Gaining User Trust Without Brand Recognition

An analysis of how startup developer teams can gain the trust of their users without being an established brand

Toby Smith

toby.smith@ebrisoft.com
www.ebrisoft.com

ABSTRACT

Apps today are easily disposable; this means they need to not only be feature-rich, but they need to look and feel trustworthy for users to install and keep them. While app stores make it easy to trust apps by default, poor UI designs, the choice of words used, or feature implementation specifics can all lead to users losing that trust. Detailed in this paper are techniques previously used by the companies Monzo, Myki, and Telegram to influence the trustworthiness felt by those who install their apps. An evaluation of these techniques is given detailing whether newly formed, unheard of, companies should be using these techniques today as they launch their first products to market.

KEYWORDS

Security features, Brand Transparency, Data Protection, Start-Up, User Interaction

1. INTRODUCTION

Amongst other factors, the success or failure of a product heavily depends on its ability to draw in and retain a userbase of sufficient size. It can be tricky for newly formed companies to find and retain users because they have not shown themselves to be trustworthy. For companies which are already seen to be trustworthy in the public eye, amassing a userbase for a new product is more straightforward as they have a pre-existing relationship with their users.

This paper details the designs, technologies, and security features that newly formed companies have used to aid the expansion of their userbase and how those techniques affect the trust a user might have in their product.

2. FIELD

The introduction of app stores on mobile platforms has enabled users to search and find many apps which have the same or similar functionality. The high volume of apps within these stores, along with how trivial the installation process is, enables users to install, trial, and remove apps quickly and easily to find the one they like the most. If an app contains flaws, users will uninstall it and try one developed by a competitor. Should a user not perceive an app as trustworthy, then they will most likely not install it in the first place or discard it as soon as they decide that it cannot be trusted. As modern desktop operating systems also continue to implement app store models (LeBlanc, 2015), desktop software will experience this same cycle of being replaceable in favour of more trustworthy or feature-rich competitors. As such, this paper focuses on software written for all user platforms while converging on applications which specifically require large amounts of trust - typically due to the nature of the user data they store.

3. KEY PLAYERS

Monzo, Myki, and Telegram have all begun operating within the last ten years, and all offer services which gather and store large amounts of personal data: financial, passwords, and personal communications respectfully. Despite the dangers of giving personal information to a company that is ‘unheard of’, all three have been able to persuade their users that they should be trusted.

3.1 Monzo / Monzo Bank Ltd

Formerly called Mondo, Monzo is a challenger bank in the UK which launched its initial Alpha trial on the 30th of October, 2015 (Blomfield, 2015). The primary selling point of Monzo over traditional high street banks is that it is branchless. Having no branches to run or maintain helps keep costs low and allows Monzo to focus their time building the best banking app possible - which is entirely how the userbase interact with their accounts. The trust that a user gives an app is usually with regards to their data - typically their name and email address, but often includes their date of birth, address, and payment information. Launching a challenger bank comes with an exponentially more significant amount of required trust because the users need to trust the app developers to look after their money for them. Monzo was able to overcome this challenge as demonstrated by the fact that within the 136 day alpha period of their release, it’s users spent more than £1.5m (Thomas, 2016).

3.1.1 A Pre-established Following

Long before the Monzo banking app was available for download, their online presence and user engagement with it were strong. Rather than launching an app and then trying to amass a userbase from the ground up, they used a relentless online blogging presence to write about the app they were currently developing. This communication channel allowed them to receive feedback from potential users before they had access to the app (Monzo, n.d.). Doing so meant that by the time of the app’s launch, there were already many users waiting to try out the product who were willing to share positive opinions about the company.

3.1.2 Breaking the Mould

Traditionally user accounts are protected by a password that the user sets and is sometimes backed up with two-factor authentication. Monzo has taken a different route to secure their users’ accounts, a method that has proven popular with their userbase. When a user downloads the app and wishes to log in for the first time, they enter their email and press submit; this sends them an email with what Monzo brand as a ‘Magic URL’. The opening of this URL on the same device triggers the app to open and to log in (Borbon, 2019).

This security model means that Monzo accounts are protected by the security put into place by their users’ email providers and inform their users to enable two-factor authentication on their

email account wherever possible. Monzo makes it clear to their users that magic URLs are only ever sent when requested by the user; they see having no password as a way to reduce the amount of potentially vulnerable data that a user can have (Borbon, 2019). Should an attacker gain access to a user's email account, they could gain access to their Monzo app; however, this access is read-only as each transactional interaction requires the user to insert their debit card's PIN.

