June 16, 2017

BHIM App Vulnerability Analysis Preliminary Disclosure

1 Executive summary

We conduct this study to assess the state of security of financial applications both from the context of the application and the Trusted Execution Environment of the device. BHIM was one of the apps chosen for the study. The entire testing process was carried in a manner befitting a malicious actor with the goal to determine if an attacker can:

1. Bypass any of the authentication controls on the app
2. Leak sensitive data
3. Ease with which an attacker can launch an attack against a compromised user

The study is being conducted in phases. This report presents the preliminary results of our analysis on the authentication subsystem of BHIM app. The report is our initiative towards responsibly disclosing the bugs and design flaws of the app, some of which are critical. We detail the attack that has resulted in a leak of **OTP, passcode** and **UPI PIN, and demonstrate how the stolen details were used to compromise a user by logging into his BHIM account from an attacker device.** The attacks were carried out on the authentication subsystem with privileges granted to a regular user. The detailed findings and remediation recommendations for these assessments may be found later in the report.

Credits: XXX

2 Summary of Vulnerabilities

The assessment utilized tools for both static and dynamic analysis and also involved considerable amount of manual reverse engineering of the app. During manual analysis, we attempted to leverage vulnerabilities discovered using the tools to test for key security flaws. The following vulnerabilities were found in the authentication subsystem.

CWE ID	Vulnerability	Remediation
319	Cleartext Transmission of Sensitive Information	Sensitive user inputs must be hidden from other apps in the same device
346	Origin Validation Error	Validate origin of the OTP message
288	Authentication Bypass Using an Alternate Path or Channel	Check for integrity of the app
287	Improper Authentication	Use correct form of three factor authentication
282	Improper Ownership Management	Implement checking the integrity of the app

3 Process Methodology

A comprehensive methodology was employed to perform security assessment of BHIM. The process began with detailed scanning and research into the architecture of the app. It was discovered that BHIM app does not use any TEE infrastructure.

3.1 Reconnaissance

The primary goal in this process was to discover any known vulnerabilities in BHIM app. Reconnaissance was carried out using an automated framework called MobSF. The following is the report generated by MobSF.

Enumeration	Description
Assessment Type	Black-Box
Automated Tool	MobSF
Native	False
Dynamic	False
Reflection	True
Crypto	True
Use of WebView	Yes
Number of exported Activities	0
Number of exported Services	0
Number of exported Receivers	1
Number of exported Providers	0

3.2 Tools Used

Static disassembly and decompilation tools such as JEB, JD-GUI, dex2jar, JADX, were used for static analysis and manual reversing. This analysis serves as a base for finding the possible vulnerabilities, as well as to author the repackaged variant of the app. APKtool was used to repackage the app. AndroidMonitor was used to trace dynamic method invocations.

3.3 Exploitation

Exploitation of BHIM involves the use of a legitimate version of the app on the victim's device and a repackaged version of the app on the attacker device. A Trojan on the victim device was used to leak sensitive information it. The exploit that is detailed below will demonstrate how an attacker could login to a compromised user's BHIM account from his device.

4 Overview of Authentication Subsystem of BHIM App

The multi-factor authentication subsystem as publicized by BHIM app is discussed below.

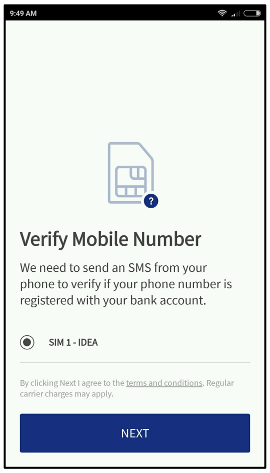
1. The app binds with a device's ID and mobile number (device authentication)
2. User must sync his bank account (UPI or non-UPI enabled) with the app to operate.
3. User must set a passcode (4-digit number) to login to the app.

Steps 1 and 2 above is performed as a part of BHIM app configuration. For a user to log in, only the passcode is required. A conventional multi-factor authentication system requires multiple factors (what you know, what you have, what you are) for authentication. For example, a two-factor authentication used by financial systems involves a password and OTP to login. This is in contrast to what BHIM offers. This can be easily bypassed.

