An Empirical Study of SMS One-Time Password Authentication in Android Apps

Siqi Ma, Runhan Feng, Juanru Li, Yang Liu, Surya Nepal, Diethelm Ostry, Elisa Bertino, Robert H. Deng, Zhuo Ma, Sanjay Jha

Data61 CSIRO, Shanghai Jiao Tong University, Xidian University, Purdue University, Singapore Management University, University of New South Wales
Outline

• Authentication in Android

• One-time password

• SMS OTP Analyzer

• Evaluation

• Conclusion
Authentication in Mobile Phones

Single-factor Authentication

- Password-based Authentication – GLACIATE (ESORICS’19)

Weak passwords
Authentication in Mobile Phones

Two-factor Authentication

- Password-based Authentication + One-time password

Username + Password
Mobile Phone Number

Error-prone

Heavyweight
Inconvenient

SMS OTP Authentication

Token OTP Authentication

Username + Password
OTP

SMS OTP

OTP
Outline

• Authentication in Android

• One-time password

• SMS OTP Analyzer

• Evaluation

• Conclusion
OTP Authentication

HMAC-based OTP (HOTP)

- An incrementing counter value (C) and a secret key (K):
  \[ \text{HOTP}(K, C) = \text{Truncate}(\text{HMAC}(K, C)) \]

- Requirements:
  - Maximum number of possible attempts per login session.
  - An additional delay for each failed attempt.
  - Length should be at least Six digits

Timestamp-based OTP (TOTP)

- A time step (C_T) and a secret key (K):
  \[ \text{TOTP} = \text{Truncated} \left( \text{HMAC}(K, C_T) \right) \]

- Requirements:
  - Set the time step for network delay to 30s.
Security Requirements for OTP

RFC Requirements

- True randomness OTP or strong cryptographic PRNG
- Secure network channel (SSL/TLS)

- Brute force attacks
- Replay attacks
## OTP Rules

<table>
<thead>
<tr>
<th>Security Rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1: OTP Randomness</td>
<td>Use a random value as an OTP for authentication.</td>
</tr>
<tr>
<td>Rule 2: OTP Length</td>
<td>Generate an OTP value with at least six digit.</td>
</tr>
<tr>
<td>Rule 3: Retry Attempts</td>
<td>Set a limit on the number of validation attempts.</td>
</tr>
<tr>
<td>Rule 4: OTP Consumption</td>
<td>Only allow each OTP value to be consumed once.</td>
</tr>
<tr>
<td>Rule 5: OTP Expiration</td>
<td>Reject expired OTP values generated by the TOTP algorithm.</td>
</tr>
<tr>
<td>Rule 6: OTP Renewal Interval</td>
<td>OTP values generated by the TOTP algorithm should be valid for at most 30s.</td>
</tr>
</tbody>
</table>

RFC 4226 – HOTP, RFC 2289 – OTP, RFC 6238 – TOTP
## Rule Violations – Single

<table>
<thead>
<tr>
<th>OTP Rules</th>
<th>Violations/Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1: OTP Randomness</td>
<td>Replay attacks</td>
</tr>
<tr>
<td>Rule 2: OTP Length</td>
<td>Brute-force attacks</td>
</tr>
<tr>
<td>Rule 3: Retry Attempts</td>
<td>Brute-force attacks</td>
</tr>
<tr>
<td>Rule 4: OTP Consumption</td>
<td>Replay attacks</td>
</tr>
<tr>
<td>Rule 5: OTP Expiration</td>
<td>Unlimited time to discover the OTP</td>
</tr>
<tr>
<td>Rule 6: OTP Renewal Interval</td>
<td>A long time window to crack the OTP</td>
</tr>
</tbody>
</table>
## Rule Violation – Multiple

<table>
<thead>
<tr>
<th>Rule Combination</th>
<th>Violation/Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 + any other rules</td>
<td>Replay attacks</td>
</tr>
<tr>
<td>R2 + R3</td>
<td>Brute-force attacks</td>
</tr>
<tr>
<td>R4 + R5</td>
<td>Replay attacks</td>
</tr>
<tr>
<td>R2 + R3 + R6</td>
<td>Brute-force attacks</td>
</tr>
</tbody>
</table>
Outline

• Authentication in Android

• One-time password

• SMS OTP Analyzer

• Evaluation

• Conclusion
Design Challenge – SMS OTP Analyzer

Blackbox Analysis – execute apps to trigger the OTP validation functionalities.

