[redacted]

Voatz
Mobile Applications
Final Report

Document Name: Final Report
Date: July 31, 2018

CONFIDENTIAL

[redacted]

Table of Contents

[redacted]

[redacted]

[redacted]

Contact Information

[redacted]

Executive Summary

[redacted] performed a source code review and thorough security audit of the Voatz applications on behalf of Voatz. This report summarizes the testing that was performed and the issues that were uncovered.

Voatz is a mobile eVoting system which allows users to vote in upcoming elections and polls from an end-user's mobile device. A user can provide their phone number and email to create an account. From there, by providing valid photo-id and fingerprint/retina, a user can be registered for voting in their local precincts.

Rooted Android phones and Jailbroken iOS devices were used for dynamic testing and OS level interaction that would be impossible on an unrooted phone. Burp Suite was used to find out how applications interact with the back-end servers. Multiple Android and iOS decompilers and static analyzers were used to fully explore the functionality of the mobile applications. These tools, listed in the toold section below, helped provide a close representation of the source code and allowed Security Innovation to verify how the application would act in theoretical situations that are difficult to reproduce.

Overall, the Voatz mobile applications functioned as intended with only a few lower-priority issues discovered. The applications have been designed and built with security in mind, leading to a relatively strong security posture.

The results of our testing are as follows:

- A total of 4 lower-priority security issues were identified:
    - Problem Report 1:- Password Policy
    - Problem Report 2:- Insufficient Boundaries for Registration Validation
    - Problem Report 3:- Brute Force Protection
    - Problem Report 4:- Insecure Cookie Settings

- From a STRIDE perspective, issues were found in the Spoofing, Information Disclosure, and Elevation of Privilege categories.

Introduction

[redacted] created a test plan, and performed security testing and a thorough source
ile Applications on behalf of Voatz.

Exploratory tests were performed to gain an understanding of the system that cannot be obtained through public knowledge or specification documents. Later, a test plan was developed to guide the attack and test execution process, ensuring every avenue of attack was thoroughly covered.

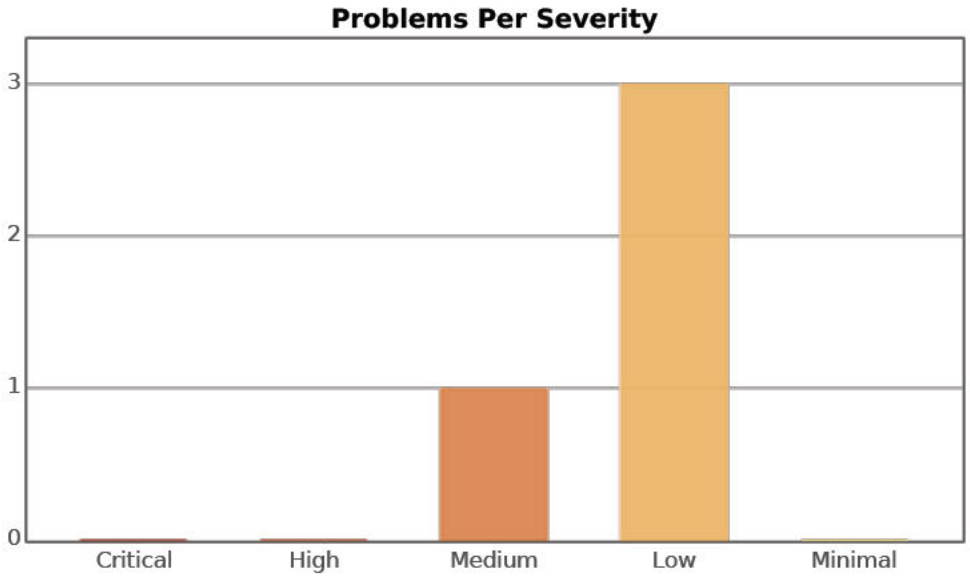
The attacks in the test plan and the subsequent testing were focused on the following areas:

