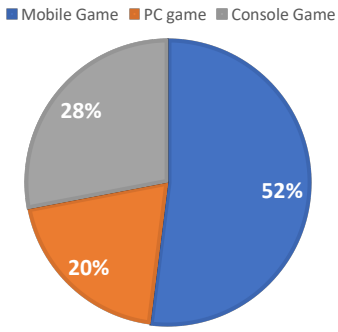
Playing Without Paying: Detecting Vulnerable Payment Verification in Native Binaries of Unity Mobile Games

Chaoshun Zuo, Zhiqiang Lin

Department of Computer Science and Engineering
The Ohio State University

USENIX Security 2022

Game Industry



Video game market revenue worldwide 2021

Source: www.statista.com

Revenue: 175.8 billion

- Mobile Game: 52%
- PC game: 20%
- Console game: 28%

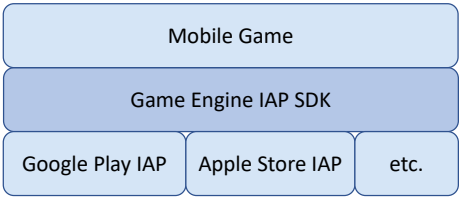
Mobile Game Monetization

Monetization strategies

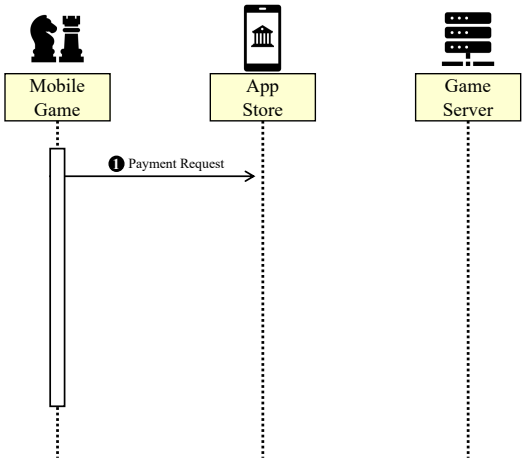
- Pay-to-play
- In-game advertising
- In-game purchase (most popular one)

