You Only Live Twice or “The Years We Wasted Caring about Shoulder-Surfing"

Joseph Maguire
School of Computing Science
University of Glasgow
Scotland
Glasgow, G12 8QQ
UK
www.dcs.gla.ac.uk/∼joseph
joseph@dcs.gla.ac.uk

Karen Renaud
School of Computing Science
University of Glasgow
Scotland
Glasgow, G12 8QQ
UK
www.dcs.gla.ac.uk/∼karen
karen@dcs.gla.ac.uk

Passwords, in theory, are a good idea. They have the potential to act as a fairly strong gateway to protected information and services. In reality, lots of problems emerge when passwords are used by normal people in their everyday lives. The great thing about passwords is that they are universally accessible, and very easy to add to any application. They are also well-established and a mechanism that users readily understand. However, passwords are also (1) easily shared, (2) trivial to observe and (3) maddeningly elusive when forgotten. Various alternatives to passwords have been proposed, many of which try to address at least one of the problems mentioned above. Alternatives to passwords are often judged harshly because, whereas they might well demonstrate superior memorability, they might well fail in terms of universal accessibility, or require extra effort on the part of the user. There seems to be a reluctance to lose the benefits of passwords and to switch to another mechanism, even if the most glaring flaws of passwords have been addressed. Graphical authentication mechanisms are a case in point. They have demonstrated their superior memorability but still suffer from observability aws. The question we pose in this article is whether users care about this, and whether this is really the deal-breaker it appears to be in terms of adopting alternatives which are indeed susceptible to observation.

authentication

1. INTRODUCTION

None of password’s problems are new: they are the stuff of legend, as is evident from the tale below:

Two brothers lived in a small town. Cassim was wealthy and wanted for nothing, Ali Baba was poor and cut wood tirelessly to support his family. While cutting wood in the forest one day, Ali Baba spotted a flock of fierce men on horses heading his way and so hid in a nearby tree. The leader walked right-up to the tree and uttered a simple, secret word. A huge rock rollbacked to reveal a vast cave, covered in treasure.

Stunned and shaken, Ali Baba stayed silent in the tree as the men entered the cave and the rock rolled shut. The men eventually left, at which point Ali Baba leaped from the tree and spoke the simple, secret word. The rock rollbacked and Ali Baba grabbed as much treasure as he could and ran back to his family.

Cassim, shocked by Ali Baba’s sudden wealth, demanded answers. Ali Baba recanted the story and agreed with his brother to return in the morning. Cassim, always the greedier, set out earlier. Once at the cave, Cassim spoke the simple, secret word and, to his amazement, the rock rollbacked. Cassim entered and the rock rolled shut, he began grabbing as much treasure as possible. However, when Cassim wanted to leave, he could not recall the simple, secret word. He tried in vain to remember, spouting word after word.

The flock of thieves returned to the cave to deposit more treasure. The leader spoke the simple, secret word. The rock rollbacked to reveal Cassim cowering in the corner of the cave. Cassim never left.

The tale of Ali Baba and Cassim is more commonly known as *Ali Baba and the Forty Thieves* from *1001 Arabian Nights*.

Some lessons emerge from this narrative:

- The first paragraph illustrates that passwords are *not resistant to surveillance*. Ali Baba, after all, was able to enter the cave because he was eavesdropping. The problem still persists today, as attackers can easily observe individuals entering their passwords by shoulder-surfing. Passwords need to be resistant to eavesdropping and observation, otherwise their strength is weakened. Passwords need to be secret and kept safe.
- The second paragraph showcases secrets being *shared with others*. Ali Baba is able to communicate the password to his brother and, as a result, Cassim was also able to access the cave. Passwords should be known only to the individual accessing the cave, no-one else. Otherwise, it weakens the strength of the password as it is no longer a secret.
- The third paragraph demonstrates that passwords are *not memorable*. Cassim's life literally depended on his remembering the password that Ali Baba shared with him. A good password is a lengthy string of gibberish with no meaning. The lack of meaning is what makes such passwords difficult to remember. If a user does not write it down, they will almost certainly reuse it. While a memorable password has meaning, it also makes it predictable as it has some structure or pattern. Making it easily compromised through brute-force or social-engineering attacks.
- The final paragraph exemplifies the fact that users and organisations only know when the attacker is unsuccessful. When the leader of the thieves rolls back the rock, he is confronted with Cassim. The secret has been compromised, the password is pointless. In modern-day electronic systems there is often no way to detect that an attacker has successfully used another person's password.

