

Rangzen Android Application Security Assessment

Denovo Group

December 18, 2015 - Version 1.0

Prepared for
Barath Raghavan

Prepared by
Luis Ramirez



Synopsis

During May 2015, Denovo Group engaged Matasano Security to conduct a security assessment of the Rangzen Android application. The application provides a distributed anonymous messaging system, allowing for communication when centralized network services have been disabled. The original assesment produced 9 findings of medium to low severity.

In December of 2015, Denovo Group engaged NCC to retest the original findings. This assesment was open ended and time boxed. Source code was provided, and only the original findings were in scope.

Scope

Matasano Security's original evaluation included the full Rangzen Android application. Testing was performed using NCC test devices running the Rangzen code, simulating a real Rangzen deployment. Components examined include all wireless communications, the anonymity guarantees, and features mentioned in the Rangzen paper. Common Android issues, such as secure file storage, network communication, and library issues were also tested. The cryptography was covered by a separate assessment.

Key Findings

The original assessment uncovered a set of common application flaws. Some of the more notable were:

- A vulnerability in peer detection allowing an attacker to efficiently prevent any user in a wide area from connecting to other real users, stopping all communications.
- Multiple components of the paper were either partially or not implemented, leaving Rangzen in a compromised state.
- Several issues related to secure handling of stored data allow for easier compromise of a given user

Limitations

There were no significant limitations in the Rangzen retest.

The original assesment had several limitations. Many components of the Rangzen paper were not implemented in the application at the time of the engagement. For example, message trust decay, known-peer prioritization, deleting messages, and unfriending users were partially or not implemented. Vulnerabilities that were a result of an unimplemented feature are called out in the report.

Target Metadata

Name	Rangzen
Type	Android Application
Platforms	Android
Environment	Production

Engagement Data

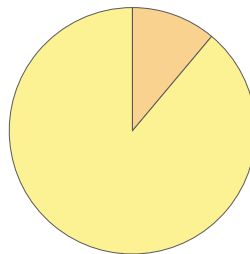
Type	Android Application security assessment
Method	Code-assisted
Dates	2015-12-14 to 2015-12-16
Consultants	1
Level of effort	3 person-days

Targets










Android Application	Rangzen Android Application
----------------------------	-----------------------------

Vulnerability Breakdown

Critical Risk issues	0
High Risk issues	0
Medium Risk issues	1
Low Risk issues	8
Informational issues	0
Total issues	9



Category Breakdown

Access Controls	2	 
Configuration	1	
Data Exposure	2	 
Data Validation	2	 
Denial of Service	2	 

Key

Critical		High		Medium		Low		Informational	
----------	-------------------------------------------------------------------------------------	------	-------------------------------------------------------------------------------------	--------	-------------------------------------------------------------------------------------	-----	-------------------------------------------------------------------------------------	---------------	---------------------------------------------------------------------------------------

Synopsis

Prior to the first day of testing, NCC received access to a current version of the Rangzen source code. Testing proceeded without incident for the duration of the assessment.

Test Plan

NCC's test plan followed the NCC Group Android application testing method, which is a superset of the OWASP top 10 mobile risks. Testing was limited to findings reported in the original assessment.

Testing Methods

NCC used the Android Studio development environment to compile the Rangzen source code into an APK. This APK was loaded onto NCC test devices to provide a production environment with which to test the Rangzen application. NCC used code base modification and source code inspection to verify findings.

Limitations

No significant limitations were encountered during the Rangzen application retest.

Table of Vulnerabilities

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and vulnerability categorization, see [Appendix A on page 16](#).

Title	Status	ID	Risk
No Limit on the Number of Peers Added from WiFi Direct Detection	Not Fixed	007	Medium
Compromised Nodes Cannot be Removed from the Network	Fixed	001	Low
Application Does Not Require Authorization Before Sending Messages	Not Fixed	002	Low
Maximum Number of Exchanged Messages Not Enforced	Not Fixed	003	Low
Rangzen Private Keys Stored Insecurely	False-Positive	004	Low
Application can be Backed Up	Fixed	005	Low
Attackers Can Send Arbitrarily Long Messages	Fixed	006	Low
Message Priorities May Reveal Friendship Connections Between Nodes	Fixed	008	Low
Attackers Can Artificially Boost Message Priority	Not Fixed	009	Low

Vulnerability No Limit on the Number of Peers Added from WiFi Direct Detection

Identifier NCC-RANG15-007

Risk **Medium** Impact: Medium, Exploitability: Medium

Category Denial of Service

Status Not Fixed

Impact An attacker can spoof Wifi Direct device names, causing the Rangzen app to try and pair with a huge list of fake peers, preventing it from communicating with the rest of the network.

