网络游戏如何做竞态测试

------基于时序分析的测试方法

OWASP 中国

The Open Web Application Security Project

个人介绍

OWASP 中国
The Open Web Application Security Project

- 赵振兴
- 游戏测试工程师
- 专注在游戏安全测试，可测性

Tencent 腾讯 | 一切以用户价值为依归

网游背景介绍

OWASP 中国
The Open Web Application Security Project

- 网络游戏种类非常多

  MMORPG(大型多人在线角色扮演游戏)

  FPS(第一人称射击游戏)

  多人对战游戏

  休闲类游戏，以及WEBGame与手机网游等等

ps.可以参考下页的截图

角色扮演 (28)			竞技游戏 (17)		网页游戏 (9)	手机游戏 (9)	平台游戏 (4)
地下城与勇士	御龙在天 H	剑灵 H	穿越火线	英雄联盟 H	七雄争霸	怪物大作战 H	QQ游戏
寻仙	轩辕传奇 H	上古世纪 H	战地之王	QQ飞车	部落守卫战	洛克通天塔 H	3366小游戏
幻想世界	斗战神	天涯明月刀 H	生化战场	QQ炫舞	就要K歌 H	三国来了	小熊梦工厂
自由幻想	QQ仙侠传	第九大陆	逆战	炫斗之王	倚天	节奏大师	QQ企鹅
QQ西游	QQ幻想	刀剑2	烈焰行动	Call of Duty OL	夜店之王 H	圣犬帕拉	
QQ仙境	QQ三国	QQ仙灵 H	战争前线	枪神纪	大话神仙	乐斗派	
大明龙权	QQ封神记	疾风之刃	NBA2K OL H	QQ炫舞2 H	Q宠大乐斗	三国塔防魏传	
万王之王3	QQ华夏	天刹	英雄岛		QQ水浒	欢乐斗地主	
绿色征途	天堂	怪物猎人OL H	QQ堂		蜀山传奇	QQ餐厅	
九界	天堂II		QQ音速		› 更多	› 更多	

