避免SQL注入攻击。

- 过滤特殊符号

  对于无法确定固定格式的变量，一定要进行特殊符号过滤或转义处理。

- 绑定变量，使用预编译语句

  不同的程序语言，都分别有使用预编译语句的方法，绑定变量使用预编译语句是预防SQL注入的最佳方式，使用预编译的SQL语句语义不会发生改变，在SQL语句中，变量用问号?表示，无法改变SQL语句的结构，可从根本上杜绝了SQL注入攻击的发生。

跨站脚本

跨站脚本攻击是攻击者利用应用程序的动态展示数据功能，在html页面里嵌入恶意代码。当用户浏览该页之时，这些嵌入在html中的恶意代码会被执行，用户浏览器被攻击者控制，从而达到攻击者的特殊目的。严格意义上说，网站引入第三方的url也存在跨站风险，但这种场景较少，WEB应用引用最多的是公共的一些js库，比如jquery等，这里重点讲述XSS：

XSS即跨站脚本攻击(Cross Site Scripting)，为不和层叠样式表(CSS)的缩写混淆，故缩写为XSS。

XSS的危害主要有：

- 突破浏览器的限制：弹框限制、Activity限制；
- 用户信息：私密信息，日志等；
- 蠕虫攻击：XSS利用系统漏洞通过网络进行自我传播
- DDoS攻击：分布式拒绝服务攻击
- 入侵服务器：获取后台地址，管理员帐号甚至直接通过Ajax上传Shell
- 劫持客户端：获取用户http refer、cookie、浏览器版本信息等

跨站脚本

XSS按照行为划分为反射型、存储型；进一步按照其载体可分为通过HTML、JS、CSS上下文，基于DOM、FLASH和突变型XSS。

测试方法：

1、基本攻击<script></script>标签

<sCriPt>alert(1)</ScRipt>

2、回调函数：

<form action="Javascript:alert(1)"><input type=submit>

<keygen autofocus onfocus=alert(1)>

3、当=();:被过滤的时

<svg><script>alert&#40/1/&#41</script>

4、当< >被过滤时

" autofocus onfocus=alert(1)//

5、通过编码绕过防御

URL编码：%3Cscript%3Ealert(1)%3C%2Fscript%3E

base64：<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsbG8iKTs8L3NjcmlwdD4=">

跨站脚本

防范措施：

- 业务代码应对有输入的数据进行过滤或转义；
- 根据CSS类型对输出进行严格校验，例如字体大小，必须为数字，图片地址不允许出现非法字符；
- Dom结构中在读取属性之后，对属性中的特殊字符进行二次过滤；
- 对与服务器进行交互的输入中要严格控制字符长度；
- 始终遵循白名单优于黑名单的做法；
- 使用UTF-8为默认字符编码，设置content为text/html；
- 采用成熟的XSS防御框架，例如OWASP的JAVA_Encoder_Project

跨站请求伪造

跨站请求伪造攻击，即CSRF（Cross-Site Request Forgery）。攻击者在用户浏览网页时，利用页面元素（例如img的src），强迫受害者的浏览器向Web应用程序发送一个改变用户信息的请求。

跨站请求的危害：

- 可以造成用户信息被迫修改
- 可以构建钓鱼网站
- 严重者引发蠕虫攻击

跨站请求伪造

测试方法：
1、从业务角度，CSRF攻击可以从站外和站内发起
1.1、从站内发起CSRF攻击，需要利用网站本身的业务，比如“自定义头像”功能，攻击者指定自己的头像URL是一个修改用户信息的链接，当其他已登录用户浏览攻击者用户头像时，会自动向这个链接发送修改信息请求。
1.2、从站外发送请求，攻击者在自己的服务器上，放一个自动提交修改个人信息的html页面，并把页面地址发给受害者用户，受害者用户打开时，会发起一个请求。

2、从请求类型上，CSRF可分为GET型和POST型
2.1、伪造一个url在受害者的浏览器上直接执行
2.2、伪造一个表单在受害者的浏览器上提交至服务器

跨站请求伪造

测试方法：

■ 验证码

优点：每次操作都需要用户进行互动，简单粗暴有效。

缺点：输入验证码会严重影响用户体验，一般只出现在特殊操作场景，如注册

■ 校验refer

优点：通过检查Referer的值，我们就可以判断这个请求是合法的还是非法的

缺点：服务器不是任何时候都能接受到Referer的值，且Referer极易伪造，所以Referer Check一般用于监控CSRF攻击，而不用来抵御攻击。

■ 加入Token校验（荐）

添加一个参数Token，其值是随机的，并有时效性。这样攻击者无法构造出合法的请求进行攻击。

文件上传

文件上传漏洞是指攻击者上传了一个可执行的文件，并通过此文件获得了执行服务器端命令或其他有危害的能力。这种攻击方式是最为直接和有效的，“文件上传”本身没有问题，有问题的是文件上传后，服务器怎么处理、解析文件。如果服务器的处理逻辑做的不够安全，则会导致严重的后果。

