逻辑漏洞检测

潘玺廷 2019.3.1

逻辑漏洞定义和分类

- 如何定义逻辑漏洞
- 业务逻辑漏洞示例
- 逻辑漏洞的场景和类型

基于黑盒扫描进行检测

- 确定目标，找出核心检测原理
- 重新抽象黑盒扫描器
- 实现过程
- 优缺点分析

逻辑漏洞定义和分类

- 如何定义逻辑漏洞
- 业务逻辑漏洞示例
- 逻辑漏洞的场景和类型

CWE 840：Business Logic Errors

Business logic errors are “**weaknesses** [...]

that commonly allow attackers to **manipulate**

the business logic of an application

OWASP: Business logic vulnerability

- Access Control
- Input Validation
- Authentication
- etc...

找回密码问题

找回密码功能设计缺陷

客户端

服务端

获取短信验证码

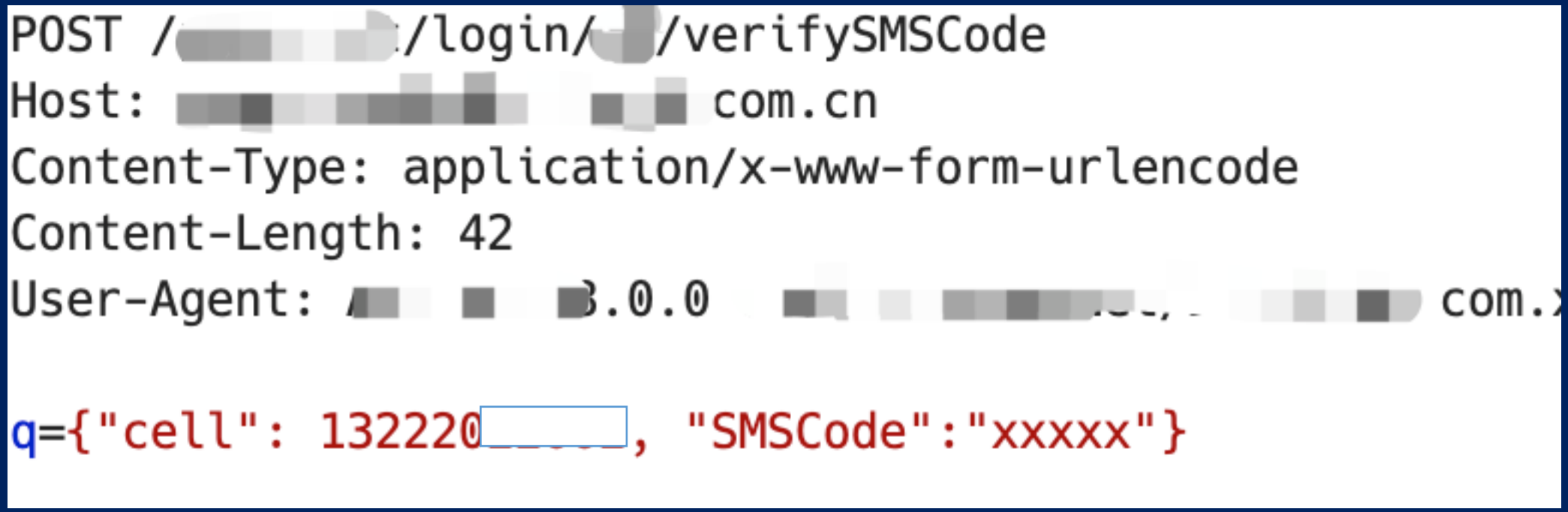
给手机发验证码

手机号 + 验证码

校验手机号与验证码绑定关系

校验验证码有效

找回密码功能设计缺陷—绑定校验缺失



找回密码功能设计缺陷

客户端 服务端

获取短信验证码 → 给手机发验证码

手机号 + 验证码 → 校验手机号与验证码绑定关系

返回凭证

新密码 + 凭证 → 校验凭证 根据凭证修改手机号对应账户密码

新密码 + 凭证 + 手机号 → 校验凭证 根据手机号修改密码

找回密码功能设计缺陷—绑定校验缺失

```
POST /[redacted]/login/[redacted]/setPassword
Host: [redacted]com.cn
Content-Type: application/x-www-form-urlencode
Content-Length: 72
User-Agent: [redacted]/8.0.0 [redacted] com.[redacted]2

q={"cell": 13222[redacted], "new_password":"XinMiMa", ticket":"233333xxxxx"}
```

