Using Receipts to Protect Your Digital Sales

Session 308

James Wilson
Mac App Store

These are confidential sessions—please refrain from streaming, blogging, or taking pictures
In-App Purchases
In-App Purchases

96%

Of the Top-Grossing Apps
Agenda
Agenda

- Introducing the Receipt
- Understanding Receipts
- Validation and Inspection
- Implementing Validation
- Testing with Receipts
Introduction

The receipt

• Trusted record of purchase
  ▪ Issued by the App Store
  ▪ Stored on device

• Signed and verifiable
• For your app, on that device
  ▪ Copy protection
  ▪ In-App purchase verification
Introduction

• Free or paid
  • In-App Purchases
• Know exactly what the user has paid for
Unified Receipt
on iOS 7 and OS X
Introduction
What’s new

• iOS 7
  ▪ Grand Unified Receipt
  ▪ Same receipt format as OS X

• Receipt now includes
  ▪ Volume purchase information
  ▪ Support paid to free with in-app purchase
Introduction
Protect your purchases

• Apple provides you with
  ▪ The receipt format specification
  ▪ The receipt itself
  ▪ Instructions for On-Device Receipt Validation
  ▪ Online service for Server-to-Server Validation

• You chose a security level appropriate for your products
  ▪ You decide the complexity of the implementation
Understanding Receipts
Understanding Receipts

Receipt workflow

• Receipt is issued when
  ▪ App is purchased or updated
  ▪ In-App purchase completed or restored
  ▪ Volume Purchase license revoked
  ▪ On-Demand Refresh API
    ▪ Receipt is not present
    ▪ Receipt is not valid on that device
Understanding Receipts

Inside the receipt

• Certificates and signatures

• Information that ties your app to this device

• Purchase information
  ▪ App and in-app purchases
  ▪ Product, quantity, and version
  ▪ Volume Purchase Program
  ▪ Initial purchase date
Understanding Receipts
Inside the receipt

• Transition from paid to free with in-app purchases
  ▪ Receipt contains the initial purchase date
  ▪ Use this date to determine eligibility for paid content
Transition from iOS 6 to iOS 7
Transition from iOS 6 to iOS 7

• iOS 7 is binary compatible with iOS 6
  ▪ Both receipt formats are issued
  ▪ Both APIs will work
  ▪ iOS 6 receipt API is deprecated

• iOS 7 and OS X manage the receipt for you
  ▪ Receipt is stored on device, in the app bundle

• Supporting both iOS 6 and iOS 7
  ▪ Weak link to iOS 7 API
Transition from iOS 6 to iOS 7

Weak linking

• Example of weak linking

```swift
NSURL *receiptURL = nil;
NSBundle *bundle = [NSBundle mainBundle];
if ([bundle respondsToSelector:@selector(appStoreReceiptURL)]) {
    receiptURL = [bundle performSelector:@selector(appStoreReceiptURL)]
}
```

• Do NOT check the system version
  ▪ Use the run-time to determine which API to use
Validating and Inspecting Receipts
Validate On Device

The receipt file

• Stored in the App Bundle
  ▪ API to get the path
• Single file
  ▪ Purchase data
  ▪ Signature to check authenticity
Three Step Process
Validating Receipts

Verify Signature  Confirm Device  Check Purchases

Authentic and trusted  For this device  What the user paid for
Validating Receipts

- Verify Signature
- Confirm Device
- Check Purchases

Authentic and trusted For this device What the user paid for
Validate On Device

Verify authenticity

• Use signature to confirm the receipt is authentic and unaltered
  1. Locate the file
  2. Read the contents
  3. Verify the signature

• PKCS #7 Container
  - Can use OpenSSL to verify
Validate On Device

Verify authenticity

• Use signature to confirm the receipt is authentic and unaltered
  1. Locate the file
  2. Read the contents
  3. Verify the signature

// Locate the Receipt
[[NSBundle mainBundle] appStoreReceiptURL];