Without source code
Design Challenge – SMS OTP Analyzer

Trigger OTP Validation System

Semantic Analysis – use Login Activity declarations and function information.
Design Challenge – SMS OTP Analyzer

Perform login

Code Analysis – decompile the apk and collect widget information.
Design Challenge – SMS OTP Analyzer

Text Analysis – Examine altered fields in each message

Analyze SMS message
SMS OTP Analyzer – AUTH-EYE

Login Code Detector:
✓ App Decompilation
✓ Login Activity Locating

Auth Message Analyzer:
✓ OTP Login Execution
✓ Evaluating Rule Violations
SMS OTP Analyzer – AUTH-EYE

Login Code Detector:
✓ App Decompilation
✓ Login Activity Locating

Auth Message Analyzer:
✓ OTP Login Execution
✓ Evaluating Rule Violations
AUTH-EYE: Login Code Detector

- App Decompilation: JEB Android Decompiler
- Login Activity Locating:
  - Customized package selection

Identity of the developer

Functions related to an Activity
AUTH-EYE: Login Code Detector

- App Decompilation: JEB Android Decompiler

- Login Activity Locating:
  - Customized package selection
  - Login Function Identification

Diagram:
- Code corpus
  - Reference set
  - Name Comparison

NLP
AUTH-EYE: Login Code Detector

- App Decompilation: JEB Android Decompiler

- Login Activity Locating:
  - Customized package selection
  - Login Function Identification
  - SMS OTP Identification

Diagram:

1. Layout XML file
2. SMS widget
3. Widget information
   - e.g., “smscode”, “mobilephone”
Design

SMS OTP Analyzer – AUTH-EYE

Login Code Detector:
✓ App Decompilation
✓ Login Activity Locating

Auth Message Analyzer:
✓ OTP Login Execution
✓ Evaluating Rule Violations
AUTH-EYE: Auth Message Analyzer

• OTP Login Execution:
  • Monkey tool – trigger SMS OTP login Activities.
  • Response Message Analysis

```java
widget location
```
```
call dispatchString()
```
```
response message mining
```
AUTH-EYE: Auth Message Analyzer

- OTP Login Execution
- Evaluating Rule Violations
  - R1: OTP Randomness

30 OTP requests

Consume each OTP before sending a new request

Send login requests without consuming OTPs
AUTH-EYE: Auth Message Analyzer

- OTP Login Execution

- Evaluating Rule Violations
  - R1: OTP Randomness
  - R2: OTP Length

Check the length of each OTP
AUTH-EYE: Auth Message Analyzer

• OTP Login Execution:

• Evaluating Rule Violations
  • R1: OTP Randomness
  • R2: OTP Length
  • R3: Retry Attempts

Request a valid OTP

Generate a fake OTP

Submit the incorrect value for n times

“Too many errors”

“x times”

Unlimited attempts
AUTH-EYE: Auth Message Analyzer

• OTP Login Execution:

• Evaluating Rule Violations
  • R1: OTP Randomness
  • R2: OTP Length
  • R3: Retry Attempts
  • R4: OTP Consumption

Request a valid OTP

Resubmit the same OTP
AUTH-EYE: Auth Message Analyzer

• OTP Login Execution:

• Evaluating Rule Violations
  • R1: OTP Randomness
  • R2: OTP Length
  • R3: Retry Attempts
  • R4: OTP Consumption
  • R5: OTP Expiration

Request a valid OTP

“expire” keyword search

Resubmit the same OTP until it expires
AUTH-EYE: Auth Message Analyzer

- OTP Login Execution:

- Evaluating Rule Violations
  - R1: OTP Randomness
  - R2: OTP Length
  - R3: Retry Attempts
  - R4: OTP Consumption
  - R5: OTP Expiration
  - R6 OTP Renewal Interval

Request a valid OTP

Submit the OTP after a time interval

[0, 30s] [30s, 60s] [60s, 5min] ……
Outline

• Authentication in Android

• One-time password

• SMS OTP Analyzer

• Evaluation

• Conclusion
Evaluation

• Dataset
  • From: GooglePlay Store and Tencent App Store
  • Total: 3,303 apps
  • Successfully analyzed 1,364 apps (648 failed to be decompiled, 1,298 crashed during SMS OTP analysis).
Results – OTP Login Activity Identify

- AUTH-EYE identified 1,069 (out of 1,364) apps with login activities, we manually inspected the apps and found 934 implemented login activities.