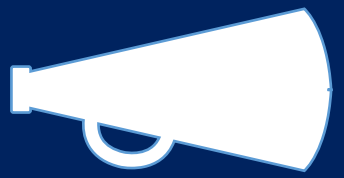
支付结算设计缺陷-金额校验

```
POST /[redacted]OrderAppService/submit[redacted]Order?15[redacted]
Host: order.[redacted]com
Conetnt-Length: 215
Content-Type: application/x-www-form-urlencodeed; charset=UTF-8
Cookie: JESSIONID=xxxxxxxxxxxxxxxxxx; xxxxx=xxxxxx

memberId=[redacted]&order[redacted]=[redacted]
orderPrice=-233.00&orderMsg=11111&invType=-[redacted]
```

常见的场景分类

身份认证



活动

交易订单

即时通讯

支付结算

其他

常见的逻辑漏洞类型

权限校验/访问控制

未授权访问

水平/垂直越权

账号关联覆盖

功能缺失

无效的反自动化手段

缺少数据并发读写锁

请求重放

失效的签名算法

校验不当

变量覆盖

上下限校验

校验标准不一致

缺少绑定校验

业务逻辑漏洞痛点

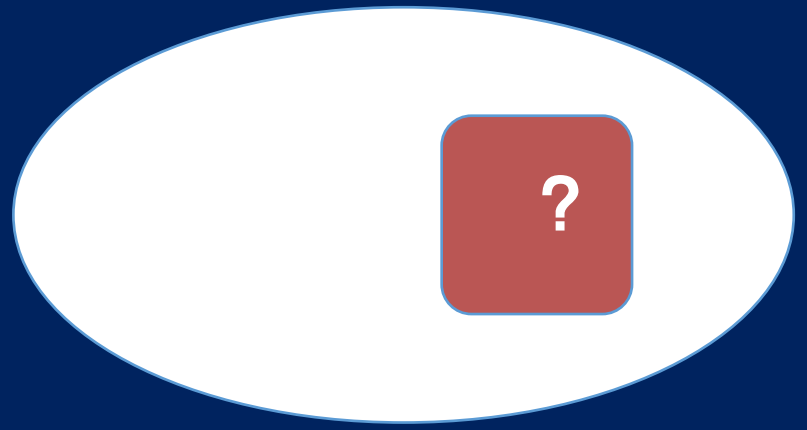
- 再资深的程序员也会犯错
- 一般的功能测试中很难覆盖
- 无视防御设备

基于漏洞类型的安全测试

- 黑盒扫描器
- 白盒审计
- 人工挖掘

基于业务逻辑的安全测试

- 人工挖掘
- ？？？



基于黑盒扫描进行检测

- 确定目标，找出核心检测原理
- 重新抽象黑盒扫描器
- 实现过程
- 优缺点分析

判断服务端应用是否正确鉴权

- 水平越权
- 用户敏感信息泄露
- 未授权访问

检测核心原理-已知鉴权方式

- 前提条件，对于某一业务 Web 应用，我们拥有两个不同的测试账号：Ua，Ub
- 要已知如何登陆此应用，并已知此应用如何鉴权：Aa，Ab
- 对于此应用的接口，构造 FUZZ 测试集，尝试获取数据：

RESP1 = REQ(IDUa + Aa)

RESP2 = REQ(IDUa + Ab)

