SAVVIcode: Preventing Mafia Attacks on Visual Code Authentication Schemes (Short Paper)

Jonathan Millican and Frank Stajano
jrm209@cantab.net, frank.stajano@cl.cam.ac.uk

University of Cambridge Computer Laboratory, Cambridge, UK

Abstract. Most visual code authentication schemes in the literature have been shown to be vulnerable to relay attacks: the attacker logs into the victim's "account A" using credentials that the victim provides with the intent of logging into "account B". Visual codes are not human-readable and therefore the victim cannot distinguish between the codes for A and B; on the other hand, codes must be machine-readable in order to automate the login process. We introduce a new type of visual code, the SAVVIcode, that contains an integrity-validated human-readable bitmap. With SAVVIcode, attackers have a harder time swapping visual codes surreptitiously because the integrity check prevents them from modifying or hiding the human-readable distinguisher.

1 Introduction

A current area of research in the field of authentication is *visual code authentication schemes* (VCASs). In such schemes, the user logs into a remote system by acquiring a visual code with a scanning device rather than (or, sometimes, in addition to) typing a password. Examples of such systems include Snap2Pass [5], our own Pico [9], tiQR [10], QRAuth [1] and SQRL [6], as well as patents filed by GMV solutions [2] and Google [4]. The "visual code" is a kind of two-dimensional barcode — often the QR-code that modern smartphones are already equipped to decode. Indeed, in many of the authentication schemes above, the device that scans the visual code is the user's smartphone.

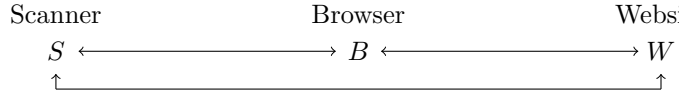
Jenkinson et al [7] have shown that, without an additional out-of-band secure communication channel from the device to the terminal, these schemes are inherently vulnerable to "Mafia fraud relay attacks" when they use existing types of visual code.

We propose an extension to visual codes that prevents an attacker from undetectably substituting one visual code for another. By making any attacks obvious to the user, we hope this mechanism will greatly reduce the likelihood that such attacks can succeed.

Authors' preprint of 2015-07-06.
Final version to appear in *Proceedings of Passwords 2014*, © Springer LNCS.

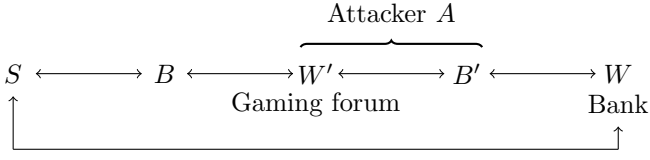
2 The Problem: Mafia Fraud Relay Attacks

Jenkinson et al [7] outline the typical operation of a VCAS, which we summarize as the following exchange of messages between Website (W), Browser (B) and Scanning device (S) owned by user (U), using session identifying nonce $n_{recipient}$:



1. $W \longrightarrow B : W, n_B$
2. $B \longrightarrow S : W, n_B$ (visual channel)
3. $S \longleftrightarrow W$: authentication
4. $S \longrightarrow W : n_B, U$
5. $B \longrightarrow W : n_B$
6. $W \longrightarrow B$: authenticates as U

A mafia attack, first described as such by Desmedt et al [3], introduces a two-faced man-in-the-middle attacker A. With one face, A masquerades as a website W' to browser B and its user U, while with the other it masquerades as a browser B' to the original website W. In the example of Jenkinson et al [7], website W is a bank where user U has an account, whereas the attacker-controlled website W' is an online gaming forum where U has another account. The attacker A thus tricks the victim U into believing they are logging into forum W', whereas in fact they are giving A the credentials to log into bank W.



1. $W \longrightarrow A : W, n_A$
2. $A \longrightarrow B : W, n_A$
3. $B \longrightarrow S : W, n_A$ (visual channel)
4. $S \longleftrightarrow W$: authentication
5. $S \longrightarrow W : n_A, U$
6. $A \longrightarrow W : n_A$
7. $W \longrightarrow A$: authenticates as U

Note in step 3 that the browser B need not be aware that it is transmitting information to the scanning device S, as it may simply show a pre-rendered image of a visual code, so cannot detect that it is allowing authentication to a different service.

The practical implication of this attack is that, if a scanning authenticating device S holds credentials for multiple services, one of which an attacker can

successfully impersonate or modify, then the attacker could cause S to authenticate the attacker's browser B' to a high-value service W, while the user imagines themself to be authenticating to a lower-value service W'. This is similar to a phishing attack, but differs in that the user cannot distinguish between being asked to authenticate to the bank or to the gaming forum, because visual codes are *not* human-readable and all look alike to users.