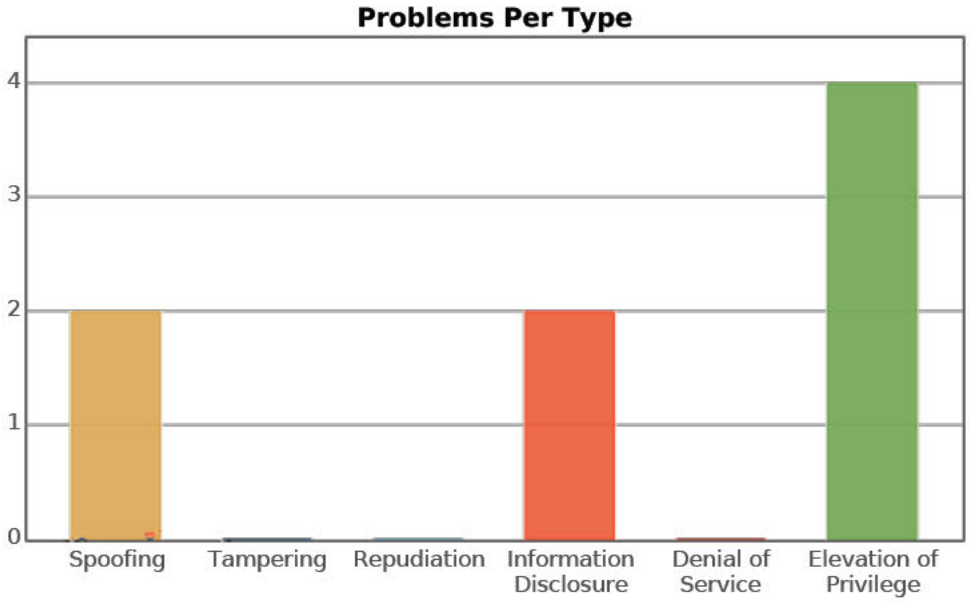
- Authentication mechanisms
- Caching of sensitive information
- Communication between the application and the back-end database
- Flawed business logic
- Validation of user-input
- Secure usage of authentication tokens
- Secure session management
- Cryptographic attacks
- Attack on binary protections

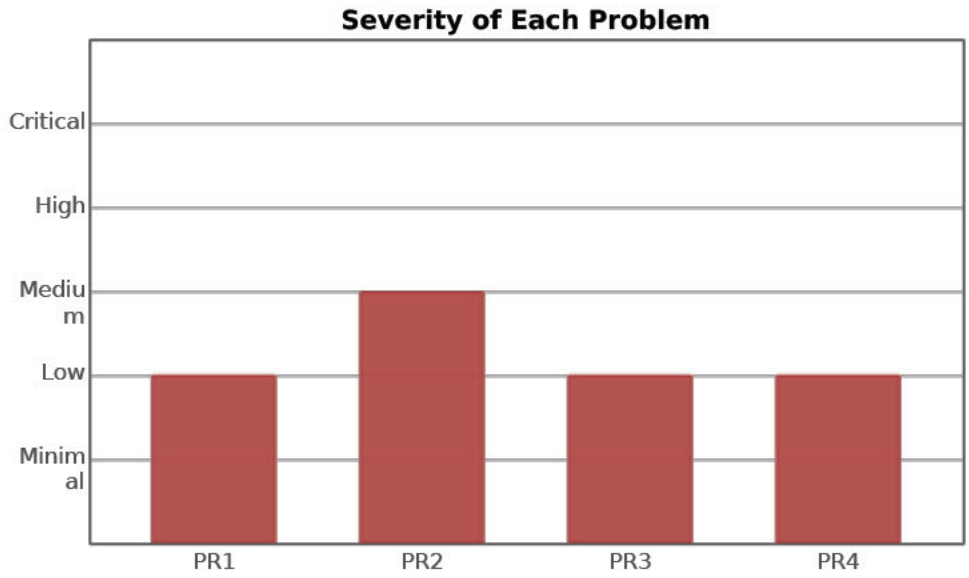
This report provides a summary of the observations made while reviewing and testing Mobile Applications. The "Threat Analysis" section goes into more depth on the initial analysis work performed. The "Observations" section describes ad-hoc observations and conclusions about the security of the controls based on the testing performed. The "Problem Report Summary" section summarizes the issues discovered during the engagement. The "Executed Test Cases" section describes all testing that was performed on the applications together with the observed results. The "Future Testing" section provides our recommendations on additional future testing on this system. The "Problem Reports" section contains the full text of each bug report.

Problem Report Summary

A total of 4 problems were identified. This section describes, at a high level, each of the problems discovered. See the Problem Summaries section for a table of each problem discovered, its severity, description and consequences. The following charts display the number of problems for each level of severity, the number of problems for each STRIDE type (note that a problem may have more than one type), and each problem's overall severity.







Problem Summaries

The problem report summaries are sorted by problem report ID. The format of the problem report table is as follows:

- The problem report ID
- The component in which the issue was discovered
- The severity of the issue
- A short description of the issue
- The consequences of the issue

PR #	Component	Severity	Description	Consequence
1	Android	Low	The password policy for the Mobile Applications allows the use of a PIN. This increases the risk of guessing or brute force attacks.	For users who do not have FaceId or Fingerprint ID enabled, when an attacker guesses a user's PIN, they will have access to the user's account
2	Android, iOS	Medium	The Voatz application does not rely on out-of-bounds validation for re-registering an account on their application and setting a new PIN.	Using this, an attacker can bypass the authentication step and log into a user's account once they have access to their device.

