# 羊毛党利用群控和接码平台薅尽羊毛

## 本节课用时

羊毛党作恶完整路径（3 分钟）　　知己知彼

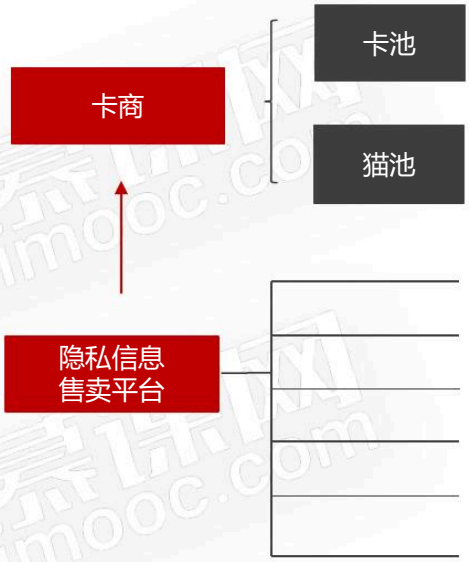
# 羊毛党作恶完整 路径

## 能产生**利益**的地方，就离不开黑灰产

即使利润很低，黑灰产依然会想办法通过**机器**，**批量**操作来**规模获利**

**假设我是羊毛党，去刷无门槛优惠券获利，需要经过哪些步骤？**

1. **搜集优惠信息**：新手福利，优惠券，红包...
2. **数量庞大的账号**：通过这些账号去批量拿优惠券
3. **自动的拿优惠券**：一个账号拿一张优惠券，不可能手动
4. **销赃渠道**：批量获取优惠券就是为了获利



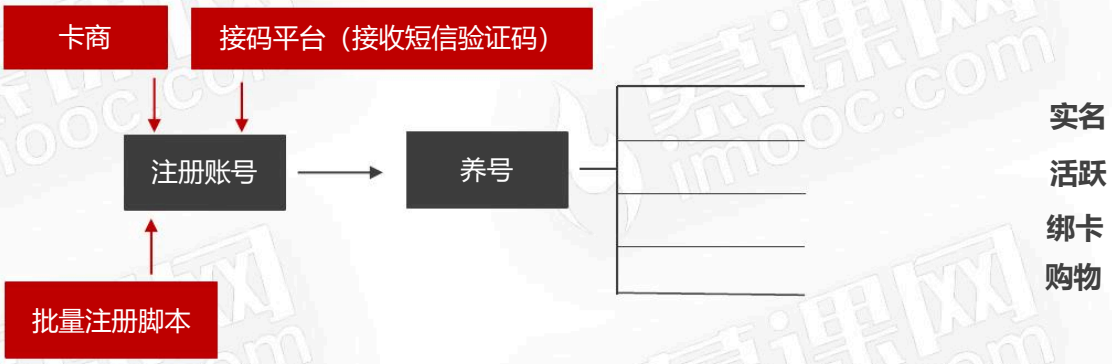

回收卡

物联网卡

黑卡

虚拟卡

实体卡


卡商
接码平台（接收短信验证码）
注册账号
养号
批量注册脚本
实名
活跃
绑卡
购物


成品号
作弊
销赃
(淘宝, 闲鱼, 各种群)
搜集优惠信息
(各类群，爬虫工具)
自动工具
(自动抢优惠券)