These problems have always plagued passwords, even when they were first deployed on the Compatible Time Sharing System (CTSS) in 1962 (1). The key difference between then and now is that CTSS was aimed at scientists and engineers. The password pitfalls could be combatted with training and education. Passwords are undeniably powerful, if used properly. Organisations can craft secure locations, make surveillance difficult, and use security policies and training to mitigate the aforementioned problems. The real problem is everyday users, managing an increasing number of passwords to achieve more and more tasks. Popular products like the Nintendo Wii and Apple iPhone have pushed passwords into untrained hands and spaces, never initially envisioned or planned for.

What is needed is a *password for plebs*. Such an authentication approach would be tailored to the needs of the average user of popular consumer products such as the Nintendo Wii and Apple iPhone. Such an approach should tackle the problems of *observation*, *memorability* and *sharing of secrets*. This authentication approach would be targeted at consumers and popular products rather than professionals and tele-type terminals. Such an approach would fundamentally target the problems of surveillance, sharing and memorability. Shigeru Miyamoto, a leading light behind the Nintendo Wii, claims "A great idea solves multiple problems at the same time". The great idea we thought we had was a *graphical authentication mechanism* called Tetrad (2). This mechanism utilised an image-based secret that users could enter in a way that was resilient to surveillance. Tetrad appeared to solve a number of the problems of passwords: the images were memorable, and observation of a user did not leak the secret. We thought we had found a viable alternative.

The initial prototype we constructed of Tetrad showed promise. Unfortunately, Tetrad, like many other graphical authentication mechanisms, has yet to be adopted by any commercial product. Plebs still use passwords and those passwords still suffer from the age-old problems. For example, the Nintendo Wii and Apple iPhone have yet to make any changes to their authentication approach in response to their systems being hacked and passwords leaked. In particular, neither have bothered to tackle the problem of shoulder-surfing even though their users often enter their passwords in full view of other users and both rely on an on-screen keyboard for password entry. This makes observation trivial and secrets public . Is this evidence of negligence or wisdom? Here we will explain that it is actually a perfectly justified decision.

Dunphy *et al.* argues the reason for slow uptake in graphical authentication approaches, like Tetrad, is due to designers and researchers not tackling practical issues such as deployment (7). This is certainly true of the initial incarnation of Tetrad. The initial implementation comprised only the mechanism itself; there was no registration process and it served no authenticating purpose. Since it was not protecting any application it could be argued that the evaluation we carried out to prove its efficacy was unrealistic. The implementation did not target an actual platform but rather a mythical Apple television context of use. Furthermore, the mechanism relied on a stock collection of celebrity faces. Personal images were avoided because it is difficult to 'bootstrap' a graphical authentication mechanism with these but celebrity faces suffer from predictability issues so the choice was less than optimal. These prickly deployment issues were neatly avoided in our early prototype.

In this paper we discuss the prototyping of two versions of Tetrad. We will refer to them as *Jack* and *Jill*. Both versions rely on personal images from Facebook. Jill is designed to make Tetrad even more resistant to shoulder-surfing. We will discuss deployment choices related to image bootstrapping procedure and the choice of application both mechanisms were bundled with. We will outline the authentication scenario and explain how the mechanism interacted with the application.

The following section presents related research, regarding the problems of passwords and the concerns of observaility of graphical secrets. The next section outlines the design of the application, followed by a section detailing the image-bootstrapping procedure for both versions of Tetrad. Finally, we discuss the problems we encountered in image-bootstrapping and address the relevancy of obfuscation in graphical authentication. The paper concludes that shoulder-surfing may well be a non-issue in authentication design.

2. RELATED RESEARCH

There are many problems with passwords. Good passwords are system-generated lengthy strings of gibberish with no meaning. However, such passwords are difficult for a user to manage in the short-term, never-mind store long-term. Users, therefore, adopt coping mechanisms (3), such as writing passwords down or reusing them, to avoid inconvenience. User-generated passwords are an alternative option but people are ‘lazy’ and often create short and memorable secrets (8). Memorable passwords have meaning, i.e. pattern and structure, that brute-force and social-engineering attacks can easily exploit. One solution to all these problems is an alternative authentication approach, e.g. graphical passwords.

Graphical passwords are based on images rather than characters. Humans use a dual-coding approach to process verbal and visual stimuli, generating analogue codes for visual information and symbolic codes for verbal (4). This approach results in images having superior recognition for the words that name them (5). Individuals are also good at recognised previously encountered images, even in pairs containing a distractor (6).