PR #	Component	Severity	Description	Consequence
3	iOS and Android Application	Low	Mobile Applications does not implement measures to limit authentication requests after multiple failed attempts.	For users who have disabled or do not have access to FaceID or Fingerprint ID, this makes the application more susceptible to brute force attacks.
4	iOS application	Low	The application does not set all cookie attributes in a secure manner.	An attacker can easily steal cookie information if application communicates over HTTP and may be able to reuse the cookie to access sensitive content in other applications. **NOTE:** This issue has been confirmed as remediated before the conclusion of this testing.

Executed Test Cases

The following table shows the breakdown of executed test cases, including any problem reports relevant to that item, and gives a brief summary of the methodology used to check that item and any other observations.

Column descriptions are as follows:

- ID - An identifier for quick test case reference
- Title - A title describing the test case
- Description - A short description of the test case and why it was performed
- Outcome - Either 'Pass' or a reference to the Problem Report Number

ID	Title	Description	Outcome
1	Administrative Functions	Confirm that the application does not provide administrative functions which can be accessed by bypassing the mobile UI.	PASS
2	Android Backup Vulnerability	Ensure that backups are not enabled for the application. If backup is enabled, make sure it does not reveal sensitive information and cannot be exploited to perform malicious operations.	PASS
3	Application Isolation	Confirm application isolation (i.e. other applications cannot access the application assets).	PASS
4	Authenticated Transmissions	Confirm that all transmissions (TCP/IP, port communications, etc.) to servers are authenticated (e.g. ensure a mobile application cannot default to an unauthenticated state).	PASS
5	Authentication Vulnerable to Brute Forcing	Confirm that there is a mechanism in place such as a CAPTCHA or login throttle to hinder brute force attacks.	PR 3
6	Automatic Login Due to Login Request Timeout	Verify that the application does not automatically authenticate when a response from the server is not received by the device within a stipulated time.	PASS
7	Binary Protection Enabled	Ensure that the application binary is protected with recommended controls (e.g. ASLR, ARC, and PIE stack protection).	PASS
8	Business Logic Attacks	Confirm that the application's business logic or functionality cannot be used in unintended ways for malicious purposes.	PR 2
9	Bypass iOS Security Mechanisms	Confirm that the application's functionality does not decrease the security provided by the device itself (e.g. if the application can be called directly via dynamic hooking, ensure that the iOS device passcode check is not bypassed).	PASS
10	Certificate Validation	Confirm that the application properly validates server certificates (e.g. validate certificate chains, domains, date ranges) and does not allow bypassing certificate checks.	PASS

ID	Title	Description	Outcome
11	Client-Side Validation	Confirm that all client-side validation is performed as strictly or more strictly on the server.	PASS
12	Code Injection	Confirm that malicious user input cannot be injected and executed.	PASS
13	Command Injection	Confirm that user input is not subsequently included within a system command which is provided to the OS.	PASS
14	Cookie Security	Confirm that all cookies handling sensitive information (e.g. session tokens) include appropriate attributes - including flags/paths/expirations.	PR 4
15	Copy and Paste	Confirm that sensitive fields do not allow copy and paste functionality.	PASS
16	Cryptography Strength	Verify that all cryptographic methods and libraries used by the application use appropriate defaults and follow industry best standards.	PASS
17	Data Caching	Confirm that no sensitive data is cached/stored onto the mobile device (e.g. in caches, cookies, preferences, webkit).	PASS
18	Data in Code	Confirm that no sensitive information (including embedded encryption keys/application codes, etc.) is contained in client code and accessible via decompiling (e.g. using otool, cycript, hopper, IDAPro, etc.).	PASS
19	Data in SMS/MMS	Confirm that no confidential or sensitive information is passed via SMS/MMS.	PR 2
20	Direct Request	Confirm that authorization checks are performed on all requests and that resources are not able to be requested directly by an identifier or URL.	PASS
21	Displaying User Data	Confirm that sensitive customer data (e.g. SSNs, authentication tokens, etc.) is not unnecessarily displayed on-screen.	PASS
22	Dynamic Hooking	Verify that you are not able to perform any attacks by means of dynamic hooking tools in iOS.	PASS
23	Flawed Broadcast Receivers	Ensure that broadcast receivers that listen for specific intents have permissions set so that they accept intents only from specific applications.	PASS
24	GPS	Confirm that the use of any GPS features has been approved as a design decision and that any associated location tracking is both responsible and well-understood.	PASS
25	Horizontal Authorization Bypass	Confirm that no flaws exist in the enforcement of business logic that can be used to bypass authentication or authorization checks and perform actions reserved for other users at the same privilege level.	PASS

[redacted]