Description The Wifi Direct peer discovery process adds all potential matches discovered to the list of peers. To pose as a potential match, an attacker has to have a device name of the form RANGZEN-11:22:33:44:55:66, where the sequence of numbers is the device's Bluetooth MAC address. There is no filtering or rate-limiting of these devices on the client-side. By flooding a client with many spurious peers, an attacker can make it extremely unlikely for a client to connect to a real peer.

Recommendation Prioritize existing known-good peers over new peers. Cap the number of discovered peers. When unable to connect to a peer, immediately search for another peer, rather than delaying the search process further. When saving peers, consider building a Bloom filter of known-working peers to prevent leaking which peers are known by a given device.

Retest Results There is still no maximum number of nodes that can be added via this method.

Vulnerability	Compromised Nodes Cannot be Removed from the Network
Identifier	NCC-RANG15-001
Risk	Low Impact: Low, Exploitability: Low
Category	Access Controls
Status	Fixed
Impact	A compromised node can continue to spread disinformation unchecked even if the compromise is detected. Lack of unfriending ability and lack of trust decay mean a compromised node may stay active even after compromise is detected
Description	A trusted node could be compromised at any time. One likely route of compromise would be via theft or capture of the trusted device. Once compromised, the attacker could use the device to continue to send messages. These messages would appear to other nodes to be completely legitimate, and indistinguishable from messages coming from other trusted parties. The root of this problem lies in the fact that trust does not ever decay and cannot be forcibly revoked, as there is no unfriending ability in the application. Even if the legitimate owner of the device detected the theft, they (and their network of trust) would have no recourse.
Recommendation	Allow removal of existing trusted friends, although this is complicated as friends do not have a mapping from name to ID. Consider making trust partially decay over time, requiring contact with other users to restore full trust.
Retest Results	Denovo Group added the ability to remove friend nodes to the Rangzen application.

Vulnerability	Application Does Not Require Authorization Before Sending Messages
Identifier	NCC-RANG15-002
Risk	Low Impact: Low, Exploitability: Low
Category	Access Controls
Status	Not Fixed
Impact	An attacker with (even brief) physical access to a trusted device can post disinformation that will be spread with that device's trust level.
Description	There is no authorization check before messages are posted on Rangzen. If an attacker is able to achieve physical access to a device, even for a brief period, they can easily post messages from that device. The messages sent from the device will go out with the trust level associated with the device. Due to the anonymous nature of Rangzen, not even the person whose device is compromised in this attack will know that the message was sent by an attacker.
Recommendation	One possible countermeasure here is to require users to enter a PIN to post a new message is posted. If multiple incorrect PIN entries are detected, the application can wipe the secret keys to prevent an attacker from obtaining them. Another option is to force users to have a passcode or lock pattern enabled. By reading the value of LOCK_PATTERN_ENABLED in <code>android.provider.Settings.Secure</code> , Rangzen can determine whether a user's phone is protected.
Retest Results	The application still does not check the value of LOCK_PATTERN_ENABLED.

Vulnerability	Maximum Number of Exchanged Messages Not Enforced
Identifier	NCC-RANG15-003
Risk	Low Impact: Low, Exploitability: Low
Target	java/org/denovogroup/rangzen/Exchange.java (line 81)
Category	Denial of Service
Status	Not Fixed
Impact	An attacker can monopolize the bandwidth of a legitimate node by sending a flood of messages above the intended exchange threshold.
Description	As designed, only 100 messages should be exchanged in each Rangzen transaction. However, the number of messages exchanged is only checked on the sender side. The overall amount of data exchanged is capped at 1MB per exchange, and appropriately checked on both ends of the transaction. Since the software is intended to be open source, it is plausible that an attacker can make a minor modification to the code in order to send an arbitrarily large number of messages. They can also then programmatically generate a corresponding number of messages in their local message store. After establishing this base, when the attacker comes into contact with an innocent node, the innocent node will be flooded with the spam messages. This may cause the innocent node to miss legitimate updates from other nodes in the area, and may also cause unexpected UI problems within the Rangzen application itself.
Recommendation	Check the number of messages exchanged on the receiver's end. Either close the connection after 100 messages have been exchanged, or drop additional messages instead of adding them to the message store.
Retest Results	The maximum number of messages exchanged is still enforced by the sender. This can be exploited by a modified malicious client.