Source: blog.instabug.com

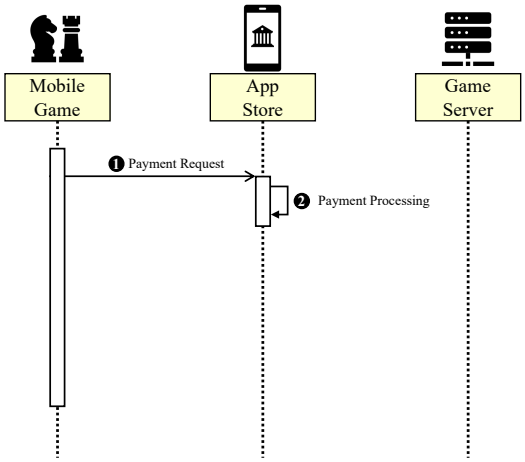
In-game Purchase Implementation



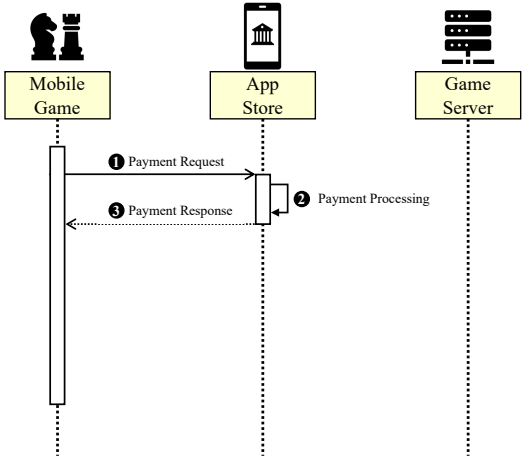
How In-game Purchases Work



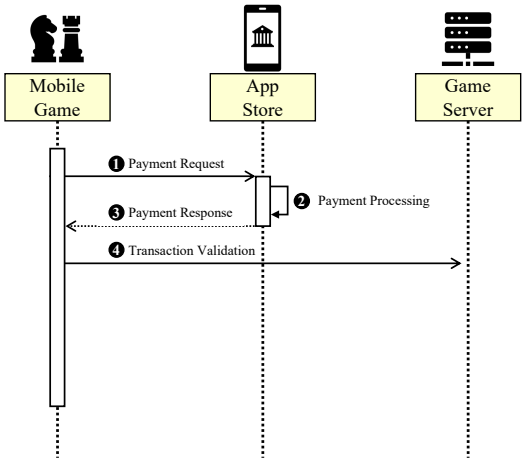
How In-game Purchases Work



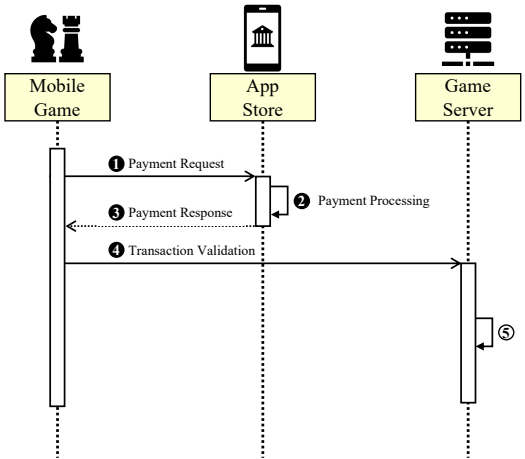
How In-game Purchases Work



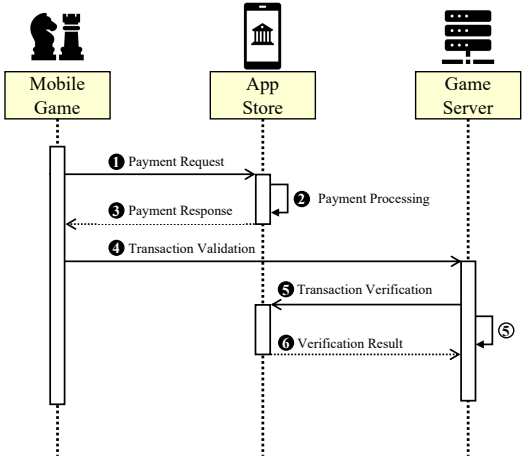
How In-game Purchases Work



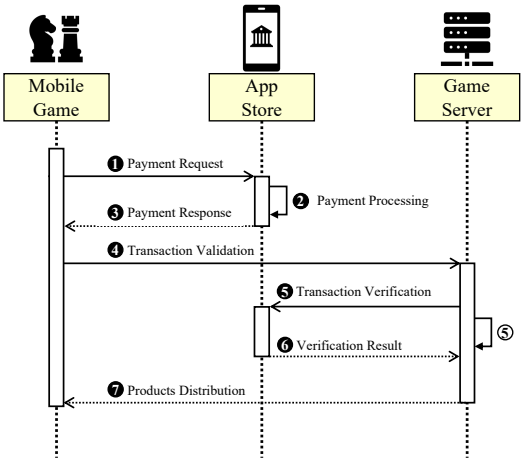
How In-game Purchases Work



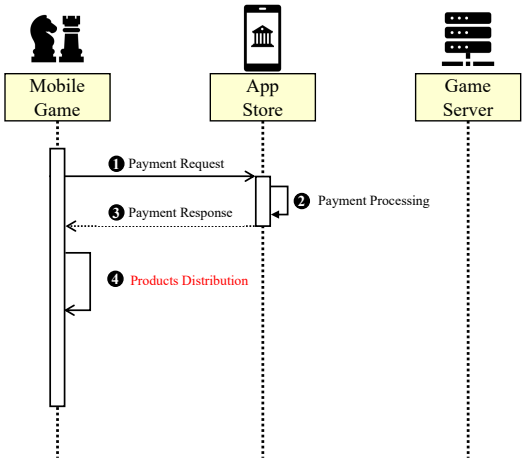
How In-game Purchases Work



How In-game Purchases Work

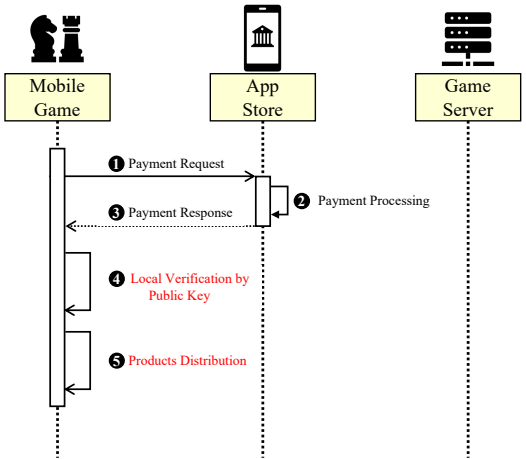


Vulnerable In-game Purchases



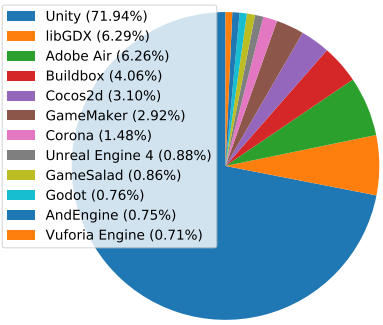
- Lack-of-verification

Vulnerable In-game Purchases



- Local Verification

Game Engines



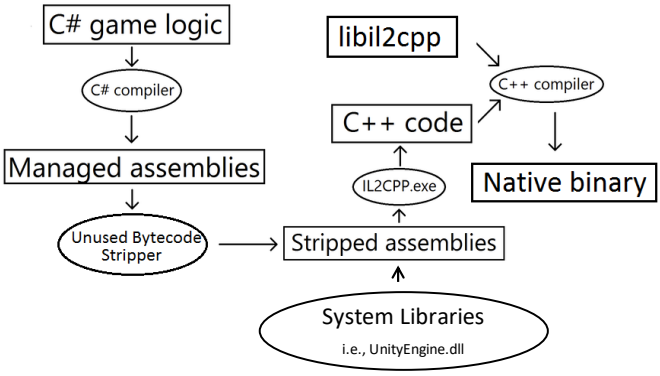
Popularity

- By analyzing 293,019 mobile games, we found that Unity is the most popular game engine in Android games

