Wiki

[[Stegdetect Research]]
PixelKnot is available on:

- GP site https://guardianproject.info/apps/pixelknot/
- Github https://github.com/guardianproject/PixelKnot/
- Transifex https://www.transifex.com/projects/p/pixelknot/
- Google Play https://play.google.com/store/apps/details?id=info.guardianproject.pixelknot
- Releases:
- Repo:

2nd Round Development Roadmap

This second round of development will address issues raised in the completed code review completed by iSEC Partners in November 2013, as well as from user feedback from Google Play Store, and help requests from the broader public. These concerns can be consolidated thusly:

1. Constant salt value used in PBKDF2 algorithm

Severity: High

A salt value is required to "seed" the cryptographic algorithm used to generate the resulting image. In order for an image to be successfully decrypted, the same seed is required to reverse the process. This seed is used to create a key that is applied across the data to encrypt or decrypt. With the seed "hard-coded" into the current app, there is a known possible key space; each key can be used in a brute-force attack to reverse encryption. The key space being limited gives adversaries an unnecessary advantage when trying to crack an intercepted image.

While any app wishing to decrypt an image must know the corresponding seed, PixelKnot will now include the option for users to set a custom seed, or generate a random one. If none is supplied, the app will fall back to the default, hard-coded one. Should the user choose a custom seed (which can be set as a default settings for reuse) then images circulating among users who know which seed to use (knowledge communicated through some channel outside of PixelKnot's purview) will not be subject to this attack. Custom-seeded images will use a key outside the known key space created by the default seed.

2. No PGP Encryption

Severity: Low

Because steganography is the act of hiding messages in plain sight, it is exemplary of "security through obscurity". The app would better serve users if it encouraged them to take advantage of encryption ("security through security!") in addition to what it currently offers. PixelKnot currently has an option for users to password-protect their messages using AES encryption, which is a good start. However, PixelKnot should make it easier for more advanced users to include PGP encryption. The UX should provide interaction with PGP-enabled apps on the device via intents. The newer version will include an additional option on the message-generation screen that launches a PGP-capable app (should the user have on installed on their device.) When a user encrypts their message on their other app, and returns to PixelKnot, the app will embed the encrypted PGP message.

3. Confusing Workflow When Entering App via Share Intent

Severity: High

Initial criticisms have shown that a number of users did not know how to open the app to decrypt an image after it had been encrypted. When the user shares an image into the app, PK automatically assumes the user wants to decrypt. However we can gauge from user response that this is not always clear; sometimes a user shares an image into the app with the intent of inserting a new message into it (encrypting). Users going down the wrong path would have experience either a crash or, worse, a hanging process that failed to gracefully terminate on error.
To address this, PixelKnot's entry screen (on share intent) will be reorganized so that a user knows clearly the difference between encrypting and decrypting; the screen will explicitly ask the user whether they want to encrypt or decrypt when entering the app this way, and further handle errors that would occur when data is malformed for whatever reason.

4. Processing in Foreground Requires Too Much of Users' Attention

Severity: Medium

Currently, PixelKnot has a splash screen with a progress bar that indicates the app's status as it generates the final image. The screen was designed to be as entertaining as possible, but a much better user experience sought; users don't want to have to watch the app perform its calculations. Backgrounding the process and using notifications to alert users when finished would make the experience that much better.

This next round of development will include a minor improvement to processing image encrypting/decrypting in the background, so it is non-blocking to app users. We will implement notifications in the notification bar in addition to the splash screen view, and make sure users are ambiently notified when their image is finished if they are not currently in-app.

5. The Very Possibility of the F5 Algorithm Becoming Obsolete/Being Compromised

Severity: High

We are aware of research being done to break f5 steganography in particular. We must prepare for the day when we must replace this algorithm for another, newer, more robust one. We propose a moderate change to its current architecture that will allow us to modularize the steganography algorithm used. Ultimately the app should be algorithm-agnostic-- should we need to replace the chief algorithm, we should be able to easily perform substitution. This opens up the possibility of treating the steganography package as a plugin; users might be able to choose between a variety of installed algorithms on-the-fly, or through the app settings. (Please note, a formal plugin structure is not on the roadmap at this time.)

Additionally, we know that there are ways in which the algorithm is ineffective. One particular ways in which the algorithm fails is when a user chooses a source image that has a limited color palate. (Certain users have even mentioned that the app fails when an image is all black-- indicators of the very problem described here.) However, while we know that users will get more out of the app if they choose images containing multiple areas of color, this is not obvious. A bit of user education with tool tips and in the app's "About" screen will go a long way. Users will have better trust in the application if disclosure of best practices and associated risks are more highly visible.