Summary

The Citizen Lab is an academic research group based at the Munk School of Global Affairs & Public Policy at the University of Toronto in Toronto, Canada.

We analyzed Xiaomi's pre-installed keyboard apps as part of our ongoing work analyzing popular mobile and desktop apps for security and privacy issues. We found that they all include vulnerabilities that allow network eavesdroppers to decrypt network transmissions from the keyboards. This means that third-parties can obtain sensitive personal information, including what users have typed.

Platform	Keyboard name	Package Name	Version analyzed
MIUI 14.0.31	百度输入法小米版	com.baidu.input_mi	10.6.120.480
MIUI 14.0.31	搜狗输入法小米版	com.sohu.inputmethod.sogou.xiaomi	10.32.21.2022102 21903
MIUI 14.0.31	讯飞输入法小米版	com.iflytek.inputmethod.miui	8.1.8014

Table 1: The versions of the Xiaomi keyboard apps analyzed.

Findings

In this section we detail vulnerabilities in three different keyboard apps included with MIUI 14.0.31 in which users' keystrokes can be, if necessary, decrypted, and read by network eavesdroppers.

百度输入法小米版 (com.baidu.input_mi)

The Baidu-based keyboard app encrypts keystrokes using a modified version of AES. Normally, AES when used with a 128-bit key performs 10 rounds of encryption on each block. However, we found that the version of AES implemented by this app uses only 9 rounds but is otherwise equivalent to AES encryption with a 128-bit key.

The app encrypts keystrokes using the above 9-round AES algorithm in the following manner. First, a key is derived according to a fixed function (see Figure 1). Note that the function takes no input nor references any external state and thus generates the same static key: k_f = "\xff\x9e\xd5H\x07Z\x10\xe4\xef\x06\xc7.\xa7\xa2\xf26".

```
def derive_fixed_key():
    key = []
    x = 0
    for i in range(16):
        key.append((~i ^ ((i + 11) * (x >> (i & 3)))) & 0xff)
        x += 1937
    return bytes(key)
```

Figure 1: Python code equivalent to the code the app uses to derive its fixed key.

Key k_f is used to decrypt bytes 28 until 44 of the UDP payload using 9-round AES in electronic codebook (ECB) mode, resulting in k_m, another 128-bit key which encrypts the actual message. Key k_m is used to decrypt bytes 44 until the end of the payload using the same 9-round AES algorithm in ECB mode, resulting in the plaintext message. Among the bytes of the plaintext, in the first four, stored in little-endian byte order, is the length of the decrypted message. Immediately following this length is a snappy-compressed protobuf serialization. When deserialized, we found that this protobuf includes our typed keystrokes as well as the name of the application into which we were typing them (see Figure 2).

```
[...]
2 {
  1: "nihaonihaoqqwerty"
}
3 {
  1: 53
  2: 10
  3: 1080
  4: 2166
  5: 5
}
4 {
  1: "DC0F75E6809F0FAAB46EDE2F2D6302ED%7CVAPBN4NOH"
  2: "p-a1-3-66|2211133C|720"
  3: "10.6.120.480"
  4: "com.miui.notes"
  5: "1000228c"
  6: "\346\242\205\345\267\236"
}
[...]
```

Figure 2: Excerpt of decrypted information, including what we had typed (“nihaonihaoqqwerty”) and the application into which it was typed (“com.miui.notes”).

A vulnerability exists in this protocol that allows a network eavesdropper to decrypt the contents of these messages. Since AES is a symmetric encryption algorithm, the same key used to encrypt a message can also be used to decrypt it. Since k_f is fixed, any network eavesdropper with knowledge of k_f can decrypt k_m and thus the plaintext contents of each message encrypted

in the manner described above. As we found that users' keystrokes and the names of the applications they were using were sent in these messages, a third-party who is eavesdropping on a user's network traffic can observe what that user is typing and into which application they are typing it by exploiting this vulnerability.

搜狗输入法小米版 (com.sohu.inputmethod.sogou.xiaomi)

The Sogou-based keyboard app is vulnerable to a vulnerability which we have already publicly disclosed in Sogou Input Method (搜狗输入法) in which a network eavesdropper can decrypt and recover users' transmitted keystrokes. Please see the corresponding details in this report for full details. Tencent responded by securing Sogou Input Method transmissions using TLS, but we found that 搜狗输入法小米版 (com.sohu.inputmethod.sogou.xiaomi) remains unfixed.

讯飞输入法小米版 (com.iflytek.inputmethod.miui)

We found that the payload of each HTTP request that the iFlytek-based app sends to `pinyin.voicecloud.cn` is encrypted with the following algorithm. Let *s* be the current time in seconds since the Unix epoch at the time of the request. For each request, an 8-byte encryption key is then derived by first performing the following computation:

$$x = (s \ \% \ 0x5F5E100) \ \hat{} \ 0x1001111$$

The 8-byte key *k* is then derived from *x* as the lowest 8 ASCII-encoded digits of *x*, left-padded with leading zeroes if necessary, in big-endian order. In Python, the above can be summarized by the following expression:

$$k = \text{b'\%08u'} \ \% \ ((s \ \% \ 0x5F5E100) \ \hat{} \ 0x1001111)$$

The payload of the request is then padded with PKCS#7 padding and then encrypted with DES using key *k* in ECB mode. The value *s* is transmitted in the HTTP request in the clear as a GET parameter named "time".

Since DES is a symmetric encryption algorithm, the same key used to encrypt a message can also be used to decrypt it. Since *k* can be easily derived from *s* and since *s* is transmitted in the clear in every HTTP request encrypted by *k*, any network eavesdropper can easily decrypt the contents of each HTTP request encrypted in the manner described above.

```
{"p":{"m":53,"f":0,"l":0},"i":"nihaoniba"}
```

Figure 3: Excerpt of decrypted information, including what we had typed ("nihaoniba").

Notably, we found that users' keystrokes were sent to `pinyin.voicecloud.cn` and encrypted in this manner, seemingly to implement the keyboard's cloud-based recommendation

feature. Therefore, a network eavesdropper who is eavesdropping on a user's network traffic can observe what that user is typing by exploiting this vulnerability (see Figure 3).

Finally, the DES encryption algorithm is an older encryption algorithm with known weaknesses, and the ECB block cipher mode is a simplistic and problematic cipher mode. The use of each of these technologies is problematic in itself and opens the app's communications to additional attacks.

Mitigation

In order to address the reported issues, Xiaomi should secure all of the keyboard apps' transmissions using a popular, up-to-date implementation of HTTPS or, more generally, TLS instead of relying on custom-designed cryptography to secure the transmission of sensitive user data.

We previously disclosed similar vulnerabilities to Baidu (百度), Tencent (腾讯), and iFlytek (讯飞) in their corresponding keyboard apps, Baidu IME, Sogou IME, and iFlyTek IME. Tencent responded by securing Sogou IME (搜狗输入法) transmissions using TLS.