课时10 逻辑漏洞（上）

大家好，本节课介绍逻辑漏洞，关于逻辑漏洞我们将分两个课时完成。

概述

逻辑指的是思维的规律。
逻辑漏洞指是的业务逻辑漏洞，是由于网站设计、开发人员的思维定势或考虑不足所导致的设计缺陷与漏洞。这些漏洞往往与业务紧密相关。

逻辑漏洞覆盖面很广，一直以来对于逻辑漏洞尚未产生明确的分类。其大致包括了：绕过功能限制、遍历、越权、弱口令、信息泄漏、任意用户密码重置、竞争性问题等。
在一些场景下，由于功能本身设计复杂、易于出现纰漏，导致针对特定场景下的特定逻辑漏洞问题讨论，此类风险场景有：支付安全、验证码安全等。

首先逻辑指的是思维的规律。逻辑漏洞就是在设计业务逻辑时，由于一些思维定势或考虑不足所导致的设计缺陷与漏洞。一直以来对于逻辑漏洞尚未产生明确的分类。其大致包括了以下这些场景，本节课内容较多，我们接着来看具体的细节。

限制

身份识别限制： 门锁、门禁、保险箱、人脸识别、登录密码、看门的狗

付费识别限制：点餐小票、地铁闸机、年费会员、一只干净的碗

顺序限制：号码牌、预约短信、九连环、栈与队列、弹夹、象棋规则

行为限制：法律、列车烟雾报警、围墙、屏风、足球规则

逻辑漏洞本质上是突破限制，那么限制都有哪些呢？关于限制会分为以下几类：

- 身份识别类的这种限制，基本上就是要去识别你是谁这个问题。最后列举了一个比较特殊的，看门的狗，假如说这只狗是专门用来开门的，一般都会去识别主人的气味，如果是其他陌生人过来，他肯定会不会轻易的就放行。它是一个比较特殊，但它确实在这个地方作用也是识别身份用的。
- 付费识别类，是为了证明你有没有付费来限制。这里也有一个也比较特殊，一只干净的碗，这个是什么意思呢，这个主要是在我家那边吃饭喝羊肉汤的话，付完钱之后他就会给你一只碗，然后你拿这只碗端着到另外一个地方去盛汤，盛汤的碗其实就表示你已经付了钱了。你如果是从其他地方也能买到这样一只碗，或者说你从别人桌子上拿了一只碗刷干净去盛汤也是可以的，但是一般人也不会这样做，为什么？因为你所花的成本高，所以一般也不会去做这种绕过，这也是基于付费识别一种标志。
- 顺序限制，就是去限制一些东西，它必须按顺序去排列。比如说我们去等位的号码牌，预约短信，有时候我们去医院预约挂号，我们要凭这个短信，如果没有预约的话，它是就没办法去直接看病的。还有一种顺序限制，我不知道大家有没有玩过，九连环，这也是很有意思的一个游戏，它在本身的物理构造上做了一些限制，你只能从最右边或者最左边把这个环取下来，它规则就是这样的，这其实也是一种顺序。还有我们的计算机比较熟悉计算机的人，经常会遇到的这种栈和队列。然后我们子弹压进弹夹之后，然后一颗一颗打出去，也是有顺序的。包括象棋规则，你只能按照这规则，马走日象走田这样的也不能去轻易的破坏，相当于本身受到了一个制约，包括象棋规则的，你走一步我走一步。
- 然后还有行为的一个限制，比如法律的限制，还有列车烟雾报警，约束了我们在车厢内不能吸烟，然后还有围墙，就限制了外面的小偷或强盗，没办法去直接的闯入家里面去去盗窃，或者说另外一个角度我们养的猪鸡这种小动物，它们也是做了一个围墙围着限制他们出去，还有屏风，限制别人看的这个行为，也是对行为的一种限制，包括足球规则不能用手。

我们在自然的生活中有这样各种各样的限制，不知道大家以前遇到限制是一个什么样的一个心态，可能以前会思考这个限制设置的是否合理，是否公平。这里还有一个角度，是思考能不能绕过，在设计上会不会有一些漏洞，存在一些缺陷，关于限制的绕过我们来去看一下，这个是也是不完全的总结，因为这个现实生活中这些东西太多了，只能就是依据经验简单的来总结和整理一下。