Advantages

- Cross platform
- C# programming language
- Powerful and easy-to-use IDE

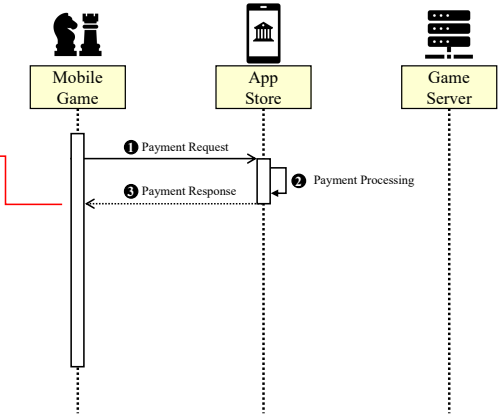
Unity - IL2CPP Compatible



- Faster
- Securer

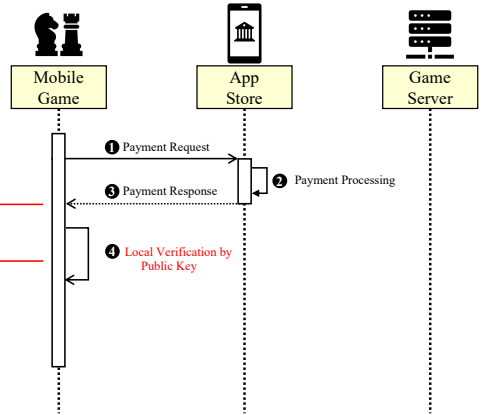
Running Examples

```
1 class IAPManager : IStoreListener
2 {
3   public PurchaseProcessingResult ProcessPurchase(
4                                   PurchaseEventArgs args)
5   {
6     CrossPlatformValidator validator = new CrossPlatformValidator(
7                       GooglePlayTangle.Data(), AppleTangle.Data(),
8                       Application.identifier);
9     try
10    {
11      validator.Validate(args.purchasedProduct.get_receipt());
12      ...
13    }
14    catch (IAPSecurityException)
15    {
16      Debug.Log("Invalid receipt, not unlocking content");
17    }
18    return PurchaseProcessingResult.Complete;
19  }
20 }
```