Device Authentication Workflow

There are three ways to authenticate a device:

1. An SMS is sent from the device to verify if the user's phone number is linked to a bank account. If the phone number is linked to a bank account, then the user is prompted for a passcode.
2. If (1) fails (i.e. SMS sending fails), then the user is prompted to enter the mobile number that is registered with the bank account. An OTP is sent to the now entered phone number.
3. If (2) fails then the phone is verified using USSD code.



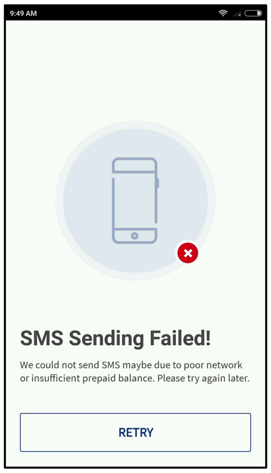
Verify Mobile Number

We need to send an SMS from your phone to verify if your phone number is registered with your bank account.

● SIM 1 - IDEA

By clicking Next I agree to the terms and conditions. Regular carrier charges may apply.

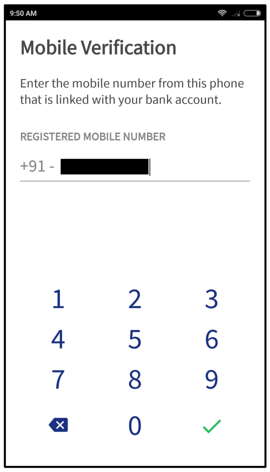
SMS Verification



SMS Sending Failed!

We could not send SMS maybe due to poor network or insufficient prepaid balance. Please try again later.

SMS Verification Failed



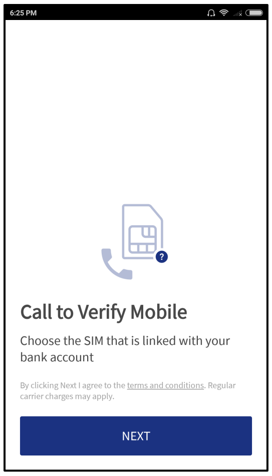
Mobile Verification

Enter the mobile number from this phone that is linked with your bank account.

REGISTERED MOBILE NUMBER

+91 - [redacted]

OTP-based Verification



Call to Verify Mobile

Choose the SIM that is linked with your bank account

By clicking Next I agree to the terms and conditions. Regular carrier charges may apply.

USSD Verification

5 Attack Workflow

This section details the end-to-end attack on BHIM app.

5.1 Setup & Assumptions

The attack involves a victim device and an attacker device. The victim device has the legitimate version of BHIM app installed and it is assumed that the user has gone through the device authentication process entailed by BHIM. At some point the device has been compromised resulting in a Trojan running in the background. The Trojan has some basic features such as:

1. Intercept SMS messages.

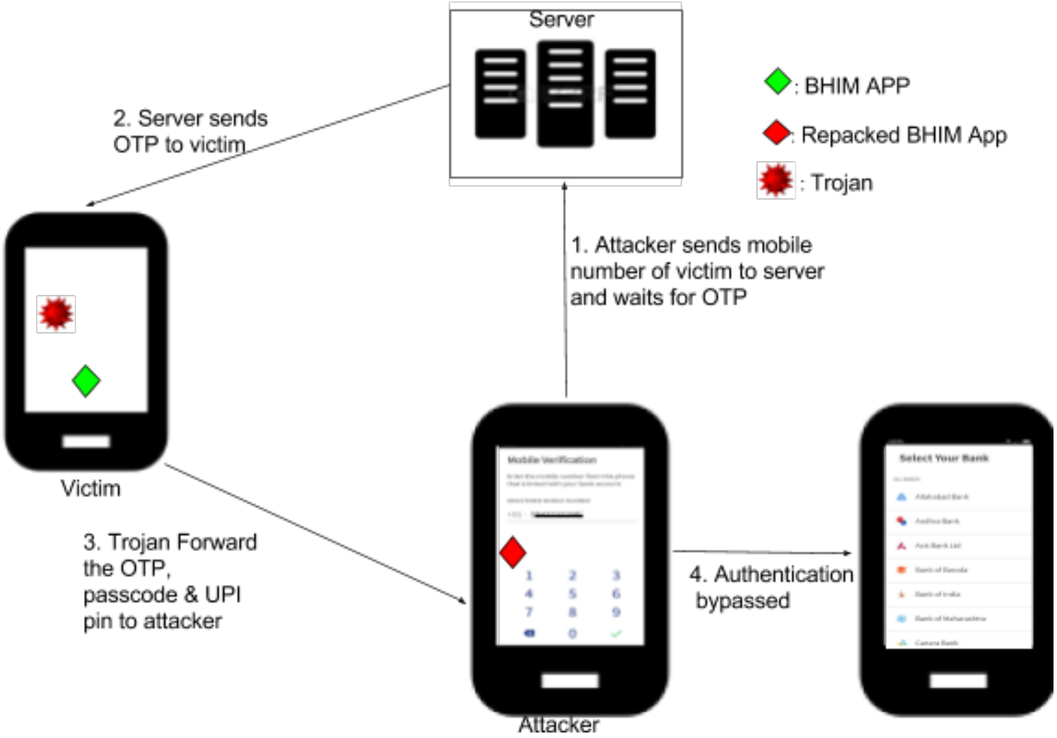
2. Sniff and log passcode into a file.
3. Forward OTP and passcode to attacker.