打破限制的场景

伪造。 饭票、车票、公交卡伪造

伪装。明修栈道暗度陈仓

重用。 凭证重用

重放。 录音重放、信号重放

劫持。 信号劫持、运输劫持

嵌套。 诱饵、货车嵌套

并行。 芯片、地铁跟随

外带。 ATM机

暴力枚举。 破解密码

条件竞争。高铁站验票口

状态还原。购物领红包。

另辟蹊径。翻窗户、拆卸车轮

局部突破。铁栅栏

宏观突破。刷票、薅羊毛

博弈突破。出租车调价案例

修改限制。 暴力开锁、收费站冲卡

- 第一个绕过方法叫伪造，伪造也是欺骗。比如说需要一个凭证，吃饭的话需要一饭票，坐车的话需要车票，我们能不能去伪造去制造跟他一样的东西？这个东西是否唯一的无法再次制造的，就像假币其实也是一种伪造，就做一个跟真的很像的，去欺骗一些识别能力比较低的这种人群。
- 伪装的话，其实就是欺骗监控，这种就是时刻防备的这种限制，比如说明修栈道暗渡成仓，表面上它限制了你从陆地上去越过军事边界，其实你可以去挖一个地道去从下面来走。
- 重用主要是凭证重用，比如说去公园的时候有时候需要一个票，然后这个票都一般都是那种不刷的，一次性的进去就算了，但是你进去之后可以从公园围墙扔到外面，然后外面一个人简直在继续进，然后再扔出来，再去尽量反复来利用就是重用。
- 然后重放的话，比如说录音重放，有时候一个大家去超市买东西时候，就不是有一个支付宝到账什么多少钱吗？这段话带把录下来，然后下次买东西时候播放一下。还有信号，有一个例子汽车钥匙，他是会发出一个特定的信号来锁车，也发出一个特定信号来解锁，然后这个信号如果被一些设备截获的话，再重放就可以把这个车解锁掉。
- 劫持，信号劫持刚才说的这个也包括偷车的时候，主人锁车的时候，站在车的周围，用这种信号屏蔽器去把锁车的信号屏蔽掉，主人误以为按了一下已经锁上了，但其实没有锁上就走了，等他走了之后就去打开车门去偷东西。我
- 嵌套也有一个比较有意思例子，在高速上有一个货车里面装了一个小轿车，他过高速路口监控的时候并没有拍到这个小轿车，但这个小轿车进入高速之后，它就从货车里面出来就在那段路上开，然后里面会有一些行凶杀人什么的，隐藏自己不在场证明，但其实他小轿车藏在货车里面了，就这样嵌套来欺骗外面的监控。
- 并行，进地铁的时候，并行一下，两个人一起进入。
- 外带，ATM机上一个人存钱时候，然后就把钱一百块钱上面涂上胶水，每一张都涂上胶水，然后存进去500块钱，取的时候这500块钱洗完之后就和其他的机器里面的钱做