• PKCS #7 Container
  • Can use OpenSSL to verify
Verify Receipt Signature

BIO *b_receipt;
BIO *b_x509;

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate
Verify Receipt Signature

```
BIO *b_receipt;
BIO *b_x509;

// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);
```
Verify Receipt Signature

```c
BIO *b_receipt;
BIO *b_x509;

// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt, NULL);

// Create the certificate store
X509_STORE *store = X509_STORE_new();
X509 *appleRootCA = d2i_X509_bio(b_x509, NULL);
X509_STORE_add_cert(store, appleRootCA);
```

Load the Receipt and Apple Root CA Certificate
Binary data from receipt plus certificate
Verify Receipt Signature

```c
BIO *b_receipt;
BIO *b_x509;

// Convert receipt data to PKCS #7 Representation
PKCS7 *p7 = d2i_PKCS7_bio(b_receipt,
NULL);

// Create the certificate store
X509_STORE *store = X509_STORE_new();
X509 *appleRootCA = d2i_X509_bio(b_x509,
NULL);
X509_STORE_add_cert(store, appleRootCA);

// Verify the Signature
BIO *b_receiptPayload;
int result = PKCS7_verify(p7,
NULL, store, NULL, b_receiptPayload,
0);
if (result == 1)
{
    // Receipt Signature is VALID
    // b_receiptPayload contains the payload
}
```
Validating Receipts

- Verify Signature
- Confirm Device
- Check Purchases

Authentic and trusted
For this device
What the user paid for
Validating Receipts

Verify Signature  Confirm Device  Check Purchases

Authentic and trusted  For this device  What the user paid for
Validate On Device
Confirm app and device

- Reading the receipt
- Series of attributes
  - Type, version, value
- ASN.1
  - Abstract Syntax Notation
Reading ASN.1

• Receipt Payload Format Definition

ReceiptModule DEFINITIONS ::= BEGIN

ReceiptAttribute ::= SEQUENCE {
    type INTEGER,
    version INTEGER,
    value OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END

• Use asn1c to generate boiler plate code
Reading ASN.1

• Using boiler plate from *asn1c*

```c
Payload_t *payload = NULL;
asn_dec_rval_t rval = asn_DEF_Payload.ber_decoder(NULL,
&asn_DEF_Payload,
(void **)&payload,
pld, pld_sz, 0);

// Walk the attributes
for (i = 0; i < payload->list.count; i++) {
    ReceiptAttribute_t *entry = payload->list.array[i];
    switch (entry->type) {
        case 2: // 2 = Bundle ID
            bundle_id = &entry->value;
            break;
        ...
    }
}
```
Validate on Device

Confirm app and device

• Check the **Bundle Identifier**
• Check the **Bundle Version**

• Check **Device Identifier** hash
  - iOS - Vendor Identifier
  - OS X - Machine GUID
    - See documentation for Example

Receipt
Purchase Information
Attribute
Type 2  Bundle Identifier
Type 3  Bundle Version
Type 4  Opaque Value
Type 5  SHA-1 Hash
Type 17 In-App Purchases
Validating Receipts

- Verify Signature
- Confirm Device
- Check Purchases

- Authentic and trusted
- For this device
- What the user paid for
Validating Receipts

Verify Signature  Confirm Device  Check Purchases

Authentic and trusted  For this device  What the user paid for
In-App Purchases

Receipt

Purchase Information

Attribute
Type 2 Bundle Identifier

Attribute
Type 17 In-App Purchases

Attribute
Type 17 In-App Purchases

Attribute
Type 17 In-App Purchases

In-App Purchase Record

Type 1701 Quantity
Type 1702 Product identifier
Type 1703 Transaction identifier
Type 1704 Purchase date
In-App Purchases

Receipt

Purchase Information

Attribute

- Type 2 Bundle Identifier

Attribute

- Type 17 In-App Purchases

Attribute

- Type 17 In-App Purchases

Attribute

- Type 17 In-App Purchases

In-App Purchase Record

- Type 1701 Quantity
- Type 1702 Product identifier
- Type 1703 Transaction identifier
- Type 1704 Purchase date