ID	Title	Description	Outcome
26	Inactivity Logout	Confirm that the application invalidates session tokens after an appropriate amount of inactivity and that manual logout requests result in the session being invalidated server-side.	**PASS**
27	Insecure Content Provider Access	Ensure that content providers are not public and not callable directly by any other malicious APK. Also ensure that the provider is secure against SQL injection and path traversal.	**PASS**
28	Insecure Memory Error Checking	Confirm that the application makes use of the malloc function securely.	**PASS**
29	Insecure Random Number Generator	Confirm that all random number generation is done using appropriate sources, defaults, and libraries (e.g. avoid use of arc4random).	**PASS**
30	Intent Sniffing	Identify if the application is using intents to send sensitive information to other applications. If it is, ensure that explicit intents are used or permissions are set correctly.	**PASS**
31	JSON Injection	Confirm that untrusted input is not interpolated into JSON in a way that changes the context of data.	**PASS**
32	JSON Tampering	Confirm that untrusted input is not capable of breaking JSON parsing in a way that results in denial of service, enumeration of resources, or other vulnerabilities.	**PASS**
33	Jailbreak Detection	Ensure that if sensitive data is stored or privileged access is available through the application, jailbreak detection is in place to safeguard against an attacker.	**PASS**
34	Keychain Permissions	Verify that permissions on the iOS keychain do not allow backing up and restoring sensitive data.	**PASS**
35	Least Privilege	Confirm that application permissions conform to the principle of least privilege (i.e. the mobile application should not have access to phone, email, SMS, Bluetooth, etc. functionality on the device if it is not utilized by the app).	**PASS**
36	Lockout Reset	Confirm that account lockout reset utilizes an out-of-band communication channel not linked to the mobile device (i.e. does not use SMS).	**PR 2**
37	Login Screen Bypass	Verify that the login screen cannot be bypassed.	**PASS**
38	Login Timing	Verify that application response times do not change, regardless of whether the username/password exists in the data store.	**PASS**
39	Manual Session Invalidation	Confirm that manually logged out session tokens cannot be reused.	**PASS**
40	Multitasking	Confirm that the mobile application's multitasking functionality does not circumvent authentication, session management, data exposure, etc. controls.	**PASS**

ID	Title	Description	Outcome
41	Password Storage	Confirm that passwords are stored securely (either remotely on the server or locally on the client) and that no functionality indicates the insecure storage of passwords such as overly restrictive password character or length limitations or the ability to recover existing passwords.	**PASS**
42	Password Strength Policy	Confirm that there is a configurable method to control password strength.	**PR1**
43	Password-Protected Data	Confirm that local passwords are used to encrypt locally-stored data (i.e. the password is not just used as a screen lock).	**PASS**
44	Path Traversal	Confirm that internal directories and their contents cannot be accessed by manipulating the URL or any parameters.	**PASS**
45	Plaintext Transmission of Credentials	Confirm that all login pages are protected by TLS.	**PASS**
46	Predictive Text and Autocorrection	Confirm that sensitive fields do not allow autocorrect and that predictive text/word suggestions are not enabled.	**PASS**
47	SQL Injection	Confirm that queries sent with SQL meta-characters do not cause recognizable errors via error messages / code snippets / reproducible timing differences.	**PASS**
48	Secure Authorization Mechanism	Confirm that the application does not rely on a weak .ipa file or an OS-level authorization technique.	**PASS**
49	Sensitive Data Sent to a URL	Confirm that sensitive data is not passed via URL query string parameters (i.e. sensitive data is sent via POST requests over TLS).	**PASS**
50	TLS Configuration	Verify that the TLS configuration does not allow for insecure protocols, cipher suites, or features.	**PASS**
51	TLS Pinning	Ensure that the application relies on TLS pinning.	**PASS**
52	Transport Security of All Traffic	Confirm that all traffic, regardless of the sensitivity of the data, is protected by TLS.	**PASS**
53	Verbose Error Messages	Confirm that no invalid input triggers errors with information useful for attacking the system, such as stack traces or path information.	**PASS**
54	Version Information Disclosure	Confirm that version information is not disclosed through headers or error pages.	**PASS**
55	Vulnerable Activity Components	Ensure that the activities in the application are not public and not directly callable by any other malicious APK. Verify that activities cannot be called when the screen is locked by the user.	**PASS**
56	Vulnerable Services	Verify that services that perform background tasks cannot be activated by other malicious applications.	**PASS**

ID	Title	Description	Outcome
57	Forgot Password Implementation	Confirm that the Forgot Password" feature is implemented securely. Ensure that the application is safe against known forgot password related attacks."	**PR 2**
58	Reset Password Implementation	Confirm that the Reset Password feature is implemented securely. Ensure that the application is safe against known reset password related attacks.	**PR 2**

Tools

While testing the Voatz Mobile Applications, the following tools were employed.