了一个粘连，就两张粘成一张，这样在取的时候取500就变成了一千。

- 暴力枚举，就是破解密码的时候，像摩拜单车四个密码，然后大家从头到尾按一遍，基本上就能把解开
- 条件竞争就是这是比较特殊的，在高铁站验证，有用票来验证，也有就是人工检票，但这两种形式往往放在一起，就是最左边的几列是自动验票，然后右边的一列人工，这时候就看到有一个人他拿来，他拿了一张票，然后走的是人工通道，但是他用左手把票塞到了闸机验票的里面，然后给另外一个人放行，然后一拿另外一个人坐过去，然后这时候他再拿出这张票之后再走人工通道再验一遍，他就一张票用了两次，他是并行的，就是相当于条件竞争在同时过去的时候就可以用两次来突破这个限制。
- 状态还原，也是很经典的，比如说有些商场你购物就可以领一个红包，购物是可以不满意是可以退掉的，你就一直买一个东西，然后退买个东西退就把红包都给用完了。
- 另辟蹊径，比如说你可以不走这条路，这条路上有限制你就可以走其他的翻窗户，或者如果是有自行车上锁的话，就把车轮拆掉，然后后面的一部分就可以骑走，这种类似的这种方法。
- 局部突破，一般我们看到电影里面关押犯人的时候，这个门是铁栅栏一样，然后如果我们想突破限制的话，可能没办法直接出去，这时候就可以把手伸出去拿一些东西，它并没有突破限制，它只是去从一部分就是局部的这个地方去突破，只是把手伸出去，并没有整个人出去，而且他限制的本身去防御的也是这个人的进出，然后我们通过这个就实现了一部分的这种突破。
- 还有一些是宏观上突破，比如说刷票这个场景很经典，假如说我们只能他限制了，就是你一个人只能投一次票，对吧？然后这时候假如说限制是很严格的，没办法去跨越，我这时候能不能去找找多个这样的相同的这样的，比如说找用代理一个ip只能投一次票，我能不能找很多这样的代理ip，用宏观的这种方法去去投票。
- 最后是修改限制，就是我们限制我们没办法绕过，也突破不了，我们就去修改限制，然后比如说暴力开锁或者收费站的直接冲这样去充卡。

突破功能限制漏洞

突破功能限制漏洞是指攻击者通过一些手段绕过网站设计本身的功能限制并可以产生实际危害的一类漏洞。

所属分类 逻辑漏洞

- **风险等级及危害** 低～严重 视具体情况而定

之前介绍的**前端校验漏洞**也属于突破功能限制漏洞的一种。

刚才我们介绍了很多生活中的这种例子，然后回到我们的互联网的世界，跟其实比生活中的问题还要严重的多，互联网上设计出来的这种缺陷的地方反而会更容易去突破，为什么？因为在现实生活中你看到一个地方有漏洞，你可以走旁边绕过，但是你会觉得这样不好意思，会不会被别人看到，会不会这样不太道德。但是在互联网上大家都不知道互相都不知道对方身份，所以做这些事情的时候反而你可以去很自由的做这样突破，所以在互联网上就有很多这样的突破功能限制的这种案例。

在这里我们学到第一个漏洞叫做突破功能限制漏洞，这种漏洞它很广泛，它在网络中也有很多这样的漏洞，然后我们提到的只是一种叫前端校验漏洞，然后前端校验的话，我们知道在后端只要我们用brup的抓包去修改，或者说用浏览器的审查，这种功能都是可以很容易很轻易的去绕过的。

还有其他的一些突破，不是前端交易也是可以突破的，这种漏洞总的我们把它归为一类，就叫突破功能限制漏洞，然后这种漏洞它风险等级也有低危的，也有严重的，它中间各种情况都有，是要视具体情况而定。我们举几个例子。

突破功能限制漏洞举例

突破服务器限制每个订单只能最多只能使用1张优惠券的限制。

突破网银限制用户当天最大交易限额是10000元的限制。

突破某游戏中的出牌规则限制，可以在条件不允许的情况下打出手牌。

突破交易时需要输入交易密码的限制。

绕过人脸识别的验证功能。

绕过USB Key校验。

……

- 第一个是就是服务器限制，每个订单最多只能用一张优惠券，但是通过一些手段有可能就能用两张三张优惠券，当达到目的的时候，其实就已经产生了一些经济上的利益，这种漏洞一般也是很严重的，也算是突破功能限制的一种。
- 还有网银限制，用户当天最大交易额是1万元，一般网银都会做一个这种交易限额设置，假如说我们能用一些手段突破限制能够交易2万元3万元，这个其实也是一种突破，但是这种突破其实它的危害程度就没有那么高。
- 然后还有在网上有一些游戏，举个例子，就比如说斗地主，我们都知道斗地主的话，五是比四大的，对吧？如果别人出了一对五，我们这时候能出一对四，是不是也算突破了规则的限制，通过这种突破限制来打破游戏原本的规则
- 还要突破交易时需要校验的交易密码。有一些交易是需要输入交易密码的，我们通过一些手段能够绕过不输入交易密码就能完成交易，这也算突破交易密码的限制。
- 绕过人脸识别，绕过USB KEY等等我们不展开去说，然后具体会当我们遇到的时候，我们始终带着疑问想想办法去突破一下，这个其实是跟每个人思维习惯是有关的，有些人拥有逆向思维，经常就能想到在哪些点很容易突破，哪些点设计的不合理。

