I bought a new security token and all I got was this lousy phish— Relay attacks on visual code authentication schemes

Graeme Jenkinson, Max Spencer, Chris Warrington, Frank Stajano
{graeme.jenkinson, max.spencer, chris.warrington, frank.stajano}@cl.cam.ac.uk

University of Cambridge Computer Laboratory, Cambridge, UK

Abstract. One recent thread of academic and commercial research into web authentication has focused on schemes where users scan a visual code with their smartphone, which is a convenient alternative to password-based login. We find that many schemes in the literature (including, previously, our own) are, unfortunately, vulnerable to relay attacks. We explain the inherent reasons for this vulnerability and offer an architectural fix, evaluating its trade-offs and discussing why it has never been proposed by other authors.

1 Introduction

We consider a relatively new class of web authentication schemes, currently attracting significant academic and commercial interest, which we refer to as *visual code authentication schemes*. A user may log into a website which supports such an authentication scheme by scanning a visual code, such as a Quick Response (QR) code [1], using their hand-held authenticator device, henceforth *scanner*. The scanner is generally a smartphone, but might be a dedicated hardware gadget. The user carries their scanner at all times, or at least whenever they might want to authenticate to a website; the scanner may have a mechanism to prevent its misuse if lost or stolen. Our own Pico system [2] is of course in this class too.

Such schemes are interesting because they have some important usability benefits which passwords do not; specifically, there is nothing for users to remember or type[1]. Furthermore these schemes are resilient to conventional phishing[2] because the long-term secrets never leave the scanner and so an attacker cannot trick the victim into revealing them. However, visual code authentication schemes present a new risk. Because the information in a visual code is not human-readable, and visual codes are easily relayed, a user may be tricked into scanning a visual code displayed outside its intended context.

[1] Cfr. definitions of *Memoryless*, *Scalable-for-Users* and *Nothing-to-Type* in the Usability, Deployability and Security (UDS) framework of Bonneau *et al.* [3].

[2] As defined in the UDS framework [3]

We have surveyed a range of schemes currently in the literature, some commercial [4,5] and some academic [2,6,7]. We find there are significant structural similarities between the schemes and that they all are[3] susceptible to attacks in which the victim inadvertently authenticates a session for the attacker.

This paper makes the following contributions:

– We show that the architecture of many visual authentication schemes currently in the literature leaves them inherently vulnerable to attacks that relay the visual code, allowing an attacker to gain control of sessions authenticated by other users.
– We present our proposed solution, *session delegation*, now adopted by Pico, which uses an additional communication channel to prevent the aforementioned relay attacks, and discuss why no other scheme has adopted anything similar.
– We discuss extensions to our session delegation protocol and some alternative means of mitigating these attacks, while considering their impact on the usability, deployability and security of the system.

2 Visual code authentication schemes

All of the schemes surveyed offer a similar user experience and share some crucial architectural features. In this section we describe these commonalities and then review the individual schemes in more detail.

2.1 User experience when authenticating

When a user visits a website in their web browser, the login page includes a visual code, possibly alongside a traditional username and password login form. In order to login, the user scans the code with their scanner, which authenticates to the website identified by the visual code. After the scanner has authenticated on the user's behalf, the web browser receives a *session cookie* which grants access to the user's account through their browser.

The scanner is responsible for the generation and retention of all keys and secrets. As a result, using the scanner with arbitrarily many accounts and services does not require additional cognitive (and in particular memory) effort on the part of the user.

2.2 Protocol

Crucially, these authentication schemes must also be able to authorise a web browser running on a different host than the scanner. We call this process *browser authorisation.* Without this ability users would be restricted to logging into, and using, websites only on their scanner. But it is this crucial feature, and the

[3] Or were, in the case of Pico.

common approach taken by the surveyed schemes to provide it, that leads to the security vulnerabilities we describe in section 3 below.

The schemes surveyed use differing protocols for the actual authentication between the scanner and the website. However, we found extensive similarities between the mechanisms used to perform the browser authorisation and, below, we show a generalised version of the overall protocol, to which the concrete implementations can all broadly be mapped[4]. Fig. 1 shows the sequence of messages sent between the user's web browser, B, their scanner, S, and the website, W.