网游背景介绍

OWASP 中国
The Open Web Application Security Project

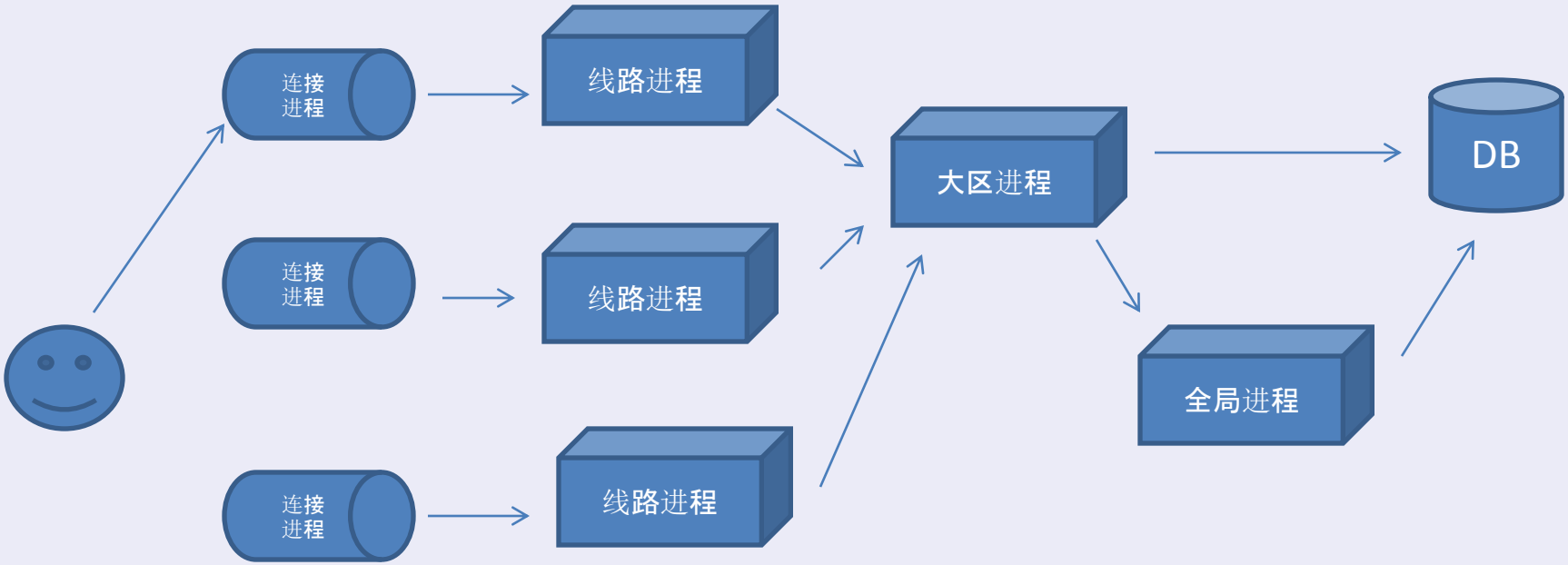
- 各种游戏技术也是种类繁多

  eg. mmog

OWASP 中国
The Open Web Application Security Project

网游后台通常是什么样子的？

MMOG网游后台架构



OWASP 中国
The Open Web Application Security Project

- 网游的安全测试与传统web application的安全测试有什么不同么？

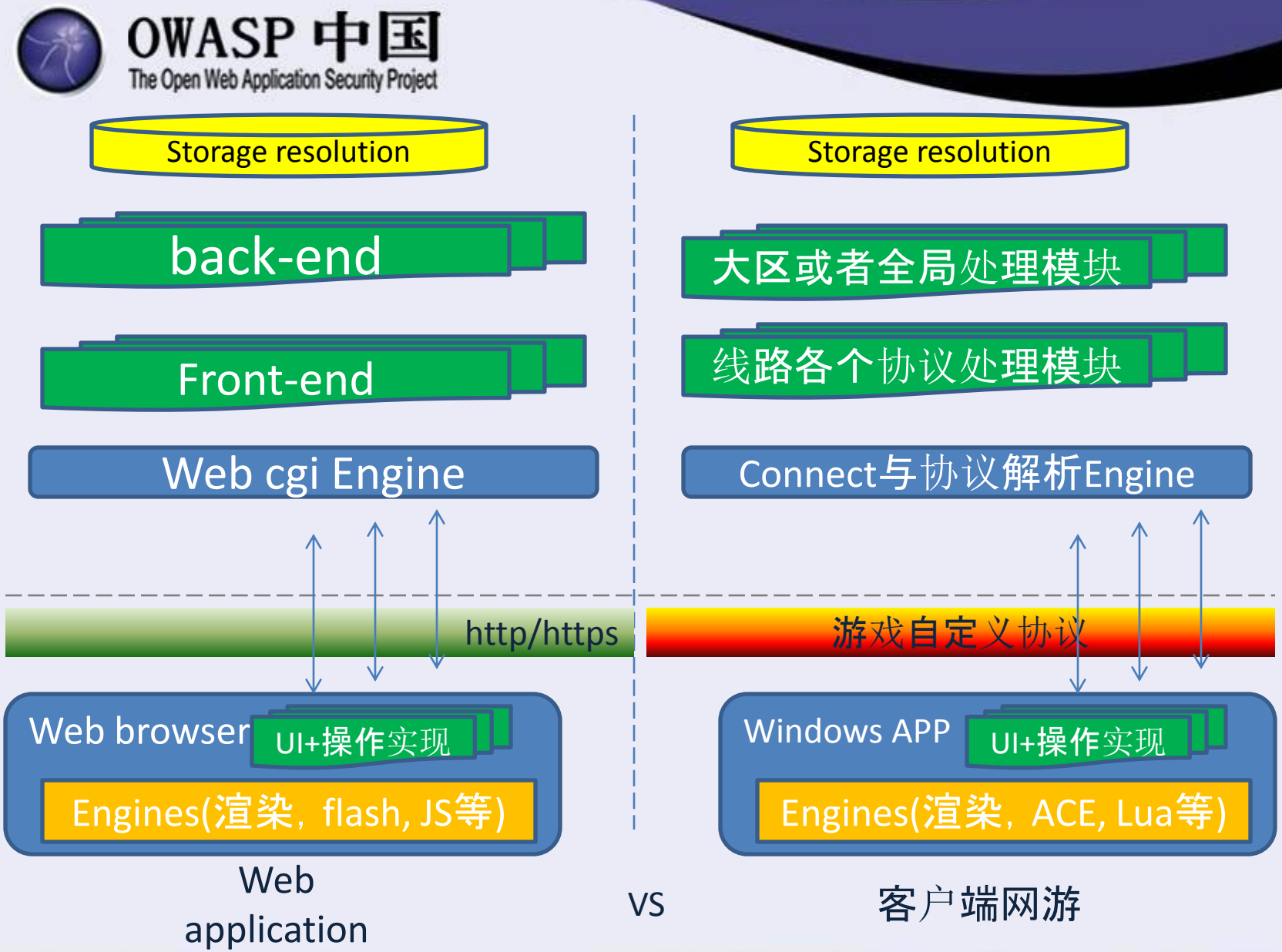
网游测试与web测试的区别

OWASP 中国
The Open Web Application Security Project

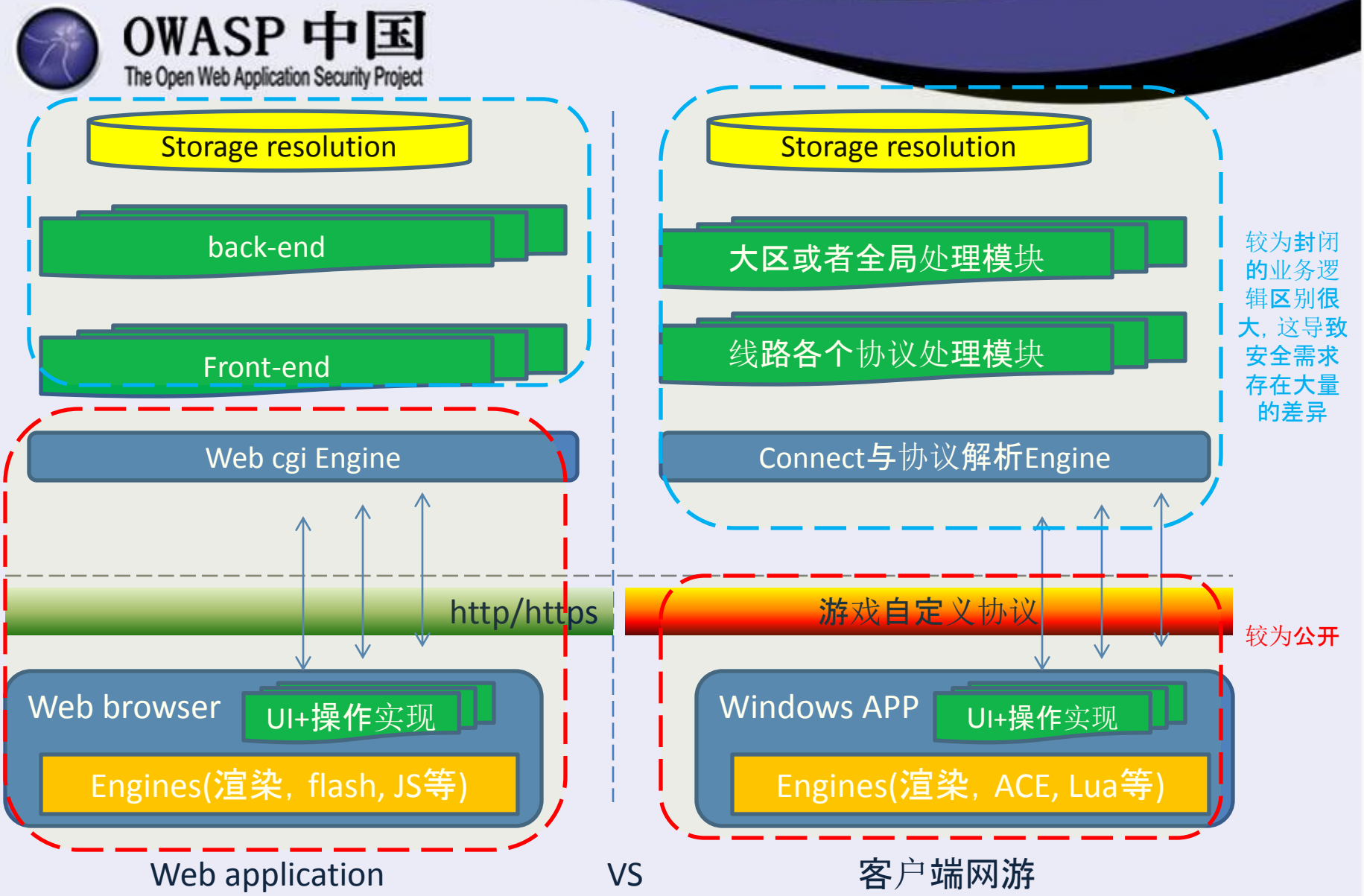
- 首先我来思考下，网游与web应用，在实现上技术有哪些区别？
- 我们对网游的安全需求 与 web应用的安全需求有那些区别？
- 测试这些安全需求的方法应该有那些区别呢？

- 带着这三个问题，我们来一起分析下……

实现上的技术区别

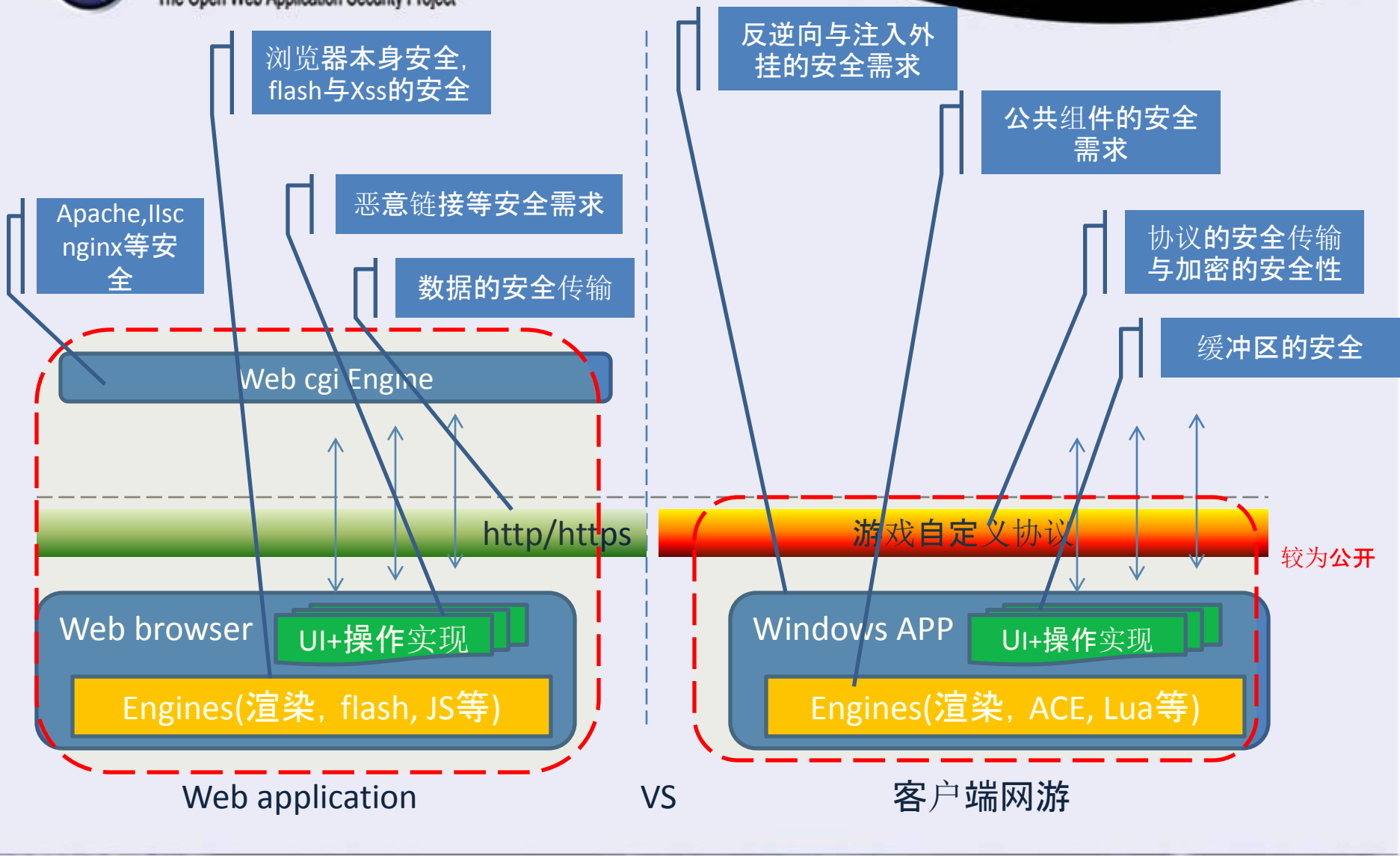

Storage resolution
back-end
Front-end
Web cgi Engine
http/https
Web browser
UI+操作实现
Engines(渲染，flash, JS等)
Web application
VS
Storage resolution
大区或者全局处理模块
线路各个协议处理模块
Connect与协议解析Engine
游戏自定义协议
Windows APP
UI+操作实现
Engines(渲染，ACE, Lua等)
客户端网游