In theory, this attack should only apply to visual codes: users would not type their banking password into their gaming forum login page because the contexts do not match. But in practice the attack would also work, without visual codes, against victims who reuse passwords across high- and low-value web sites.

Our proposal is to bind some context to the visual code, thus alerting users when an authentication visual code is being presented in the wrong context.

3 Our Solution: the SAVVIcode

The mafia attack is possible when a visual code does not have to match its context. If the scanner could recognise when a code is displayed in the wrong context, it could stop this attack. In the general case, this is a hard problem involving computer vision and contextual knowledge. We simplify it by observing that:

1. A scanner that can read a visual code must inherently be able to read "pixels" of a given size, as it must read the matrix of the visual code itself.
2. As in a visual code, a pixel matrix can be serialised into a string of bits.
3. Even if text is pixellated and blocky, humans can very easily read it – and will do so effortlessly.
4. It is difficult for a person to scan a visual code without looking at it, therefore any obvious text alongside the code will most likely be read by the user.

These observations suggest a solution initially outlined by the first author [8] whereby an image is placed above the visual code at the same block-resolution. It should appear as text or an image identifying the service requiring authentication.

At this point we assume that the remote service has an "identity keypair" which it uses to prove its identity — either through a Public Key Infrastructure, or without one as in Pico. This key can be used to sign the serialised image and the payload of the visual code. This signature should be included in the visual code, allowing a scanner to determine whether the scanned image matches the one intended for display alongside the visual code. If we assume that a hacker cannot, in reasonable time, forge a valid signature, then, under this scheme, *a valid visual code cannot be displayed without revealing the image that its creator wanted the user to see.* This ensures that a mafia attack is easily evident to the user before it occurs. We name the image described a *Serialised And Visually Verified Image Code* or SAVVIcode, and suggest that it should be hyphenated with the visual code type used, e.g. QR-SAVVIcode when used with a QR code.

It is worth noting that multiple systems could very easily choose to display the same image at the point of logging in. This is, however, no risk to the scheme as their authentication credentials would be different, and thus the authentication device would not authenticate to one as if it were the other. In other words, the malicious online forum operator may well say "I'm your bank" in the picture of his SAVVIcode (but that is not going to entice the user to log in to the forum); what he cannot do is to modify *the bank*'s SAVVIcode to say "I'm the online forum", because then the bank's signature would not verify.

Figure 1 exemplifies how a QR-SAVVIcode might look. It consists of a QR code containing a payload, a public key and a signature, and an image above containing text and a number of patterns that may be used to detect the bounds. As the QR code itself contains finding patterns which will align the reader, only rudimentary additions should be required on the SAVVIcode.

Fig. 1. How a SAVVIcode might look

4 Challenges and Future Work

Many visual codes, including at least QR-code, Aztec, Data Matrix and Maxicode, use Reed-Solomon error correction to compensate for inaccuracies in the decoded bit-stream,[1] thus allowing a significantly less than perfect scan to recover the original contents. For a SAVVIcode to be effective, however, we do not

[1] Another mechanism used to improve the scanning accuracy of visual codes is to use encodings that break up large contiguous blocks of the same colour. The SAVVIcode does not use this method for resynchronisation because we want the bitmap to be immediately readable, even without meaning to, by the person scanning the code. We thus want a high-contrast bitmap in which the black ink stands out against a background of white space.

want error correction to allow an attacker to pass a modified code for a genuine one, so we must be careful.

To clarify, the attack would consist of taking the bank's SAVVIcode, whose human-readable text says "I'm the bank", then modifying that text to say "I'm the online forum" but with sufficiently few pixel changes that the error correction would still accept it as a slightly noisy version of "I'm the bank", thus allowing the signature to verify. The attacker would then have succeeded in persuading the user that they are logging in to the online forum (because the text says so and the digital signature verifies successfully) while instead they would be logging the attacker into the bank. We must absolutely prevent this.

To implement a SAVVIcode, we propose two approaches, whose trade-offs should be evaluated and compared after writing prototype implementations. In both cases a free-form bitmap is drawn in an area above the main visual code, within a border with reference markers, as in the previous picture. The recognition software that turns the main visual code image into a boolean matrix is modified to recognize and digitize this extended region as well.

In the first approach, the free-form bitmap is also compressed and appended to the original payload; the whole payload is then digitally signed before being encoded in the main visual code. The SAVVIcode thus contains two versions of the bitmap: one that is also human-readable but whose integrity is not digitally protected, and one embedded in the code and digitally signed. The recognition software must compare the two versions and the challenge is to ensure that all fraudulent modifications of the human-readable version are detected by the recognizer, while allowing for some quantity of non-malicious bit errors.