# 优惠券场景下被薅羊毛的业务逻辑漏洞复盘

# 本节课大纲

**代码不严谨**，导致了优惠券漏洞　　**羊毛党倾巢出动，大**

**代码不严谨，导致了优惠券漏洞**

**代码不严谨，无门槛优惠券场景下引发的业务逻辑漏洞：**

1. 发券漏洞
2. 用券漏洞
3. 库存漏洞
4. 监控漏洞
5. 运营漏洞

## 无门槛优惠券场景下**发券漏洞**：

1. 无门槛优惠券没有**限制总数**
2. 无门槛优惠券没有**严格限制领取资格**
3. 无门槛优惠券不知道哪些用户(**身份核实**)领取了券

## 无门槛优惠券场景下**用券漏洞**：

1. 使用券时，没有核验**用户的异常行为特征**
2. 使用券时，没有**横向比较用户的历史下单信息**
3. 使用券时，没有**核验异常用户的共同特征**

## 无门槛优惠券场景下**库存漏洞**：

1. 没有在库存机制**设定一个上限**
2. 订单进入**待发货状态**时，没有**再做一次订单核验**

## 无门槛优惠券场景下运营漏洞：

1. 优惠券人工设置，没有控制权限
2. 优惠券人工设置，没有大于1次的审核机制
3. 订单实时数据指标监控和警报没有做好

# 基于领域驱动分析优惠券场景下风控的架构设计

# 本节课大纲

为什么需要领域驱动设计

## 架构思想

**分层设计**：Service 层, Controller 层，Dao 层

**面向对象**：User 对象，Order 对象

## 设计账户之间的转账

**分层设计**

Service 层：1. 转账服务接口，2. 接口实现类，3. 账户转账的业务逻辑

Controller 层： 调用转账服务接口

**面向对象**

1. 每个账户实体是 Account 对象

2. 账户交易的数据是 AccountInfo 对象

## 账户之间的转账增加需求

1. 添加账户的信用等级判断：在 Service 层添加具体业务逻辑
2. 记录转账明细：在 Service 层添加具体业务逻辑

Service 层过于臃肿, 各个 Service 间关系错综复杂

# 这不是真正的面向对象的编程

# 领域驱动设计( DDD )的架构思想

账户实体（Account 对象）:

**账户的属性**：姓名，**信用等级 （信用等级对象类）**

**账户的行为**:

借贷，转账 等**账户应有的行为**

每种行为的**每一笔交易都对应交易明细**

## 使用领域驱动设计( DDD )的架构思想设计账户间转账需求

Service 层不再关注转账的细节，只负责将实体对象组织起来

具体的转账细节由对应的领域对象处理

1. 转账交易：调用账户实体的转账行为
2. 账户信用等级判断：由账户的信用实体对象处理
3. 转账细节记录：账户实体的转账行为处理

领域驱动设计的架构思想能够让代码具有高的内聚性


业务人员
从业务角度（流程，功能，交互）
讨论需求
提取模型
设计模型
从模型角度（业务维度被抽象隐藏）
研发人员
代码实现

## 规则域

1. 规则上下文
2. 决策上下文

## 用户特征域

1. 用户画像
2. 变量指标上下文

## 用户接口层

（ 请求应用服务 ）

## 应用层

1. 事件发布和订阅
2. 请求领域服务

## 领域层

1. 封装值对象（ Value Object ）
2. 通过 仓储接口 和数据库交换

## 基础层

（ 工具类, 配置类, 公共类 ）

# 羊毛党利用群控和接码平台薅尽羊毛

## 本节课用时

需求背景（1 分钟）

风控需求（2 分钟）

为什么要单独搭建风控引擎

后面课程风控引擎的架构思路会围绕这些需求进行搭建

# 需求背景：电商风控面临的 挑战

## 1. 业务场景多，风险点多

**场景风控**：不同的场景会有不同的业务风险；风控措施也不同

## 注册场景涉及的风险点

| | |
|---|---|
| 1. 手机号是否真实的手机号 | 措施 1. 发送验证 |
| 2. 输入别人的手机号 | 措施 2. IP限制 |
| 3. 短信通道被大量的恶意刷短信 | 措施 3. 短信数量 |
| 4. 绕过验证码，直接发起请求 | 措施 4. 用户行为 |
| 5. 通过程序漏洞进行拖库 | 措施 5. 堵塞能获 |