RESP3 = REQ(IDUa)

- 如果 RESP1 ~= RESP2 则判断此接口可能存在平行越权。
- 如果 RESP1 ~= RESP3 则判断此接口可能存在未授权访问。
- 如果 RESP3 中存在 用户敏感信息 则判断存在用户敏感信息泄露。

检测核心原理-未知鉴权方式

- 重放包含有 IDUi 的请求，无视鉴权方式。
- 构造 IDUi - IDUn 的请求。替换值 IDUf(i) 由遍历函数 f 控制，
  替换时可能是整数递增，手机号，城市甚至是经纬度。
- 尝试剔除鉴权标示，再次重放。

RESP1 = REQ(IDUi + Ai)

RESP2 = REQ(IDUf(i) + Ai)

… …

RESPn = REQ(IDUn +Ai)

RESP = REQ(IDUi)

- 如果 RESP2 — RESPn 中存在 ~ = RESP1 则判断此接口可能存在平行越权。
- 如果 RESP1 ~ = RESP 则判断此接口可能存在未授权访问。
- 如果 RESP 中存在 用户敏感信息 则判断存在用户敏感信息泄露。

检测核心原理-响应相似度计算

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 221
Content-Type: application/json
Date: Sun, 24 Feb 2019 18:53:11 GMT
Server: nginx