Tool	Description	Link
Burp Suite Professional	Interactive HTTP/S proxy server that can intercept, inspect and modify data between the browser and target web server	https://portswigger.net/burp/
Android Studio	The standard IDE for Android software development kits	https://developer.android.com/studio
QARK	Open-source Android application analyzer	https://github.com/linkedin/qark
Wireshark	Gold standard of network sniffers and analysis	https://www.wireshark.org/
cURL	Command line tool for transferring files with URL syntax, supporting FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS and FILE	https://curl.haxx.se/
Python	High level, general purpose, scripting language	https://www.python.org
OpenSSH	A utility to provide users the ability to connect remotely to the iOS FileSystem	https://www.openssh.com/
Otool	Binary Analysis tool available in Xcode command line tools	https://developer.apple.com/downloads/index.action
Keychain Dumper	Dumps all the information stored in the keychain on iOS devices	https://github.com/ptoomey3/Keychain-Dumper

[redacted]

Problem Reports

Below are the complete Problem Reports for all discovered issues.

Problem Report 1 - Password Policy

The password policy for the Mobile Applications allows using a PIN. This increases the risk of PIN guessing or brute force attacks. For users who do not have FaceId or Fingerprint ID enabled, when an attacker guesses a user's PIN, they will have access to the user's account.

Component	Android
STRIDE	Elevation of Privilege
CWE	CWE-521: Weak Password Requirements
CAPEC	CAPEC-16: Dictionary-based Password Attack CAPEC-49: Password Brute Forcing
CVSS Score	7.5 (AV:N/AC:L/Au:N/C:P/I:P/A:P)
OWASP Reference	OWASP Top 10 2010-A6: Security Misconfiguration

Overall Severity	**Low**
Vulnerability Type	**Exploitable**
Impact	**Medium**
Confidentiality	An attacker could access the data of a user with a weak password/PIN.
Integrity	An attacker could modify the data of a user with a weak password/PIN.
Availability	An attacker could lock an individual user out of their account.
Exposure	An attacker can brute force or guess the PIN of a legitimate user after gaining access to their device.
Affected Users	Any individual user in the system could be affected.
Likelihood	**Low**
Skill Required	An attacker would need knowledge of common passwords/PINs or know how to script a brute force attack.
Conditions and Complexity	An attacker would need access to the user's physical device and the user would have to either opt out or not have access to FaceID or Fingerprint ID.
Discoverability	This is easy to discover.
Reproducibility	This is reproducible.

Background Information

The password policies of applications aim to stop the use of insecure passwords and to deter dictionary and brute force attacks.

Typical attacks against passwords include dictionary and brute force attacks. Dictionary attacks aim to gain access to accounts by using lists of popular passwords, words, phrases, and any personal information the attacker may have on the owner of the user account. Brute force attacks attempt to obtain the user's password by trying every possible combination of characters until the correct one is found.

Systems that use PINs as a substitute for passwords are more vulnerable to attack, as they have an even more limited subset of numbers/characters that can be used to authenticate into the application. This increases the potential of an attacker being able to bypass the authentication checks through brute force techniques.

Problem Details

The Voatz applications currently use a 6-digit PIN to authenticate users on the application. This makes it easier for an attacker to guess or brute force their way into the application and modify a user's account or register votes on their behalf.

Affected Areas

This was found in both the Android and the iOS version of the Voatz application.

Remediation

Use a password instead of a PIN. This will significantly increase the potential complexity capable for a authentication request, while minimally impacting the end-user.

Enforce a password complexity policy. At a minimum, the policy should satisfy the following requirements:

- Passwords must contain at least 8 characters.
- Passwords must not contain the user's display name, username, or email address.
- Passwords must contain at least 1 letter from each character type: lowercase letters, uppercase letters, numbers, and special characters.
- Users should not be able to reuse passwords. Store every user's last 5 password hashes and prevent them from reusing previous passwords during a password reset.
- For administrative accounts, passwords should contain at least 14 characters and expire after a certain amount of time. When the password expires, force the user to create a new one.

[redacted]

Problem Report 2 - Insufficient Boundaries for Registration Validation

The Voatz application does not rely on out-of-bounds validation for reregistering an account on their application and setting a new PIN. Using this, an attacker can bypass the authentication step and log into a user's account once they have access to their device.

Component	Android, iOS
STRIDE	Elevation of Privilege
CVSS Score	3.6 (AV:L/AC:L/Au:N/C:P/I:P/A:N)
OWASP Reference	OWASP Top 10 2017-A5: Broken Access Control

Overall Severity	**Medium**
Vulnerability Type	**Exploitable**
Impact	**Medium**
Confidentiality	An attacker with access to a user's device can gain access to that user's account information using this vulnerability.
Integrity	An attacker with access to a user's device can modify that user's account information with this vulnerability.
Availability	There is no direct impact on availability.
Exposure	This vulnerability allows attackers to bypass the authentication process on any stolen device with Voatz installed.
Affected Users	This vulnerability affects all Voatz users.
Likelihood	**Medium**
Skill Required	This vulnerability requires familiarity with the application registration workflow.
Conditions and Complexity	This vulnerability would require an attacker to have stolen or otherwise gained access to a user's device.
Discoverability	This vulnerability is easy to discover.
Reproducibility	This vulnerability is reproducible.

