Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023
Introduction

Payment Systems:
- Card Present (CP)
- Card Not Present (CNP)

CP Acceptance Terminals:
- Traditionally: Point of Sale (PoS)
- Recently: mobile PoS (mPoS)
mPoS Terminals

**mobile PoS Terminals**: small, compact, low-cost, wireless, easy to configure

**Accept various payment methods**: Contact, Contactless, QR Code

**Accept various devices**: card, mobile, watch, wearables
Introduction

Payment Systems:
- Card Present (CP)
- Card Not Present (CNP)

CP Acceptance Terminals:
- Traditionally: Point of Sale (PoS)
- Recently: mobile PoS (mPoS)
Some Potential Risks

- Lock-screen bypass for mobile payments [22]
- PIN bypass for over the contactless limit [3-5]
- and ...
- Relay (Digital Pickpocketing) [17]
Some Potential Risks

- Lock-screen bypass for mobile payments [22]
- PIN bypass for over the contactless limit [3-5]
- and ...
- Relay (Digital Pickpocketing) [17]
Some Potential Risks

- Lock-screen bypass for mobile payments [22]
- PIN bypass for over the contactless limit [3-5]
- and ...
- Relay (Digital Pickpocketing) [17]
Introduction

Payment Systems:
- Card Present (CP)
- Card Not Present (CNP)

CP Acceptance Terminals:
- Traditionally: Point of Sale (PoS)
- Recently: mobile PoS (mPoS)
Ecosystem

- **Crucial Component:** Mobile Phone
- **Roles:**
  - Communication with mPoS terminal
  - Connection to payment provider
  - Mobile Application
- **Proof of concept:** SumUP
Ecosystem

• Crucial Component: Mobile Phone

• Roles:
  • Communication with mPoS terminal
  • Connection to payment provider
  • Mobile Application

• Proof of concept: SumUP
Introduction

Payment Systems:
- Card Present (CP)
- Card Not Present (CNP)

CP Acceptance Terminals:
- Traditionally: Point of Sale (PoS)
- Recently: mobile PoS (mPoS)
### Related Work

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>Frisby et. al. [10]:</td>
<td>disable the magnetic stripe reader in audio-jack magnetic stripe reader (AMS) by arbitrary application running and obtain cryptographic keys</td>
</tr>
<tr>
<td>2014</td>
<td>MWR Lab [15]:</td>
<td>utilize USB and Bluetooth interfaces, get root access, 1) execute arbitrary command 2) full control over screen (&quot;Try again&quot;)</td>
</tr>
<tr>
<td>2015</td>
<td>Mellen et. al. [18]:</td>
<td>bypass the encryption by crushing the encryption chip, recording unencrypted swipes and transmit the credit card information to an external server</td>
</tr>
<tr>
<td>2018</td>
<td>Galloway and Yunusov [11]:</td>
<td>exploit BLE interface, send arbitrary commands (&quot;please swipe card&quot;) and tamper with amounts (Sumup transmitted commands in plaintext!)</td>
</tr>
</tbody>
</table>
Introduction

Payment Systems:
- Card Present (CP)
- Card Not Present (CNP)

CP Acceptance Terminals:
- Traditionally: Point of Sale (PoS)
- Recently: mobile PoS (mPoS)
Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023
Encryption Security

- Communication: BLE
- Protocol Stack: Controller, (HCI), Host, Application
- Our interest: Security Manager Protocol (SMP)
  - Contains pairing
  - Generates and distributes keys
- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys
Encryption Security

- Communication: BLE
- Protocol Stack: Controller, (HCI), Host, Application
- Our interest: Security Manager Protocol (SMP)
  - Contains pairing
  - Generates and distributes keys
- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys
Encryption Security

- Communication: **BLE**

- Protocol **Stack**: Controller, (HCI), Host, Application

- Our interest: **Security Manager Protocol (SMP)**
  - Contains pairing
  - Generates and distributes keys

- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys
Phase 1:

- Pairing Request (I/O, OOB, BF, SC, Key size, ...)
- Pairing Response (I/O, OOB, BF, SC, Key size, ...)