Advertising this feature to the userbase can, and has, given an impression to the average user that Monzo is thinking about their user's security and is willing to implement what they believe is right rather than merely following the trends.

3.1.3 Transparency

All banks experience the occasional technical glitch, and Monzo is no exception. One of the reasons that their userbase continues to trust them is their policy to be as transparent as possible. A large portion of the transparency they have put into place is through their online blog; whenever Monzo encounters a problem, they are quick to post updates reassuring users that they are aware of the situation. When Monzo resolves issues, they typically post further updates explaining what went wrong and what has been put into place to stop reoccurrences. A recent example is when most Monzo systems stopped working on the 29th of July, 2019. A routine database upgrade didn't go as planned and left users unable to use their debit cards as well as the app. Monzo released a short blog post the following day acknowledging the issue and then a longer more detailed one 10 days later once they had concluded what happened more specifically.[1,2]

To achieve as much transparency as possible with their userbase, Monzo has launched a transparency dashboard on their website functioning as a central location for users to see analytics, submit user reports or suggestions, as well as check the availability and uptime of Monzo systems.[3]

3.2 Myki / Myki Inc.

The usage of password managers is ever-increasingly on the rise (LastPass, 2019) as more users are having to memorise rapidly expanding lists of passwords for accounts within their personal and professional lives. Incorporated in 2015 (Anderson, 2016) and initially launching their password manager service in 2016 (Lynley, 2016), Myki has released their product for many different platforms including all conventional desktops, phones, and web browsers (Myki, n.d.). As well as the traditional password management functionality, Myki software enables the automatic auto-filling of multi-factor authentication passes, debit/credit card details, and government-issued identification details like passport or driving licence details.

3.2.1 No Cloud Storage

A traditional password manager stores a user's passwords within a single encrypted file in the cloud which their client downloads and decrypts on demand. This implementation model opens up the potential, no matter how unlikely, for a malicious agent to gain access to the encrypted passwords and then attempt to decrypt them. Myki differs from these traditional password managers by storing the secured password file locally on the client machine only - never in the cloud (Myki, n.d.). Both their desktop application and their mobile apps function as both password manager clients and as storage for the secured password file. When the user updates a password on one client, it communicates the change with any others to ensure that they are all up to date. The browser plugins created by Myki only act as traditional password manager clients, meaning that they do not store the secured password file. When a user logs in to an account via a browser plugin, it securely communicates with either a desktop application or app to retrieve the password. This implementation allows the user to install the browser plugin on a computer which isn't their own without storing their passwords on that machine while still avoiding storing them in the cloud (Myki, n.d.).

3.2.2 Optical Peer-To-Peer Communication

To encrypt and decrypt securely sent information, the sending and receiving clients must share the same key. Since its publication in 1976, one of the most popular ways for clients to agree on a single key is the Diffie-Hellman Key Exchange (University of Nottingham, 2017). While it is still popular and secure in the modern cryptography industry, it relies on there being only two clients used in the exchange. When devices communicate via a cloud server, this is not a problem as they can each have a one-to-one relationship with the server. Myki servers, however, do not directly play a role in the communication between user devices (Myki, n.d.) - this means a different solution is required. The keys used by Myki clients are instead transmitted optically through a QR code displayed on one device and scanned by another (Myki, n.d.). This offline communication means that Myki never sends the keys, or any of their components, through an internet connection which is where a malicious agent would typically intercept them; in this instance, they would need physical access to a device to intercept a key.

3.2.3 Frontpage Endorsements and Accreditations

Should a user visit the website of an app or company looking to see whether they should be trusted or not, one of the most decisive factors is to see if others already trust them. Myki has implemented this concept by displaying on their front page the icons of well-known newspapers and tech review websites which have said positive words about their products. The icons are also hyperlinks to the articles themselves so that users can quickly learn what the reviewers said about Myki.

In the footer of the Myki website, there are further images to quickly inform the visitor that Myki is GDPR compliant and that the whole company 'Complies with the requirements of the Cyber Essentials Scheme', certified by Tresor Security (Akhtar, 2019).