文件上传的危害:

- 上传文件是Web脚本语言，服务器的Web容器解释并执行了用户上传的脚本，导致代码执行。
- 上传文件是Flash的策略文件crossdomain.xml，黑客用以控制Flash在该域下的行为。
- 上传文件是病毒、木马文件，黑客用以诱骗用户或者管理员下载执行。
- 上传文件是钓鱼图片或为包含了脚本的图片，在浏览器中会被作为脚本执行，被用于钓鱼和欺诈。
- 上传文件是大小马，可开启服务端后门，进而威胁整个服务端安全。

文件上传

测试方法：
1、直接上传风险文件测试
2、客户端验证绕过
　通过代理工具绕过
3、服务端验证绕过Content-type检测
　通过代理添加Content-type字段进行绕过
4、服务端验证绕过-扩展名检测
4.1、寻找漏网之鱼
4.2、大小写绕过，如aSp，pHp
4.3、特别文件名构造
4.4、利用容器解析漏洞，例如双扩展名绕过
5、服务端验证绕过-文件完整性检测
　文件头检测绕过，通过代理工具添加
6、图像代码注入后的攻击
　代码注入到图片中，在通过远程控制图片

文件上传

防范措施：

- 检查上传文件扩展名白名单，不属于白名单内，不允许上传。
- 上传文件的目录必须是http请求无法直接访问到的。如果需要访问的，必须上传到其他（和web服务器不同的）域名下，并设置该目录为不解析脚本语言的目录。
- 上传文件要保存的文件名和目录名由系统根据时间戳生成，不允许用户自定义。
- 图片上传，要通过处理（缩略图、水印等），无异常后才能保存到服务器。
- 上传文件需要做日志记录。

文件下载

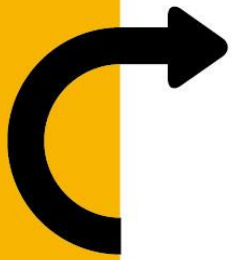
文件下载安全风险主要是任意文件下载攻击和目录遍历攻击。处理用户请求下载文件时，允许用户提交任意文件路径，并把服务器上对应的文件直接发送给用户，这将造成任意文件下载威胁。如果让用户提交文件目录地址，就把目录下的文件列表发给用户，会造成目录遍历安全威胁。恶意用户会变换目录或文件地址，下载服务器上的敏感文件、数据库链接配置文件、网站源代码等。

测试方法：
1、已知某下载页面URL（例如 http://www.exmaple.com/download/usera/salary.xls）
2、猜测并更改URL路径是否存在非法获取他人文档（例如 http://www.exmaple.com/download/userb/salary.xls）
3、更改URL为上一级路径并尝试查看是否为目录（例如 http://www.exmaple.com/download/）
4、尝试根据服务器操作系统特点猜测物理结构目录
5、尝试使用目录探测工具进行测试等

文件下载

防范措施：

- 要下载的文件地址保存至数据库中。
- 文件路径保存至数据库，让用户提交文件对应ID下载文件。
- 下载文件之前做权限判断。
- 文件放在web无法直接访问的目录下。
- 不允许提供目录遍历服务。
- 记录不符合规范的上传文件日志。
- 不允许将代码放到服务器端。



第一部分 WEB安全测试概念

第二部分 WEB安全测试技术

第三部分 业务逻辑安全测试技术

业务逻辑安全漏洞

业务逻辑安全漏洞，即利用应用的业务、功能的**逻辑缺陷**的漏洞。业务逻辑安全漏洞一直是安全测试中令人头疼的话题。相比SQL注入、XSS漏洞等传统安全漏洞，现在的攻击者更倾向于利用业务逻辑层的应用安全问题，这类问题往往危害巨大，可能造成了企业的资产损失和名誉受损，并且传统的安全防御设备和措施几乎对此不起任何作用。