Background Information

Many mobile applications require an additional validation step when registering an account. This validation is often in the form of an email or SMS message containing a validation code which is required to complete the registration process. This step is often repeated when a user registers the account on a new device, or goes through the process of changing or recovering the password. This validation step has insufficient boundaries when it is sent to the same device that the application is installed on. An attacker can use this process to change the password on a stolen device, using the validation sent to the stolen device to complete the authentication steps and gain access to the user's account on the application.

Problem Details

The Voatz mobile applications uses an SMS message as the only validation step for registering or reregistering an account. In addition, the only method to reset a password is by reregistering the account on the device.

An attacker who gains access to a user's device can bypass the authentication step of the

[redacted]

Voatz application by reregistering the account, using the validation code sent to the same device, and creating a new password/PIN. This will allow them to access the user's account, validate other information about the user, and potentially submit votes as the user.

Additionally, the system does not notify a user through any means when the password/PIN has been changed. An additional notification would alert a user to major account changes and can serve as additional warning in the case of an account compromise.

This list may not include all instances of this vulnerability in the application. It is suggested that, upon remediation, the development team review other areas of Voatz where similar techniques are used in order to find and fix related vulnerabilities.

Test Steps

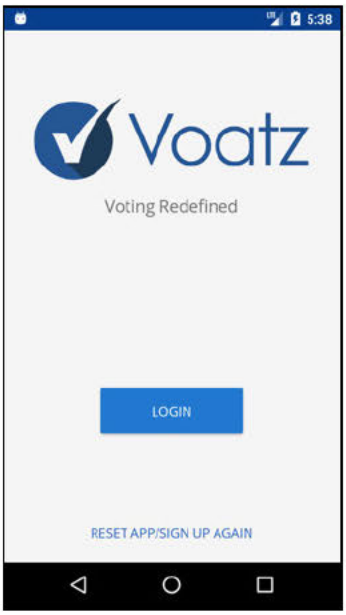
Test Configuration

The following is needed to reproduce this issue:

- A mobile device with Voatz installed
- A valid Voatz account pre-authenticated on the mobile device
- A phone number associated with the mobile device

Steps to Reproduce

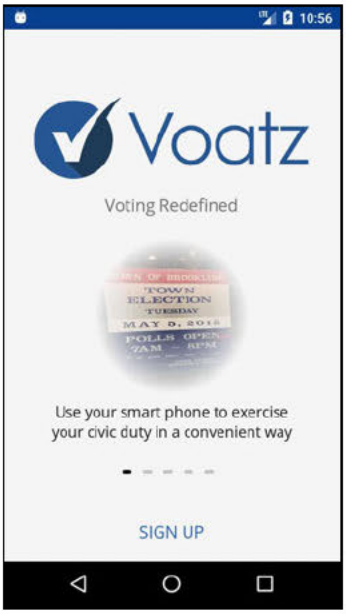
1. Open the Voatz application.



2. Click on *Reset App/Sign Up Again*.

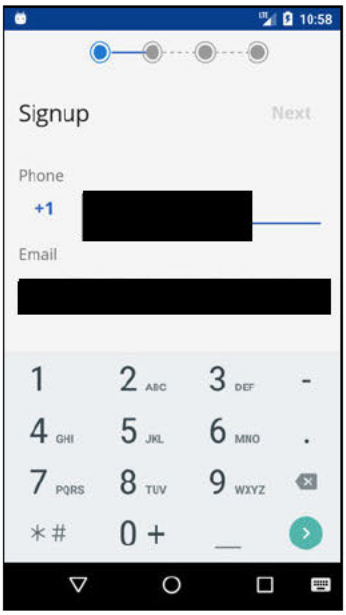
3. The Voatz application will now close, so click on the Voatz icon to reopen it.