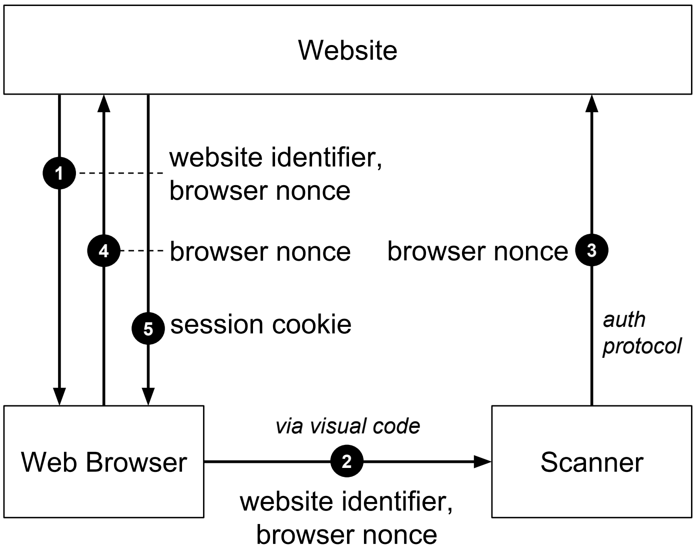

Fig. 1. Generalisation of the flawed browser authorisation protocol used by the reviewed visual code authentication schemes. The figure shows how a session cookie, c_U, for user U is installed into their browser B.

1. The user navigates to the login page of a website W in their web browser B. The login page returned by the website includes a visual code containing the website's address (or identifier) W and a fresh *browser nonce*, n_B. Note that the request and response are performed over HTTPS and therefore encrypted under a TLS session key, K_{BW}.

$$W \rightarrow B : \quad \{W, n_B\}_{K_{BW}}$$

[4] At least as far as we can infer—some schemes have not openly published a complete specification.

2. The website address W and browser nonce n_B are transfered to the scanner, S, when the user scans the visual code on the website's login page.

$$B \to S : \quad W, n_B \quad \text{(visual channel)}$$

3. The scanner authenticates to the website using a scheme-specific authentication protocol. The website looks up the user account, U, associated with the identity authenticated by the scanner. Subsequently, or as part of the authentication protocol itself, the scanner securely sends the browser nonce to the website.

$$S \to W : \quad \{n_B\}_{K_{SW}}$$

The website is then able to associate the nonce n_B with the account U.

4. The browser makes another request to the website, again over HTTPS, which includes the browser nonce n_B, with the intention of "trading in" this nonce for a session cookie.

$$B \to W : \quad \{n_B\}_{K_{BW}}$$

5. After looking up the user account U associated with n_B, the website W returns a session cookie c_U. This cookie grants access to user U's account, as if the user had authenticated using a typical username-and-password-based scheme.

$$W \to B : \quad \{c_U\}_{K_{BW}}$$

If the browser had sent its second request (4) before the scanner had validated the nonce in (3) with the authentication protocol, the website would not have granted a session cookie in this step.

2.3 Schemes in the class

Method and System for Authenticating a User by Means of a Mobile Device (2009) This patent [4], held by GMV Soluciones Globales Internet S.A., describes a visual code authentication scheme in which users authenticate to remote services with a trusted application running on a mobile phone.

In this scheme the user selects the service to authenticate to by acquiring a visual code displayed by an untrusted device. The visual code contains both a random challenge and an identifier of the service to authenticate to. On scanning the visual code the mobile device creates a response to the challenge by signing it with a private key. The scheme uses Identity Based Encryption (IBE) allowing the response to be verified using the user's public identity such as an email address.

In common with the other schemes described here, the scheme links the login session on the untrusted device with the scanner's response using the random challenge contained in the visual code. Thus, this random challenge is directly equivalent to the browser nonce shown in Fig. 1.

Snap2Pass Snap2Pass [8] is a visual code authentication system for web applications in which users authenticate using a smartphone application.