造成业务逻辑漏洞的原因：

- 业务发展迅速，产品设计考虑不周
- 开发水平不一，功能设计考虑不周
- 内部监管不力，对业务逻辑漏洞的重视不足
- 第三方引入的缺陷

业务逻辑漏洞的特点：

- 资深的程序员也易犯错
- 业务逻辑漏洞没有天敌
- 逃逸各种防护
- 技术手段无法有效监控和防范

身份认证安全

一、身份认证安全
即利用技术手段破解或绕过服务端对用户身份的认证

1. 暴力破解
1.1在没有验证码限制或者一次验证码可以多次使用的地方，使用已知用户对密码进行暴力破解或者用一个通用密码对用户进行暴力破解。
1.2利用工具或脚本，例如使用Burpsuite的intruder工具进行靶向攻击，或者利用htpwdScan进行撞库攻击等

2.利用session&cookie
2.1会话固定攻击：利用服务器的session不变机制，借他人之手获得认证和授权，冒充他人。
2.2Cookie仿冒：修改cookie中的某个参数可以登录其他用户。

身份认证安全

3.弱加密
3.1未使用https，在传递用户名和口令时易被中间人攻击，例如通过伪基站的技术手段。
3.2前端加密，用密文去后台校验，并利用smart decode解密，易被攻击者解密造成泄密。

防范措施：

- 所有认证用户身份的业务场景均需考虑用户的尝试次数和错误次数限制
- 尽可能的禁止认证用户身份时的服务器回显信息
- 对Session和Cookie的管理要规范
- 说好用HTTPS进行数据传递
- 采用高强度的加密算法，并在后台进行强校验

业务一致性安全

二、业务一致性安全

即利用业务处理中未对业务的一致性进行校验的逻辑漏洞进行攻击

1.手机号篡改

通过代理抓包修改手机号码参数为其他号码尝试，例如在办理查询页面，输入自己的号码然后抓包，修改手机号码参数为其他人号码，查看是否能查询其他人的业务。

2.邮箱或者用户篡改

2.1抓包修改用户或者邮箱参数为其他用户或者邮箱。

2.2篡改管理员身份

3.订单id篡改

查看自己的订单id，然后修改id（加减一）查看是否能查看其它订单信息。

业务一致性安全

4.商品编号篡改
例如支付商品时，通过篡改商品编号试图达到获利的目的。

5.用户id篡改
抓包查看自己的用户id，然后修改id（加减 1）查看是否能查看其它用户信息。

防范措施：
- 做好用户、资源等参数的权限校验，防止越权
- 对前台提交的业务相关的参数须校验其合法性
- 业务相关的参数应确立一一对应关系

业务数据篡改

三、业务数据篡改

即利用业务处理中未对业务数据的完整性和正确性进行校验的逻辑漏洞

1.金额数据篡改

抓包修改金额等字段，例如在支付页面抓取请求中商品的金额字段，修改成任意数额的金额并提交，查看能否以修改后的金额数据完成业务流程。

2.商品数量篡改

抓包修改商品数量等字段，将请求中的商品数量修改成任意数额，如负数并提交，查看能否以修改后的数量完成业务流程。

3.最大数限制突破

很多商品限制用户购买数量时，服务器仅在页面通过js脚本限制，未在服务器端校验用户提交的数量，通过抓包修改商品最大数限制，将请求中的商品数量通过代理改为大于最大数限制的值，查看能否以修改后的数量完成业务流程。

业务数据篡改

4.本地js参数修改
部分应用程序通过Javascript处理用户提交的请求，通过修改Javascript脚本，测试修改后的数据是否影响到用户。

防范措施：

- 对传递的参数首先进行合法性校验
- 对参数的数据前后台应进行比对，保证数据的正确性
- 对有逻辑处理的数据应校验其数据前后逻辑关系
- 禁止在页面通过js处理提交参数

密码找回漏洞

四、密码找回漏洞

密码找回功能本意是设计给那些忘记密码的用户，以便他们能够找回自己的密码。然而，攻击者可以利用此项功能非法窃取他人的账户。此类漏洞的利用方式非常之多，以下仅列举出常见的一些攻击方式：