Running Examples

```
1 class IAPManager : IStoreListener
2 {
3   public PurchaseProcessingResult ProcessPurchase(
4                                   PurchaseEventArgs args)
5   {
6     CrossPlatformValidator validator = new CrossPlatformValidator(
7                       GooglePlayTangle.Data(), AppleTangle.Data(),
8                       Application.identifier);
9     try
10    {
11      validator.Validate(args.purchasedProduct.get_receipt());
12      ...
13    }
14    catch (IAPSecurityException)
15    {
16      Debug.Log("Invalid receipt, not unlocking content");
17    }
18    return PurchaseProcessingResult.Complete;
19  }
20 }
```



Running Examples

```
 1 Class UnityEngine.Purchasing.IStoreListener
 2
 3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
 4 {
 5   ...
 6   String receiptstr = args.purchasedProduct.get_receipt();
 7   StorePurchaseReceipt receipt = JsonUtility.FromJson<StorePurchaseReceipt>(receiptstr);
 8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
 9   httpRequest.AddField("signature", gpayload.signature)
10   ...
11 }
```

```
12 public class StorePurchaseReceipt
13 {
14   public string Store; // 0x10
15   public string TransactionID; // 0x18
16   public string Payload; // 0x20
17 }
18
19 public class GooglePayload
20 {
21   public string json; // 0x10
22   public string signature; // 0x18
23 }
```

Running Examples

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)


```
1 Class UnityEngine.Purchasing.IStoreListener
2
3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
4 {
5   ...
6   String receiptstr = args.purchasedProduct.get_receipt();
7   StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(receiptstr);
8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
9   httpRequest.AddField("signature", gpayload.signature)
10  ...
11 }
```

Running Examples

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)


```
1 Class UnityEngine.Purchasing.IStoreListener
2
3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
4 {
5   ...
6   String receiptstr = args.purchasedProduct.get_receipt();
7   StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(receiptstr);
8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
9   httpRequest.AddField("signature", gpayload.signature)
10  ...
11 }
```

Running Examples

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)


```
1 Class UnityEngine.Purchasing.IStoreListener
2
3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
4 {
5   ...
6   String receiptstr = args.purchasedProduct.get_receipt();
7   StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(receiptstr);
8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
9   httpRequest.AddField("signature", gpayload.signature)
10  ...
11 }
```

Running Examples

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)


```
1 Class UnityEngine.Purchasing.IStoreListener
2
3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
4 {
5   ...
6   String receiptstr = args.purchasedProduct.get_receipt();
7   StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(receiptstr);
8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
9   httpRequest.AddField("signature", gpayload.signature)
10  ...
11 }
```

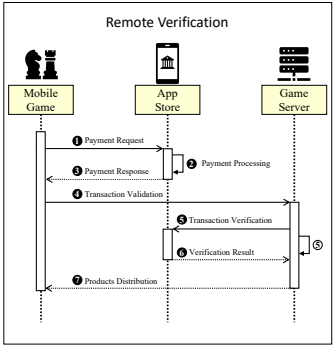
Running Examples

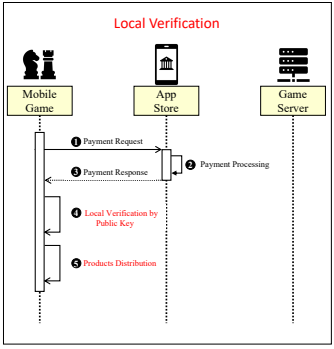
Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)

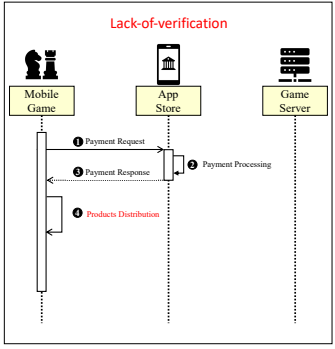
GHIDRA

```
1 Class UnityEngine.Purchasing.IStoreListener
2
3 public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs args)
4 {
5   ...
6   String receiptstr = args.purchasedProduct.get_receipt();
7   StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(receiptstr);
8   GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(receipt.Payload);
9   httpRequest.AddField("signature", gpayload.signature)
10  ...
11 }
```