In this scheme the user selects the service to authenticate to by acquiring a visual code displayed by an untrusted device. The visual code contains both a random challenge and an identifier of the service (the relying party) to authenticate to. On scanning the visual code, the mobile device creates a response to the challenge comprising of the HMAC-SHA1 hash of the entire challenge message using a pre-shared secret as a key. The provider verifies this responds and, if successful, the browser session is authenticated with the appropriate account.

The challenge nonce (sometimes referred to as a session key) in the visual code is directly equivalent to the browser nonce. Snap2Pass explicitly acknowledges that it does not mitigate active man in the middle attacks such as those discussed in this paper.

Pico (2011) Stajano's Pico [2] is a visual code authentication scheme intended for a dedicated hardware device, although it could also be implemented, trading off security for convenience, as an application running on a smartphone[5]. Pico also includes a novel locking mechanism dependent on the proximity of a number of other, smaller devices referred to as *Picosiblings*, as well as on proximity detection between the user's Pico and their web browser (or rather the computer it is running on), allowing the user's session to lock automatically when they are away.

A smartphone-based research prototype created by Tian [] under Stajano's supervision . Another smartphone-based prototype independently developed by Fu [9] includes a *sessionID* in the visual code that is directly equivalent to the browser nonce above.

tiQR (2011) van Rijswijk's tiQR [6] is a prototype smartphone-based visual code authentication system. The scheme uses the OAuth Challenge Response Authentication (OCRA) [10] protocol to authenticate the user to services. The user has a four-digit PIN, in addition to a secret held by the phone, for logging in to each of their accounts.

In this scheme the visual code contains a random challenge that is directly equivalent to the browser nonce.

Login Using QR Code (2012) This patent [5], held by eBay Inc., describes an authentication scheme that uses a visual code authenticator to broker secure log-ins to websites from devices that may be insecure. The scheme uses a trusted application, running on the mobile device to authenticate to a single third-party Identity Provider (IdP).

The contents of the visual code displayed on the untrusted device are passed to the IdP by the authenticator. The contents of the visual code are encrypted

[5] As already envisaged in the original paper [2] as well as in our other paper in these proceedings, "Bootstrapping adoption of the Pico password replacement scheme".

and can only be read by the IdP. Upon validating the contents of the visual code, the IdP issues a challenge to the trusted device, such as requesting a password. Once the user has authenticated, the IdP informs the relying web service which maintains an association between the QR code contents to active login sessions. The relying web service then updates the status of that login session.

The information contained in the visual code is sent from the IdP to the website to identify the authenticated web session. Although the description given in the patent [5] is a bit vague and obscure, it seems reasonable to assume that this data is somewhat analogous to the browser nonce.

QRAuth (2013) Howard's QRAuth[6] [7] is a research prototype visual code authentication system with significant similarities to Pico. The authenticator has a shared secret for each service, unlike Pico which uses asymmetric cryptography and QRAuth uses a mobile application rather than a dedicated hardware device as the visual code authenticator.

In this scheme the *login identifier* is directly analogous to the browser nonce.

Secure Quick Reliable Login (2013) Another recent smart-phone-based scheme is Secure Quick Reliable Login (SQRL), proposed by Gibson [12]. Visual codes used in the SQRL scheme contain a URL which includes a *session id* and points to an authentication service. The SQRL app signs this URL and then sends the signature to the authentication service over HTTPS. It uses a different public-private key pair for each service but, unusually, these key pairs are not stored, but are derived from a master secret and master password when needed. The system specifies a revocation protocol to be used when a SQRL device is lost or stolen.

The *session id* contained in the URL in the visual code is directly equivalent to the browser nonce.

3 Attacks

3.1 Core vulnerability

Visual codes are not human readable; so, whilst acquiring a visual code reflects the user's intent to authenticate, it is unclear to the user what they are authenticating *to*, or whether the information in the visual code is fresh. Although the visual channel itself can reasonably be assumed to be unmodifiable, the user's web browser is not a trusted display. Specifically it does not prevent *relayed* visual codes from being displayed.

The attacks we describe below all exploit the same core vulnerability. In all cases the attacker (who uses browser B') seeks to obtain a cookie, c_U, which will give them access to victim U's account for a given website W.