突破功能限制漏洞防御

1. 不依赖于前端校验。 大多数突破功能限制的案例往往都是服务器只进行前端校验而忽视了后端校验。
2. 对于复杂逻辑的设计，在设计前应当进行反复推敲。复杂逻辑的产生一定是由于服务器本身的逻辑需要，而这种逻辑关系是否能做到缜密，是否能无懈可击，需要进行大量的测试和推理验证。
3. 怀疑前提和假设（假设检验）。演绎推理前，应当对条件进行充分地怀疑，只有不信任前提假设才能制定完整而严密的逻辑。

- 然后我们用的理解到就是在跟网络相关最多的就是前端校验，它是很大的一类，我们遇到大多数这种突破功能限制，案例都是前端校验。我们之前讲过，首先要防御这类功能突破限制漏洞，第一个就是不要依赖于前端校验，然后大多数突破功能限制的案例，都是服务器的管理员缺乏网络安全的知识，认为在前端用JS做这种校验，就能够让用户限制它输入很多东西，然后到服务器的数据就不会有恶意的这样的字符，但其实我们已经学过了，这种是前端校验都是无效的，所以这个要重视。
- 然后第二个是对于复杂逻辑的这种设计，在设计之前，这个逻辑一定要经过一个反复的推敲。为什么网站设计时候要设计这么复杂逻辑，网站业务功能逻辑为什么要设计这么复杂，然后它的逻辑关系它是否合理，是否能够做到证明是否无懈可击，这个都是需要很多人去推敲，并不是一个设计人员就可以考虑到很全面的，需要经过大量的测试和验证。在网站功能设计好之后，首先就是功能验证，然后是安全验证，都会有这样的一个验证来去检查是否涉及到位。
- 然后还有就是要去怀疑前提和假设，在做演绎推理的时候，举个例子，比如说安服工程师，然后都会测试网站，因为安服工程师里面还有一些不会测试网站，那就不是安服工程师了么？大家在做假设之前，要先问一下自己的前提是否是对的，好多逻辑设计可能前提都是错的，造成了后面很严重的逻辑错误。

越权漏洞

越权漏洞是指攻击者跨越网站应用原本的权限管理设计而读取或操作其他用户或管理员的数据和设置的一类漏洞。

所属分类　　逻辑漏洞

- **风险等级及危害**　　低～严重　　视具体情况而定

一般情况下，越权漏洞可分为**水平越权**和**垂直越权**。

下面一个逻辑漏洞叫越权漏洞，这个漏洞大家可能经常听到，越权漏洞也是很简单，就是跨越了网站应用原本的权限设计规则。如果规则能够通过一些方法绕过，就产生了越权漏洞，它的危害也是要视情况，有些越权是很严重的，比如说越权添加管理员就很严重，添加管理员账号就意味着已经拥有了后台权限，有些比如说越权为他人支付订单，这种漏洞可能就没有什么危害，一般SRC也不会收这样的漏洞。
越权漏洞可以分为水平越权和垂直越权。水平越权就是我们的级别是相同的，但是我可以操纵你的一些东西；
垂直越权，一般都是网上垂直权限比我高的人，比如会员，比如管理员的权限。

越权漏洞的原理

网站设计时，为了标识用户，往往会给不同的用户分配不同的Session来维持用户会话。以读取个人信息为例，当用户提交读取个人信息请求时，网站应当判断当前用户是否拥有读取的权限，而此时的判断依据应当源自于用户本身的Session，而不应当以用户提交的id作为参考，不应当在没有校验的情况下返回用户提交id所对应的用户信息。

一般情况下，低级越权漏洞产生的原因是由于大多数网站设计者认为前端校验可以避免越权，认为前端固定用户id即可信任用户所提交的参数，或是在功能设计时将id进行简单的base64编码来认定用户身份。