In the second approach, a detached error detection and correction (EDC) code for the free-form bitmap is generated. The concatenation of the serialized bitmap, the detached EDC code and the original payload is digitally signed. The concatenation of the detached EDC code, the original payload and the digital signature (but not the serialized bitmap) is then encoded in the main visual code. The recognition software thus extracts the bitmap, the EDC, the original payload and the signature; then applies the EDC to the bitmap to correct errors; then verifies the signature. This approach seems at first more robust, and carries only one copy of the bitmap, but it is still vulnerable to the same attack. A malicious adversary could subtly change the bitmap in a way that tricked the human viewer; but then the EDC would "undo" those changes and allow the verification to pass, yielding a false accept and thus a successful attack.

Fine-tuning the false accept (fraud) rate against the false reject (failure to admit honest customers) rate is, as ever, going to be the crucial security trade-off. The two approaches must be evaluated experimentally against malicious alterations to determine which one offers the best characteristics.

Although the SAVVIcode makes the visual code partially human-readable, a structural limitation of our strategy is its reliance on the alertness of the user. A "rushing user" (the kind who wants to get on with their real work and presses the OK button no matter what the annoying dialog box says) may well scan the code without paying any attention to the bitmap. We recognize this

limit. The SAVVIcode is not an absolute defence against the man-in-the-middle attack on visual code authentication, but we believe it is still an improvement on the status quo: even though users may not pay attention to the bitmap, it is cognitively difficult to scan the visual code without reading it (as in "Do not think of an elephant!"). The SAVVIcode thus makes it difficult to mount a man-in-the-middle attack without someone noticing.

5 Conclusions

Most visual code authentication systems (Pico excepted [7]) are vulnerable to middleperson attacks. The SAVVIcode adds an integrity-protected human-readable bitmap to the visual code. It is hard for a person to scan a SAVVIcode without reading this bitmap and thus, while the scheme relies on the user's cooperation, such cooperation happens almost automatically.

In this paper we have not laid out a specific protocol in which SAVVIcode should be used, as we do not wish to limit its potential uses. Further research is required for implementation details of the technique to be defined and, when this has occurred, new protocols using SAVVIcode can be drafted.

While the protection offered by the SAVVIcode does not claim to be absolute, it might prove remarkably effective compared to its modest cost of deployment. We believe its adoption would greatly reduce the impact of the inherent vulnerability to Mafia relay attacks that affects most existing visual schemes.

Acknowledgements

We are grateful to the Pico team for their feedback and to Andy Rice for helpful discussions on visual code scanning technology.

The Pico team is also working on an alternative "augmented reality" approach in which the human-readable tag is displayed by the scanner rather than being shown alongside the visual tag.

The second author is partly supported by European Research Council grant 307224 (Pico).

References

1. L. Batyuk, S.A. Camtepe, and S. Albayrak. Multi-device key management using visual side channels in pervasive computing environments. In *Proc. BWCCA 2011*, pages 207–214, Oct 2011.
2. J.J.L. Cobos and P.C. De La Hoz. Method and system for authenticating a user by means of a mobile device, September 4 2012. US Patent 8,261,089.
3. Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In *Proc. CRYPTO '87*, LNCS. Springer, 1987.
4. D.B. DeSoto and M.A. Peskin. Login using QR code, August 22 2013. US Patent App. 13/768,336.

5. Ben Dodson, Debangsu Sengupta, Dan Boneh, and Monica S. Lam. Secure, consumer-friendly web authentication and payments with a phone. In *Proc. Conf. on Mobile Computing, Applications, and Services (MobiCASE10)*, 2010.
6. Steve Gibson. Secure quick reliable login. `https://www.grc.com/sqrl/sqrl.htm`.
7. Graeme Jenkinson, Max Spencer, Chris Warrington, and Frank Stajano. I bought a new security token and all I got was this lousy phish—Relay attacks on visual code authentication systems. In *Proceedings of Security Protocols Workshop 2014*, LNCS. Springer, 2014.
8. Jonathan Millican. Implementing Pico Authentication for Linux. Undergraduate final year dissertation, May 2014.
9. Frank Stajano. Pico: No more passwords! In *Proc. Security Protocols Workshop 2011*, LNCS, pages 49–81. Springer, 2011.
10. Roland M. Van Rijswijk and Joost Van Dijk. Tiqr: A novel take on two-factor authentication. In *Proc. LISA'11*, pages 7–7. USENIX Association, 2011.