A concern with graphical passwords is the predicability of passwords created by users. Users are influenced by elements such as attraction, race and gender when selecting faces. Therefore, an attacker can potentially identify likely images that form part of a user’s secret by simply knowing something about the user. This makes sourcing images or ‘bootstrapping’ a mechanism with images, difficult, as some collection of images may not be suitable. Celebrity images for example, a male user may select attractive female users around his age. The pool of images would be nearly the same for every user, making it easy for an attacker to identify a likely password. Graphical passwords based on personal images have strong memorability (9) and may be alternative solution, as they would be tailored to the user. Furthermore, social network services have made accessing and using such personal images much easier.

Therefore, we decided to construct Tetrad prototypes that used images of faces, extracted from personal images, sourced from Facebook.

3. APPLICATION DESIGN

Purchasing a movie in front of friends and family is a familiar authentication scenario for many. Users peruse the catalogue, pick a product and then enter a password to purchase it. Current consumer products expect users to enter the password using an on-screen keyboard, essentially exposing it to onlookers. This burdens the user with the decision of either sacrificing their secret to maintain good vibrations or being paranoid and demanding that everyone exit the room.

Tetrad is designed to tackle this scenario, allowing an individual to authenticate without worrying about onlookers. The mechanism presents 45 images within a grid and asks users to position four secret images so that they align either horizontally, vertically or diagonally. Interaction does not involve moving individual images but manipulates rows and columns of them, resulting in entry being resilient to observation from onlookers.

Tetrad was implemented in Objective-C and originally only targeted Mac OS X. The implementation made extensive use of key frameworks, such as Core Animation, and users interacted with it using an Apple Remote Control. The prototype was designed for use on a television and was aimed at tackling the aforementioned authentication scenario.

Translating the implementation to the iPhone was relatively simple, as the iPhone’s operating system is essentially Mac OS X. Interaction had to be re-thought as the remote control was no longer relevant. Remote control events thus became multi-touch gestures. Users swipe up, down, left and right on a particular row or column to reposition images. A double-tap gesture submits an authentication attempt.

The first step in the design of the application was to define an **authentication scenario**. Numerous scenarios were considered, including a password manager and location tracker, with the conclusion being to continue with the original one, a digital content store, with the addition of a media player. A digital content store is a natural fit for Tetrad as both heavily depend on screens. Digital stores rely on high-quality screens that are spacious enough for a user to peruse a product catalogue and splendid enough to showcase content. Tetrad, equally, relies on screens with enough space and quality to showcase images.

The alternative scenarios were seriously considered but ultimately rejected. Deploying a password manager or location tracker puts the user at risk. The application will, inevitably, contain bugs and flaws. When it fails, and it will, the user’s passwords and locations would be exposed, an unacceptable risk. A digital store puts content at risk, not the user, a far preferable scenario.

The second step in the design process was to define the **consequences of authenticity**. Users need to understand the actions that follow authentication. Screen real-estate is a premium on smartphones, making text descriptions and instructions a luxury. Consistent interaction should ensure that the mere presentation of the authentication mechanism communicates what actions follow. Sky serve as a bad example, using a single PIN for (1) accessing age-rated content and for (2) purchasing premium content. Sky would argue the purpose is the same in both cases, confirming the user is the bill-payer. However, PIN entry actually has two possible consequences, leading to ambiguity and confusion. Users may accidentally purchase premium content while assuming they are accessing age-rated subscription content.

The consequences of authentication in the application had to be clear. Tetrad was coupled with a content store, consequences were therefore limited to either (1) payment or (2) access. The first approach is the one adopted by iTunes while the second is akin to Spotfiy and Netflix.

The latter approach was favoured, as the application comprised of a digital store and media-player. Purchased content would be kept within the application and individuals would use the application’s media-player to consume content. Tetrad regulated access to application usage.

This design was used for both prototypes. What separated the prototypes was the image-bootstrapping procedure used.

4. IMAGE BOOTSTRAPPING

Personal images were sourced from Facebook. The first step in both prototypes required users to connect their Facebook account with Tetrad and grant permission for the mechanism to access personal images of friends.

4.1. Jack — Face Recognition

The first challenge for the Jack prototype was to analyse the images. The iPhone operating system, at the time, did not provide support for detecting faces within images. The Open Source Computer Vision Library or OpenCV was used to process personal images on the iPhone. The cross-platform library has a vibrant community, providing documentation and support for any number of generic image processing algorithms, including detecting faces within images.