然后我们来简单看一下，越权漏洞为什么会产生？网站在设计的时候，假如有用户系统的网站，它肯定是要去区分出每一个用户的，不同的用户在这个系统中它肯定有不同的数据，比如说我们有自己的头像，就自己的ID等个人信息，这些信息是归用户个人所有的，假如说我想读取我自己的信息是可以的，假如说我想读取另外一个用户的信息，这个行为应该是被禁止的。我提交了一个读取个人信息的请求请求服务器，它接收到我这个请求之后，应当是先去校验我要读的对象和我本身的权限是否匹配。我们登录这个网站，要获取session，session是用来维持用户登录状态的，可以对应到服务器的筛选，筛选就标识了每一个用户。

但如果网站设计的不够安全，以用户的ID作为唯一判断，那我抓包后，把ID改成别人的，网站就会返回别人的个人信息，这就产生了一个最简单的越权漏洞，也可以算是前端校验，漏洞具体归哪一类，我们不具体去说，他都可以去算。

然后还有一种是网站设计时认为简单的ID你可能一眼就能看出来，那我做一个bese64编码，你就猜不到了。但其实bese64编码根本没有起到加密作用，就是因为它的编码后和编码前可以任意的去还原的，它就构不成对用户的一个数据的保护。

常见的越权漏洞

修改表单中的id 直接读取、修改他人信息。

通过 view_controller 推理出 edit_controller 从而获取管理员编辑权限。

通过 xiaoming201901011234 猜测到他人的数据可能是 用户名+年月日+4位随机数，再进行暴力破解

通过阅读应用源码、HTML、JS等获取到隐藏的缺乏校验的接口。

在Cookie中修改 edit=false 为 edit= true 以获取某项权限。

用户注销后形成幽灵用户，再注册相同用户名的用户就拥有原来用户的权限

构造数组以绕过对第一个元素的校验。例如： user[0]=11332&user[1]=11333&user[2]=11334

常见的越选漏洞有哪些?

- 比如从表单中直接读到一个ip修改，这个ID就能读其他人信息，这是很常见的。
- 然后还有从view_controller 推理出 edit_controller，假如说编辑这个接口确实存在，有没有做权限验证这事，我就越权我就可以修改这个这个内容，就获得了一个管理员的编辑权限。
- 还有就是通过一些简单的逻辑推理，比如说我看到这样一串序列代表着我个人的一个简历或者一个东西，上传完之后服务器就返回了，您已上传成功，您的简历编号是小明2019001234就可以去猜我是19年1月1号上传的，然后名字叫小明，再进行暴力破解得到随机数信息。
- 通过阅读应用源码、HTML、JS等获取到隐藏的缺乏校验的接口，有些应用它是开源的程序，所以我能够很轻易地读到它后台的一些接口是在哪个地方。
- 在Cookie中修改 edit=false 为 edit= true 以获取某项权限。
- 还有一种越权比较有意思，当用户注销以后，这个用户并没有从数据库中删除，而是在这个数据库中给他打上了一个已注销的这样一个标记，用户就无法再登录了，这时候认为这个是很安全的。但是当在注册用户相同的用户名相同的用户的时候，这个用户就又被激活了，他把参数就给改掉，然后剩下的账户用户名密码重新刷新一下，他就又可以登录了，这时候我们只需要去注册相同用户名的话就可以碰撞出已经存在的这种系统用户
- 还有一种可能遇到比较少，举一个极端例子，比如说在一个可以下载自己简历的地方，我没发现自己的编号的11332，那试一下11333可能会获得了别人的简历。它的简历下载有可能是为了管理员方便批量下载这样设计。

越权漏洞的逻辑错误

越权漏洞的逻辑错误主要有两点：

1. **网站设计者面向的是大多数普通用户，但网站安全的设计者应当面向所有情况。**以为“普通用户不会看到hidden隐藏的值，自然也不会修改”的观点本身就是错误的。
2. **在任何情况下都不应当偷懒。**“用户是否能找的到该接口”和“是否校验”在本质上是有区别的。因为忽视了搜索引擎爬取的力量，忽视了暴力破解的力量，忽视了逻辑推理的力量。