On the attackers device, a repackaged version of the app is installed with certain modified functionalities so as to login to the victims account.

5.2 Attack Explained

1. **Victim>** During some run of BHIM on the victim device, the trojan saves passcode of BHIM app into a text file. The trojan also sniffs the UPI PIN when the victim makes a transaction.
2. **Attacker>** Installs a repackaged BHIM app. The attacker bypasses SMS verification of device. This is accomplished by exploiting the airplane mode or by leaving insufficient funds to send the SMS. This will subsequently launch the second method of authentication using OTP.
3. **Attacker>** For OTP-based verification of the attacker device, the attacker is prompted for the phone number of the device he wishes to authenticate. At this point, the attacker enters the victims phone number (since the phone number is verified with the phone number the victim has registered with the bank). At this point an OTP is sent to the victims device
4. **Victim>** The Trojan on victim device intercepts the OTP and forwards it along with passcode and UPI PIN to the attacker as SMS.
5. **Attacker>** The repackaged app on attacker device allows verification of OTP.

The diagram below shows the overall attack workflow.


Server
🟩: BHIM APP
🟥: Repacked BHIM App
✹: Trojan
2. Server sends OTP to victim
1. Attacker sends mobile number of victim to server and waits for OTP
Victim
3. Trojan Forward the OTP, passcode & UPI pin to attacker
4. Authentication bypassed
Attacker

At this point the attacker has

a. OTP to bypass the device authentication
b. Passcode for entering to the app each time
c. UPI PIN which helps in completing a transaction

6 Vulnerability Findings

6.1 Cleartext Transmission of Sensitive Information

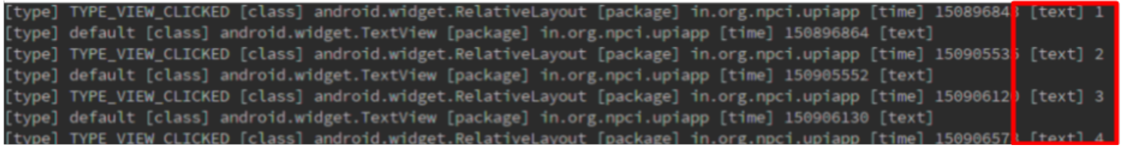
Description (from CWE website): The software transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.

BHIM app uses a custom keypad for password and PIN entry. Customization introduced a bug wherein it allows a third-party app to sniff the "passcode" and "UPI pin". The attribute on the input textbox of the keypad is categorized as "not password" thus allowing it to be sniff-able through accessibility service.

Exploit:

1. Create an Trojan app that has permission to use accessibility service.
2. The accessibility service intercepts all events and data (sensitive or not) from BHIM app.

The digits highlighted in red below shows the a sample output of sniffed passcode.



```
[type] TYPE_VIEW_CLICKED [class] android.widget.RelativeLayout [package] in.org.npci.upiapp [time] 15089684 [text] 1
[type] default [class] android.widget.TextView [package] in.org.npci.upiapp [time] 150896864 [text]
[type] TYPE_VIEW_CLICKED [class] android.widget.RelativeLayout [package] in.org.npci.upiapp [time] 15090553 [text] 2
[type] default [class] android.widget.TextView [package] in.org.npci.upiapp [time] 150905552 [text]
[type] TYPE_VIEW_CLICKED [class] android.widget.RelativeLayout [package] in.org.npci.upiapp [time] 15090612 [text] 3
[type] default [class] android.widget.TextView [package] in.org.npci.upiapp [time] 150906130 [text]
[type] TYPE_VIEW_CLICKED [class] android.widget.RelativeLayout [package] in.org.npci.upiapp [time] 15090657 [text] 4
```

Fig : Output from logcat as the app receives the passcode

Remediation

The keypad must be designed to handle sensitive data (as is the case for password fields). The passcode and UPI pin are to be masked prior to use.

