



游戏上线安全自查指南

代码、资源、运行时与服务端的 10 类 24 项安全问题



游戏上线前，安全检查容易漏在哪里？

游戏上线前，功能、兼容性、性能、支付等环节通常都有明确的测试流程。而游戏安全因相关风险分很难靠常规功能测试逐一排查。

比如，游戏能正常启动，关键代码却可能很容易被定位；资源热更新没有报错，更新后的文件却未必还在保护范围内。客户端上传的数据也是如此：格式正确，只说明数据符合接口要求，内容是否可信，还需要进一步校验。等到外挂、修改器、破解包或异常数据出现后再回头排查，处理起来往往就比较被动了。

为了帮助游戏团队在版本上线前发现容易遗漏的安全问题，FairGuard 撰写了《游戏上线安全自查指南》，将常见风险整理为 10 类、24 项检查问题，供研发和测试人员对照排查。

检查时，除了确认是否接入安全产品，还需要看清几个问题：

01 攻击者可以看到什么？	02 客户端里有哪些内容可以被修改？
03 修改之后能不能继续生效？	04 关键结果有没有其他可信依据可以核验？

不同平台、引擎、玩法和联网架构，面临的风险各有差异，保护方式也需要结合项目情况选择。这 24 项可以作为上线前的安全自查清单，帮助团队快速梳理：哪些地方已经有了保护，哪些位置还存在明显缺口。

10 类 24 项游戏安全问题总览

01 代码保护 关键业务函数是否容易被定位？ 符号、方法名和字符串是否暴露明显业务含义？ IL2CPP 代码与 metadata 是否容易重新建立关联？	02 资源保护 游戏资源是否能够被直接提取？ 热更新后的资源是否继续受到保护？
03 内存与运行时 关键运行时数据是否容易被修改？ 是否关注调试、Hook 和非法注入？ 核心代码和资源是否存在完整性风险？	04 网络通信 使用 HTTPS 之后，客户端就可信吗？ 关键请求是否考虑重放和重复提交？
05 本地存档 本地关键数据是否能够直接修改？ 是否考虑复制和回滚？	06 运行环境 Root 和越狱环境是否纳入风险判断？ 模拟器、多开与云手机是否符合项目风险策略？ 是否关注异常进程、模块和环境变化？
07 服务端校验 关键业务结果是否由服务端独立确认？ 参数合法，是否代表业务合理？ 是否关注异常行为组合？	08 内购与支付 支付结果是否由服务端确认？ 是否考虑订单重放和异常状态？
09 重打包与渠道版本 修改、重签名或重新打包后是否存在识别能力？ 多渠道包的安全状态是否一致？	10 上线监控与应急 上线之后是否持续观察风险变化？ 出现外挂或破解后，是否具备响应能力？

安全检查真正关注的并不是“有没有某项功能”，而是攻击链上有没有明显可以被利用的缺口。

01 代码保护

01 关键业务函数是否容易被定位？

代码经过 Native 编译或 IL2CPP 转换，并不意味着业务逻辑已经无法分析。战斗计算、奖励结算、角色属性、购买校验、安全检测等关键逻辑，如果仍能通过调用关系、数据结构或运行时行为快速定位，攻击者就可能进一步分析和修改。上线前真正需要关注的，不只是“能不能反编译”，而是反编译或分析之后，关键逻辑还能不能被快速理解和关联。

02 符号、方法名和字符串是否暴露明显业务含义？

即使代码本身已经进行一定处理，明显的符号和业务字符串仍可能成为分析线索。奖励、购买、伤害、安全检测等具有明确含义的信息，都可能帮助分析者迅速缩小目标范围。因此，代码保护不仅要看函数本身，也要关注周围是否仍存在大量可用于理解业务含义的信息。

03 IL2CPP 代码与 metadata 是否容易重新建立关联？

Unity IL2CPP 项目中，global-metadata.dat 包含大量类型、方法、字段和字符串等元数据信息，并与 Native 代码存在对应关系。只保护其中一侧，并不一定能够解决整个逆向链路。需要关注的是：metadata、Native 代码以及运行时信息之间，是否仍然能够被快速重新建立映射关系。

02 资源保护

04 游戏资源是否能够被直接提取？

游戏资源风险并不只是“图片被扒”。AssetBundle、配置表、Lua 脚本、关卡数据、活动内容以及未公开资源，都可能暴露业务逻辑或尚未上线的信息。上线前应确认核心资源是否能够通过常见方式直接识别、读取和提取。