这些常见的越权漏洞，在逻辑设计上有什么错误呢？我们来分析一下。

- 第一个是网站设计者，它面向的是大多数的普通用户，但是网站安全的设计者他面向的应该是所有的用户，所有的情况它不应该是考虑大多数人都不会改，所以这个地方不需要校验，大多数人都不会这样，所以不会他如果是这样想的话，就可能犯了一个很低级的逻辑错误，就以偏概全了。
- 然后在任何情况下都不应当偷懒，就是有些用户的接口就是后台接口，可能认为我就接口写得很特殊，什么26abc什么三七就是一串字母数字，比密码还长的一串，你是不可能找到这个接口的，所以我这接口不用做任何交易也是很安全的。其实有些这种接口很容易就被搜索引擎爬取到了。

越权漏洞防御

1. 对网站权限进行合理划分，除了明确被允许的功能外，其他功能都应设置验证，若验证失败则禁止访问。
2. 权限最小化原则，限制用户不必要的权限，用户权限过期后应当予以回收。
3. 在服务器端进行验证，对于用户身份的识别应当基于Session而不是简单地使用用户提交的参数。

然后我们看越权漏洞防御：

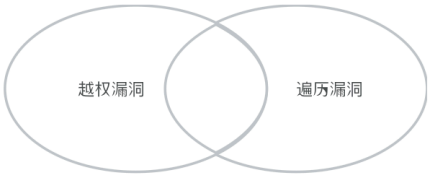
- 第一个是要对网站权限进行一个合理划分，除了明确哪些允许做的之外，其他的都应该设置为不允许，而不应该直接忽视，每一项不允许都要做一个校验。
- 还有就是权限最小化原则，然后每个用户他所拥有的权限是多大，就应该约束到这个范围内，如果用户有一些不必要的权限应当被禁止掉，然后当用户权限过期之后，应当把这个权限回收。
- 在服务端进行验证，不要在客户端进行验证，对用户的身份的识别应该是基于session的，而不是用户传递的参数。

遍历漏洞

遍历漏洞是指攻击者通过归纳总结服务器资源唯一标识变化规律后，通过一定手段获取到大量可产生利用价值资源的一类漏洞。

所属分类　　逻辑漏洞

- **风险等级及危害**　　低～严重　　视具体情况而定



然后下一类漏洞跟越权漏洞很像，叫遍历漏洞。这个漏洞和越权漏洞有什么区别？如果是那种查看信息的，大概率同时存在。遍历漏洞是指通过归纳或者总结服务器资源标识的变化规律以后，就获得了这个变化规律，获得规律以后，通过一定手段可以大量的获取到这样的一个资源，然后这些资源并且能产生一定价值，这时候就这个问题就导致了他有可能不越权也能获取到这样的一些资源。而越权漏洞是指修改越权，越权漏洞不一定是遍历。
他们是一个下图这个关系，他们中间有交集，有些漏洞，它同时是越权和遍历。大概是这样的一个关系。
它危害也是实际情况定。

遍历漏洞原理

网站在设计时，未对有价值的数据进行合理的访问控制，导致攻击者可以按整数递增等其他方式逐个获取到服务器的数据和资源。

有些数据，如果没有形成规模，并不算严格意义上的敏感数据，但如果达到一定规模，就可能给企业带来不可估量的损失。所以，对于敏感数据的定义和保护也是安全研究的一个十分重要的范畴。