- 544 apps used OTP authentication

- 354 (out of 544) apps use two-factor authentication

<table>
<thead>
<tr>
<th>Login Activity Names</th>
<th># of apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>105</td>
</tr>
<tr>
<td>LoginSuccess</td>
<td>53</td>
</tr>
<tr>
<td>doLogin</td>
<td>37</td>
</tr>
<tr>
<td>smsLogin</td>
<td>18</td>
</tr>
<tr>
<td>onLogin</td>
<td>16</td>
</tr>
<tr>
<td>requestLogin</td>
<td>14</td>
</tr>
<tr>
<td>startLogin</td>
<td>14</td>
</tr>
</tbody>
</table>
### Results – OTP Rules Violations

<table>
<thead>
<tr>
<th>Rules</th>
<th># of apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>R6: OTP Renewal Interval</td>
<td>536</td>
</tr>
<tr>
<td>R3: Retry Attempts</td>
<td>324</td>
</tr>
<tr>
<td>R2: OTP Length</td>
<td>209</td>
</tr>
<tr>
<td>R4: OTP Consumption</td>
<td>106</td>
</tr>
<tr>
<td>R1: OTP Randomness</td>
<td>71</td>
</tr>
<tr>
<td>R5: OTP Expiration</td>
<td>41</td>
</tr>
</tbody>
</table>
Results – R6 : OTP Renewal Interval

• Only 8 apps follow this requirement.
• 165 apps do not renew their OTP values.
Results – R3: Retry Attempts

• Only 220 (40.44%) apps have OTP validation complying with the rule.

• AUTH-EYE was set to send a fake OTP at most 20 times. It identified that 126 apps still work after 20 times of retry.

• 97 apps have the delay protection for OTP validation.
Results – R2: OTP Length

• 209 apps use OTP values with the length < 6

• Although the OTP length could be set at 10 digits, all validation systems generate OTPs with at most 6 digits.
Results – R4: OTP Consumption

• Apps violated this rule are only from 8 categories: Shopping, Video Player & Editor, Books & Reference, Music & Audio, Travel & Local, Entertainment & Productivity.

• 37.7% and 18.9% vulnerable apps are from Books & Reference and Video Players & Editor, respectively.
Results – R1: OTP Randomness

• Two types of errors are identified: repeated values and static values.

• Repeated values: 56 apps generate repeated OTP values
  • 21 apps generate a sequence of unique values and then repeat the same sequence.
  • 35 apps repeat the same OTP values for n times (n = 2 or 3).

• Static Values: 15 apps use static OTP values.
Results – R5: OTP Expiration

• 33 apps reject the OTP value if it is expired.

• 40 apps accept expired OTP values.

• 471 apps do not have any expiration set for OTP values.
## Results – Multiple Rules Violation

<table>
<thead>
<tr>
<th># of apps</th>
<th>Multiple-rules violated</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>R2 (OTP Length) &amp; R4 (OTP Consumption)</td>
</tr>
<tr>
<td>13</td>
<td>R1 (OTP Randomness) &amp; (R2 or R3 (Retry Attempts))</td>
</tr>
<tr>
<td>9</td>
<td>R4 (OTP Consumption) &amp; R5 (OTP Expiration)</td>
</tr>
<tr>
<td>2</td>
<td>R2 &amp; R3</td>
</tr>
</tbody>
</table>
Outline

• Authentication in Android

• One-time password

• SMS OTP Analyzer

• Evaluation

• Conclusion
Conclusion

• We listed 6 OTP rules based on RFC documents.

• We designed AUTH-EYE to check for violations of OTP rules.

• An empirical study is conducted, and most Android apps are found with incorrect OTP implementations.

• The validation systems of apps in security-critical domains, such as Finance, Shopping, and Social are not secure.
Thank You

Q & A