安全需求的区别

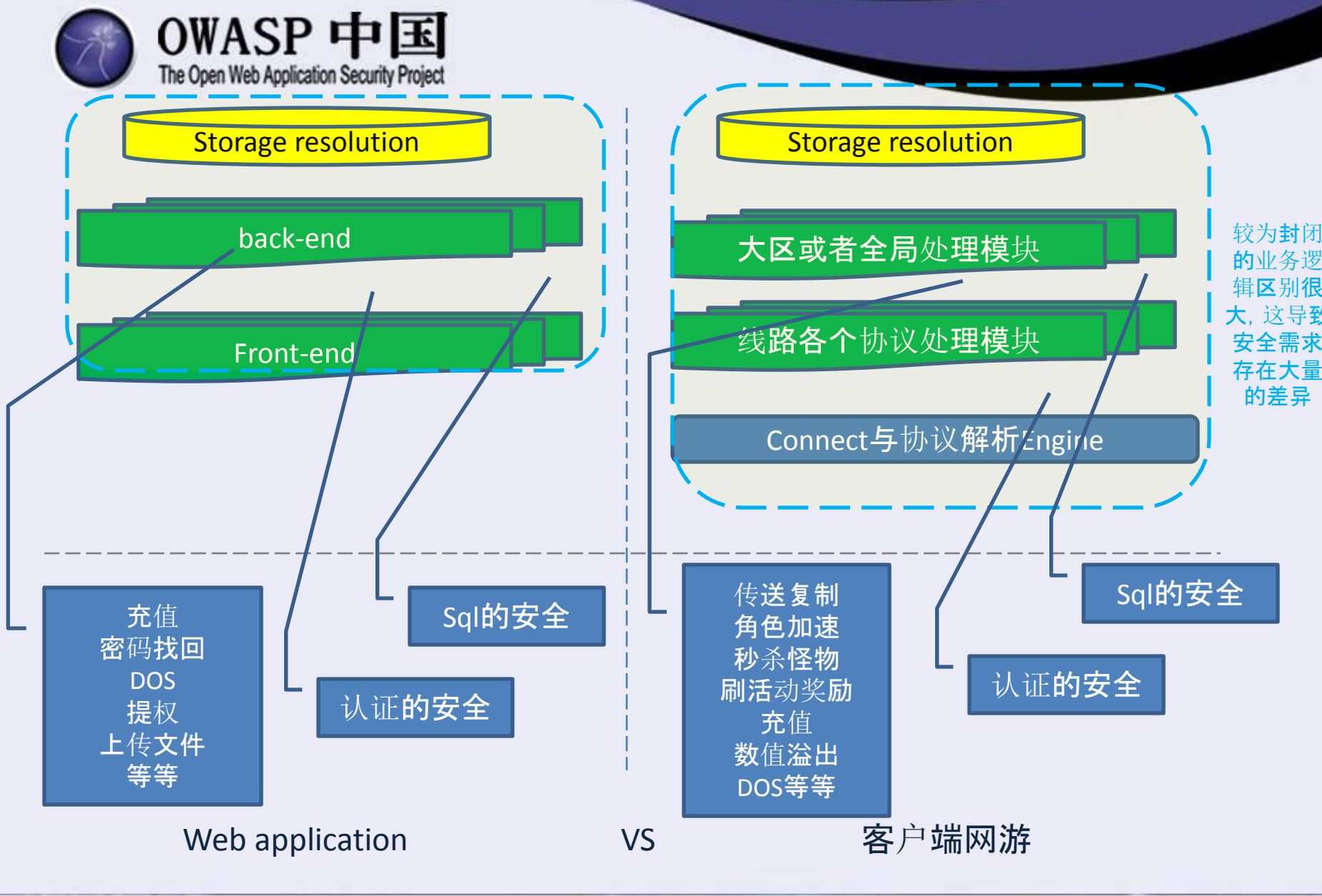

OWASP 中国
The Open Web Application Security Project
Storage resolution
back-end
Front-end
Web cgi Engine
http/https
Web browser
UI+操作实现
Engines(渲染，flash, JS等)
Web application
VS
Storage resolution
大区或者全局处理模块
线路各个协议处理模块
Connect与协议解析Engine
游戏自定义协议
Windows APP
UI+操作实现
Engines(渲染，ACE, Lua等)
客户端网游
较为封闭的业务逻辑区别很大，这导致安全需求存在大量的差异
较为公开

安全需求的区别枚举

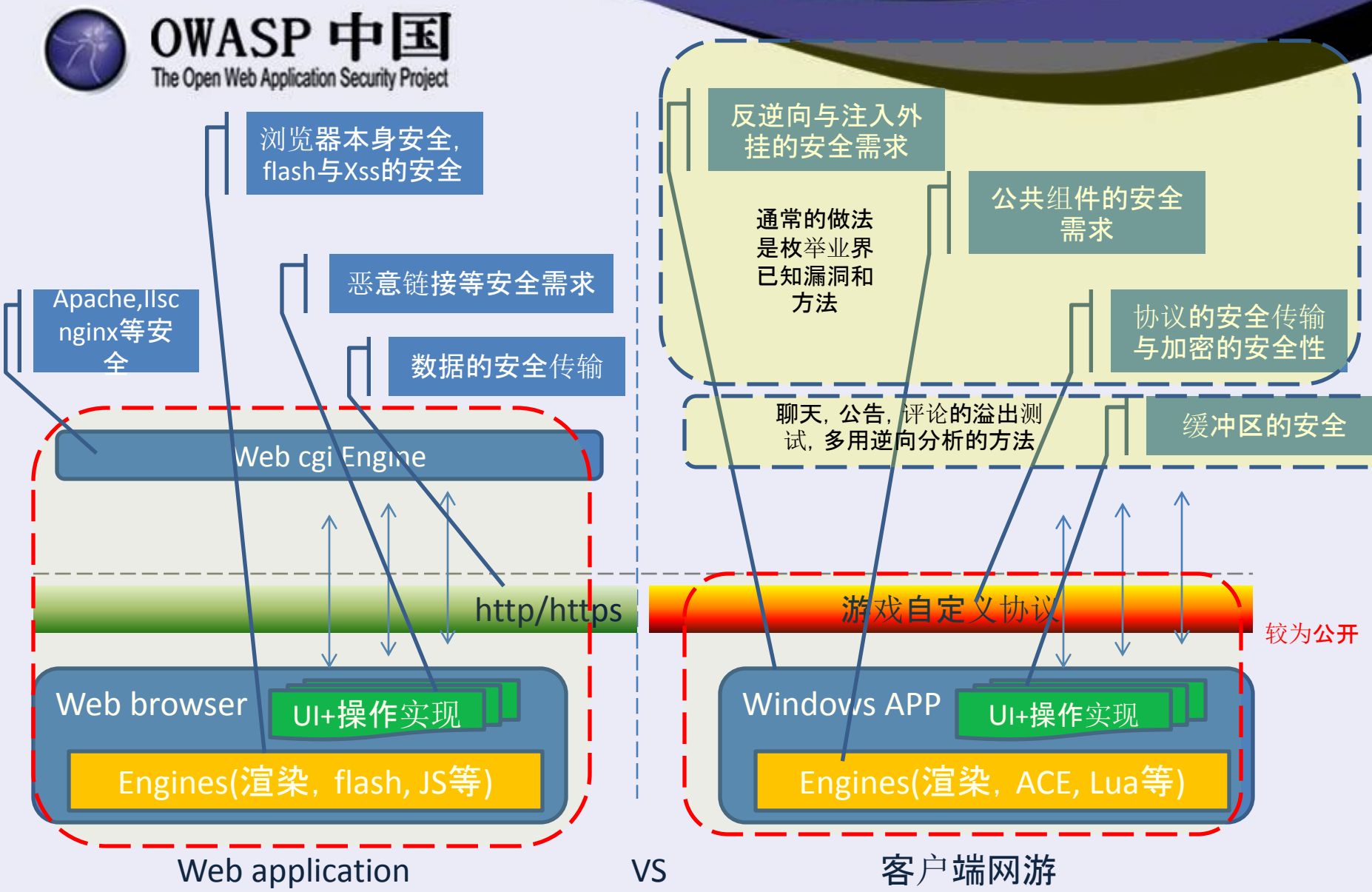
OWASP 中国
The Open Web Application Security Project



安全需求的区别枚举


OWASP 中国
The Open Web Application Security Project
Storage resolution
back-end
Front-end
充值
密码找回
DOS
提权
上传文件
等等
Sql的安全
认证的安全
Web application
VS
Storage resolution
大区或者全局处理模块
线路各个协议处理模块
Connect与协议解析Engine
传送复制
角色加速
秒杀怪物
刷活动奖励
充值
数值溢出
DOS等等
Sql的安全
认证的安全
客户端网游
较为封闭的业务逻辑区别很大，这导致安全需求存在大量的差异