Vulnerability	Rangzen Private Keys Stored Insecurely
Identifier	NCC-RANG15-004
Risk	Low Impact: Low, Exploitability: Low
Target	org/denovogroup/rangzen/FriendStore.java (line 225)
Category	Data Exposure
Status	False-Positive
Impact	An attacker with physical access to the device or access to a user's phone backup could steal the Rangzen private key and use it to masquerade as the phone's owner in future communications.
Description	The public / private keypair for Rangzen, which is used as the user's anonymous identity in the Rangzen network, is stored as plaintext in RangzenData.xml. This file is located in the shared_preferences directory for the application. This directory is accessible if the phone has been rooted, or if a backup is made of Rangzen. If an attacker obtains brief physical access to a rooted device, they can pull RangzenData.xml off of the phone using a few simple adb commands.
Reproduction Steps	1. Load Rangzen onto a rooted test device. 2. Connect to the device and browse to /data/-data/org.denovogroup.com/rangzen. 3. Open RangzenData.xml and note that it contains both the public and private key in plaintext, encoded as base64 strings.
Recommendation	Use the Android Keystore Provider to store keys. Private keys can be stored either in software or in hardware (on supported devices), and are only exposed to the application through an interface that allows signing and encryption, preventing accidental key leakage. Note that on devices with software key storage, keys can be recovered on a rooted device. See https://developer.android.com/training/articles/keystore.html for more information.
Retest Results	Denovo Group informed NCC that the public keys used by the Rangzen application are used entirely for generating randomized identification tokens. The private keys are not used for authentication or authorization.

Vulnerability	Application can be Backed Up
Identifier	NCC-RANG15-005
Risk	Low Impact: Low, Exploitability: Low
Target	ui/Rangzen/AndroidManifest.xml (line 22)
Category	Configuration
Status	Fixed
Impact	An attacker finds a backup of Rangzen's data on a user's computer. They use the contained keys and friends to deanonymize and impersonate the user.
Description	The application manifest currently specifies that Rangzen can be backed up by a user. Allowing a user to backup Rangzen data exposes them to further attacks against their computer in addition to their phone.
Recommendation	Set <code>android:allowBackup="false"</code> on the application element in the <code>AndroidManifest.xml</code> to disallow all application backups. See http://developer.android.com/guide/topics/manifest/application-element.html#allowbackup for more details.
Retest Results	Denovo Group disabled backups of the Rangzen application.

Vulnerability	Attackers Can Send Arbitrarily Long Messages
Identifier	NCC-RANG15-006
Risk	Low Impact: Low, Exploitability: Low
Target	ui/Rangzen/res/layout/makepost.xml (line 45) java/org/denovogroup/rangzen/MessageStore.java (line 206)
Category	Data Validation
Status	Fixed
Impact	An attacker can drown out messages from anyone with a trust level equal to or below theirs by crafting extremely long messages.
Description	<p>It is possible for an attacker to send a message of an arbitrary length by hardcoding a message in the app and rebuilding it. This is achievable because the only check on individual message length is built into the UI. The TextView field is limited to 140 characters, so an attacker cannot send an arbitrarily long message through the normal means; however, since the software is open source, it is plausible for the attacker to download the source, hardcode a disruptively long message or set of messages, and rebuild the app, thereby circumventing the check entirely. Due to the small form factor of the mobile devices Rangzen is intended for, the presence of overly long messages severely hampers the usability of the application. By also exploiting knowledge of the organization method (messages are sorted first by priority, then sorted alphabetically), an attacker can use overly long inputs to effectively drown out any messages at or below their own trust level. Additionally, we have observed that overly long messages cause UI problems long before the maximum message size is reached. So at present, this feature could also be used to render Rangzen totally unusable on a victim's device, by merit of causing the device to crash every time the UI is accessed.</p>
Reproduction Steps	<ul style="list-style-type: none"> Insert the following code into Opener.java around line 130 (after creation of the MessageStore). Note that this code could be also be added through lower-level means such as smali / baksmali. <pre> char[] somebytes; somebytes = new char[522244]; while(i < 522242) { somebytes[i] = '*'; somebytes[i+1] = 'o'; i = i + 1; } String somestring; somestring = new String(somebytes); messageStore.addMessage(somestring, 1L); </pre> <ul style="list-style-type: none"> Rebuild experimentalApp and reinstall it on the test device. Open the UI, wait for it to load, and observe that the overly long message appears. Note that it may crash if you attempt to interact with it before the message appears. Bring another device running Rangzen into proximity and observe that the overly long message is successfully received on that device as well. Notice that any messages with the

Recommendation

same (or lower) priority are extremely difficult to see.

Check the message length before storing it in the message store. Reject any messages over a certain length. Ideally, the maximum length stored will correspond with the maximum length accepted by the UI.

Retest Results

Denovo Group added a message length check in the `MessageStore.addMessage` method.

Vulnerability	Message Priorities May Reveal Friendship Connections Between Nodes
Identifier	NCC-RANG15-008
Risk	Low Impact: Low, Exploitability: Low
Category	Data Exposure
Status	Fixed
Impact	An attacker can identify nodes by mapping their friendship graphs using message priorities.
Description	An attacker may be able to capture the priorities going both directions and use those priorities to determine the network of friends connecting the two users. This may be used to reveal the identity of any given node. Priorities can be collected either via an attacker who is actively participating in Rangzen, or by an attacker who is eavesdropping but not using the Rangzen app. This vulnerability exists because neither the noise factor proposed in the Rangzen white paper nor per-message encryption have been implemented at present.
Recommendation	Add a noise factor to the message priorities to obscure friendships further as proposed in the Rangzen white paper. Also consider encrypting messages to prevent non-participating, eavesdropping nodes from collecting the priority data.
Retest Results	Noise factor has been implemented.