## 1. 业务场景多，风险点多

**场景风控**：不同的场景会有不同的业务风险；风控措施也不同

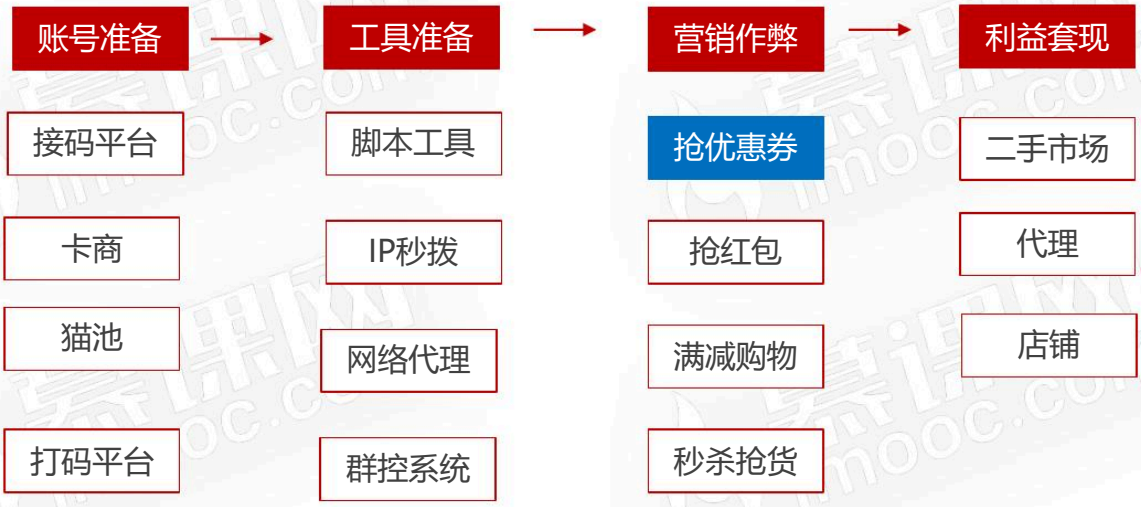
**场景风控 = 场景定义 + 场景风险点**

1. **业务场景多**，风险点多

**场景风控**：不同的场景会有不同的业务风险；风控措施也不同

**场景风控 = 场景定义 + 场景风险点**

2. 黑灰产**工具不断升级**


账号准备
工具准备
营销作弊
利益套现
接码平台
卡商
猫池
打码平台
脚本工具
IP秒拨
网络代理
群控系统
抢优惠券
抢红包
满减购物
秒杀抢货
二手市场
代理
店铺

## 1. 业务场景多，风险点多

**场景风控**：不同的场景会有不同的业务风险；风控措施也不同

**场景风控 = 场景定义 + 场景风险点**

## 2. 黑灰产工具不断升级

**一站式**工具包；

## 1. 业务场景多，风险点多

场景风控：不同的场景会有不同的业务风险；风控措施也不同

场景风控 = 场景定义 + 场景风险点

## 2. 黑灰产工具不断升级

一站式工具包； 工具使用门槛和成本的降低；

## 3. 真人作弊

**黑产新利器：真人众包**

# 需求 1：风控要 低成本 对接业务

1. **接入**成本

业务系统不需要明显感知到风控的存在

2. **无感**验证

验证服务需要毫秒级响应；正常用户完全无打扰；

# 需求 2：风控策略要 健壮和灵活

1. 人工制定和算法策略结合

业务系统不需要明显感知到风控的存在

2. 不同的场景可以配置不同的风控策略

策略可以灵活定制，并可以设置不同优先级

# 需求 3：风控决策效率要做到 实时

1. 任何决策都能**实时响应**

业务系统不需要明显感知到风控的存在

2. 在**大量数据并发**时也能保持调用

策略可以灵活定制，并可以设置不同优先级


场景1
场景2
拖库撞库
规则1
规则2
抢优惠券
抢红包