网游的测试关注点和方法



网游的测试关注点和方法

Storage resolution

back-end

Front-end

充值
密码找回
DOS
提权
上传文件
等等

Sql的安全

认证的安全

Web application

VS

Storage resolution

大区或者全局处理模块

线路各个协议处理模块

Connect与协议解析Engine

较为封闭的业务逻辑区别很大，这导致安全需求存在大量的差异

传送复制
角色加速
秒杀怪物
刷活动奖励
充值
数值溢出
DOS等等

Sql的安全

认证的安全

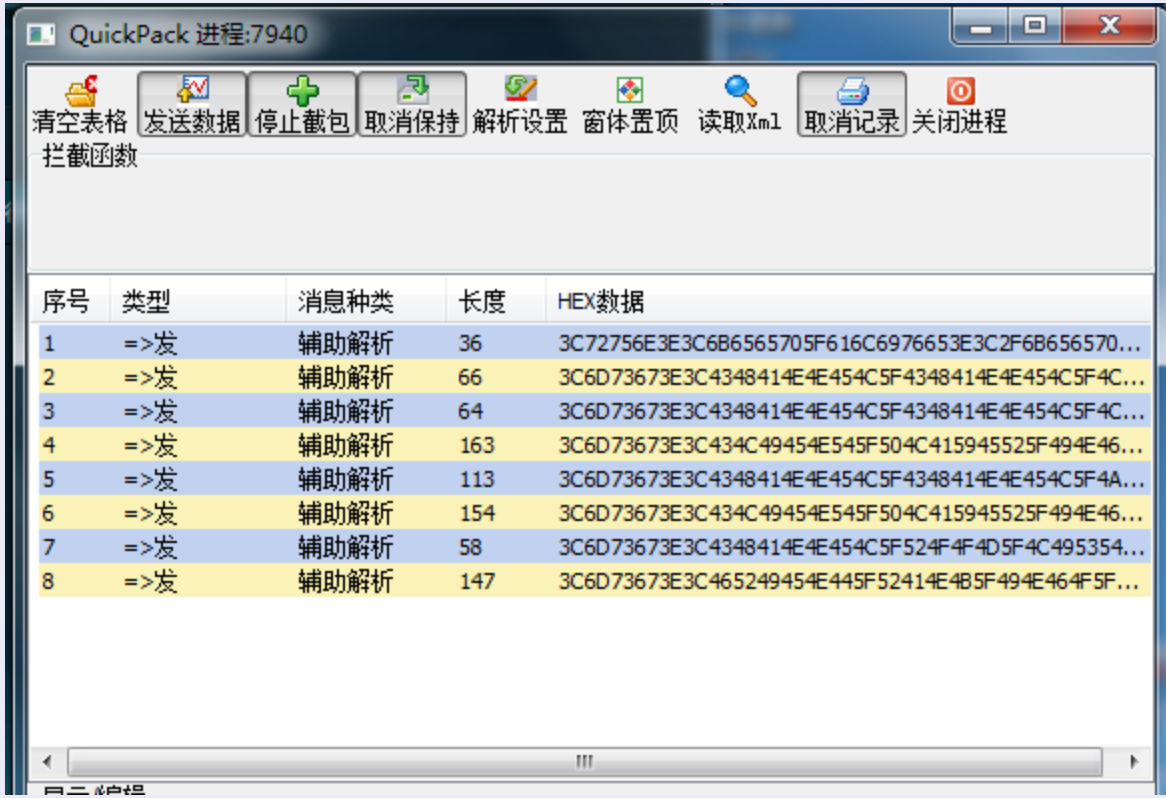
可用的方法：
协议测试，时序测试，code审计

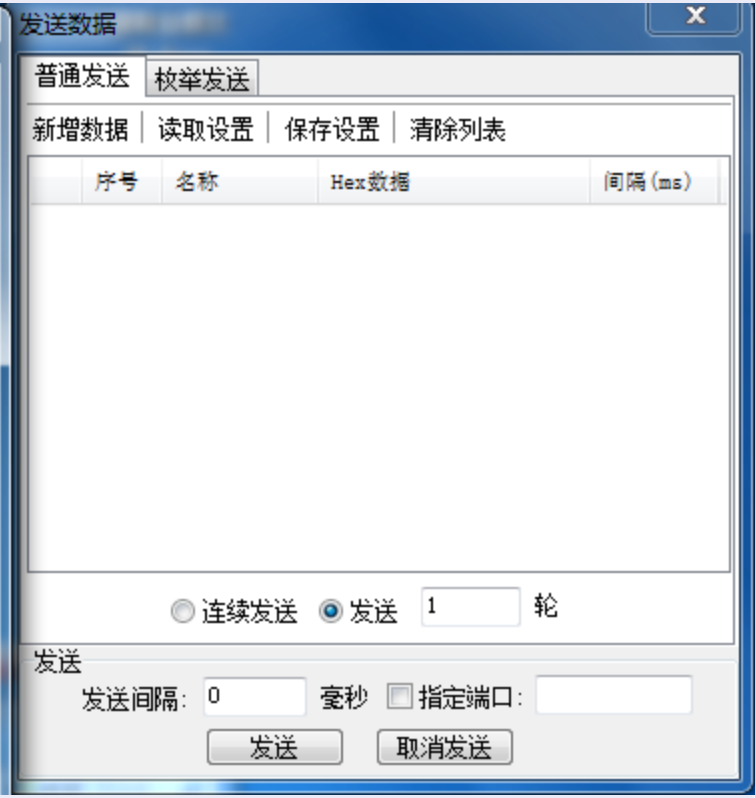
客户端网游

- 下面我来看下最常用的游戏分析与测试工具

基于协议分析的测试

OWASP 中国
The Open Web Application Security Project



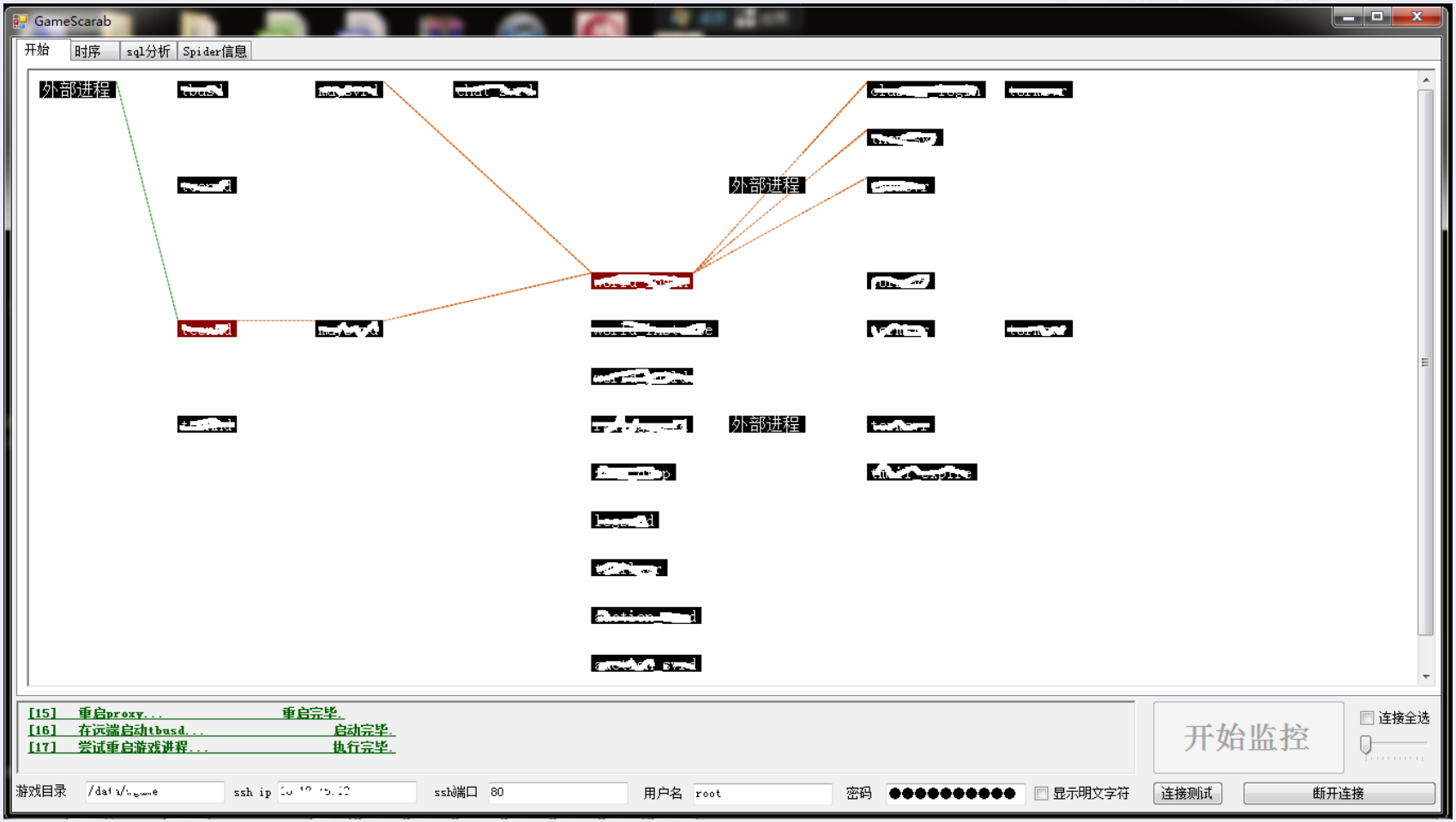


逆向分析工具

OWASP 中国
The Open Web Application Security Project