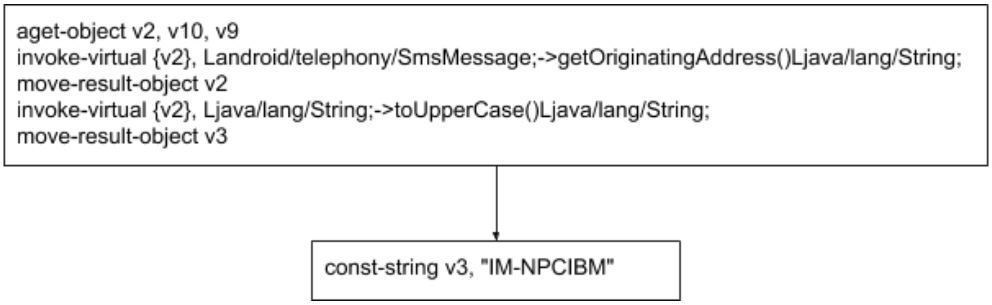
6.2 Origin Validation Error

Description: The software does not properly verify that the source of data or communication is valid.

The source of the OTP SMS is validated by comparing it with strings hard-coded into the application. This can be exploited to enable verification.

Exploit:

Manual reversing of BHIM app revealed that the SMS was verified by comparing with one of the strings { IM-NPCIBM, AK-NPCIMB, VK-NPCIBM}. BHIM uses *getOriginatingAddress()* API call to validate SMS. Smali code containing a hard-coded string "IM-NPCIBM" was injected into the app to enable successful verification by string comparison. BHIM app was subsequently repackaged for use by the attacker. When an OTP is received, originating address will be the string IM-NPCIBM, which will evaluate to true.



```
aget-object v2, v10, v9
invoke-virtual {v2}, Landroid/telephony/SmsMessage;->getOriginatingAddress()Ljava/lang/String;
move-result-object v2
invoke-virtual {v2}, Ljava/lang/String;->toUpperCase()Ljava/lang/String;
move-result-object v3
```

```
const-string v3, "IM-NPCIBM"
```

Remediation
Proper validation of source of OTP message.

6.3 Authentication Bypass Using an Alternate Path or Channel

Description: A product requires authentication, but the product has an alternate path or channel that does not require authentication

An alternate channel using OTP was used for authentication. We used a message from a different source (victim device) other than the source address. We used one of the following strings { IM-NPCIBM, AK-NPCIMB, VK-NPCIBM}.

Exploit: Described above

6.4 Improper Ownership Management

Description
The software assigns the wrong ownership, or does not properly verify the ownership, of an object or resource.

Exploit
A repackaged app modifies the ownership of the app. Table below shoes the signatures of the original app and the repackaged app. BHIM app does not check for the authenticity of the app.

Signature of BHIM App	**Signature of Repackaged BHIM App**
Signature Algorithm: sha256WithRSAEncryption **Issuer**: C=IN, ST=KARNATAKA, L=BANGALORE, O=NPCI, OU=NPCI, CN=BHIM **Validity** Not Before: Dec 30 02:53:53 2016 GMT Not After: Dec 24 02:53:53 2041 GMT **Subject:** C=IN, ST=KARNATAKA, L=BANGALORE, O=NPCI, OU=NPCI, CN=BHIM **Subject Public Key Info:** Public Key Algorithm: rsaEncryption Public-Key: (2048 bit)	**Signature Algorithm:** sha256WithRSAEncryption **Issuer:** ST=Gotham, L=Gotham, O=Cyber, OU=Amrita, CN=JMR **Validity** Not Before: Nov 6 02:45:12 2016 GMT Not After : Oct 31 02:45:12 2041 GMT **Subject**: ST=Gotham, L=Gotham, O=Cyber, OU=Amrita, CN=JMR **Subject Public Key Info:** Public Key Algorithm: rsaEncryption Public-Key: (2048 bit)

7 Other Suggestions

Provide an option to change the passcode within the app.