Vulnerability	Attackers Can Artificially Boost Message Priority
Identifier	NCC-RANG15-009
Risk	Low Impact: Low, Exploitability: Low
Category	Data Validation
Status	Not Fixed
Impact	An attacker can increase or decrease the priority of a message before sending it onward to legitimate nodes.
Description	<p>Due to the open source nature of the project, an attacker can manually insert priorities into messages programmatically. The insertion of attacker-selected priorities can result in boosting or drowning out important messages on the receiver's end. Currently, the application will check the priority assigned to a message on the receiver's end and assure that it is between -1.0 and 2.0. An attacker can choose any value in that range to apply to a message, however, whether or not the message originated with the attacker. The problem arises because the receiver has no way of knowing what the original priority was. The attacker-chosen priority will be used in calculating the new display priority instead. The boosting attack works as long as the receiver trusts the attacker even slightly. If the receiver and the attacker have at least one friend in common, the message's priority can be boosted as high as 199 priority (higher even than a self-post). The extent of the impact of the tampering scales with the size of the legitimate node's network and the number of friends the legitimate node has in common with the attacker. For a legitimate node that has only 1 friend, who is also a friend of the attacker, the impact of the tampering is large; the attacker can send messages with a priority up to 199. When the legitimate node has a larger network with few shared friends, the attack's impact is lessened; however, it may still be used to raise the attacker's messages relative to other legitimate messages. The downvoting attack (priority decrease) works only in a scenario where the legitimate receiver has never seen the message the attacker is sending before. This limitation is due to the fact that Rangzen stores the maximum priority seen for a given message. If the receiver has already seen the message with a higher priority, Rangzen will ignore the attacker's new, lower priority; however, if the receiver has not seen the message previously, the receiver will accept the lowered priority. This could lead to the legitimate node missing truly high priority messages. This has potential to be devastating when combined with other attacks such as the Wi-Fi direct peer suppression attack.</p>
Reproduction Steps	<ol style="list-style-type: none"> 1. Obtain two devices and have them each add a third device (a fake QR code is fine) as a friend. . Modify <code>getPriority's</code> return (<code>MessageStore.java</code>, line 251) to this: <code>return 1.99;</code> Rebuild the source and install the resulting app on one device. This device now represents an attacker. . Force a refresh by doing something like posting a new message. Observe that all messages on the attacker device will now have priority 199. Observe that the second device should now receive any messages posted by the attacker with priority 199 as well.
Recommendation	Without compromising anonymity, this problem does not have a clear solution.
Retest Results	As mentioned previously, there is no clear solution to this issue.

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing vulnerabilities. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a vulnerability poses to the target system or systems. It takes into account the impact of the vulnerability, the difficulty of exploitation, and any other relevant factors.

- Critical** Implies an immediate, easily accessible threat of total compromise.
- High** Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
- Medium** A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
- Low** Implies a relatively minor threat to the application.
- Informational** No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable vulnerability.

Impact

Impact reflects the effects that successful exploitation upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

- High** Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
- Medium** Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
- Low** Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a vulnerability. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

- High** Attackers can unilaterally exploit the vulnerability without special permissions or significant roadblocks.
- Medium** Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the vulnerability.
- Low** Exploitation requires implausible social engineering, a difficult race condition, guessing difficult to guess data, or is otherwise unlikely.

Category

NCC Group groups vulnerabilities based on the security area to which those vulnerabilities belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

- Access Controls** Related to authorization of users, and assessment of rights.
- Auditing and Logging** Related to auditing of actions, or logging of problems.
- Authentication** Related to the identification of users.
- Configuration** Related to security configurations of servers, devices, or software.
- Cryptography** Related to mathematical protections for data.
- Data Exposure** Related to unintended exposure of sensitive information.
- Data Validation** Related to improper reliance on the structure or values of data.
- Denial of Service** Related to causing system failure.
- Error Reporting** Related to the reporting of error conditions in a secure fashion.
- Patching** Related to keeping software up to date.
- Session Management** Related to the identification of authenticated users.
- Timing** Related to race conditions, locking, or order of operations.

The NCC Group team has the following primary members:

- Luis Ramirez – Security Consultant
luis.ramirez@nccgroup.trust, (239) 240-1964

The Denovo Group team has the following primary members:

- Barath Raghavan – Denovo Group
barath@denovogroup.org