Phase 2:

- Pairing mechanism:
  - Legacy Pairing (TK ==> STK ==> LTK)
  - SC: Secure Connection (ECDH: LTK)

- Pairing method:
  - Just Works Unauthenticated (TK=0)
  - Out of Band (OOB)
  - Passkey (TK: 6 digit)
  - Numeric Comparison
Encryption Security

- Communication: BLE
- Protocol Stack: Controller, (HCI), Host, Application
- Our interest: Security Manager Protocol (SMP)
  - Contains pairing
  - Generates and distributes keys
- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys
Eavesdropping

- **Threat Model:** malicious merchant or eavesdropper

- **Tools:**
  - HCI Snoop Log
  - BLE Over-the-air Sniffer
Encryption Security

- Communication: **BLE**
- Protocol **Stack**: Controller, (HCI), Host, Application
- Our interest: **Security Manager Protocol (SMP)**
  - Contains pairing
  - Generates and distributes keys
- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys
Extract Cryptographic Keys

- Request: Keyboard & Display ~
  Response: No I/O

- Pairing: LE Legacy

- Key Generation: Just Works (Unauthenticated)

- Temporary Key (TK): Zero

- Extract LTK!

- Crack: "Decrypt with LTK"
  - Input: encrypted file + LTK
  - Output: decrypted file
- Request: Key Generation
- Response: None

- Pairing Information

- Key Generation (Unauthorized)

- Temporary Key
<table>
<thead>
<tr>
<th>Field</th>
<th>Pairing Request Value</th>
<th>Pairing Request Meaning</th>
<th>Pairing Response Value</th>
<th>Pairing Response Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>0x01</td>
<td>Pairing Request</td>
<td>0x02</td>
<td>Pairing Response</td>
</tr>
<tr>
<td>I/O</td>
<td>0x04</td>
<td>Keyboard/Display</td>
<td>0x03</td>
<td>No I/O</td>
</tr>
<tr>
<td>OOB</td>
<td>0x00</td>
<td>NOT Present</td>
<td>0x00</td>
<td>NOT Present</td>
</tr>
</tbody>
</table>

**Authentication Request**

<table>
<thead>
<tr>
<th>Bonding</th>
<th>0x1</th>
<th>Bonding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MITM</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>SC</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>KP</td>
<td>0</td>
<td>False</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x0</td>
<td>False</td>
</tr>
<tr>
<td>Max Enc.</td>
<td>16</td>
<td>Max Enc. Size</td>
</tr>
</tbody>
</table>

**Initiator Key Distribution**

<table>
<thead>
<tr>
<th>LTK</th>
<th>1</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRK</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>CSRK</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>Link Key</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x0</td>
<td>False</td>
</tr>
</tbody>
</table>

**Responder Key Distribution**

<table>
<thead>
<tr>
<th>LTK</th>
<th>1</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRK</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>CSRK</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>Link Key</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x0</td>
<td>False</td>
</tr>
</tbody>
</table>
Extract Cryptographic Keys

- Request: Keyboard&Display ~ Response: No I/O
  - Pairing: **LE Legacy**
  - Key Generation: **Just Works (Unauthenticated)**
  - Temporary Key (TK): **Zero**
  - Extract **LTK**!
  - Crackle: "**Decrypt with LTK**"
    - Input: encrypted file + LTK
    - Output: decrypted file
iZettle:

- Secure Connection
- Numeric comparison

Not Common Practice!
Extract Cryptographic Keys

- Request: Keyboard&Display ~
  Response: No I/O

  - Pairing: **LE Legacy**
  - Key Generation: **Just Works (Unauthenticated)**
  - Temporary Key (TK): **Zero**
  - Extract **LTK**!

- Crackle: "**Decrypt with LTK**"
  - Input: encrypted file + LTK
  - Output: decrypted file
Encryption Security

- Communication: **BLE**
- Protocol **Stack**: Controller, (HCI), Host, Application
- Our interest: **Security Manager Protocol (SMP)**
  - Contains pairing
  - Generates and distributes keys
- Pairing Phases:
  - Phase 1: exchange pairing feature
  - Phase 2: determines pairing mechanism
  - Phase 3: distributes keys

Pairing
Eavesdropping
Attack
Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023
Network Security

- Communication: **HTTPS** (uses TLS)
- Threat model: **man-in-the-middle (MITM)**

- Proxy server: **mitmproxy**
  - intercept and decrypt traffic
Network Security

- Communication: **HTTPS** (uses TLS)
- Threat model: **man-in-the-middle (MITM)**

- Proxy server: **mitmproxy**
  - intercept and decrypt traffic
HTTPS Interception

1. Set up manual **proxy** configuration on the phone

2. Install mitmproxy **Certificate Authority (CA)** on the phone

3. Bypass **Certificate Pinning** (allowing user-added certificate) by modifying the app

4. The modified app now **trusts** the mitmproxy certificate!
The modified app now trusts the MitM proxy certificate!
HTTPS Interception

1. Set up manual **proxy** configuration on the phone

2. Install mitmproxy **Certificate Authority (CA)** on the phone

3. Bypass **Certificate Pinning** (allowing user-added certificate) by modifying the app

4. The modified app now **trusts** the mitmproxy certificate!
Network Security

- Communication: HTTPS (uses TLS)
- Threat model: man-in-the-middle (MITM)
- Proxy server: mitmproxy
  - intercept and decrypt traffic
Tampering Attack

- **Tampering** with the (protected) messages

- Command "**PINPLUS SHOW DEFAULT MESSAGE**" coded in **plain text** Hexadecimal

- Inserting **arbitrary commands** to force the terminal to change the displayed message ("Declined!")