[1] https://monzo.com/blog/2019/07/30/we-had-issues-with-monzo-yesterday

[2] https://monzo.com/blog/2019/09/08/why-monzo-wasnt-working-on-july-29th

[3] https://monzo.com/transparency

3.3 Telegram / Telegram FZ LLC

Telegram is an instant messaging and voice over IP app originally developed in 2013. While Telegram is not considered new in 2019, and nor are its technologies currently emerging, they were when it was released, and they remain relevant today as they are still unique amongst its competitors. As of March 2018, Telegram celebrated 200 million monthly active users (Durov, 2018). As well as offering traditional messaging app features including chats, group chats, voice calls, and video calls, Telegram also has a feature it brands *channels*. Channels are feeds which a single user can post messages into, other users can subscribe to the channel to receive the messages, but they cannot reply (Telegram, n.d.). This model of communication is suitable for businesses or celebrities to reach large crowds. Some examples include the football team *Real Madrid* who use a channel to share updates about their players (Real Madrid, n.d.) and BBC News who share articles from their website (BBC, n.d.).

3.3.1 Open-Sourcing

Telegram has many official clients, including ones which run on Android, iOS, Windows, macOS, Linux, and Chrome (Telegram, n.d.). All official clients for Telegram are open source and are publically available on Github (Telegram, n.d.). Alongside the clients is the Telegram public API. The use of this API allows anyone to create their own Telegram client to directly compete with the official ones (Telegram, n.d.). Communication between the Telegram servers and any unofficial clients can be direct via the API or can travel through the Telegram Database Library (TDLib). TDLib is a fully functional Telegram client written mostly in C++ which developers can integrate into their unofficial clients; it can also be found on Github (Telegram, n.d.).[4] Open-sourcing code helps users (especially technically aware users) to trust a company and their software as all the code is available to be seen. This visibility means that issues are more likely to be found and resolved, as well as ensuring that the company is not hiding anything malicious. Telegram has also stated plans to open-source its server-side code in the future (Telegram, n.d.).

3.3.2 Custom Encryption Protocol

There are many posts on internet forums from people asking why one should not design and implement their own security protocols and every time the same answer is returned. To do so takes a very large and complex understanding of cryptography, mathematics, and statistics- skills which the average developer does not possess to a required degree, even if they understand the basics of how common cryptography protocols function.[5,6,7] The question of *why* is also commonly raised; there are several encryption protocols already in use within the industry, all of which have passed vigorous peer-review processes as well as passing the test of time such as the *Signal Protocol* (Cohn-Gordon et al., 2019). Telegram, however, decided during the development of their systems that they would design and implement their own encryption protocol called MTProto v1.0 (Telegram, n.d.). This protocol supports both client-to-server communication in cloud-based conversations as well a client-to-client communication in secret conversations (Telegram, n.d.). The goal of developing their own encryption protocol was to increase reliability over unstable mobile networks as well as to increase the speed that large files are handled (Telegram, n.d.).

In 2015 two theoretical vulnerabilities were discovered in MTProto where padding could be replaced or extended in such a manner that the intended recipient would still accept and successfully decrypt the ciphertext (Jakobsen and Orlandi, 2015). Such vulnerabilities where a ciphertext can be manipulated but still accepted mean that the protocol cannot be considered IND-CCA secure[8]. Telegram acknowledged these vulnerabilities but did not regard them as an issue. They released a statement saying that while there are use cases where IND-CCA compliance is essential for security, MTProto is not one of them and that the security of Telegram messages remains unaffected (Telegram, n.d.).

From version 4.6 onwards, all official Telegram clients have upgraded to MTProto 2.0, which Telegram claim is IND-CCA secure (Telegram, n.d.).

3.3.3 Bug Bounties

The open-sourcing of codebases means that anyone who wishes to can help by reporting issues they find within the code. Users can also directly contribute to the code by fixing any previously raised issues or by adding new content. Telegram has taken this concept a stage further and offer monetary rewards for people who report security issues specifically with either their clients or their secure protocol (Telegram, n.d.). Depending on the significance of the issues, bounties can vary between $500 to more than $100,000 (The Telegram Team, 2015). One notable example is a user referred to as *x7mz*, who was awarded $100,000 for discovering a vulnerability with the implementation of secret chats at the time (The Telegram Team, 2013). In the past Telegram have also held competitions for developers to try and crack their custom encryption protocol with rewards as high as $300,000. However, no one has ever been successful (Telegram, n.d.).

4. Evaluation

All three of the analysed companies in this paper feature blogs on their websites, these blogs appear to be excellent places for the brands to reach out to their users about new features, issues, and related company news.[9,10,11] Many other third parties also appear to quote blog posts when they are writing articles about the companies and their products. For at least Monzo specifically, the success of their blog before their product was released was very advantageous for them to increase the number of users they had when their app was first released.