[redacted]



4. Click *Sign Up*.

5. Register a new account by providing an email address and the phone number.

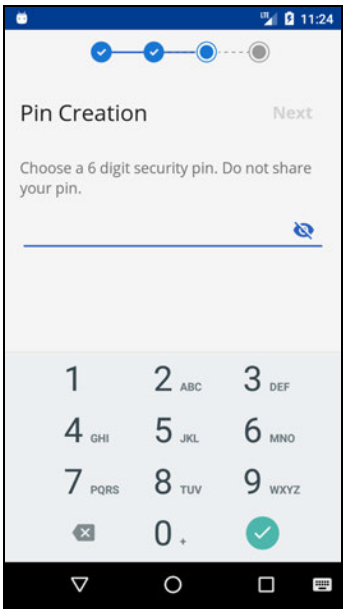


6. Click *Continue*.

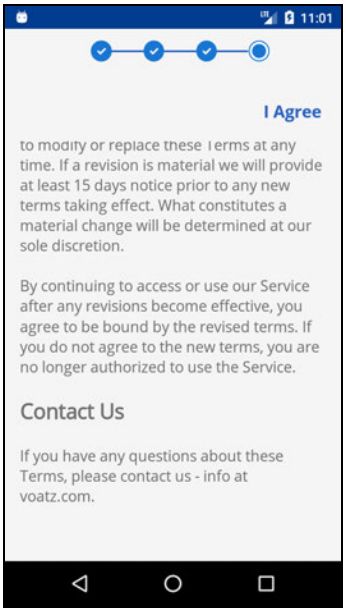
7. Wait for the SMS message to arrive with the validation code, then enter it into the Voatz application.

8. Set a new PIN of your choosing.

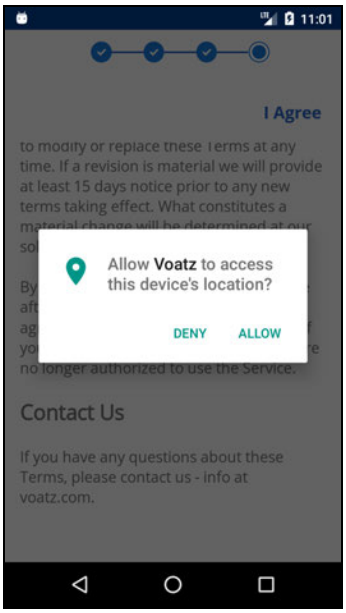
[redacted]



9. Scroll through the Terms and Conditions and click *I Agree*.



10. Click Allow.



11. Note that the application then launches and provides details about the end-user, along with access to the rest of the application's functionality.

12. Verify that no email with notice of a password change has been sent to any email address.

Remediation

Do not use in-band validation steps for mobile application validation. This includes SMS messages to the same device as the application installation. Instead require validation to an out-of-band method that is not directly related to the device it is being installed on, such as email.

Submit alerts when authentication credentials are changed. Whenever a device is registered with a new password/PIN, send an email to the end-user informing them of this change. This will give them additional warning in the case of an account compromise.

Problem Report 3 - Brute Force Protection

For users who have disabled or do not have access to FaceID or Fingerprint ID, this makes the application more susceptible to brute force attacks. An attacker who successfully guesses a user's PIN will gain access to the Voatz application.

Component	iOS and Android Application
STRIDE	Spoofing, Information Disclosure, Elevation of Privilege
CWE	CWE-307: Improper Restriction of Excessive Authentication Attempts
CAPEC	CAPEC-49: Password Brute Forcing
CVSS Score	4.0 (AV:N/AC:H/Au:N/C:P/I:P/A:N)
OWASP Reference	OWASP Top 10 2013-A2 : Broken Authentication and Session Management

Overall Severity	**Low**
Vulnerability Type	**Defense in Depth**
Impact	**Medium**
Confidentiality	An attacker can try and guess the PIN to gain access to a specific user's account.
Integrity	The compromised user's data could be modified by an attacker.
Availability	Availability is not affected.
Exposure	The application would be exposed to brute force attacks.
Affected Users	All users of the application are affected.
Likelihood	**Low**
Skill Required	No special skills needed but scripting skills or knowledge of brute forcing tools would increase the likelihood of success.
Conditions and Complexity	An attacker would need to get the user's phone and should have access to Voatz application on phone.
Discoverability	An attacker can discover this issue by trying multiple failed login attempts.
Reproducibility	The attack is reproducible.

Background Information

The Voatz applications do not mitigate against brute force and password guessing attacks on the login page. This allows a malicious attacker unlimited attempts at authenticating to a valid user's account without knowledge of their password. This increases the likelihood that an attacker can gain access to their account.

Problem Details

Affected Areas

The Login Page of the Voatz application was found to be vulnerable.

Test Steps

[redacted]

Test Configuration

The following is needed to reproduce this issue:

- Access to the Voatz application
- An account with FaceID and/or Fingerprint ID disabled

Steps to Reproduce

1. Open the Voatz application.
2. Attempt to authenticate while supplying the wrong PIN.
3. Repeatedly supply the wrong PIN multiple times in quick succession.
4. Observe that application does not throttle the login attempts and after providing valid PIN it allows login.

Remediation

After a sufficient number of unsuccessful login attempts have occurred, one of the following actions can be taken.