Goal







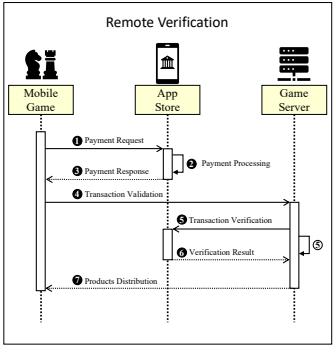
Problem Statement

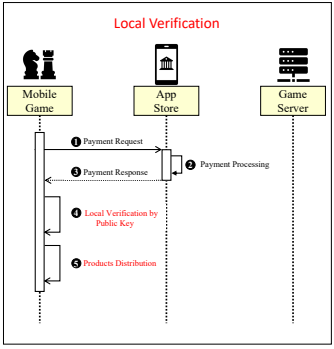
- In-game purchase implemented w/ Unity Engine
- Local verification and lack-of-verification

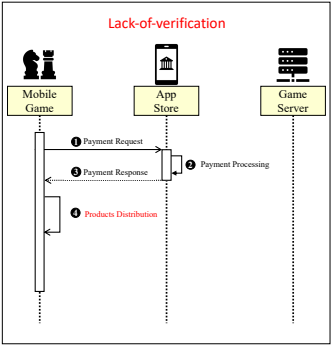
Assumption

- Android games
- IL2CPP

Challenges and Insights







Challenge

- How to identify vulnerable in-game purchases

Challenges and Insights

```
1 class IAPManager : IStoreListener
2 {
3   public PurchaseProcessingResult ProcessPurchase(
4                                   PurchaseEventArgs args)
5   {
6     CrossPlatformValidator validator = new CrossPlatformValidator(
7                     GooglePlayTangle.Data(), AppleTangle.Data(),
8                     Application.identifier);
9     try
10    {
11      validator.Validate(args.purchasedProduct.get_receipt());
12
13    }
14    catch (IAPSecurityException)
15    {
16      Debug.Log("Invalid receipt, not unlocking content");
17    }
18    return PurchaseProcessingResult.Complete;
19  }
20 }
```

CrossPlatformValidator.Validate(String receipt)

```
1 class PurchaseManager : IStoreListener
2 {
3   public PurchaseProcessingResult ProcessPurchase(
4                                   PurchaseEventArgs args)
5   {
6     ...
7     StoreReceipt receipt = JsonUtility.FromJson<StoreReceipt>(
8                                   args.purchasedProduct.get_receipt());
9     GooglePayload gpayload = JsonUtility.FromJson<GooglePayload>(
10
11    httpRequest.AddField("signature", gpayload.signature)
12
13  }
14  ...
15 }
```

WWWForm.AddField(String key, String value)

Challenge

- How to identify vulnerable in-game purchases

Insight

- Operating System and Unity Engine APIs

Challenges and Insights

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)

Challenge

- How to pinpoint target APIs from game binaries

Challenges and Insights

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)

Challenge

- How to pinpoint target APIs from game binaries

Insight

- Extracting meta-data file generated during compilation

Challenges and Insights

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)

0x0163cdf4 → UnityEngine.JsonUtility$$FromJson<object>

Challenge

- How to identify the payment-data definition, use, and propagation

Challenges and Insights

Address	Bytes	Disassemble	P-code Output	OP Code	P-code Input
0130d49c	c0f24794	bl 0x02509f9c	(register, 0x4000, 8)	CALL	(ram, 0x02509f9c, 8), (register, 0x40b0, 8), (const, 0x00, 8)
0130d4ac	52be0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d4c0	c01240f9	ldr x0, [x22, #0x20]	(unique, 0x100004b4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x20, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004b4, 8)
			(register, 0x4000, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d4cc	4abe0c94	bl 0x0163cdf4	(register, 0x4000, 8)	CALL	(ram, 0x0163cdf4, 8), (register, 0x4000, 8), (register, 0x4008, 8)
0130d510	d70e40f9	ldr x23, [x22, #0x18]	(unique, 0x100004d4, 8)	INT_ADD	(register, 0x4000, 8), (const, 0x18, 8)
			(unique, 0x0c90, 8)	CAST	(unique, 0x100004d4, 8)
			(register, 0x40b8, 8)	LOAD	(const, 0x01b1, 4), (unique, 0x0c90, 8)
0130d538	67eb4794	bl 0x025082d4	--	CALL	(ram, 0x025082d4, 8), ..., (register, 0x40b8, 8), (const, 0x00, 8)