[6] There is also a commercial mobile application [11] of the same name, but it is equivalent to a password wallet and bears only a superficial resemblance to the other schemes discussed here.

For each attack, the attacker makes a request to W and gets back a visual code containing $W, n_{B'}$ (see step 1 in the description of the protocol above).

$$W \to B' : \quad \{W, n_{B'}\}_{K_{B'W}}$$

The attacker then relays this visual code and convinces the victim to scan it, thereby causing the user's scanner to authenticate to W and link U's account with $n_{B'}$ (see steps 2 and 3 above). Note that the relayed channel may or may not be re-encrypted, depending on the mode of the attack.

$$B' \to B : \quad W, n_{B'} \quad \text{(relay)}$$

$$B \to W : \quad W, n_{B'} \quad \text{(visual channel)}$$

Finally the attacker's browser can send $n_{B'}$ back to W, trading it in for the session cookie c_U they want (see steps 4 and 5 above).

$$B' \to W : \quad \{n_{B'}\}_{K_{B'W}}$$

$$W \to B' : \quad \{c_U\}_{K_{B'W}}$$

The details of how an attacker might relay a visual code and convince a user to scan it with their visual code scanner device are given below. None of these attacks involve the attacker modifying the contents of any visual code[7], only relaying them to trick victims into authenticating sessions they did not intend to. We show how two well-known types of attacks, phishing and mafia fraud, are even more insidious when applied to visual code authentication schemes.

Perhaps surprisingly, proponents of several of the schemes surveyed claim that resilience to phishing is one of their key security benefits; moreover, the Usability-Deployability-Security evaluation framework [3] for web authentication schemes does not penalize schemes that are only vulnerable to more elaborate real-time man-in-the-middle or relay attacks (*cfr.* the definition of its *Resilient-to-Phishing* benefit). However, while this definition is appropriate for the schemes presented in Bonneau *et al.*'s evaluation, it fails to tell the whole story for visual code authentication schemes.

3.2 Phishing with visual codes

In a traditional phishing attack, the victim unwittingly divulges their password to an attacker, who pretends to be or represent a website the victim trusts. The use of a scanner appears to offer some protection against phishing because the secrets used to authenticate the user are contained within the scanner and are unknown to the user and, depending on the specific authentication protocol, might even never leave the scanner. While these secrets can be revealed by physically compromising the device, this is an altogether different type of attack which doesn't scale. An attacker can only physically compromise a single device

[7] It would still be prudent to sign the contents of visual codes to prevent such attacks.

at a time, with significant effort, rather than attack many of them in parallel over the net.

However, an attacker is able to convince the victim to *use* their scanner and an attacker is able to relay a specific visual code over various communications channels, including email. For example, the attacker could send an email to the victim, purporting to be from their bank, claiming they need to scan a code to "validate" their scanner. When the victim does so, they authenticate the nonce in the visual code which the attacker knows and relayed (see n'_B above), and the attacker can now trade this nonce in for the user's session cookie. (We might view this as an instance of the "chosen protocol attack" [13].)

A visual code phishing email may come with the usual carrots ("you will be entered into a prize draw") and sticks ("your account may be locked") to persuade the victim to comply. However, there are several reasons why it would be more difficult for a user to spot a visual code phishing attempt, making the new attack more insidious. In a traditional phishing attack, the victim must either reply to the attacker's email, in which case the attacker must disclose an email address they control; or the victim must enter their password into a form on a fake version of the trusted website, which the attacker must provide. With a visual code phishing attack, neither is required of the attacker: the victim can scan the visual code right in their email client, thus contacting the legitimate website directly, and needn't reply to the attacker in any other way. By the same token, no "suspicious address" (email or web) will be found in the email that an alert user could spot to detect the fraud.

Furthermore, it is important to see that the victim's scanner also does not contact any server controlled by the attacker; the scanner really does authenticate to the website the phisherman is impersonating. If the scanner prompted the user for confirmation before each authentication, it would still not defend against this type of attack because the website identified by the visual code and contacted by the scanner for authentication is "correct"; the victim *wants* to authenticate to it. Any existing training the user may have received, to check for the right website address or the HTTPS padlock, is useless here, even if followed to the letter.