时序分析工具

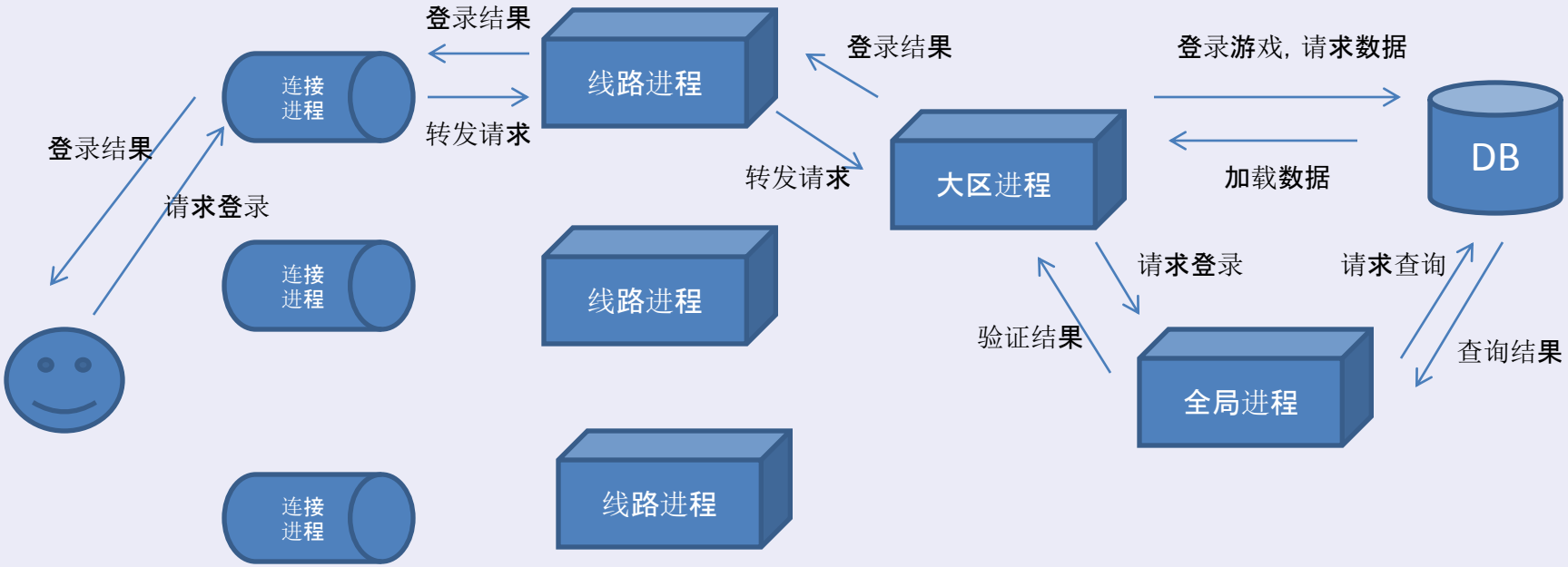
OWASP 中国
The Open Web Application Security Project



- 以上给大家介绍了网游的背景，以及网游测试与传统web application测试的区别
- 下面我来重点跟大家分享网游测试中针对系统可能出现的竞态状态，来做的专门测试，它是依赖于对游戏进程间的通信时序分析来进行的：

什么是基于协议时序的分析

OWASP 中国
The Open Web Application Security Project



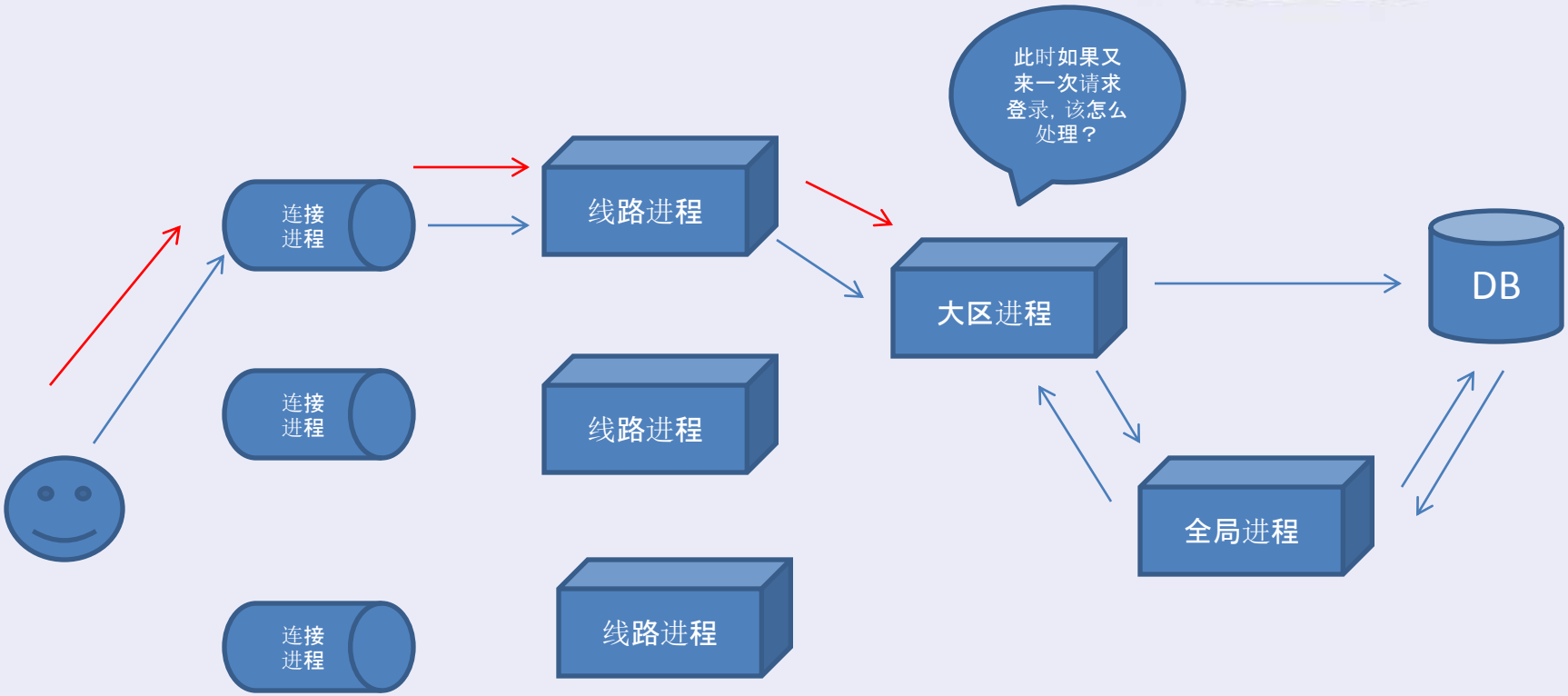
以登录游戏的过程为例子

时序分析背景

OWASP 中国
The Open Web Application Security Project

- 网络游戏的架构一般都是由许多进程协同工作，其实现较为复杂
- 在游戏中，其系统的状态也是非常多，存在一类竞态条件下的BUG，尤其是那些不易被发现的安全漏洞。
- 比如系统中某一个对象共有N(0<N<10)种状态，当它处在S3状态时会在一段时间D1后立即进入S4状态，但如果在这D1期间响应了请求R_action,那么该对象将处于一种不确定的状态中。