05 热更新后的资源是否继续受到保护？

首包中的资源已经加密，不代表后续更新资源天然拥有同样的保护。尤其是使用 AssetBundle 等热更新方案时，需要关注下载、缓存、存储和加载后的资源，是否仍然保持一致的保护状态。

03 内存与运行时

06 关键运行时数据是否容易被修改？

血量、金币、速度、技能冷却只是比较直观的例子。真正的问题在于，客户端运行在用户能够控制的设备上，本地内存中的数据天然面临被观察和修改的可能。不能因为数据没有写入文件，就默认它是安全的。

07 是否关注调试、Hook 和非法注入？

攻击进入运行时后，问题已经从“代码是否容易阅读”转向“游戏当前是不是运行在正常环境中，关键逻辑有没有被异常调用或替换”。因此静态保护和运行时检测解决的是不同阶段的问题。

08 核心代码和资源是否存在完整性风险？

文件、模块或关键资源一旦被替换、修改，游戏仍然继续正常运行，就可能给篡改留下空间。上线前需要关注的不只是“文件有没有加密”，还包括关键内容被修改之后，系统是否能够感知异常。

04 网络通信

09 使用 HTTPS 之后，客户端就可信吗？

不一定。HTTPS 主要解决传输过程中的窃听和篡改问题，但它并不能证明发出请求的客户端本身没有被修改。如果攻击者已经能够控制客户端逻辑，那么一个通过正常 HTTPS 连接发出的请求，内容本身仍然可能存在问题。因此通信安全和客户端可信度需要分开看。

10 关键请求是否考虑重放和重复提交？

一条请求本身合法，并不意味着它可以在任意时间、任意状态下重复生效。奖励领取、购买、任务完成、结算等业务尤其需要关注请求时机、次数、状态以及上下文是否合理。

05 本地存档

11 本地关键数据是否能够直接修改？

对于单机、弱联网以及部分模拟经营、Roguelike 等游戏，本地存档可能承载大量进度和业务数据。只要修改文件后游戏仍然认可其中结果，就可能形成存档篡改风险。

12 是否考虑复制和回滚？

存档风险并不只是“改一个金币数字”。复制旧存档、回滚进度、重复利用某个状态，同样可能影响道具、奖励、游戏进度和经济系统。加密主要解决“数据是否容易被看懂”，完整性和状态校验解决的是“修改之后还能不能被接受”。

06 运行环境

13 Root 和越狱环境是否纳入风险判断？

Root 或越狱会扩大用户对系统和应用运行环境的控制能力，因此通常属于需要关注的高风险环境。但“高风险”并不意味着一定作弊，实际策略仍应结合游戏类型、用户群和业务风险决定。

14 模拟器、多开与云手机是否符合项目风险策略？

模拟器、云手机、多开本身也不一定等于作弊。但在部分强竞技、工作室刷量、批量账号等场景中，它们可能与异常行为产生明显关联。重点不是“检测到就封”，而是项目是否知道这些环境存在，并已经明确自己的风险策略。

15 是否关注异常进程、模块和环境变化？

运行过程中出现异常模块、注入行为或环境变化时，如果客户端完全没有感知能力，就可能为后续修改留下空间。因此运行环境检查往往需要与运行时数据、完整性和行为信号结合判断。

07 服务端校验

16 关键业务结果是否由服务端独立确认？

这是很多客户端安全问题最终都会回到的一点。客户端可以参与战斗、表现、输入和计算，但奖励、货币、排名、支付结果等关键业务，不应该只因为客户端“告诉服务端结果是什么”就被接受。客户端可以参与计算，但不应该拥有关键业务结果的最终解释权。

17 参数合法，是否代表业务合理？

例如 reward = 100，即使 100 处于允许范围内，也不意味着玩家在当前游戏状态下真的有资格获得 100。因此服务端除了检查类型、范围、格式，还需要结合状态、上下文、业务逻辑和历史行为。

18 是否关注异常行为组合？

很多作弊并不会产生一个特别明显的“非法值”。真正值得关注的可能是行为频率异常、顺序不合理、长期结果偏离正常范围，或者多项行为组合几乎不可能由正常玩家完成。因此运行时检测和服务端行为分析通常需要形成互补。