常见的遍历漏洞有：
根据id顺序访问拉取数据。
通过修改约束条件，筛选出系统中大量其他数据。

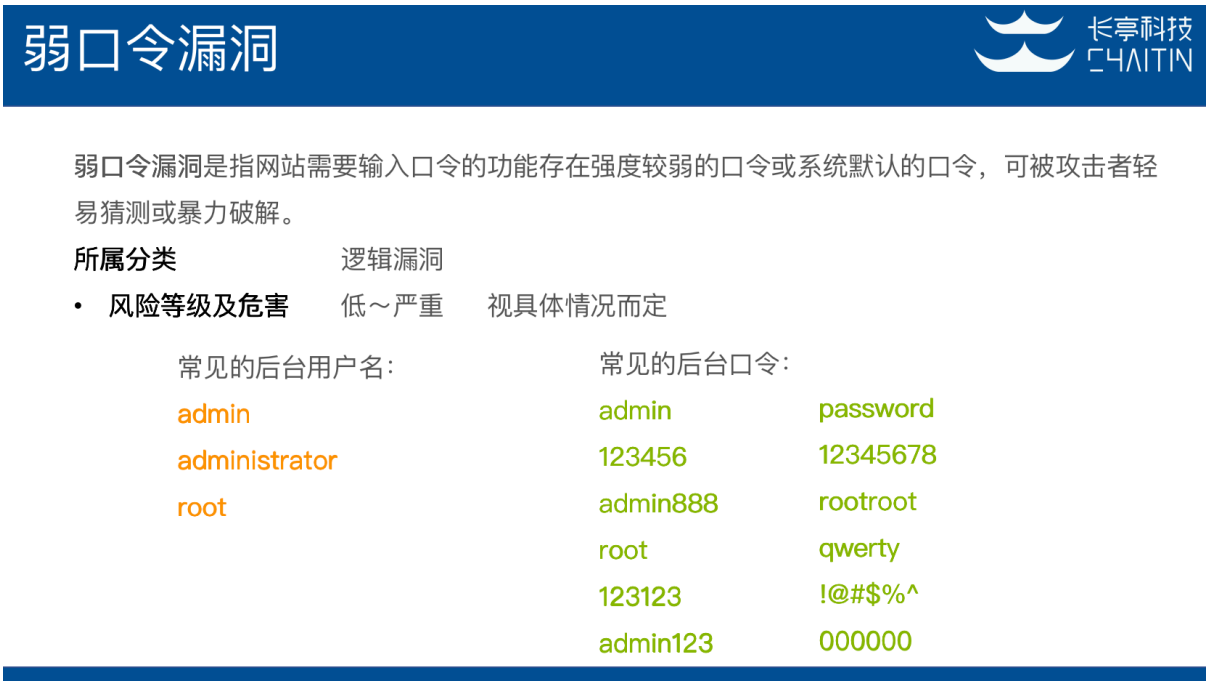
遍历漏洞产生于网站设计时，对有价值的数据，如果没有进行合理控制。攻击者就可以通过这样的，比如说整数递增，就是订单号是连续的，通过这个方式来去遍历的读取整个系统中的订单，这个时候其实也涉及到了一个越权，因为在网站设计的时候，一个人的用户的一个订单，应该是只有他自己能够看到，和商家能够看到，这两方能够看到，不应该是其他用户也可以看到。
举一个另外一个例子，打车场景中，我们知道每一个用户都能够看到接单司机，他的车牌号，他的评分，打车的人看到司机的信息也没有任何问题，这个功能本身没有问题。假如说调取车辆司机信息的接口是顺序的，序号是连续的，我们能够通过遍及这个接口，一直请求，就把整个系统中司机都拉取下来，这时候就产生了一个遍历的行为，它并不是越权。
还有一种比较特殊，这种可能也比较少见，但是确实遇到过。就是去修改它的约束条件，然后主要出现在一些有这种高级搜索的功能的应用下面。一般来说我们的搜索会变成一个查询语句，比如查看个人信息时，发起的查询SQL中包含我们的ID，这个时间如果我们能把ID这个条件去掉，再搜索出来的可能就针对数据库所有用户的。

遍历漏洞防御

1. 使用hash+salt算法加密获取资源的序号
2. 监控资源访问接口的请求频率，针对请求分布进行建模打击。

这种漏洞我们如何去防御？刚才说数字都是顺序的，或者说BASE64编码之后，也能够很容易还原，我们能不能用一些比较强的算法加密呢？

- 一般推荐就是用这种HASH加密，比如说md5这种加密算法，然后再加一个盐。这个盐其实就是保证了你加密的序列的一个长度，要达到一个固定的长度。
- 除此之外，有一些接口他就是需要这些是顺序的，或者说需要按这样展示的话，我们只就只能用一些其他手段，比如说用这种频率控制，看一下有没有一个ip多次请求接口，你就能识别出其实会有一些额外的这种请求。这个是要去建立一个专门的模型，也是在业务安全风控领域，现在越来越多会用这种大数据的这种思路去对抗业务安全上面的一些问题。我们在web安全里面就不再去深入讨论业务安全的东西。