竞争状态出现



以登录游戏的过程为例子

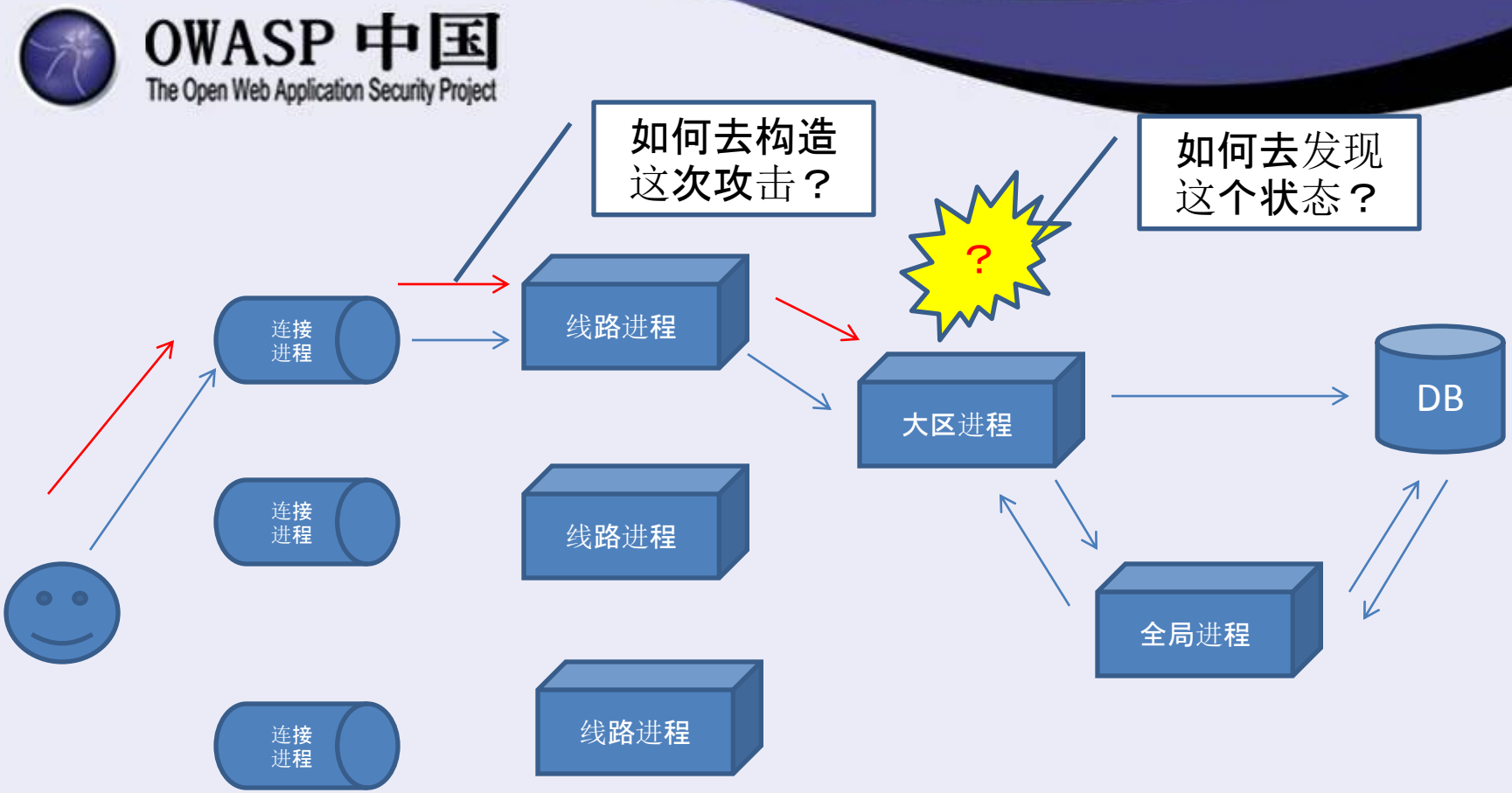
竞态测试需要解决的两个问题

OWASP 中国
The Open Web Application Security Project

第一，我们如何发现状态S3?

第二，如何在这个时间段D1内构造R_action请求来完成这次竞态测试呢？

竞争状态出现



以登录游戏的过程为例子

可以参考的方法

OWASP 中国
The Open Web Application Security Project

- 在服务器进程间架设Proxy,然后分析它们的时序，与交流的数据。这就跟我们的浏览器分析数据包工具是一个思路的。

- 下面举个实际的例子

帮派系统的测试的例子

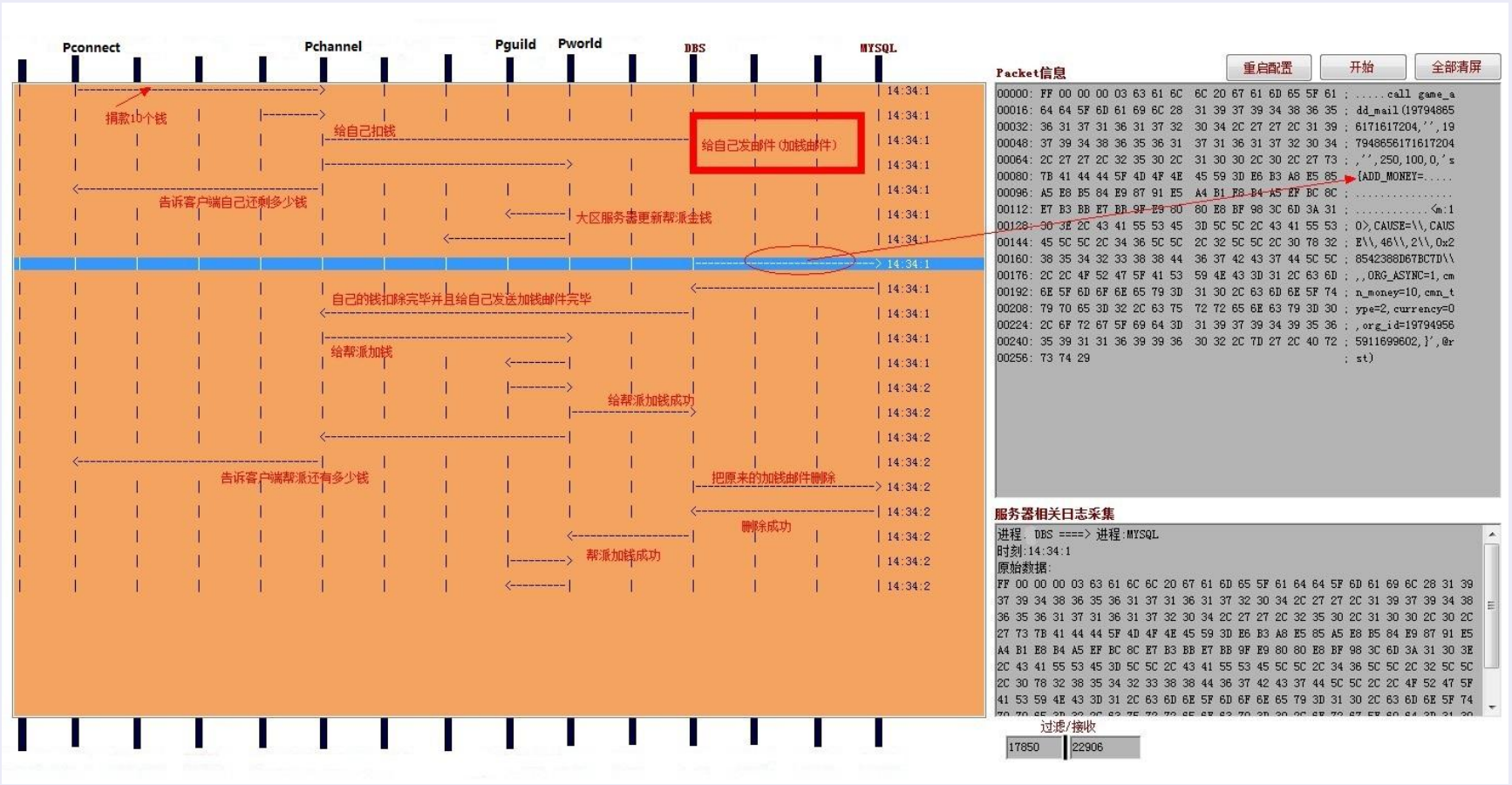
其场景是：
帮派角色，进入帮派领地中给帮派捐献金钱10个单位。

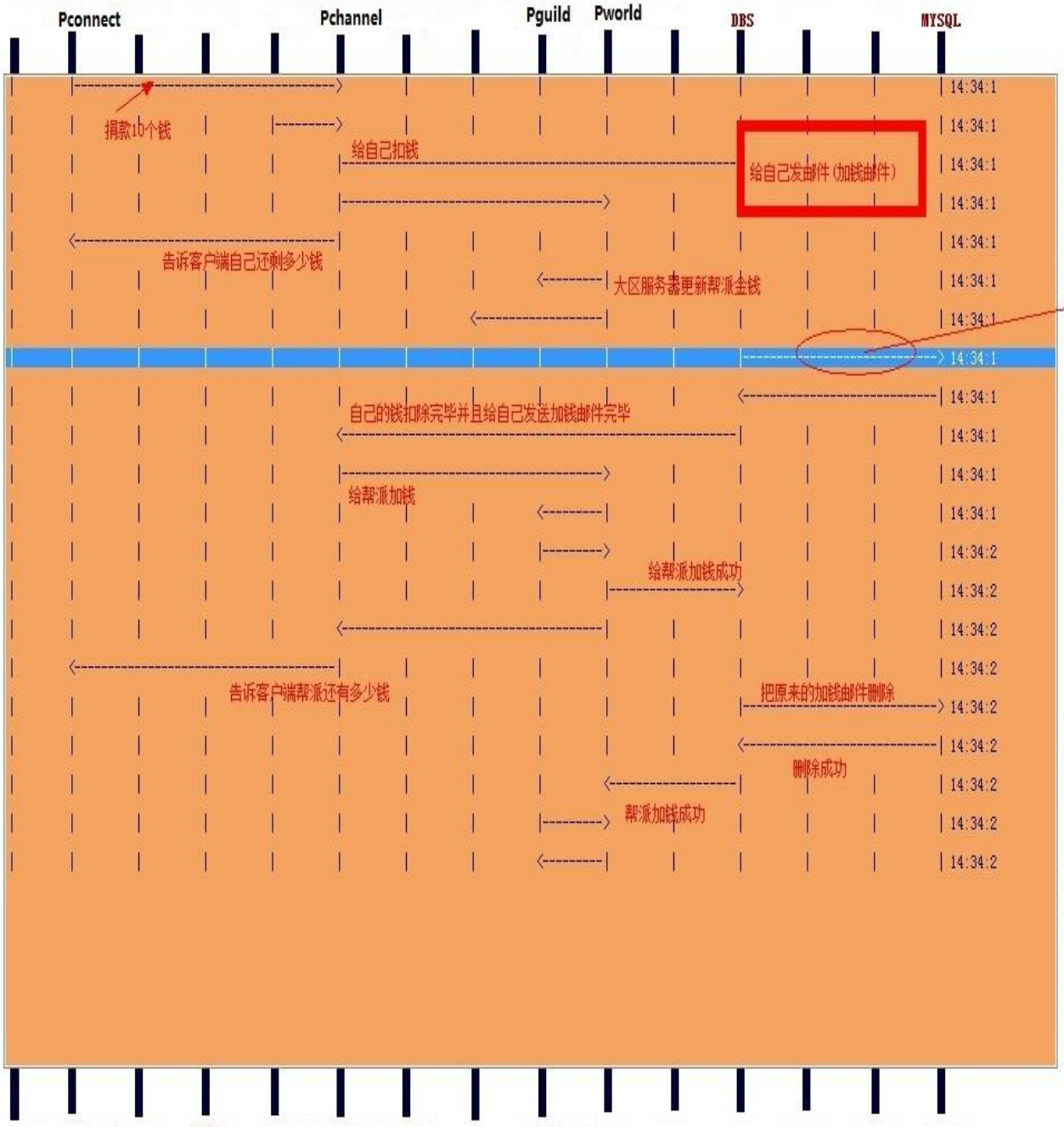
在这里主要涉及到了前端连接进程Pconnect，频道线路进程Pchannel，帮派进程Pguild，大区进程Pworld,数据处理进程DBS以及数据库进程mysql等