While the 'magic URL' system and the custom encryption protocol that Monzo and Telegram have respectively implemented appear to be working in their favours now, there is always the possibility that white or black-hat hackers could discover new security threats. Telegram almost had this happen with MTProto 1.0, and it is feasible that it might happen again. While innovation is always vital for a sector to develop, security advancements should be created by developers who have many years of experience in the security industry rather

[4] https://github.com/tdlib/td

[5] https://crypto.stackexchange.com/questions/43272/why-is-writing-your-own-encryption-discouraged

[6] https://security.stackexchange.com/questions/18197/why-shouldnt-we-roll-our-own

[7] https://www.quora.com/Is-it-possible-to-write-your-own-encryption-program

[8] Indistinguishability under chosen ciphertext attack (Möller, 2004)

[9] https://monzo.com/blog/

[10] https://myki.com/blog/

[11] https://telegram.org/blog

than from within newly formed companies or from developers with a broader knowledgebase. Also, it is notable that security developments should be excessively peer-reviewed before being implemented.

The open-sourcing of codebases appears to be an excellent way to gain the trust of other developers while also being an indirect sign to the general public that you are trying to be trustworthy. However, open-sourcing code is not an approach that all companies can take, as doing so also grants anyone the ability to take the code for themselves and do whatever they wish with it. Apps which are not free cannot be open-source because people who wish to not pay for the product can compile it for themselves, bypassing purchasing from an app store. Apps which contain in-app purchases within them can be open source provided that the back-end services remain closed-source, the back-end validates that the client is genuine, and that the client cannot use the paid-for service without a connection to the back-end. It is also highly likely that the portion of the client codebase which handles the payments will need to be closed-source too to avoid tampering.

5. Conclusion

The concept of open-sourcing as well as writing blog posts both appear to appeal to users in the same manner: they both help the company to be as transparent as possible. It is tricky to gain trust while keeping your cards close to your chest, and it is easy to trust someone when you know, or when you think you know, their true intentions. While blog posts can be lies and back-end code could be malicious or insecure, trust is generally about appearance - unless you've previously given reasons for why people should not trust you.

Blogs are only excellent methods of pushing out content provided there is an audience in place to read what is posted. Research into both search engine optimisation (SEO) along with findings on how to represent a brand effectively using third-party social media platforms will aid any blogged content by helping the growth of an initial audience.

Bug bounties and other competitions are also excellent mechanisms for a company to interact with its product's users. However, if the code is of poor quality or contains many flaws, then users will be discouraged from using the software once those flaws are public. These latter techniques are more related to strengthening the numbers a userbase rather than gaining an initial one.

Reference List

Akhtar, J. (2019). *Certificate of Assurance*. [online] *Myki*. Tressor Security. Available at: https://static.myki.co/certifications/Certificate_Cyber_Essentials_2018_-_Myki.pdf [Accessed 16 Oct. 2019].

Anderson, B. (2016). How a Digital Tool for Grandma Turned Into a Startup. *The Wall Street Journal*. [online] 20 Nov. Available at: https://web.archive.org/web/20161202105159/https://www.wsj.com/articles/how-a-digital-tool-for-grandma-turned-into-a-startup-1479697323 [Accessed 10 Oct. 2019].

BBC (n.d.). *BBC News*. [online] Telegram. Available at: https://t.me/s/bbcworldnews [Accessed 10 Oct. 2019].

Blomfield, T. (2015). We're Ready. Available at: https://monzo.com/blog/2015/10/30/we-are-ready [Accessed 10 Oct. 2019].

Borbon, B. (2019). Is Monzo Safe? Available at: https://monzo.com/blog/2019/02/06/is-monzo-safe [Accessed 10 Oct. 2019].

Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L. and Stebila, D. (2019). *A Formal Security Analysis of the Signal Messaging Protocol*. [online] Available at: https://eprint.iacr.org/2016/1013.pdf [Accessed 10 Oct. 2019].

Durov, P. (2018). 200,000,000 Monthly Active Users. Available at: https://telegram.org/blog/200-million [Accessed 10 Oct. 2019].

Jakobsen, J. and Orlandi, C. (2015). *On the CCA (in)security of MTProto*. [online] Available at: https://eprint.iacr.org/2015/1177.pdf [Accessed 10 Oct. 2019].