08 内购与支付

19 支付结果是否由服务端确认？

客户端出现“支付成功”提示，不应该直接等价于服务端立即完成发货。订单、商品、金额和支付结果应存在可信的服务端验证链路。

20 是否考虑订单重放和异常状态？

除了伪造支付结果，还应关注重复订单、重复发货、异常回调和状态不同步。支付安全本质上仍然是一个业务状态可信问题。

09 重打包与渠道版本

21 修改、重签名或重新打包后是否存在识别能力？

游戏风险并不只发生在正常安装包内部。如果客户端经过修改、重新签名甚至二次分发后仍然可以无差别运行，就可能形成破解版、修改版等问题。因此需要关注当前运行的客户端是不是开发者真正发布的版本。

22 多渠道包的安全状态是否一致？

游戏可能同时存在官方渠道、联运渠道、海外版本、测试版本和不同商店版本。不同构建和发布流程容易出现配置差异。最终需要确认某一个渠道是否因为构建流程不同而意外漏掉安全能力。

10 上线监控与应急

23 上线之后是否持续观察风险变化？

安全不是发版前检查一次就结束。新版本、新活动、新玩法，以及新的作弊和逆向工具，都可能改变原有风险面。上线以后仍然需要持续关注异常客户端、修改行为、破解包和异常玩家行为。

24 出现外挂或破解后，是否具备响应能力？

真正出现问题以后，团队至少需要能够回答：问题影响的是哪个版本？攻击发生在哪一个环节？是否需要调整客户端？是否需要服务端策略配合？新版本如何避免同一问题重复出现？游戏安全最终是一场持续对抗，而不是一次性的上线动作。

游戏上线安全自查清单

项目名称	版本号	检查日期	检查人
------	-----	------	-----

序号	检查问题	检查状态
代码		
1	关键业务函数是否容易定位？	☐ 已确认 ☐ 待处理
2	符号和字符串是否暴露业务含义？	☐ 已确认 ☐ 待处理
3	IL2CPP 代码与 metadata 是否容易重新关联？	☐ 已确认 ☐ 待处理
资源		
4	核心资源是否容易直接提取？	☐ 已确认 ☐ 待处理
5	热更新资源是否持续受到保护？	☐ 已确认 ☐ 待处理
内存与运行时		
6	关键运行时数据是否容易修改？	☐ 已确认 ☐ 待处理
7	是否关注调试、Hook 和非法注入？	☐ 已确认 ☐ 待处理
8	核心代码和资源是否存在完整性风险？	☐ 已确认 ☐ 待处理
通信		
9	敏感数据传输是否合理保护？	☐ 已确认 ☐ 待处理
10	是否考虑重放和重复提交？	☐ 已确认 ☐ 待处理
存档		
11	本地关键数据是否容易修改？	☐ 已确认 ☐ 待处理
12	是否考虑复制和回滚？	☐ 已确认 ☐ 待处理
运行环境		
13	Root/越狱是否纳入判断？	☐ 已确认 ☐ 待处理
14	模拟器、多开、云手机是否有明确策略？	☐ 已确认 ☐ 待处理
15	是否关注异常进程、模块和环境变化？	☐ 已确认 ☐ 待处理
服务端		
16	关键结果是否由服务端确认？	☐ 已确认 ☐ 待处理
17	客户端参数是否结合业务状态校验？	☐ 已确认 ☐ 待处理
18	是否关注异常行为组合？	☐ 已确认 ☐ 待处理
支付		
19	支付结果是否由服务端验证？	☐ 已确认 ☐ 待处理
20	是否考虑订单重放和异常状态？	☐ 已确认 ☐ 待处理
版本		
21	是否关注修改、重签名与重打包？	☐ 已确认 ☐ 待处理
22	不同渠道版本的安全状态是否一致？	☐ 已确认 ☐ 待处理
上线后		
23	是否持续观察风险变化？	☐ 已确认 ☐ 待处理
24	是否已经具备异常事件响应能力？	☐ 已确认 ☐ 待处理

关于 FairGuard

FairGuard 是杭州法嘉德科技有限公司创立的游戏加固产品的品牌。公司专注于游戏加固及反外挂领域，致力于帮助游戏公司解决外挂和破解问题，为游戏提供深度一体化的加密保护方案。

FairGuard 开发团队深耕技术，研发了游戏专用设备指纹方案、无导入函数 SO 加壳、无 API 签名校验、多端通用的 Unity AssetBundle 资源加密方案等多项业界独家技术。

公司始终坚持自主研发、客户至上的服务理念，秉承专注、极致、口碑的企业精神，通过专业的服务，为客户提供优质的游戏安全解决方案，赢得了客户的一致好评。

目前，产品已经被 FunPlus、三七互娱、游族、恺英网络、心动网络等多家头部游戏公司采用，接入 700+款热门游戏。



扫码关注微信公众号



扫码添加客服微信