iOS应用内支付安全分析

罗 成 工业和信息化部电信研究院
武 玥 中国互联网协会

摘 要 应用内支付是开发者盈利的重要手段。然而,由于开发者在具体实现中未对交易进行有效性验证,导致攻击者可以通过控制流劫持和会话劫持两类攻击方式实现免费内购,给开发者权益造成了极大危害。本文以iOS平台应用内支付为研究对象,介绍了其支付流程,分析了支付过程中存在的安全问题及当前存在的攻击方法,最后总结了应用内支付攻击防御思路。

关键词 iOS 应用内支付 安全

1 引言

应用内支付(In-App Purchase,IAP)是开发者从应用商店获利的重要方式,它使得开发者能够在收费或者免费应用(App)中内置数字商店,并通过该商店向消费者出售数字内容,如虚拟商品、虚拟货币或是订阅高级服务。根据咨询公司Distimo的统计,在美国iTunes App Store市场中,排名前250的应用中,有223个应用包含了应用内支付功能,而应用内支付所产生的利润更是占据了应用市场总盈利的76%,并且仍在持续增长中。

然而,随着应用内支付的兴起,针对应用内支付的攻击也日益严重,并出现了一些低门槛的自动化攻击方式,这些攻击针对应用程序对交易真实性验证不严密的逻辑漏洞,通过劫持应用程序控制流代码或网络通讯,欺骗应用让其误以为当前用户已经进行了购买,进而提供数字内容,简称为IAP攻击。IAP攻击使得攻击者可以免费获取收费数字内容,对以应用内支付为主要盈利方式的应用市场造成了重大威胁。

2 应用内支付

2.1 应用内支付交易流程

iOS应用内支付功能由StoreKit框架提供,应用只需要与StoreKit进行交互,通讯细节由StoreKit和App Store之间自动完成,支付过程通常涉及三方:iOS应用、App Store平台以及可选的第三方服务器,整个流程分为4个阶段:

(1)请求阶段:iOS应用向App Store平台请求数字商店物品清单,并向App Store平台发送交易请求。

(2)响应阶段:App Store平台返回交易对象,并调用iOS应用中的响应函数。

(3)验证阶段:iOS应用将交易对象的票据信息发给App Store平台,或经第三方服务器转交给App Store平台,由App Store平台返回票据的验证结果。

(4)支付阶段:iOS应用根据票据的验证结果进行支付,或者返回出错信息。

2.2 应用内支付票据验证

验证阶段起着保障交易真实性的重要作用,是应用内支付过程中最重要的安全环节。验证时,应用需将交易对象中的票据信息transactionReceipt直接或经第三方服务器发送给App Store平台,以验证票据信息的有效性,transactionReceipt经过了苹果公司加密,对开发者不可读,需封装成键值名为receipt-data的JSON对象,用POST方法发送给https://buy.itunes.apple.com/verifyReceipt,之后App Store会验证票据并返回验证结果和可读的票据信息,包括交易是否真实有效、交易购买的数量、商品名称和种类、购买时间等信息。

3 应用内支付安全风险分析

3.1 应用内支付逻辑漏洞成因

应用内支付逻辑漏洞存在于响应阶段和验证阶段,方法paymentQueue:updatedTransactions:标识响应阶段开始,示例代码如下所示:

```
- (void)paymentQueue:(SKPaymentQueue *)queue
updatedTransactions:(NSArray *)transactions
{
```

```
for (SKPaymentTransaction *transaction in transactions)
{
  switch (transaction.transactionState)
  {
     case SKPaymentTransactionStatePurchased:
        //[self verifyTransaction] #验证交易有效性
        [self completeTransaction:transaction];
        break;
     case SKPaymentTransactionStateFailed:
        self failedTransaction:transaction];
        break;
     case SKPaymentTransactionStateRestored:
        [self restoreTransaction:transaction];
     default:
        break;
  }
}
}
```

该响应代码会在交易状态变化时被调用，代码主体是一个switch语句，通过transactionState属性的值判断交易是成功还是失败，然后再进行后续操作，方法completeTransaction表示交易成功，进行支付操作，方法failedTransaction表示交易失败，方法restoreTransaction表示已进行过成功交易，在此恢复原来的购买。

在该代码中，应用程序缺乏支付验证逻辑，仅通过transactionState属性来判断交易的状态，并未进行进一步的操作来验证交易的真实性，如果攻击者劫持transactionState返回值，使其等于SKPaymentTransactionStatePurchased，或劫持验证通讯，返回伪造结果，则能欺骗应用支付数字内容。

此外，如果应用在调用completeTransaction方法前通过网络验证了票据的有效性（verifyTransaction），却未做进一步本地校验，此时攻击者在劫持transactionState的基础上，可以进一步篡改方法verifyTransaction的返回值，伪造票据验证结果，欺骗应用程序使其误认为交易成功，此情况为验证逻辑不严密。

3.2 应用内支付攻击

应用内支付攻击主要有两种形式，控制流劫持攻击和会话劫持攻击，两者都能欺骗应用程序达到免费内购的目的。

3.2.1 控制流劫持攻击

控制流劫持攻击针对已经越狱并获得Root权限的iOS设备，攻击者通过加载动态链接库Dylib的形式注入代码到程序的进程空间，该动态链接库会修改进程空间并劫持程序控制流代码，进而影响程序的判断逻辑，改变程序的执行流。