ReceiptModule DEFINITIONS ::= BEGIN

ReceiptAttribute ::= SEQUENCE {
  type INTEGER,
  version INTEGER,
  value OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
Validate On Device

Key technologies

- **PKCS #7 Container**
  - Signature certificates
    - Verify authenticity
    - OpenSSL can be used
  - ASN.1 format receipt data
    - Use asn1c for boiler plate
Validate Online
Server-to-Server validation

- Allows your servers to validate the receipt before issuing content
- Send the receipt to your server
  - Not directly from the device
- Your server sends the receipt to Apple
- Apple returns JSON receipt data
- Check purchases, provide content
Implementing Validation
Implementing Validation
On iOS 7

• If the receipt doesn’t exist or is invalid
  ▪ Refresh the receipt using Store Kit

• Receipt refresh will require network
• Store sign-in will be required
Implementing Validation
On iOS 7

• If the receipt doesn’t exist or is invalid
  ▪ Refresh the receipt using Store Kit

// Refresh the Receipt
SKReceiptRefreshRequest *request = [SKReceiptRefreshRequest alloc] init];
[request setDelegate:self];
[request start];

• Receipt refresh will require network
• Store sign-in will be required
Implementing Validation

On OS X

• If the receipt is invalid
  ▪ Exit with code 173 to refresh receipt

• Receipt refresh will require network
• Store sign-in will be required
Implementing Validation
On OS X

• If the receipt is invalid
  ▪ Exit with code 173 to refresh receipt

// Receipt is invalid
exit(173);

• Receipt refresh will require network
• Store sign-in will be required
Implementing Validation

In-app purchase lifecycle

• **Consumable and non-renewing** subscriptions
  - Will only appear once
  - In the receipt issued at time of purchase
  - Will not be present in subsequent receipts issued

• **Non-consumable and auto-renewable** subscriptions
  - **Always** in the receipt
  - Can be restored via Store Kit API
Implementing Validation

If the receipt is invalid

- Match the user experience to the value
- iOS apps cannot quit but can limit functionality
- OS X apps can quit or keep running
Using the Test Environment
Test Environment
Test Environment

Doesn’t work, says I haven’t paid!
Test Environment

• Test thoroughly
  ▪ No receipt
  ▪ Invalid receipt
    ▪ Valid on refresh
    ▪ Invalid on refresh
  ▪ Volume Purchase Program receipts
Test Environment

Getting a receipt

• iOS Developers
  ▪ Run the app from Xcode
  ▪ Use Store Kit API to get a receipt

• Must be signed with Development Certificate
Test Environment

Getting a receipt

• OS X Developers
  • Build the app in Xcode
  • Run the app from Finder
  • Exit with code 173 to get a receipt

• Must be signed with Development Certificate
Must be signed with Development Certificate
Test Environment

Avoid common mistakes

• Check which profile is being used to sign the app
  • Must be developer signed to use sandbox

• Sign In with Test Environment account
  • Don’t use Production Apple ID
App Submission
App Submission
With receipt validation

• Developers use Developer Certificate and Test Environment
• Store uses Production Certificate and Production Environment

• App review is different
  ▪ Production signed
  ▪ Test Environment
  ▪ Test receipts

• Do not invalidate Test Environment receipts
  ▪ App will be rejected
Summary
Protect Your In-App Purchases

• Verify and inspect the receipt
  ▪ It’s your trusted record of purchase
• Choose a model that suits the value of your products
• Validation can be done on-device or server-to-server
• Use Test Environment
  ▪ Developer signed
  ▪ Test Environment accounts
More Information

Paul Marcos  
App Services Evangelist  
pmarcos@apple.com

Documentation  
Receipt Validation Programming Guide  
http://developer.apple.com

Apple Developer Forums  
http://devforums.apple.com
| Store Kit and Receipts Lab | Services Lab B | Thursday 3:15PM |