In light of this attack, the only advice that users could be given is that they should never scan a visual code contained in an out-of-band communication, such as an email, and they should only scan visual codes found on websites that they trust. But it is well known that reliably authenticating the website to the user is a hard, unsolved problem.

Besides, users may be accustomed to scanning visual codes with their smartphone for other purposes than authentication; therefore they are unlikely to appreciate the difference between scanning an authentication visual code and a visual code on an advert. Furthermore, there are additional attacks if users trust websites which are not trustworthy and then do not pay full attention to all confirmation messages the scanner might present to them.

3.3 Mafia fraud with visual codes

A "mafia fraud" relay attack against a visual code authentication systems results from users trusting an untrustworthy (mafia-operated, in the canonical example) website. The mafia fraud, as first described by Desmedt *et al.* [14], is a type of man-in-the-middle attack in which a challenge from the verifier is relayed without modification, in real time, to an honest prover. The man-in-the-middle then transmits the honest prover's response back to the verifier as shown in Fig. 2.

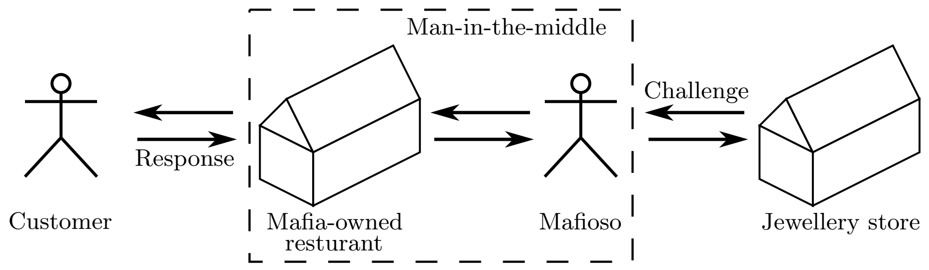

Fig. 2. Anatomy of a mafia fraud. The honest customer thinks they are paying for their meal, but is actually being tricked by the mafia into buying them some jewelry.

A mafia fraud with a visual code authentication system is slightly different because the response of the user, or rather the user's scanner, goes directly to the verifier, but the structure of the attack is otherwise the same.

As an example, consider a discussion forum website. This is a low-value site that the user trusts sufficiently to read discussion threads and sometimes post comments. If the user logged in to the forum with a username and password it would be difficult for the malicious site operator to trick the user into authenticating to the forum using their credentials for another high-value website, such as their online banking website[8]. However, with a visual code authentication system, only the non-human-readable visual code tells the scanner which website to authenticate to. The user may not detect the substitution, by the malicious operator, of a visual code for the forum with one from their online banking website.

If the consequences of such an attack were simply that the victim authenticates to a different website to the one they intended, then the advantage that a malicious actor gains is modest. However, as with the phishing attack above, the attacker can record the browser nonce, n_B, contained in the visual code, before relaying the code to the victim U and later trade this nonce in to obtain a session cookie c_U granting them access to the victim's account. Fig. 3 shows the sequence of messages sent when such a mafia fraud attack is carried out.

[8] Unless of course the victim uses the same password on every site.