UnityEngine.JsonUtility$$FromJson<object>

Challenge

- How to identify the payment-data definition, use, and propagation

Insight-direct data flow

- Using system and Unity APIs summary approaches

Challenges and Insights

```
1 class unityInAppPurchase_LS : IStoreListener
2 {
3   ...
4   private string m_LastReceipt; // 0x30
5   public PurchaseProcessingResult ProcessPurchase(
6                                   PurchaseEventArgs args)
7   {
8     this.m_LastReceipt = args.purchasedProduct.get_receipt()
9     ...
10  }
11 }
```

Class	Taint?
unityInAppPurchase_LS	
• ...	0
• ~(0x28) String	0
• ~(0x30) String	1

Global Class Table

Address	Machine Code	Disassemble	P-Code Output	Opcode	P-Code Input
0x0132946c	1c3e0294	bl 0x013b8cdc	(register, 0x4000, 8)	CALL	(ram, 0x13b8cdc, 8), (register, 0x4000, 8), (const, 0x00, 8)
0x01329470	601a00f9	str x0, [x19, #0x30]	(unique, 0x00000c90, 8)	INT_ADD	(register, 0x4098, 8), (const, 0x30, 8)
			---	STORE	(const, 0x01b1, 4), (unique, 0x00000c90, 8), (register, 0x4000, 8)

```
0x01329390 | int32_t         unityInAppPurchase_LS.ProcessPurchase(unityInAppPurchase_LS* _this, UnityEngine.Purchasing.PurchaseEventArgs* e, ...)
0x013b8cdc | System.String* UnityEngine.Purchasing.Product.get_receipt(UnityEngine.Purchasing.Product* _this, const MethodInfo* method)
```

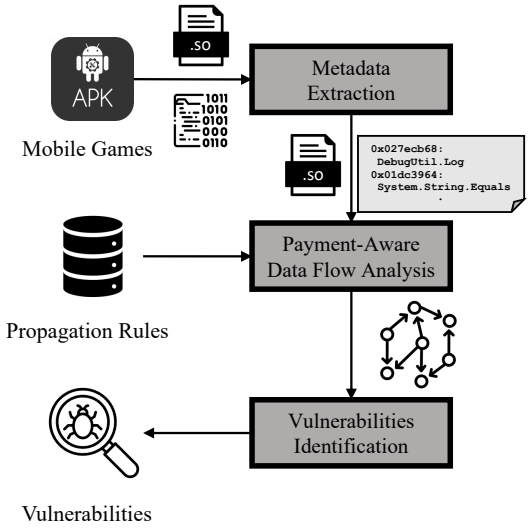
Challenge

- How to identify the payment-data definition, use, and propagation

Insight-indirect data flow

- Building global class table for propagation

PaymenScope



Three Key Components

1. Metadata Extraction
2. Payment-Aware Data Flow Analysis
3. Vulnerabilities Identification

Experiments

1. 2.1 million apps from Google Play (downloaded from AndroZoo)
2. 39,121 of them are developed based on Unity Engine and compiled by IL2CPP
3. 10,640 of them support in-game purchase.

Experiments

1. DELL server: two E5-2695 v2 CPUs (48 cores in total) and 96GB memory.
2. The experiment took 669 hours (almost 28 days) with 24 threads.

Experiments

1. DELL server: two E5-2695 v2 CPUs (48 cores in total) and 96GB memory.
2. The experiment took 669 hours (almost 28 days) with 24 threads.

Vulnerable Games

- 8,233 games with lack-of-verification
- 721 games with local verification

Experiments

1. DELL server: two E5-2695 v2 CPUs (48 cores in total) and 96GB memory.
2. The experiment took 669 hours (almost 28 days) with 24 threads.

