Show Me the Money! Finding Flawed Implementations of Third-party In-app Payment in Android Apps

Yang Wenbo, Zhang Yuanyuan, Li Juanru,
Liu Hui, Wang Qing, Zhang Yueheng, Gu Dawu

Shanghai Jiao Tong University

Introduction

- Mobile payment has developed dramatically (especially in China) in recent years
- Previous work mainly focused on security of traditional web payment
- No unified specification or assessment approach to validate the security

In-app Payment Demystified

- In-app payment
  - Merchant App (MA)
  - Merchant Server (MS)
  - 3rd-Party Payment SDK(TP-SDK)
  - Cashier Server (CS)
- China market
  - AliPay, WexPay, UniPay, BadPay
  - 1/3 use 3rd party payment

TABLE I: TP-SDK Distribution

Cashier	Number
WexPay	2260
AliPay	1299
UniPay	574
BadPay	34
Total	2679
Sample	7145

In-app Payment Process Model

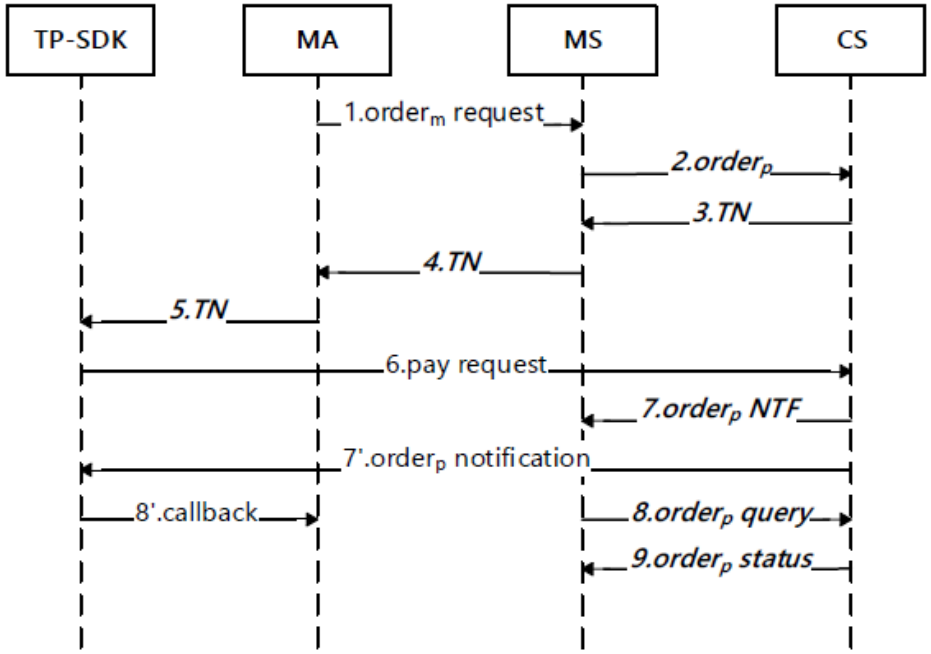

Fig. 1: In-app Payment Process Model I adopted by WexPay and UniPay

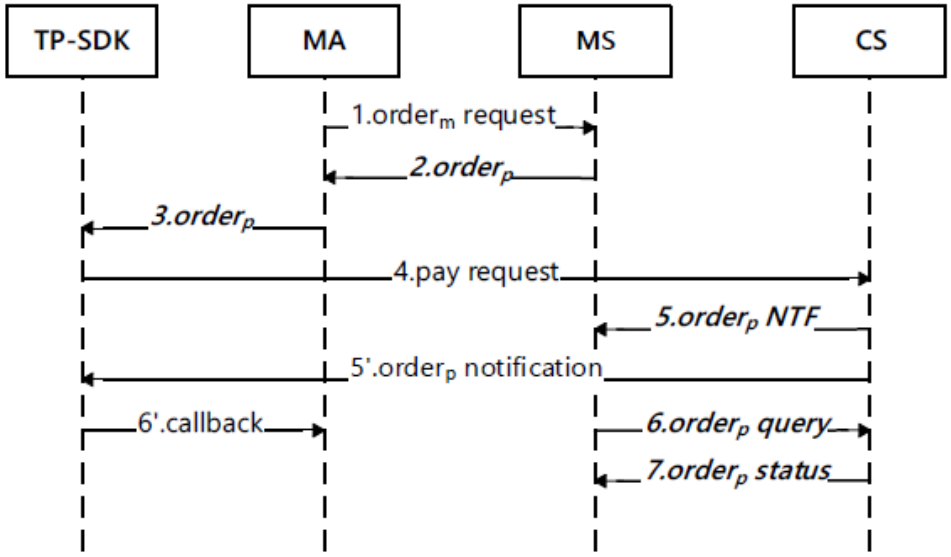

Fig. 2: In-app Payment Process Model II adopted by AliPay and BadPay

Security Analysis

- Adversary Model
  - Attackers can reverse-engineering MA and the embedded TP-SDK
  - Forge request or message to MS and CS
  - Attack targets cashier or merchant
    - Attacker plays the role of a malicious user
    - Manipulate execution or data of local app and system
  - Attack targets other users of merchant app
    - Control the data transmission
    - Perform MITM attack with ARP spoofing or malicious WiFi

Security Rules

I. Payment orders must be generated/signed by MS
II. Never expose any secret (the signing KEY)
III. TP-SDK inform user detailed information of payment order
IV. TP-SDK verify the owner (MA) of transaction
V. Use secure network communication
VI. Server verify the signature of received messages
VII. MS re-confirm the notified payment to CS

Order Tampering Attack

- Fail to generate or sign payment order in server
- Fail to re-confirm the payment to CS
- Tamper the content (total amount) in payment order and pay less money