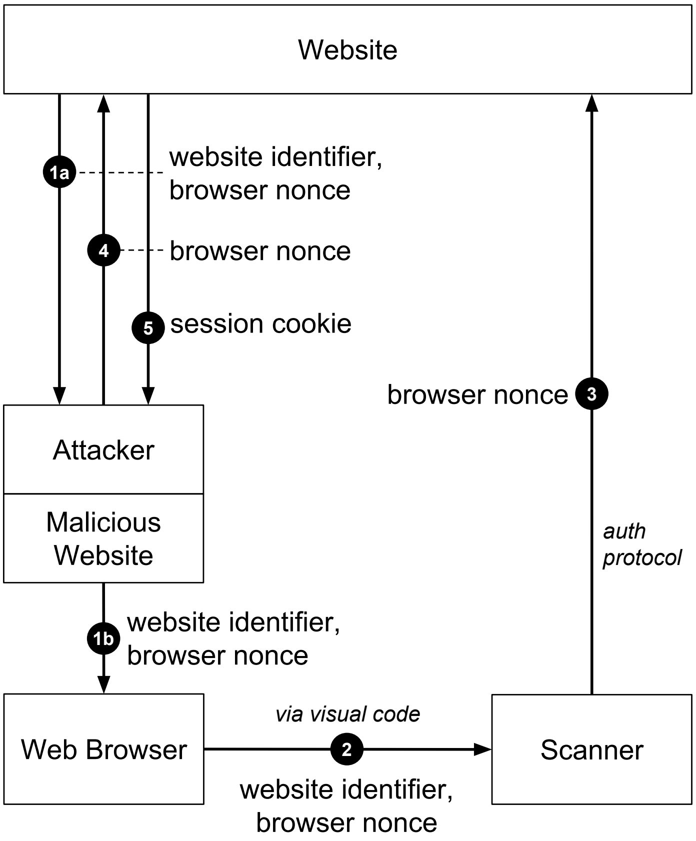

Fig. 3. Relay attack on the flawed browser authorisation protocol shown in Fig. 1. Figure shows how the attacker obtains a session cookie, c_U, for user U using a malicious website they control.

Any attacker could start up their own malicious website to perform this kind of attack, or they could hijack another website with existing users. In either case they could try to avoid detection by launching the attack only a fraction of the time, so that users would assume any discrepancy was the fault of their scanner rather than the malicious website.

The key difference between this mafia fraud attack and the previous phishing attack is that the user is not tricked into thinking that the attacker represents someone else. This means the user has an opportunity to spot that something is wrong if their scanner asks them to confirm the authentication and tells them the service that the visual code identified. However we do not think highly of protection techniques that dump back on the user the actual onus of checking. Users are conditioned by false alarms to accept or override such warning messages indiscriminately. Furthermore, because the user wants to log into the malicious website, which they trust and which they may have logged into successfully many times before, they are unlikely to be looking out for any discrepancies.

4 Solutions

4.1 Session delegation

We call "session delegation" our proposed solution to prevent these types of attacks. Instead of having the website W initially send the browser B a browser nonce n_B, which the browser can later "trade in" for the session cookie c_U after the authentication has linked n_B with a user's account U, we propose passing c_U from the website to the browser via the trusted scanner. In order to do that we need a new channel from the scanner to the browser and furthermore we propose this new channel be authenticated and encrypted, so that a scanner may only delegate to a browser with which it has previously established a trusted pairing.

A visual code authentication scheme that requires this new channel with these constraints suffers from reduced deployability, which may be why the schemes surveyed do not do so. In our ongoing work to improve deployability of this solution we are developing a *rendezvous point*. Provided that browser and scanner have an Internet connection, the rendezvous point allows them to communicate even when their net connection is heavily restricted by NATs and firewalls. We also present a fallback mechanism for the protocol so users can still log in when the browser they are using cannot be modified to carry out the cryptographic pairing procedure.

First we describe the session delegation protocol in more detail. Fig. 4 shows the sequence of messages sent when the session delegation protocol is used.

1. The user navigates to the login page of a website W in their browser B. The login page returned by the website includes a visual code containing the websites address (or identifier) W, but now no browser nonce.

$$W \to B : \quad W$$

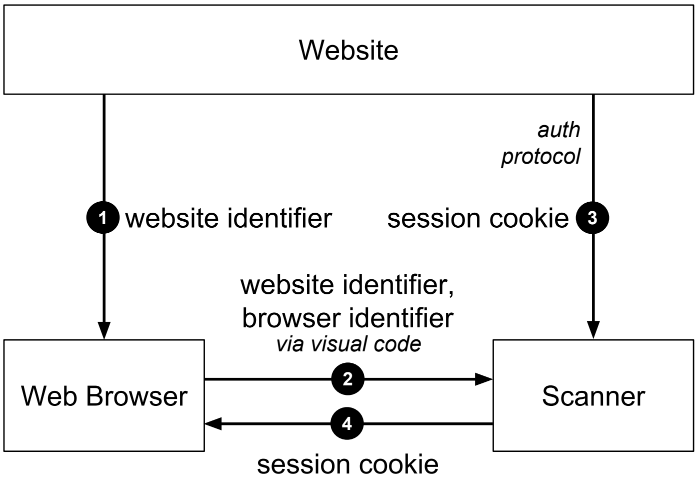

Fig. 4. Our proposed browser authorisation protocol: session delegation. Figure shows how a session cookie, c_U, for user U is installed into their browser.