The library was bundled with the application. Once the user had connected Tetrad with Facebook, the mechanism submitted a query to the social network for the user’s friend list. The response to the query was used to download the profile picture of all the user’s friends. The image processing procedure was multi-threaded in an attempt to reduce the time taken to process the images. The application spawned multiple threads at once. An individual thread, dealt with a single friend. The thread downloaded a friend’s profile picture, analysed it for faces, returned an array of possible faces and extracted a single face from the image and stored it on the device. The process did not recognition faces, i.e. the face detected in the profile picture, is not necessarily of the profile owner.

The process simply selected the first element in the array and used the coordinate information to locate the face within the image and extract it. The thread completed by adding an entry to list, the entry included the friend name and the face extracted. The instructions requested users to select 45 friends. The table-view remained responsive while image-processing was happening. Therefore, users did not need to wait until all personal images were processed, they could select 45 images quickly and move-on.

The next-step requested users to select 4 of the 45 images, as their secret. The user simply scrolled through a table-view of the 45 images and tapped the entries they wanted to make their secret. The final step was to order the sequence of the 4 images. Users were instructed to memorise the extracted images and sequences. Once the user was content with the sequence, they could complete registration.

4.2. Jill — Tags

The second prototype took a different direction to image-bootstrapping, favouring a less computational expensive approach. Instead of downloading and analysing images to detect faces, the mechanism relied on 'tags'. Tags are used to identify friends within images on Facebook. The social network service provides a tagging tool, affording users the ability to drag a box around a friend's face within an image. Users can then easily see all the photographs that include a specific friend.

The Jill prototype utilised these tags, rather than attempting to analyse and detect faces itself. The registration process was compressed into a single-stage. The aim was to make the registration process feel zippier and more responsive, than earlier versions.

The user's friend-list was presented in table view, with a table entry including the friend's name and profile picture. Users were instructed to select 45 friends, with 4 of them making-up their secret. All a user had to do was simply tap a table-entry to select a friend. When a friend was selected, users had the option to add that friend to their secret. They also had the option to order the sequence of their secret, at the same time. When the user had selected 45 friends and made 4 of them their secret - a button was enabled that allowed them to complete the registration process.

5. DISCUSSION

Designing and developing prototypes of Tetrad that relied on personal images, only served to emphasises the difficulty in deploying the mechanism in the real world. The initial incarnation of the authentication approach may have showed promise but the practicality of both prototypes is questionable.

The authentication approach relied on faces. The prototypes extracted these faces from personal images, sourced from Facebook. Putting aside the actual usability of either prototype and merely focusing on the practicality, neither seemed worthy of further investigation. Bootstrapping either prototype with images was incredibly expensive in terms of time and resources, especially so when contrasted with tried and tested, and convenient, alphanumeric approach.

The prototypes extracted and used faces in different ways. The first prototype, titled 'Jack', actually detected faces within personal images and extracted them while the second prototype, titled 'Jill', used user-generated tags from Facebook to extract faces from personal images.

The image-bootstrapping procedure for Jack was incredibly expensive as images had to be analysed to extract faces. This consumes significant resources on the user's device and takes time. Alternatively, image processing of personal images could have occurred in the cloud instead of on the device. The cloud option is desirable as image processing and analysis is an intensive task. Amazon can provide powerful cloud computing instances that allow for multi-threaded image processing operations. The cloud computing instance could locate and extract faces within a few seconds, contrasted with several minutes on a mobile device. The extracted faces could be downloaded directly to the device and the instance could be discarded. The main concerns with the cloud computing approach is the bottleneck in downloading images, the cost associated with the operation and the privacy risk to the user. These concerns led us to favour the second option, processing the personal images on the user's device.

Meanwhile, the second prototype, Jill, avoided costly computation by using tags from Facebook to extract faces. Although the prototype avoids costly computation it still requires a complex registration process. The user needs to spend time selecting images for use within the mechanism and they have to wait for those images to download. Contrast this with alphanumeric authentication, where a user can quickly tap-out a text password in a few seconds.

Therefore, the real question becomes: How big a threat is shoulder-surfing? The threat needs to be fairly big to be worth the hassle of dealing with Tetrad, instead of alphanumerics. The prototypes forced thought on deployment issues, such as how the mechanism would be bootstrapped with personal images and what sort of application would it protect.

The original scenario we envisioned was a user purchasing a movie in-front of friends and family. The reality is, however, most users do not care if these people know the password they use to purchase movies. Users want to share a movie with these people in their living room, a far more valuable and intimate thing than a password. If a friend or family member does use someone's else password, then they will work it out. The same way they would, if they had the last beer or slice of pizza. Shoulder-surfing in this context is a non-issue.