- **Challenge**: Protected messages are rejected
- **Solution**: Send “leave_protected_session” command first!

![Images with a code card and a mobile device]
Tampering Attack

- **Tampering** with the (protected) messages
- Command "PINPLUS SHOW DEFAULT MESSAGE" coded in **plain text** Hexadecimal
- Inserting **arbitrary commands** to force the terminal to change the displayed message ("Declined!")
- **Challenge**: Protected messages are rejected
- **Solution**: Send "leave_protected_session" command first!
Tampering Attack: Barbie Version!

Not a barbie girl, but stuck in the barbie world!
Not a barbie girl, but stuck in the barbie world!
Tampering Attack

- Tampering with the (protected) messages

- Command "PINPLUS SHOW DEFAULT MESSAGE" coded in plain text Hexadecimal

- Inserting arbitrary commands to force the terminal to change the displayed message ("Declined!")

- Challenge: Protected messages are rejected
- Solution: Send “leave_protected_session” command first!
Network Security

- Communication: **HTTPS** (uses TLS)
- Threat model: **man-in-the-middle (MITM)**
- Proxy server: **mitmproxy**
  - intercept and decrypt traffic
Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023

- Background
- Encryption Security
- Network Security
- Software Security
- Conclusion
Software Security

- Mobile application: manages the terminal
- Reverse-engineering to identify vulnerabilities in the code
- Proof of concept: Android phone, APK
Software Security

- Mobile application: manages the terminal
- Reverse-engineering to identify vulnerabilities in the code
- proof of concept: Android phone, APK

Reverse Engineering

Attacks
Reverse Engineering Steps

1. Download genuine APK
2. Decompile the APK
   - apktool: Smali code (main)
   - Java decompiler: Java code (complementary)
3. Make Modifications
4. Recompile the APK (e.g., apk-mitm)
5. Sign the APK (e.g., uber-apk-signer)
6. Re-install compromised App!
Software Security

- Mobile application: manages the terminal
- Reverse-engineering to identify vulnerabilities in the code
- Proof of concept: Android phone, APK
Attacks

1. Bypass Certificate Pinning
   - Replace the application's network security config to allow user-added certs.
   - Modify the code to disable cert. pinning implementation

2. Bypass Protected Messages
   - Leave Protected Session

3. Disable Security Features: Beep Sound
   - Find "AudioManager" Class ==> "PlaySoundEffect" method
   - Modify or remove
   - Sounds are muted!

4. ...
Software Security

- Mobile application: manages the terminal
- Reverse-engineering to identify vulnerabilities in the code
- proof of concept: Android phone, APK
Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023

- Background
- Encryption Security
- Network Security
- Software Security
- Conclusion
Conclusion

- mPoS terminals can be vulnerable in various ways
- The involvement of merchant’s phone can make it worse!
Potential Solutions

- Secure Pairing on BLE
- Code Obscuring
- Anti-tampering
- Abuse Detection

- Requires Further Research!
Conclusion

- mPoS terminals can be vulnerable in various ways
- The involvement of merchant’s phone can make it worse!

Potential Solutions

What’s next?

Thanks!
What's next?

Tap-to-phone!
- Potential Solution
- Susceptible to risks
- Requires further research
Conclusion

- mPoS terminals can be vulnerable in various ways
- The involvement of merchant’s phone can make it worse!
Conclusion

- mPoS terminals can be vulnerable in various ways
- The involvement of merchant's phone can make it worse!
Thank you!

Any questions?

Mahshid.mehr-nezhad@warwick.ac.uk

@MahshiidMehr

@mahshidmehr
**Conclusion**

- mPoS terminals can be vulnerable in various ways
- The involvement of merchant’s phone can make it worse!

**Potential Solutions**

**What's next?**

**Thanks!**
Security Analysis of Mobile Point-of-Sale Terminals

Mahshid Mehr Nezhad, Elliot Laidlaw, Feng Hao
University of Warwick, UK

Network and System Security 2023

- Background
- Encryption Security
- Network Security
- Software Security
- Conclusion