因此为了能够进一步的分析出游戏的逻辑，我们可以在这些进程间架设通信的Proxy, 即截获住所有来自源进程要发送的数据包，然后顺势转发给目标进程，并且在这期间打印出进程间的通信时序图。

这样，便可以让测试者发现这次捐款场景究竟在服务器做了什么操作

网络游戏的时序:帮派捐款时序





重启配置 开始 全部清屏

Packet信息

```
00000: FF 00 00 00 03 63 61 6C  6C 20 67 61 6D 65 5F 61 ; .....call game_a
00016: 64 64 5F 6D 61 69 6C 28  31 39 37 39 34 38 36 35 ; dd_mail(19794865
00032: 36 31 37 31 36 31 37 32  30 34 2C 27 27 2C 31 39 ; 6171617204,'',19
00048: 37 39 34 38 36 35 36 31  37 31 36 31 37 32 30 34 ; 7948656171617204
00064: 2C 27 27 2C 32 35 30 2C  31 30 30 2C 30 2C 27 73 ; ,'',250,100,0,'s
00080: 7B 41 44 44 5F 4D 4F 4E  45 59 3D E6 B3 A8 E5 85 ; {ADD_MONEY=.....
00096: A5 E8 B5 84 E9 87 91 E5  A4 B1 E8 B4 A5 EF BC 8C ; ................
00112: E7 B3 BB E7 BB 9F E9 80  80 E8 BF 98 3C 6D 3A 31 ; ............<m:1
00128: 30 3E 2C 43 41 55 53 45  3D 5C 5C 2C 43 41 55 53 ; 0>,CAUSE=\\,CAUS
00144: 45 5C 5C 2C 34 36 5C 5C  2C 32 5C 5C 2C 30 78 32 ; E\\,46\\,2\\,0x2
00160: 38 35 34 32 33 38 38 44  36 37 42 43 37 44 5C 5C ; 8542388D67BC7D\\
00176: 2C 2C 4F 52 47 5F 41 53  59 4E 43 3D 31 2C 63 6D ; ,,ORG_ASYNC=1,cm
00192: 6E 5F 6D 6F 6E 65 79 3D  31 30 2C 63 6D 6E 5F 74 ; n_money=10,cmn_t
00208: 79 70 65 3D 32 2C 63 75  72 72 65 6E 63 79 3D 30 ; ype=2,currency=0
00224: 2C 6F 72 67 5F 69 64 3D  31 39 37 39 34 39 35 36 ; ,org_id=19794956
00240: 35 39 31 31 36 39 39 36  30 32 2C 7D 27 2C 40 72 ; 5911699602,}',@r
00256: 73 74 29                                         ; st)
```

服务器相关日志采集

```
进程: DBS ====> 进程:MYSQL
时刻:14:34:1
原始数据:
FF 00 00 00 03 63 61 6C 6C 20 67 61 6D 65 5F 61 64 64 5F 6D 61 69 6C 28 31 39
37 39 34 38 36 35 36 31 37 31 36 31 37 32 30 34 2C 27 27 2C 31 39 37 39 34 38
36 35 36 31 37 31 36 31 37 32 30 34 2C 27 27 2C 32 35 30 2C 31 30 30 2C 30 2C
27 73 7B 41 44 44 5F 4D 4F 4E 45 59 3D E6 B3 A8 E5 85 A5 E8 B5 84 E9 87 91 E5
A4 B1 E8 B4 A5 EF BC 8C E7 B3 BB E7 BB 9F E9 80 80 E8 BF 98 3C 6D 3A 31 30 3E
2C 43 41 55 53 45 3D 5C 5C 2C 43 41 55 53 45 5C 5C 2C 34 36 5C 5C 2C 32 5C 5C
2C 30 78 32 38 35 34 32 33 38 38 44 36 37 42 43 37 44 5C 5C 2C 2C 4F 52 47 5F
41 53 59 4E 43 3D 31 2C 63 6D 6E 5F 6D 6F 6E 65 79 3D 31 30 2C 63 6D 6E 5F 74
```

过滤/接收

17850 22906

时序的分析

可以看到整个过程中有一个疑点，那就是在扣款之后，系统给自己发送了一封邮件，是用来给自己加钱的！但之后又给自己发邮件请求删除同样数量的钱。

根据这个逻辑，我们在做测试设计的时候自然会想到当加钱邮件发送成功后并且在删除邮件之前，这是一个典型的临界状态，即开篇我们提到的D1期间的S3状态。

在这个状态下，我们可以设计R_action: 让客户端向服务器发送一个取邮件的请求，测试是否可以获得这个加钱的邮件。到目前为止，我们解决了第一个问题：找到了这个状态，并且找到了这个时间段。

下面来解决第二个问题，如何让R_action请求在这个时间段内被执行？

通常我们可以考虑用C/S（客户端/服务器）协议测试工具来构造这个测试数据，但是实际发现，在现有的网络环境下，很难构造这个时间差(发邮件和删除邮件之间)，或许我们也可以试图用频繁快速的发包，或者网络延迟来构造这次攻击，但测试的效果仍然需要寄希望于时间差的利用。

如果我们在服务器端的Proxy上让所有服务器的时序都是FIFO（先进先出）的存在，那么我们便会得到一个序列，可以在任意一个时刻，插入一个数据包，来构造这次时间差的邮件获取攻击。