2. The website address W is transferred to the scanner S when the user scans the visual code on the websites login page:

$$B \to S : \quad W, B \quad \text{(visual channel)}$$

3. The scanner authenticates to the website, W. There is no longer a nonce to send at this stage. The website looks up the user account, U, associated with the identity authenticated by the scanner. The website creates the session cookie c_U and returns to the scanner:

$$W \to S : \quad \{c_U\}_{K_{SW}}$$

4. Via a new authenticated and encrypted channel, the session cookie c_U is transferred to the browser. c_U grants access to user Us account through the browser as previously.

$$A \to B : \quad \{c_U\}_{K_{AB}} \quad \text{(new channel)}$$

The new channel Our session delegation protocol shown in Figure 4 imposes two requirements: first, there must exist a new channel from scanner to browser[9] in order to transmit message 4. Second, this channel must be authenticated and encrypted.

We have built prototypes using two different types of channel: one using a local Bluetooth link and another using the Internet. The former does not

[9] The existing visual channel from browser to scanner is of course unsuitable because it is unidirectional in the wrong direction.

require the scanner to have its own Internet connection, reducing its hardware requirements, but it imposes requirements on the hardware of the host on which the web browser is running. For the second type, the Internet-based channel, we implemented a HTTP-based rendezvous point in the public Internet. In an ideal world, the browser would simply put the IP address of its host into the visual code and the scanner could connect to that, but this is not possible for all browser-scanner pairs due to NATs and firewalls.

To use our rendezvous point, the web browser first makes request for a "channel" and the server responds with a URL of the form:

```
http://rendezvous.example.com/channel/<channel-uuid>
```

The browser includes this URL in the visual code, and the browser and scanner may subsequently write to this channel by making HTTP POST requests and read from it by making GET requests.

We suggest that this new channel should be authenticated and encrypted so that a cookie sent over it cannot be eavesdropped and an attacker cannot have a scanner return a cookie to *their* web browser, B', simply by getting a user to scan a code containing B''s identity and address. In other words, the scanner must only send cookies to those browsers which can prove ownership of a private key corresponding to an identity the scanner trusts. Our suggestion is that before step 4 above, the authenticator and the browser carry out a mutual authentication protocol, such as the SIGMA protocol [15], which has the side-effect of generating a session key, K_{AB} thereby providing the authentication and encryption simultaneously.

For the trusted browser to authenticate some identity to the scanner they must have previously "paired". This pairing could be done through a menu in the browser which causes the browser's full public key to be displayed to the scanner in a visual code.

Increasing deployability: fallback mode Unfortunately, if the new channel is to be authenticated and encrypted, the user's web browser requires modification, harming the deployability of the system by removing the *Browser-Compatible* UDS benefit [3]. The browser must be able to receive cookies over an encrypted channel and install them as if they had been set by the website directly. This is possible using a browser addon, but installing such an addon will not be possible for all users in all situations. We propose a fallback mechanism, transcription of a URL, to be used in these, hopefully rare, circumstances.

When the website W returns cookie c_U to the scanner (see step 3 above), it also returns a special single-use login URL, l_U, which is of the form:

```
https://<domain-of-W>/?<nonce>
```

The website links the nonce in l_U with U, such that opening the URL in a web browser will cause the corresponding cookie c_U to be installed. So if the scanner is unable to write the cookie back to the browser automatically because the latter is unmodified and/or no channel is available, the scanner can instead display the login URL l_U for the user to transcribe into the browser's address bar manually and after another round-trip to W the cookie is installed. In effect the user themselves takes on the role of the new required channel. Clearly this impacts on usability, notably compromising the *Physically-Effortless* and *Infrequent-Errors* benefits of the UDS framework [3]. From a usability perspective typing out l_U, which must contain an unguessable random nonce, is at least as difficult as having to type a password, but it is just a fallback to save the user in rare cases and there is still nothing for the user to remember.