影响应用判断支付状态的方法主要有两类，一类是交易状态判断方法SKPaymentTransaction类的transactionState，该方法直接反映应用内支付交易的状态，只有当其返回SKPaymentTransactionStatePurchased时，才会进入支付流程；第二类关键方法是网络通讯类，如NSURLConnection，iOS应用通过这些类获取网络验证的结果。因此，以IAP Free为代表的控制流劫持工具就修改了这两类函数的返回结果，通过改变transactionState的值劫持应用进入交易成功分支，通过修改网络通讯类的返回结果伪造网络验证成功的信息。

3.2.2 会话劫持攻击

会话劫持攻击是通过某种手段将iOS应用发往App Store服务器的交易请求和验证请求劫持到攻击者搭建的服务器，通过该服务器返回伪造的交易结果和验证结果，欺骗应用支付商品的攻击行为。

会话劫持攻击的代表是in-appstore网站，攻击者通过手动设置将iOS设备的DNS服务器指向in-appstore网站，通过DNS欺骗，将交易请求和验证请求劫持到in-appstore网站，返回伪造的验证结果信息。由于iOS平台的应用内支付是采用HTTPS加密传输的，因此需要在iOS设备上添加自签名证书，以便iOS设备能够接收in-appstore返回的伪造数据，当前in-appstore针对iOS平台的伪造数据不仅包括了交易成功数据，还包括伪造的验证结果。

4 应用内支付攻击防御思路

当前，以IAPFree为代表的控制流劫持攻击和以in-appstore为代表的会话劫持攻击已经非常成熟，即使普通用户也只需简单的几个步骤就能实现免费内购，开发者要维护自身权益，需从如下几个方面完善支付验证逻辑，确保支付正常进行。

4.1 完善网络验证流程

添加支付票据网络验证功能，尽量采用第三方服务器验证票据是否有效，避免直接根据transactionState值进行交易，iOS应用与第三方服务器之间的通讯考虑进行加密，避免攻击者篡改或伪造验证结果。可以考虑采用第三方验证服务来验证票据有效性，如Bee-

全球信息安全的六大战略错误

杨义先　北京邮电大学信息安全中心主任,教授

1　引言

科学家是人类最聪明的群体。而科学家中,最善于彼此对抗者,又非信息安全专家莫属。事实证明,正是这批人精中的人精,在保卫信息安全方面,却办了糊涂事,使得赛搏空间越来越不安全,甚至如果再继续错下去,最终将导致"人人裸奔、个个自危"。

2　全球信息安全界所犯的几个代表性战略错误

下面简要罗列了全球信息安全界所犯的几个代表性战略错误:

●错误1(最基础的错误):忽略了"返祖"现象。构成赛搏空间的所有要素(计算系统、存储系统、传输系统、采集系统等)都是人类文明精华的最先进成果,因此按惯性思维,自然认为赛搏空间本身也最文明。然而,我们错了,并且大错特错。实事上,赛搏空间是最"返祖"的野蛮社会。在这里,甚至连文明社会的最基本准则(道德准则、关系准则、秩序准则)都是空白的,"弱肉强食"司空见惯,"损人利己"天经地义,"损人不

blex,通过调用其SDK来防止会话劫持和票据伪造。

4.2 完善本地验证逻辑

若只能由iOS应用本地验证票据有效性,需确保连接的验证服务器是App Store官方服务器,确保用于验证的票据未经篡改,在此基础上,检查:

(1)通讯服务器的SSL证书是否是自签名扩展证书。

(2)服务器返回的商品信息与用户购买的商品是否一致。

(3)交易对象中的票据信息是否含有有效的数字签名。

(4)验证结果中的交易ID值是否唯一。

4.3 其他防御思路

检查iOS系统是否已越狱并取得了Root权限,检查iOS应用程序代码是否被动态篡改,若iOS应用代码被修改,考虑禁用应用内支付功能。

5　结束语

应用内支付作为应用市场主要的盈利方式之一,已得到越来越多开发者的重视,其不仅为开发者带来了持续客观的收入,也使得应用的生命周期得以延续,目前这一模式已扩展到Android、Mac OS、Chrome、Windows 8等多个主流平台,随着应用内支付功能的流行,其安全问题也必将成为开发者和用户关注的焦点。

参考文献

1 Distimo. How the Most Successful Apps Monetize Their User Base. 2013

2 iOS Developer Library. In- App Purchase Programming Guide. 2013

3 碳基体. iOS平台游戏安全再议之IAP Free游戏内购破解的防御. 2012

An Analysis of Vulnerability in iOS In-App Purchases

Abstract In-App Purchase (IAP) has become an important profit mode for iOS developers. However, a large number of iOS applications containing IAP functionality fail to perform sufficient verification on IAP transactions. By forging a transaction response, attackers can cheat the application and obtain the digital content in in-App store without paying. In this paper, we introduce the procedure of in-App Purchases, analyze the security feathers of IAP and summarize the defensive methods of IAP attacks.

Key words iOS, application of payment, safety

(收稿日期:2013-11-05)