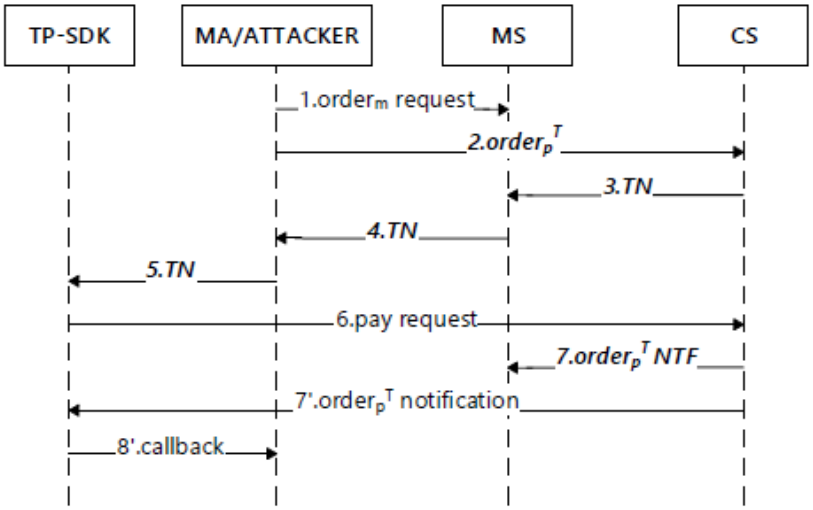

Fig. 3: Order Tampering Attack to Process Model I

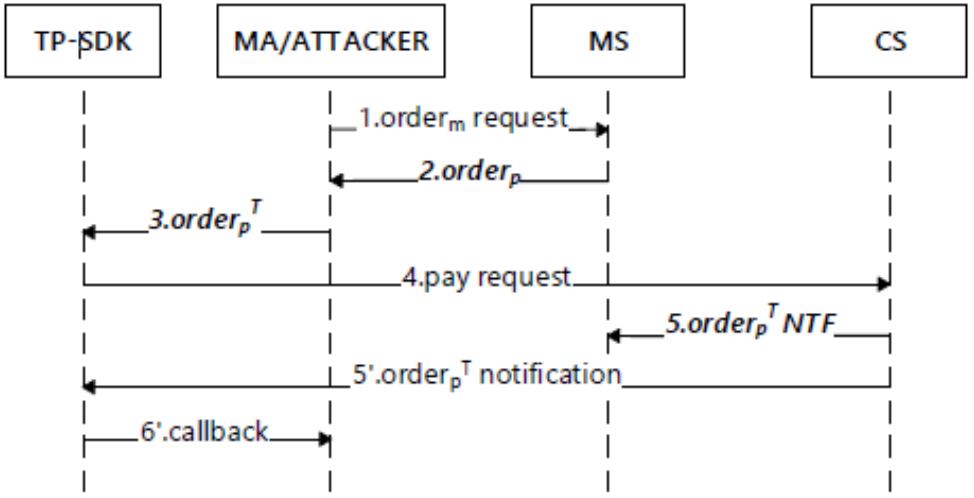

Fig. 4: Order Tampering Attack to Process Model II

Notification Forging Attack

- Fail to verify the message’ s signature/leak the KEY
- Fail to re-confirm the payment to CS
- Purchase things without paying

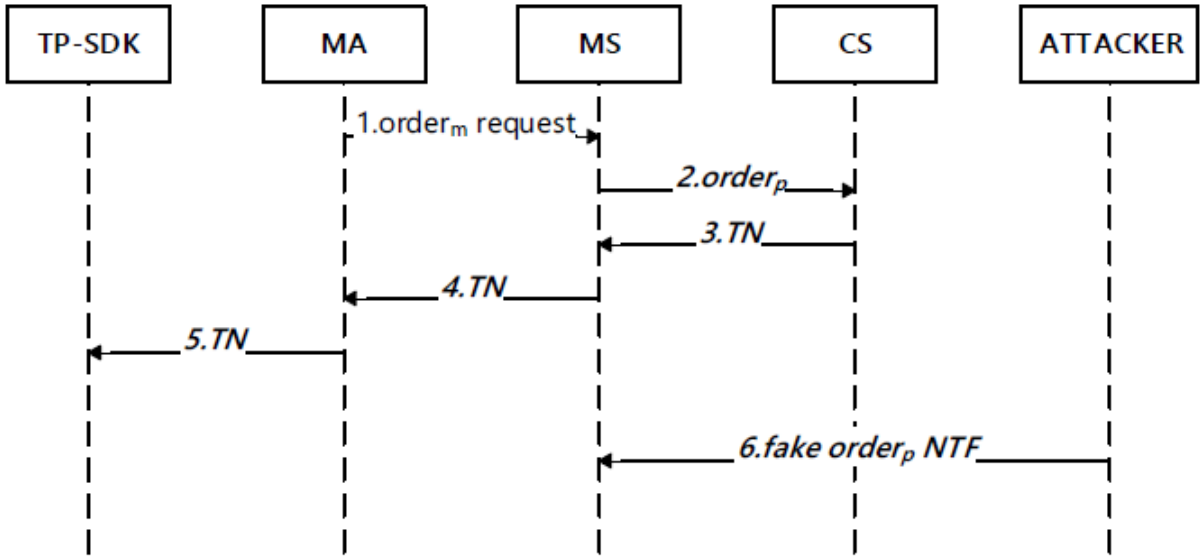

Fig. 5: Notification Forging Attack to Process Model I

Order Substituting Attack

- Target users rather than merchant
- Insecure network between MS and MA
- TP-SDK incomplete prompt and missing transaction verification
- Substitute an order of one transaction to another, mislead a victim user to pay for the attacker’s order