然后今天最后一种漏洞是弱口令漏洞。这个可能不需要多讲，我们每个人都有接触过，对吧。

今天重点介绍的其实是对于这种存在弱口令的地方，如何去暴力破解？暴力破解时候应该注意什么？如何防御？弱口令漏洞给他做一个定义的话，就存在输入口令功能的这个地方存在了强度较弱的口令或者系统默认口令，有些系统默认口令强度很强，但是它毕竟是默认的，所以也是可以称为一个弱口令，可以被攻击者轻易的猜测或者暴力破解的，这种都可以称为是弱口令漏洞。危害程度也是视情况而定。这里有一些常见的后台用户名及密码。

弱口令漏洞的形成

1. 使用系统默认口令
   tomcat：tomcat/tomcat。
   weblogic：weblogic/weblogic

2. 未对用户密码强度进行要求
   此时用户很容易设置类似“123456”的弱口令

3. 未对暴力破解进行合理防御

那么弱口令漏洞是如何形成的呢：

- 一种是使用系统默认口令，比如说tomcat默认是没有管理员的，密码默认为tomcat，有些网站管理员就为了省事，就直接默认，赋予了网站管理员的权限。
- 还有一种是对用户的密码强度进行强制要求，现在一些大型的这种网站都会强制用户去使用高强度的口令。
- 另外可以对暴力破解进行合理防御，我们可以限制暴力破解的频率等。

暴力破解

可允许暴力破解的成因

1. 系统输入用户名密码或输入密码位置缺少验证码导致用户口令被暴力破解

2. 系统未设置最大校验错误次数阈值导致用户口令被暴力破解

对于限制了最大校验错误次数阈值的场景，依然要警惕通过使用固定口令遍历用户名的攻击手法。

演示：使用Burpsuite进行暴力破解测试

下面介绍一下暴力破解，它不是一个漏洞，它是一种攻击的方法。利用正确密码和错误密码服务器返回信息的差异进行猜测。

- 当网站没有设置验证码，或密码位数较少，就可以用暴力破解的方式猜测出密码。
- 有时设置了验证码，但没有设置最大错误次数校验，导致可以无限测试，也是暴力破解成立的一个原因。
- 还有一个思路，假如一个用户名只能尝试五次密码。在五次之内用暴力破解破解一个用户难度是非常高的，这时可以考虑用一个固定的密码口令去碰撞不同的用户名。我把这个系统中所有的用户名字跑一遍，跑出一个用户列表。

弱口令漏洞防御

长亭科技 CHAITIN

拍卖行小故事

系统设计者：

1. 使用安全可靠的图形验证码
2. 一定场景下，可以设置校验失败最大次数阈值

用户：

1. 避免使用系统默认的口令
2. 避免在多个系统中使用同一套口令
3. 提高口令强度，8位以上，大小写字母、数字、特殊字符

- 如何防御弱口令漏洞，一般来说是使用一个安全可靠的图形验证码，图形验证码现在随着OCR技术的出现，验证码会存在大量的这种被识别出来的风险，有一些成本上升的对抗，比如说拼图，还有就是把一些东西传转到正确的位置，但没有说一定是有一种很安全十分安全可靠的验证码，但是我们要尽可能的使用安全考核，让暴力破解的攻击者成本上升。
- 在一定场景下，我们可以去做这种错误最大次数阈值的这种限制，比如说网银这种，一般来说它的对用户的保护性会更强一些，如果一旦这个密码失窃了，可能引发的就是用户里面的几十万几百万的存款就被盗窃，这个时候就可以给用户下发这种比较强的一天之内只能验证三次或者五次这样的错误次数，即使它被T掉线了，他也知道自己的账号已经收到一定风险，它也可以人工再去开通，对他来说也是一种保护。
- 对用户来说应该去避免使用系统默认的口令，就是对管理员来说，一般来说是管理员在安装完系统之后，避免使用默认的口令。
- 对于其他普通的网民来说，不要在多套系统中使用同一套口令，要提升这个口令的强

度设置为八位以上，然后大小写字母数字特殊字符，这也是等级保护里面对口令的要求的。

然后这边今天这个就介绍介绍内容已经结束，下节课我们来看逻辑漏洞（下）。