{
    "slideshow": {
        "author": "Yours Truly",
        "date": "date of publication",
        "slides": [
            {
                "title": "Wake up to WonderWidgets!",
                "type": "all"
            },
            {
                "items": [
                    "Why <em>WonderWidgets</em> are great",
                    "Who <em>buys</em> WonderWidgets"
                ],
                "title": "Overview",
                "type": "all"
            }
        ],
        "title": "Sample Slide Show"
    }
}
```

- 状态码
- Content-Length
- Message body 的类型，关键字段，数据类型，数据格式，深度。
- 屏蔽特殊字段（避免FP）

漏洞扫描

1. 获得目标应用的 Request 集合。

2. 添加设定好的Payload，发送给应用。

3. 分析Response，得出漏洞结论，
或是修正Payload添加方式。

逻辑分析

1. 获得目标应用的 Request 集合。（如果已知鉴权方式，增加鉴权数据的构造）

2. 没有固定的 Payload，需要动态构造。

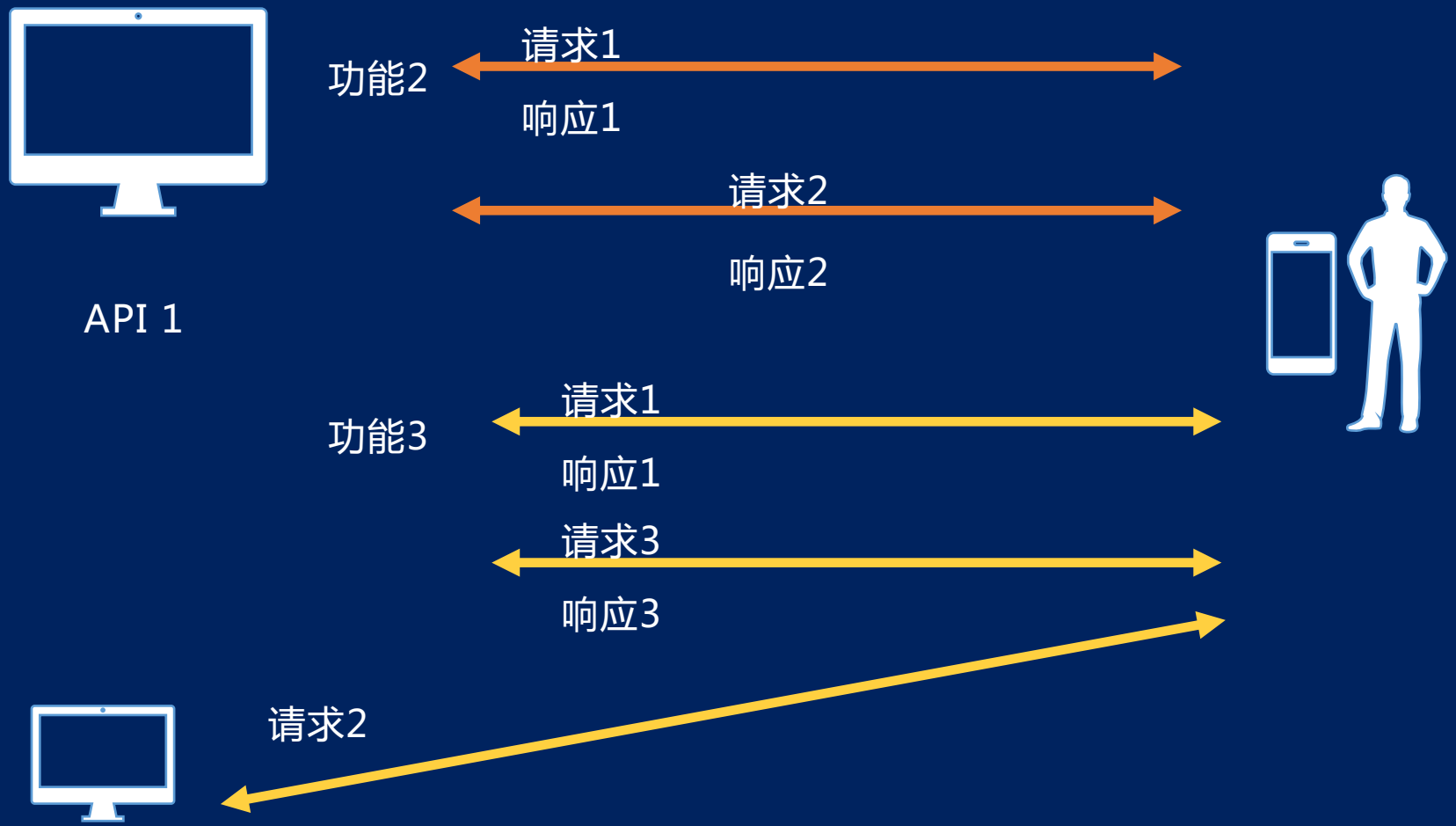
3. 分析 Response 集合，得出可能出现的鉴权问题。

按照已有的漏洞扫描的思路，逻辑分析中缺少上下文数据的替换，大量的请求将会是是无效的，并需要人工增加防误报判断，覆盖率和准确率无法接受。

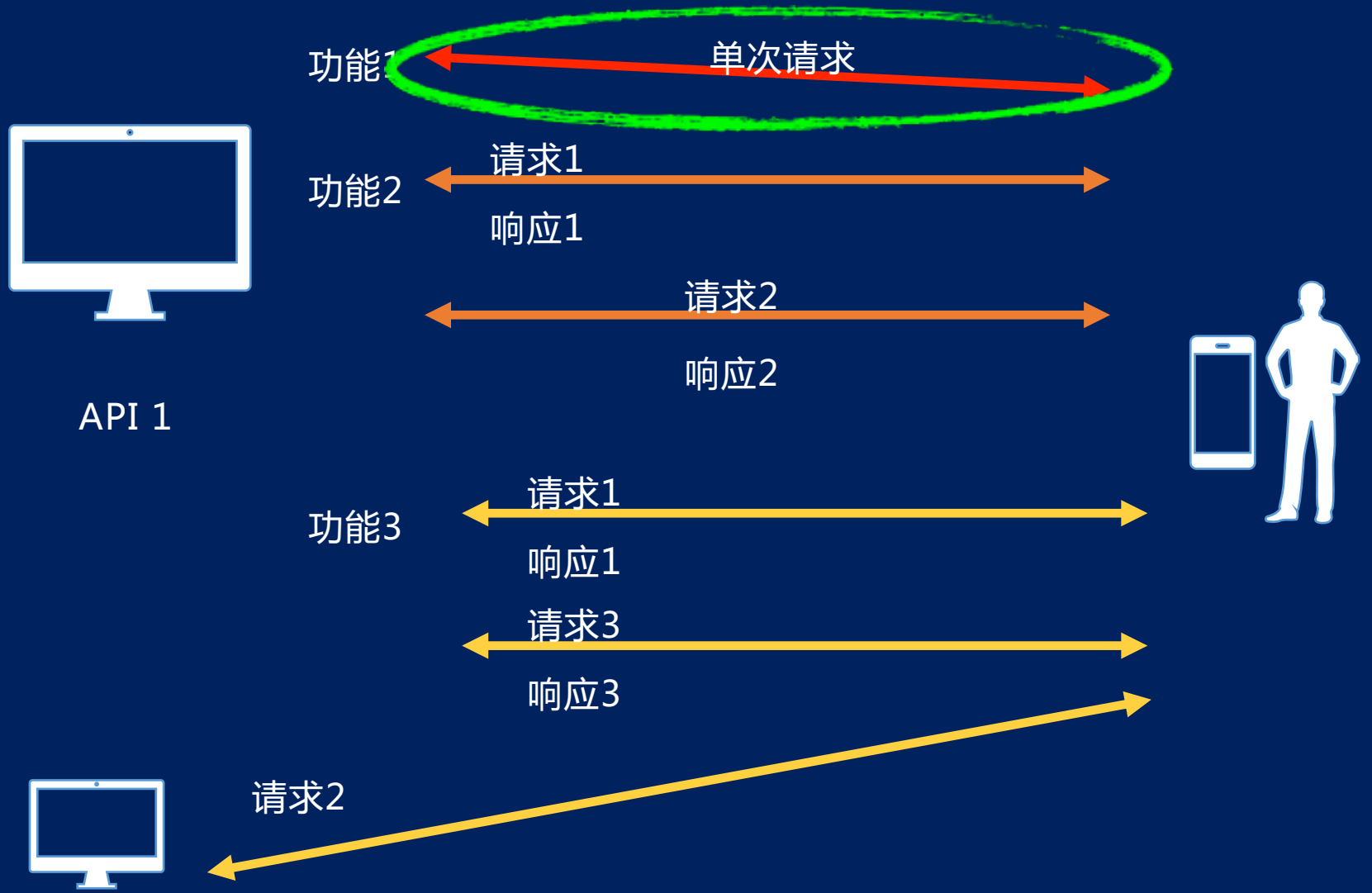
对攻击者漏洞利用的抽象

对攻击者恶意操作的抽象

检测手段-黑盒检测-重新抽象黑盒扫描器

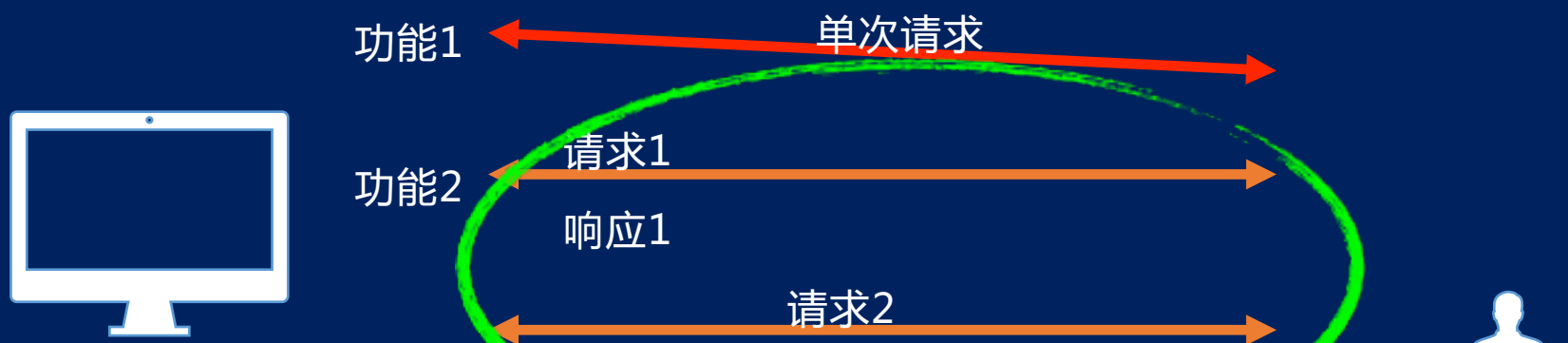


检测手段-黑盒检测-重新抽象黑盒扫描器



将请求的模式规约成模版（Template）