LastPass (2019). *The 3rd Annual Global Password Security Report*. [online] *LastPass*. LastPass. Available at: https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LMI0828a-IAM-LastPass-State-of-the-Password-Report-2019.pdf [Accessed 10 Oct. 2019].

LeBlanc, B. (2015). Delivering a single unified Store experience in Windows 10. *Windows Experience Blog*. Available at: https://blogs.windows.com/windowsexperience/2015/04/09/delivering-a-single-unified-store-experience-in-windows-10/ [Accessed 10 Oct. 2019].

Lynley, M. (2016). *Myki rolls out a password manager that locks all your info away on your phone*. [online] TechCrunch. Available at: https://techcrunch.com/2016/09/13/myki-rolls-out-a-password-manager-that-locks-all-your-info-away-on-your-phone/ [Accessed 10 Oct. 2019].

Möller, B. (2004). A Public-Key Encryption Scheme with Pseudo-random Ciphertexts. *Computer Security – ESORICS 2004*, [online] 3193, pp.335–351. Available at: https://link.springer.com/chapter/10.1007%2F978-3-540-30108-0_21 [Accessed 10 Oct. 2019].

Monzo. (n.d.). *Monzo Blog*. [online] Available at: https://monzo.com/blog/ [Accessed 10 Oct. 2019].

Myki (n.d.). *Download Myki on all of your devices*. [online] Myki. Available at: https://myki.com/download [Accessed 10 Oct. 2019a].

Myki (n.d.). *Frequently Asked Questions about the Myki App*. [online] Myki. Available at: https://myki.com/faq [Accessed 10 Oct. 2019b].

Myki (n.d.). *How Myki Works*. [online] Myki. Available at: https://myki.com/privacy/how-myki-works [Accessed 10 Oct. 2019c].

Real Madrid (n.d.). *Real Madrid*. [online] Telegram. Available at: https://t.me/s/RealMadridEN [Accessed 10 Oct. 2019].

Telegram (n.d.). *Channels FAQ*. [online] Telegram. Available at: https://telegram.org/faq_channels [Accessed 10 Oct. 2019a].

Telegram (n.d.). *FAQ for the Technically Inclined*. [online] Telegram. Available at: https://core.telegram.org/techfaq [Accessed 10 Oct. 2019b].

Telegram (n.d.). *FAQ for the Technically Inclined (MTProto v.1.0)*. [online] Telegram. Available at: https://core.telegram.org/techfaq/mtproto_v1 [Accessed 10 Oct. 2019c].

Telegram (n.d.). *Mobile Protocol: Detailed Description (v.1.0, DEPRECATED)*. [online] Telegram. Available at: https://core.telegram.org/mtproto/description_v1 [Accessed 10 Oct. 2019d].

Telegram (n.d.). *MTProto Mobile Protocol*. [online] Telegram. Available at: https://core.telegram.org/mtproto [Accessed 10 Oct. 2019e].

Telegram (n.d.). *Telegram APIs*. [online] Telegram. Available at: https://core.telegram.org/api [Accessed 10 Oct. 2019f].

Telegram (n.d.). *Telegram Applications*. [online] Telegram. Available at: https://telegram.org/apps [Accessed 10 Oct. 2019g].

Telegram (n.d.). *Telegram Cracking Contest Description*. [online] Telegram. Available at: https://core.telegram.org/contest300K [Accessed 10 Oct. 2019h].

Telegram (n.d.). *Telegram Database Library*. [online] Telegram. Available at: https://core.telegram.org/tdlib [Accessed 10 Oct. 2019i].

Telegram (n.d.). *Telegram FAQ*. [online] Telegram. Available at: https://telegram.org/faq [Accessed 10 Oct. 2019j].

The Telegram Team (2013). Crowdsourcing a More Secure Future. Available at: https://telegram.org/blog/crowdsourcing-a-more-secure-future [Accessed 10 Oct. 2019].

The Telegram Team (2015). Crypto Contest Ends. Available at: https://telegram.org/blog/cryptocontest-ends [Accessed 10 Oct. 2019].

Thomas, T. (2016). The Next Step. Available at: https://monzo.com/blog/2016/03/17/beta [Accessed 10 Oct. 2019].

University of Nottingham (2017). *Secret Key Exchange (Diffie-Hellman) - Computerphile*. *Youtube*. Available at: https://www.youtube.com/watch?v=NmM9HA2MQGI [Accessed 10 Oct. 2019].