In order of preference:

- Progressively throttle additional authentication attempts for the targeted user account, such that each consecutive attempt takes more time than the previous attempt to complete. The introduced delay should be random, last only a few seconds and must reset after a certain period of no login attempts for the user account. This approach balances usability when credentials are forgotten with the need to prevent attackers from continually guessing a valid accounts' PIN.
- Require users to successfully identify the contents of a CAPTCHA after multiple failed login attempts. The prompt for the CAPTCHA will help to protect against automated attacks. One such implementation is reCAPTCHA and additional information on this can be found at: https://developers.google.com/recaptcha/docs/start

Additionally, if account access must be critically protected and there is a strong business need, the following strategies can be implemented. **Note**, however, if these strategies are opted, an attacker can iterate through accounts, intentionally attempting bad passwords in an attempt to lockout valid users. This will result in a Denial of Service for the targeted users.

- Lock the account from further authentication attempts for a fixed period of time.
- Lock the account from further authentication attempts and require manual intervention by an administrator to re-enable the account.

In addition to these, notify the users login activity via email whenever a suspicious login attempt or too many failed attempts are made in a short period of time.

[redacted]

Problem Report 4 - Insecure Cookie Settings

An attacker who intercepts plaintext client-server traffic successfully can steal sensitive information.

NOTE: This issue has been confirmed as remediated before the conclusion of this testing.

Component	iOS application
STRIDE	Spoofing, Information Disclosure, Elevation of Privilege
CWE	CWE-614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
CAPEC	CAPEC-102: Session Sidejacking
CVSS Score	6.6 (AV:N/AC:H/Au:N/C:C/I:P/A:P)
OWASP Reference	OWASP Top 10 2013-A2 : Broken Authentication and Session Management

Overall Severity	**Low**
Vulnerability Type	**Defense in Depth**
Impact	**Medium**
Confidentiality	If the cookies are stolen, an attacker can impersonate a user, and read their data.
Integrity	If the cookies are stolen, an attacker can impersonate a user, and modify their data.
Availability	If the cookies are stolen, an attacker can impersonate a user, and delete their data.
Exposure	This can allow an attacker to gain access to the application by stealing the session-related cookies through a MITM attack.
Affected Users	All users are affected.
Likelihood	**Low**
Skill Required	An attacker would need moderate skill to perform the passive Man-in-the-Middle attack needed to exploit this issue.
Conditions and Complexity	An attacker needs a privileged network position between the user and application server.
Discoverability	This issue is easy to discover.
Reproducibility	This issue is reproducible.

Background Information

Cookies that do not have the proper security flags are significantly easier to compromise than those configured correctly. If an attacker can acquire another user's cookie, they may be able to use information stored in the cookie to authenticate as that user, perform any actions on behalf of that user, and access the user's information within the application.

Cookies missing the secure attribute will be transmitted over the network in plaintext if the application allows communication without using TLS. This can allow an attacker sniffing network traffic to read the cookie information.

Problem Details

[redacted]

Test Steps

Test Configuration

In order to reproduce this test the following is required:

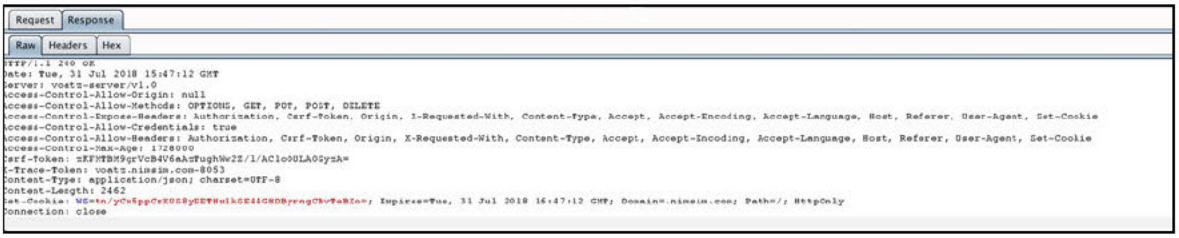
- An iOS device with Voatz installed
- An intercepting proxy (Burp Suite was used in this example)
    - Configure Burp Suite to work on the iOS device using the following the guide: https://support.portswigger.net/customer/portal/articles/1841108-configuring-an-ios-device-to-work-with-burp
- Login credentials for a user account

Steps to Reproduce

1. Follow the above guides to configure Burp Suite on a laptop and proxy traffic from a mobile device through to the laptop. Both devices will need to be on the same network.

2. Log into the Voatz App.

3. Open the response of the login request in BURP as shown below:



4. Observe that `ws` cookie is missing secure header.

Remediation

The secure attribute must be set for the cookies. This indicates to the browser that the cookie must only be sent via encrypted channels.

Ensure that all cookies are set to session-only. They should be destroyed once the session is invalidated.

Follow-up

Inconsistencies, errors, and reproducibility problems associated with this report should be directed through the customer contact person to the tester and preparer indicated at the beginning of this report.

[redacted]