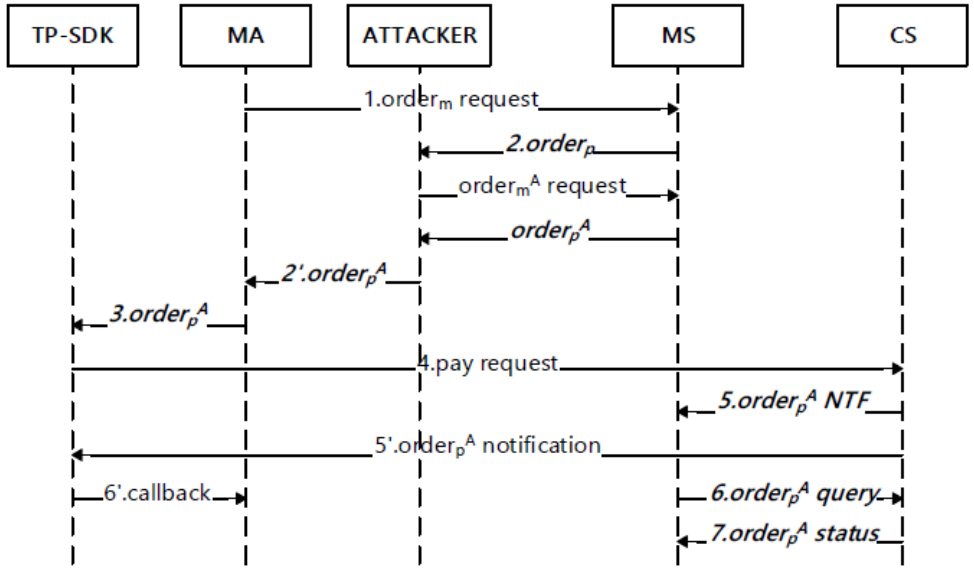

Fig. 6: Order Substituting Attack to Process Model II

Unauthorized Querying Attack

- Leak the signing KEY
- query every transaction recorded in CS, acquiring secret business information which should only be shared by cashier and merchant

Detecting Flawed In-app Payment

- Local Ordering
  - Violation of Security Rule 1
  - Search the URL of placing payment orders in MA (https://api.mch.weixin.qq.com/pay/unifiedorder for WexPay)
- KEY Leakage
  - Violation of Security Rule 2
  - Feature of KEY (Base64-encoded ASN1 private key of AliPay)
  - Web API to verify the exact signing key of WexPay

Detecting Flawed In-app Payment

- Incomplete Prompt
  - Violation of Security Rule 3
  - Check the payment orderID, commodity, owner, merchant, money
- Transaction Verification Missing
  - Violation of Security Rule 4
  - Whether TP-SDK accepts a payment order does not belong to the host MA

Detecting Flawed In-app Payment

- Insecure Communication
  - Violation of Security Rule 5
  - Set proxy to perform MITM between MA (TP-SDK) and MS (CS)
- Notified Payment Confirmation Missing
  - Violation of Security Rule 6
  - Whether the MS accepts the tampered payment order with valid signature

Detecting Flawed In-app Payment

- Signature Validation Missing
  - Violation of Security Rule 7
  - Place an order without paying for it
  - Forge an order notification to MS with invalid signature
  - Whether MS accepts it
  - Sample based on the result of notified payment confirmation missing

Empirical Study

Cashier	KEY leakage	Local Ordering
WexPay	155	104
AliPay	398	/
UniPay	0	0
BadPay	7	/

TABLE II: Flaws in Merchant Apps

Cashier	Transaction Verification	Information Prompt					Network Communication
		orderID	commodity	owner	merchant	money	
WexPay	✓	×	✓	×	✓	✓	secure private protocal
AliPay	×	×	✓	×	×	✓	HTTPS pinning
UniPay	×	✓	✓	×	✓	✓	HTTPS pinning
BadPay	×	×	×	×	×	✓	HTTPS validation

TABLE III: flaws in TP-SDKs

Empirical Study

- Flaws in MS
  - 9/15 miss the confirmation of notified payment.
  - 2/9 miss the validation of received message’s signature
- Insecure Communication
  - 49/87 apps vulnerable
  - 45 use HTTP, 42 use HTTPS
  - 4/42 fail to validate SSL certificate properly

Root cause Inquiry

- Cashier
  - Mistakes in sample code
  - Mistakes in official doc
  - Conflict between code and doc
  - Lack of sample code implementation of server
  - Compromise for business
- Merchant
  - Weak keys

Ethical Consideration

- Several case studies in paper
- Report all the findings to Tencent/Ant Financial and Baidu Security Response Center
- Return/repay items in our cases

Thank you

Group of Software Security In Progress (GoSSIP)
Lab of Cryptology and Computer Security (LoCCS)
Shanghai Jiao Tong University (SJTU)