Therefore, for the prototypes we envisioned the scenario of an individual accessing a digital store, was pushed out into the wider world. Purchasing a movie on a smartphone while having a coffee at Starbucks or waiting for train in a station. These places are bursting with strangers and a user would not want any of them to observe entry of their password. However, users are likely to authenticate on a 4" inch screen, not a 40" one. Therefore, they can position themselves and the screen to avoid observation by others.

Lets say a stranger does, somehow, manage to observe the password being entered in such a space. What are the actual consequences, in the case of a digital store? If an attacker does manage to purchase several items from a service, such as iTunes, Apple can deactivate the purchases.

Therefore, maybe the focus on digital stores was the wrong scenario. Tetrad could have been used to protect email or a financial service. However, no individual would check their personal email account or personal finance account, on a television in front of friends and family. Not necessarily because the information is sensitive but because it is boring and uninteresting. Large screens are for sharing and although some users might want to read email and check balances on a big-screen, they would wait until the television is free, rather than interrupting a movie.

Constructing prototypes, that are meant to be the manifestation of a practical, deployable product force thought on the actual application used, the people using it and the places where it will be used. Authentication is a package, it is not something that is spooned on-top of a finished product, it is part of the product. The time and energy taken to create a Tetrad secret is not necessary worth the hassle because shoulder-surfing is not really something to be concern about.

In the end we didn’t actually tackle an authentication problem; we tackled a password problem. Tetrad was not designed thinking about the users and the tasks they wanted to complete but in order to right the wrongs of passwords. Tetrad was burdened with the legacy of the password approach. The design process did not start with: people want to buy movies in the living room. Instead it started with: these are the problems with passwords in the living room.

6. CONCLUSION

You only live twice, or so it seems.
One life for yourself and one for your dreams.
You drift through the years and life seems tame
Till one dream appears and love is its name

Sung by Nancy Sinatra, the lyrics above are from the John Barry classic, ‘You only live twice’. They are inspired by the film and novel of the same name and the lyrics speak to the idea, that an individual ‘lives twice’. The life they led and the life they lead after facing death.

We thought we were testing Tetrad. The surprising result of the evaluation was that shielding secrets from surveillance, in this context, is misguided at best and pointless at worst. Shoulder-surfing is not a real threat to professionals or plebs, especially when one considers the kinds of applications that would use mechanisms like Tetrad.

The realisation from all our efforts is that shoulder-surfing is not a real problem for the authentication scenario we envisioned. Shoulder-surfing is a password problem. Solving the problems of passwords doth not a successful authentication mechanism make.

Authentication is a package. Instead authentication design should focus on solving an authentication scenario. The design process should start by considering every aspect of that package: people, places and purpose. Rather than starting with the problems of passwords. In the future, we will tackle an authentication problem rather than password problem.

7. REFERENCES

[1]Corbato, F.J. and Merwin-Daggett, M. and Daley, R.C. An Experimental Time-sharing System. In Proceedings of the Spring Joint Computer Conference, pages 335-344. ACM, May 01-03 1962.

[2]K. Renaud and J. Maguire. Armchair authentication. In BCS-HCI ’09: Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology. British Computer Society, Sept. 2009.

[3]S. Chiasson, A. Forget, R. Biddle, and P. van Oorschot. User interface design affects security: Patterns in click-based graphical passwords. International Journal of Information Security, 8(6):387-398, 2009.

[4]A. Paivio. Mental representations: A dual coding approach. Oxford University Press, USA, 1990.

[5]D. Nelson, V. Reed, and J. Walling. Pictorial superiority effect. Journal of Experimental Psychology: Human Learning and Memory, 2(5):523, 1976.

[6]R. Shepard. Recognition Memory for Words, Sentences, and Pictures. Journal of Verbal Learning and Verbal Behavior, 6(1):156-163, 1967.

[7]P. Dunphy, A. Heiner, and N. Asokan. A Closer Look at Recognition-based Graphical Passwords on Mobile Devices. In Proceedings of the 6th Symposium on Usable Privacy and Security. ACM, 2010.

[8]B. Riddle, M. Miron, and J. Semo. Passwords in use in a university timesharing environment. Computers and Security, 8(7):569-578, 1989.

[9]T. Tullis and D. Tedesco. Using Personal Photos as Pictorial Passwords. In CHI’05 Extended Abstracts on Human Factors in Computing Systems, pages 1841-1844. ACM, 2005.