Vulnerable Games

- 8,233 games with lack-of-verification
- 721 games with local verification

10,640 games support in-game purchase

84.15% games are vulnerable

FP and FN Analysis

Dataset - 280 games

- Top-10 local verification games
- Top-10 lack-of-verification games
- Top-10 remote verification games
- Randomly selected 200 games from vulnerable games
- Randomly selected 50 games from non-vulnerable games

- Injecting fake transaction using virtualization
- Disabling local-verification using code patching

FP and FN Analysis

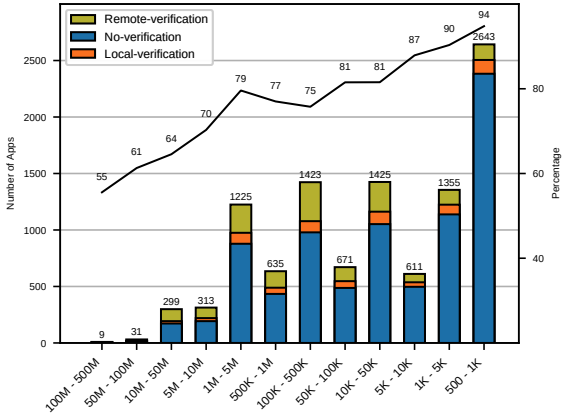
FP analysis - 220 vulnerable games

- 30 games cannot be tested
- 190 games are confirmed to be vulnerable

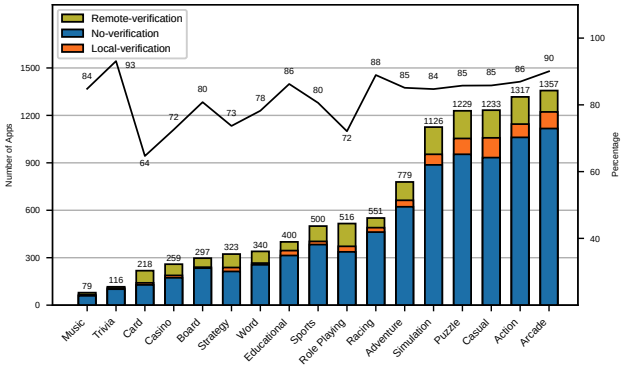
FN analysis - 60 non-vulnerable games

- 9 games cannot be tested
- 37 games are secure
- 14 games are vulnerable (i.e., FN: 29%)

Vulnerable Game Analysis



Vulnerable Game Analysis



- Off-line games

Responsible Disclosure

Unity

- Remove the local verification API
- Offer remote verification API
- Show risk in the documentation

Game developers

- Contacted 5,494 developers of the vulnerable games

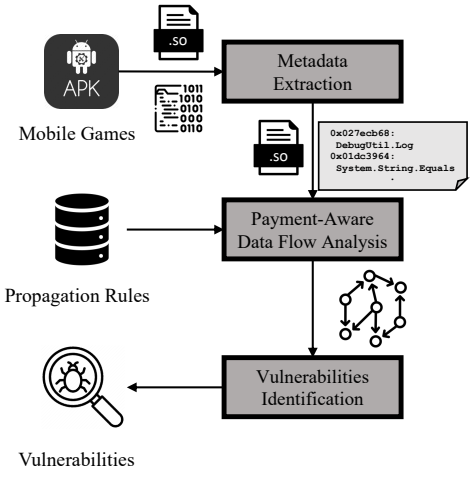
Limitation

- PaymentScope cannot handle some indirect propagations
- PaymentScope has FNs (i.e., 29%)

Related Work

1. **Game security**. Numerous efforts have been made to fight game bots [BCR08, LGZ+17, LWK+16, GWXW09]. BlackMirror [PAL20] use Intel SGX to defeat wallhacks. Tian et al. [TCM+16] studied attacks on mobile games.
2. **Binary analysis and payment security**. Over the past decades, a large body of research focusing on binary analysis [NS05, YSE+07, CLZ21, WLZ20, SWS+16, Wei84, KSS17] and payment security [WCWQ11, SXS14, WHS16, RSM+12, LH, MRK14].