检测手段-黑盒检测-重新抽象黑盒扫描器



功能1

单次请求

功能2

请求1

响应1

请求2

响应2

API 1

将请求和响应的次序规约成流（flow）

功能3

请求1

响应1

请求3

响应3

请求2

响应2

API 2

检测手段-黑盒检测-重新抽象黑盒扫描器

API 1

功能1

单次请求

功能2

请求1

响应1

请求2

响应2

将客户端控制的响应处理和请求构造规约成控制器（controller)

功能3

请求1

响应1

请求3

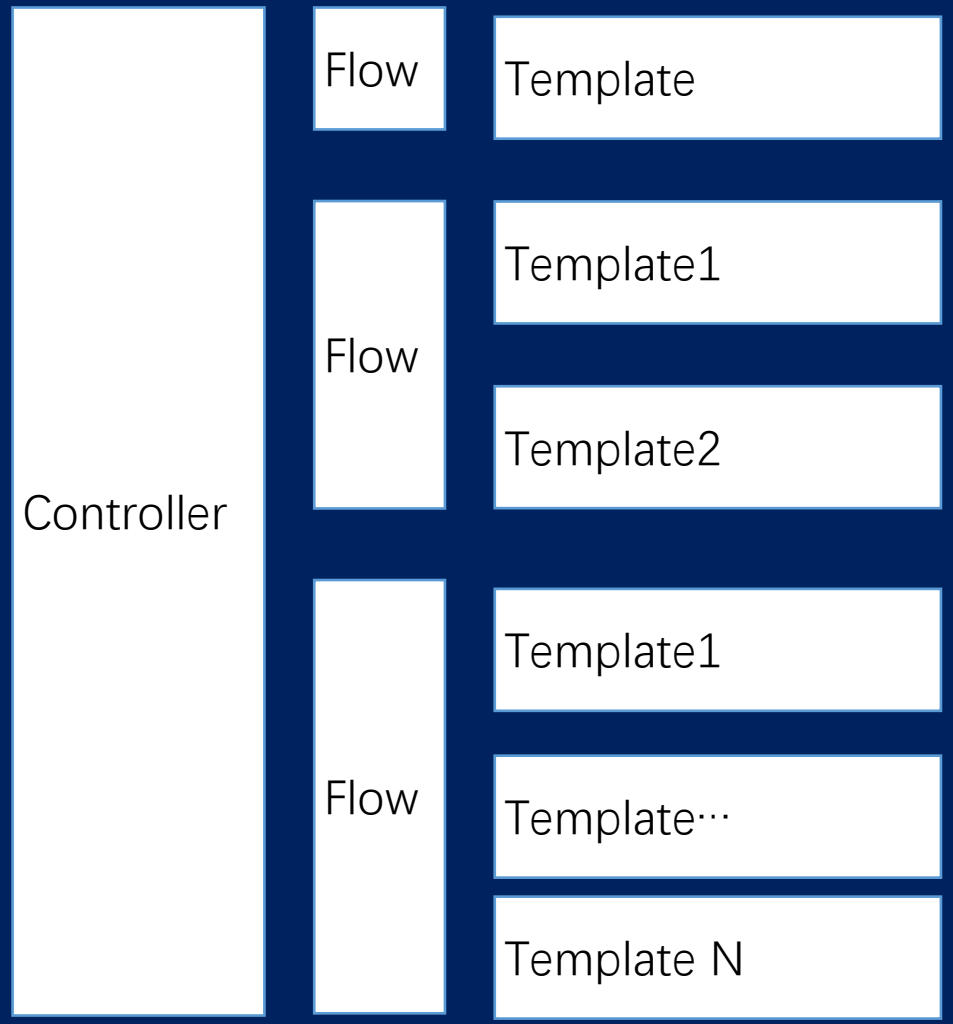
响应3

API 2

请求2

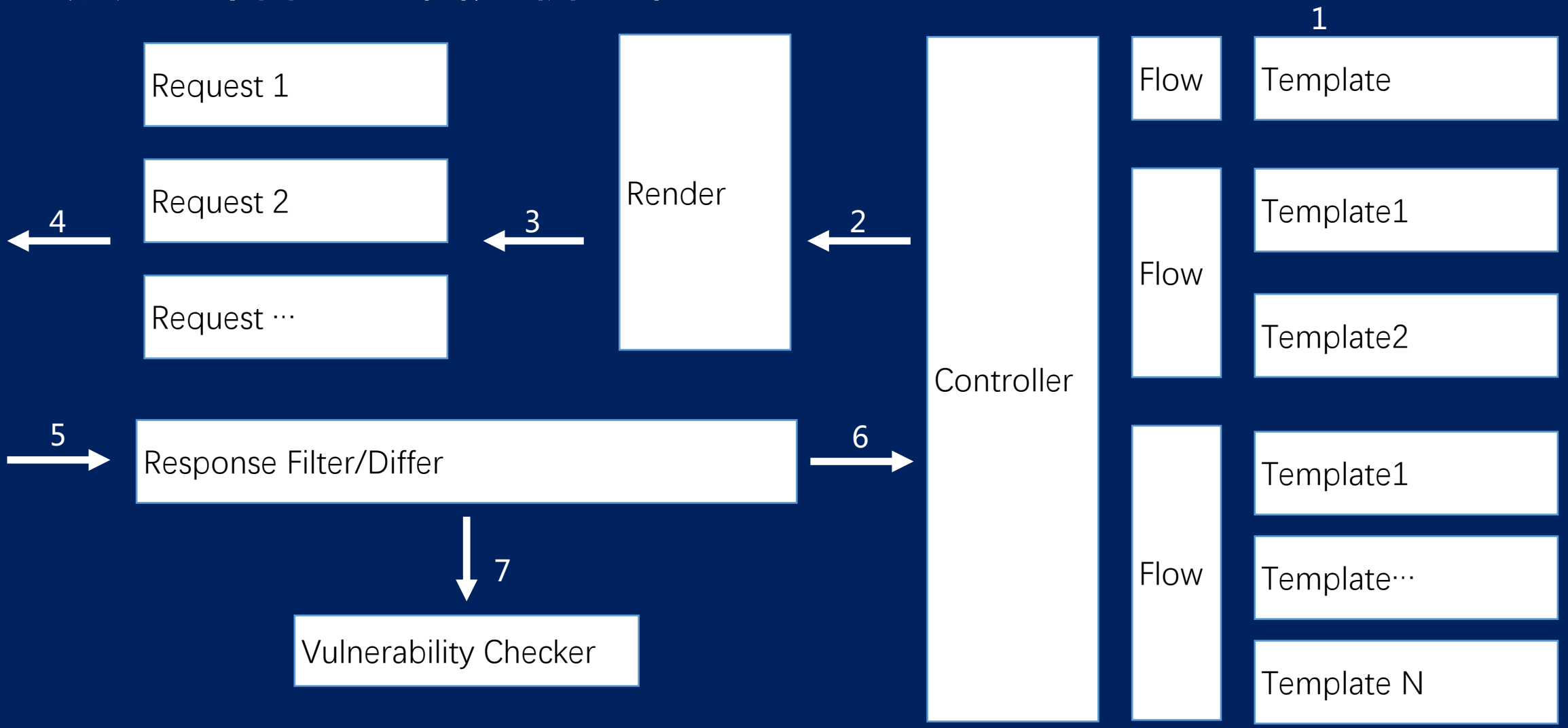
响应2

对于一个Web应用用户的抽象