在proxy上构造攻击规则

OWASP 中国
The Open Web Application Security Project

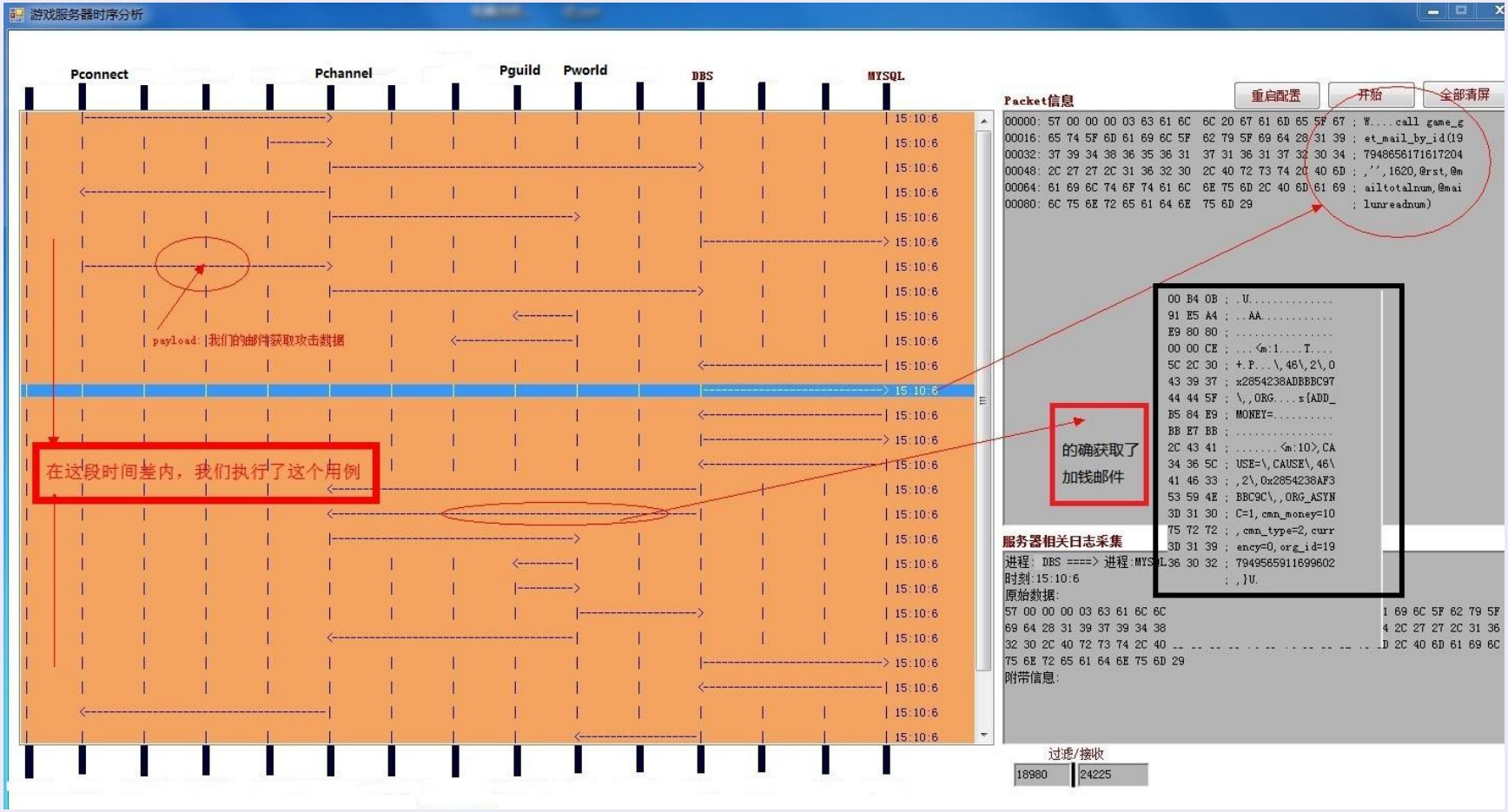
可以在Proxy上构造如下的攻击规则(json格式)：

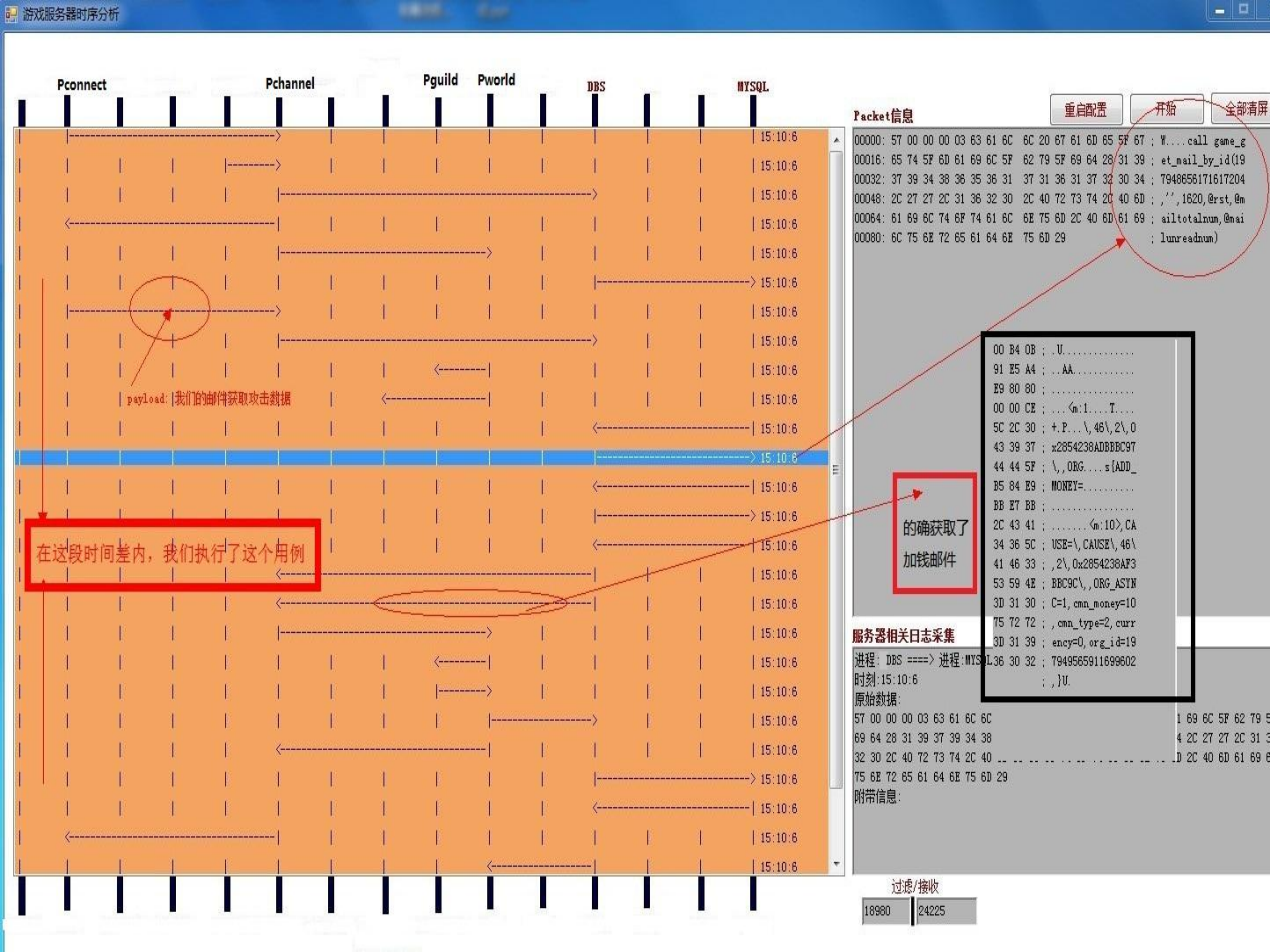
```
{
 "hack":"true",//攻击标志，true表示准备好攻击的条件和数据了
 "from":"DBS",// 场景描述，当发现数据从DBS发出的时候，准备攻击
 "to":"MYSQL",// 场景描述，当发现数据的目的是MYSQL的时候，准备攻击
 "feature":"00 01 02 03 04",//场景描述，当数据中包含有定义的数据时，发出攻
    击
 "payload":"04 03 02 01 00"//当以上条件都满足的情况下，模拟发送Pconnect到
    Pchannel(帮派线路)的攻击数据
}
```

那么，在这个规则的基础上，我们构造一次攻击：当DBS向MYSQL发送给自己添加邮件的数据包时，发送Pconnect到Pchannel的一次获取邮件请求。

攻击时的时序

OWASP 中国
The Open Web Application Security Project




游戏服务器时序分析
Pconnect
Pchannel
Pguild
Pworld
DBS
MYSQL
15:10:6
payload: 我们的邮件获取攻击数据
在这段时间差内，我们执行了这个用例
Packet信息
重启配置
开始
全部清屏
00000: 57 00 00 00 03 63 61 6C 6C 20 67 61 6D 65 5F 67 ; W....call game_g
00016: 65 74 5F 6D 61 69 6C 5F 62 79 5F 69 64 28 31 39 ; et_mail_by_id(19
00032: 37 39 34 38 36 35 36 31 37 31 36 31 37 32 30 34 ; 7948656171617204
00048: 2C 27 27 2C 31 36 32 30 2C 40 72 73 74 2C 40 6D ; ,'',1620,@rst,@m
00064: 61 69 6C 74 6F 74 61 6C 6E 75 6D 2C 40 6D 61 69 ; ailtotalnum,@mai
00080: 6C 75 6E 72 65 61 64 6E 75 6D 29 ; lunreadnum)
00 B4 0B ; .U..............
91 E5 A4 ; ..AA...........
E9 80 80 ; ....
00 00 CE ; ....<m:1....T....
5C 2C 30 ; +.P...\,46\,2\,0
43 39 37 ; x2854238ADBBBC97
44 44 5F ; \,,ORG....s{ADD_
B5 84 E9 ; MONEY=.........
BB E7 BB ; ....
2C 43 41 ; ....<m:10>,CA
34 36 5C ; USE=\,CAUSE\,46\
41 46 33 ; ,2\,0x2854238AF3
53 59 4E ; BBC9C\,,ORG_ASYN
3D 31 30 ; C=1,cmn_money=10
75 72 72 ; ,cmn_type=2,curr
3D 31 39 ; ency=0,org_id=19
36 30 32 ; 7949565911699602
; ,}U.
的确获取了
加钱邮件
服务器相关日志采集
进程: DBS ====> 进程:MYSQL
时刻:15:10:6
原始数据:
57 00 00 00 03 63 61 6C 6C
69 64 28 31 39 37 39 34 38
32 30 2C 40 72 73 74 2C 40
75 6E 72 65 61 64 6E 75 6D 29
附带信息:
过滤/接收
18980
24225

攻击结果

OWASP 中国
The Open Web Application Security Project

可以看出我们构造的R_action用例得到了执行，并且跟预期一样在这个时间段D1内触发了获取邮件的数据库请求。

这样我们便把第二个问题给解决了。

总结

OWASP 中国
The Open Web Application Security Project

1 网游的安全测试中，除了传统的协议测试外，为了能够更好的去分析系统，时序分析是一个比较好的方法和补充。

2 要实现竞态测试，必须能够先找到竞态点，这需要给测试者提供出必要的信息，比如系统的时序和系统的状态，我们可以通过proxy的方法来获取到这些信息

3 然后在这个基础上，要想去执行基于该状态的竞态测试，还需要能够找到方法使得待测系统“冷却”，这次分享采用的是用FIFO的机制来控制进程间通信。

OWASP 中国
The Open Web Application Security Project

Q&A

欢迎各位大侠批评