1.暴力破解

密码找回的凭证太弱，如只需要填入一个四位或者六位的纯数字就可以重置密码，导致可以暴力破解。

2.密码找回凭证从客户端直接获取

密码找回凭证在客户端获取，在密码找回时注意抓包查看所有url返回响应等，看是否有最终的凭证出现，这样就可以绕过手机或者安全邮箱了。

3.密码找回凭证在页面中直接获取

找回密码的答案在网页的源代码中，常见于早期的一些网站，当然也不排除现在某些网站依然存在此类问题

密码找回漏洞

4.密码找回凭证容易猜出
例如找回密码的关键凭证仅仅是时间戳的md5，极易破译

5.密码找回凭证存并非只是与单个用户并绑定的问题。
找回密码凭证发到邮箱中，url中包含用户信息以及凭证，但是这个凭证可以重置任何用户。

6.用户找回密码的邮箱地址或者手机号码被修改
是绑定安全手机的逻辑问题，导致可以使任意用户帮上自己可控的安全手机，然后就可以重置任意人的手机号码。

7.在最后提交修改的密码处的逻辑错误
例如在最后重置密码处跟随一个用户ID，改成其它用户的ID，即可把其它用户改成你刚刚修改的密码。

密码找回漏洞

8.验证码窃取
服务器只验证了对应的验证信息是否存在，没有验证是否与账号匹配，可用合法的用户凭证窃取他人的密码重置业务

防范措施：
- 密码找回凭证应做次数限制，包括发送次数和输入错误次数限制等
- 找回密码凭证够复杂并且不可猜测
- 不可存在越权，或者重要的凭证在不该出现的地方出现
- 传输的验证参数做好加密，同时对参数做好过滤
- 任何逻辑处理动作放在服务器端进行

验证码突破

五、验证码突破

验证码的突破主要是利用验证码的设计缺陷进行破译或绕过

1.验证码暴力破解

使用BrupSuite对特定的验证码进行暴力破解

2.验证码时间、次数测试

抓取携带验证码的数据包不断重复提交，例如：在投诉建议处输入要投诉的内容信息，及验证码参数，此时抓包重复提交数据包，查看历史投诉中是否存在重复提交

3.验证码客户端回显

当客户端有需要和服务器进行交互，发送验证码时，即可使用firebug就可看到客户端与服务器进行交互的详细信息查看是否存在具体的回显信息

验证码突破

4.验证码绕过
当第一步向第二步跳转时，抓取数据包，对验证码进行篡改清空测试，验证该步骤验证码是否可以绕过

5. 验证码js绕过
例如，业务流程的第一步、第二部、第三步都是放在同一个页面里，验证第一步验证码是通过js来判断的，可以通过越过js限制修改验证码在没有获取验证码的情况下可以填写实名信息，并且提交成功

防范措施：
- 验证码的需做次数限制，特别是错误尝试次数
- 禁止任何验证码的回显
- 功能设计时需考虑前后逻辑关系
- 禁止通过js来做逻辑处理

业务流程乱序

六、业务流程乱序

即利用业务流程的前后关系的漏洞进行绕过功能环节

1.顺序执行缺陷

例如某业务逻辑可能是先A过程后B过程然后C过程最后D过程，攻击者可构建数据给应用程序发送的每一个请求，因此能够按照任何顺序进行访问。于是，攻击者可以从B直接进入了D过程，就绕过了C。如果C是支付过程，那么用户就绕过了支付过程而买到了一件商品。如果C是验证过程，就会绕过验证直接进入网站程序了

解决措施：

- 服务端对每个过程都应进行二次校验，保证业务流程的顺序正确性

业务接口调用安全

七、业务接口调用安全

即利用正常的接口调用来做正常业务之外的非法功能

1.重放攻击

在调用业务或生成业务数据环节中（例如：短信验证码，邮件验证码，订单生成，评论提交等），对其业务环节进行调用（重放）。如果业务经过调用（重放）后被多次生成有效的业务或数据结果就可能导致恶意注册、短信轰炸等。

2.内容篡改攻击

例如通过利用短信平台，篡改数据包，可以修改本来发送给用户的正常的文本信息“请访问http://www.example.com”为“请访问http://www.hack.com”，造成钓鱼攻击等。

解决措施：

- 对接口的调用需限制次数以及校验调用方的合法性
- 对接口的调用传递的数据包进行正确性和完整性校验

时效绕过攻击

八、时效绕过攻击

即利用业务设置的业务数据的时效性的校验漏洞进行攻击

1.时间刷新缺陷

例如某票务网站的买票业务是每隔5s，票会刷新一次，但是这个时间是在本地设置的间隔。于是，在控制台就可以将这个时间的关联变量重新设置成1s或者更小，这样刷新的时间就会大幅度缩短（主要更改autoSearchTime本地参数）

2.时间范围漏洞

针对某些带有时间限制的业务，修改其时间限制范围，例如在某项时间限制范围内查询的业务，修改含有时间明文字段的请求并提交，查看能否绕过时间限制完成业务流程。

防范措施：

- 数据时效的校验和走秒都必须放在服务端
- 从客户端传递的参数含有时效性的数据时需进行加密或用常数进行替代并比对数据库表中数据

Thanks!