```
Origin:
/a/b?id=2333&token=$gabahey13370a2b3c4d
Template:
/a/b?id=<int-4-2000~3000>&token=<auth-common-driver>
```

黑盒扫描器逻辑分析过程



检测手段-黑盒检测-实现过程

应用抽象的构建

从应用流量中分析

1. 对于一个单一应用，用户的流量行为是趋于一致的。
2. 上下文重叠参数的跟踪，得出单一或线性的 Flow。
3. 增加用户跟踪，对用户的请求序列做规约。

从客户端分析

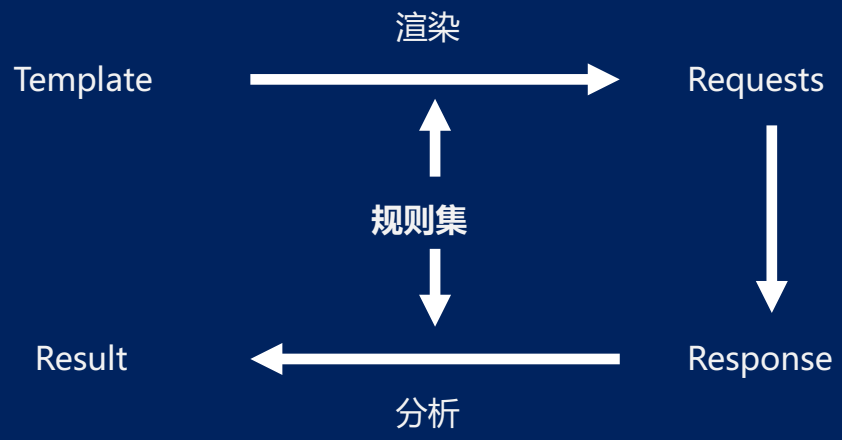
1. 配合客户端自动分析工具进行抓包
2. 通过时序和上下文的重叠参数跟踪，得出线性Flow。
3. 很方便得出三方API调用次序。

基于功能测试构建

1. 引用质量测试的 Case 和自动测试框架。
2. 使用简易的 DSL 描述 Flow 和 Controller。

准确性：由低到高
成本：由低到高

渲染机制和规则集



- **通用规则集**
  - 定义 一般模版的渲染方式。
  - 定义 response 分析的模式。
  - 接受外部的 patch，进行二次渲染。
  - 调用外部接口补齐缺失数据。

- **场景规则集**
  - 人为总结规律的体现，支持外部自定义。
  - 参数黑白名单。
  - 指定某类型参数数据填充的方式。
  - 需要调用其他接口进行二次验证（精准FUZZ）

基础设施依赖

- 技术体系依赖
  - 自动账号权限获取
  - 状态机修正
- 流程与规范依赖
  - 安全评估流程
  - 功能测试流程
  - 一致的接口开发规范

检测手段-黑盒检测-优缺点讨论

由于无法通过自动化的手段得知目标应用的**内在属性**和**外部方法**的约束关系，通过黑盒完全检测逻辑漏洞几乎是一件工程实践和理论上都行不通的方法。

检测手段-黑盒检测-优缺点讨论

但是尽可能的抽象/模拟人工的测试方式，

减少不必要的FUZZ来提高效率

增加对请求上下文的控制提高覆盖率

对规则集进行长期的积累和修正

对于一部分逻辑弱点，黑盒检测可以提供半自动化的检测。

THANKS