Summary



PaymenScope

- A fully automated system to identify vulnerable in-game purchases
- It performs payment-aware data flow analysis on Unity games binaries

Experimental Result w/ 10,640 games

- 8,954 (84.15%) games are vulnerable

Open source

- github.com/OSUSecLab/PaymentScope

Thank You

Playing Without Paying: Detecting Vulnerable Payment Verification in Native Binaries of Unity Mobile Games

Chaoshun Zuo, Zhiqiang Lin

Department of Computer Science and Engineering
The Ohio State University

USENIX Security 2022

References I


Darrell Bethea, Robert A Cochran, and Michael K Reiter, *Server-side verification of client behavior in online games*, ACM Transactions on Information and System Security (TISSEC) **14** (2008), no. 4, 1–27.


Sanchuan Chen, Zhiqiang Lin, and Yinqian Zhang, {*SelectiveTaint*}: *Efficient data flow tracking with static binary rewriting*, 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 1665–1682.


Steven Gianvecchio, Zhenyu Wu, Mengjun Xie, and Haining Wang, *Battle of botcraft: fighting bots in online games with human observational proofs*, Proceedings of the 16th ACM conference on Computer and communications security, 2009, pp. 256–268.


Uday Khedker, Amitabha Sanyal, and Bageshri Sathe, *Data flow analysis: theory and practice*, CRC Press, 2017.


Daiping Liu, Xing Gao, Mingwei Zhang, Haining Wang, and Angelos Stavrou, *Detecting passive cheats in online games via performance-skillfulness inconsistency*, 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2017, pp. 615–626.


Yeh-chi Lai and Mohammad Husain, *A holistic approach for securing in-app purchase (iap) vulnerability in mobile applications*.


Eunjo Lee, Jiyoung Woo, Hyoungshick Kim, Aziz Mohaisen, and Huy Kang Kim, *You are a game bot!: Uncovering game bots in mmorpgs via self-similarity in the wild.*, Ndss, 2016.


Collin Mulliner, William Robertson, and Engin Kirda, *Virtualswindle: An automated attack against in-app billing on android*, Proceedings of the 9th ACM symposium on Information, computer and communications security, 2014, pp. 459–470.


James Newsome and Dawn Xiaodong Song, *Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software.*, NDSS, vol. 5, Citeseer, 2005, pp. 3–4.

References II


Seonghyun Park, Adil Ahmad, and Byoungyoung Lee, *Blackmirror: Preventing wallhacks in 3d online fps games*, Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 987–1000.


Daniel Reynaud, Dawn Xiaodong Song, Thomas R Magrino, Edward XueJun Wu, and Eui Chul Richard Shin, *Freemarket: Shopping for free in android applications.*, NDSS, 2012.


Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Andrew Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, et al., *Sok:(state of) the art of war: Offensive techniques in binary analysis*, 2016 IEEE Symposium on Security and Privacy (SP), IEEE, 2016, pp. 138–157.


Fangqi Sun, Liang Xu, and Zhendong Su, *Detecting logic vulnerabilities in e-commerce applications.*, NDSS, 2014.


Yuan Tian, Eric Chen, Xiaojun Ma, Shuo Chen, Xiao Wang, and Patrick Tague, *Swords and shields: a study of mobile game hacks and existing defenses*, Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 386–397.


Rui Wang, Shuo Chen, XiaoFeng Wang, and Shaz Qadeer, *How to shop for free online–security analysis of cashier-as-a-service based web stores*, 2011 IEEE Symposium on Security and Privacy, IEEE, 2011, pp. 465–480.


Mark Weiser, *Program slicing*, IEEE Transactions on software engineering (1984), no. 4, 352–357.


Yong Wang, Christen Hahn, and Kruttika Sutrave, *Mobile payment security, threats, and challenges*, 2016 second international conference on mobile and secure services (MobiSecServ), IEEE, 2016, pp. 1–5.


Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang, *Firmxray: Detecting bluetooth link layer vulnerabilities from bare-metal firmware*, Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 167–180.

References III

Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda, *Panorama: capturing system-wide information flow for malware detection and analysis*, Proceedings of the 14th ACM conference on Computer and communications security, 2007, pp. 116–127.