The benefit of using a login URL which is typed directly into the browser's address bar, is that it's hard to send the nonce to the wrong person. Browser B making a request to a URL of the above form, is effectively the same as:

$$B \rightarrow W : \quad \{n\}_{K_{BW}}$$

Crucially the URL contains the nonce to send, n, the website to send it to, W, and the protocol to use, HTTPS, which provides the encryption under K_{BW}. If instead the user were asked to transcribe a single-use password into some specific form field on the website's login page, an attacker could coerce the user to enter it into a form field on their own fake login page (using traditional phishing techniques) and then forward it to the real site[10].

Session gifting attack Unfortunately, introducing this fallback mechanism introduces a new vulnerability[11]. An attacker, U', can use their scanner to obtain a fallback URL $l_{U'}$ and then get a victim to open it, leaving the victim with a cookie $c_{U'}$. In other words, the attacker *gifts* the victim a session for an account they (the attacker) control, just by having them open $l_{U'}$. If a user did not notice this, they might divulge sensitive information, such as credit card details, which would later be accessible to the attacker.

To defend against such attacks we augment the new session delegation protocol with something similar to the "browser nonce", n_B, from the original (flawed) protocol above (see section 2.2).

When the user navigates to the login page of the website, the website installs a fresh "browser identifying cookie", c_B, in the user's browser, B. This browser ID cookie will automatically be sent back to the website with each future request until it is deleted. The value of c_B is also included in the visual code and thus reaches the scanner. The scanner sends c_B back to the website when it authenticates, allowing the website to form a link between c_B, and the session cookie c_U and login URL l_U it returns.

Now, whenever a browser makes a request to a login URL, the website simultaneously receives the nonce in the URL, and any browser ID cookie previously set for that browser. The website can check if the correct browser ID cookie is included in any such request before granting session cookie c_U.

[10] It would still be possible for attackers listen for nonces by typosquatting on domains similar to the domain of a popular website W.

[11] We thank Olgierd Pieczul for pointing this out during the workshop.

With this countermeasure, the session gifting attack is no longer possible. The attacker may acquire the fallback URL $l_{U'}$, but if the victim opens it they will not be granted cookie $c_{U'}$, because their browser doesn't have the required browser ID cookie.

4.2 Other solutions

It may be argued that the challenges in the visual code should only remain valid for a limited period to reduce the window of vulnerability. However we consider this to be merely an implementation feature that does not fundamentally address the underlying security issue. The attacker can relay the visual code more quickly, perhaps requesting it on-demand, or they can relay the same code to many targets simultaneously to improve the chances of a catch before it expires.

Trusted visual code display An alternative solution to attacks where the visual code is relayed from one site to another would be to extend the trusted computing base to include the browser. In such a scheme the browser, or browser addon, verifies that the domain of the website presenting the visual code matches the website identifier or address being transmitted to the scanner in that code. Unfortunately we deem this to be a non-solution for several reasons.

It is not in general possible for the web browser to tell when it is displaying a visual code. An attacker clearly wouldn't helpfully tag their relayed visual codes to make them easier for the browser to find, so it would have to run a detection algorithm on every displayed image. But then an attacker might not use an actual embedded image, but create a visual code by arranging other HTML elements[12].

Alternatively, the browser could provide a special trusted display area specifically for visual codes somewhere in the chrome of the browser window and offer some kind of API to allow websites to have visual codes for their own domain displayed there. However this does not prevent other visual codes being displayed in the normal, non-trusted browser window and experience with mechanisms such as the HTTPS padlock shows that such signals are not fully understood by users. A user may not understand the difference between a visual code in the trusted display area and one in the normal web page.

Furthermore visual codes might be present in any number of other locations including physical locations; we already discussed an email-based session phishing attack above. There is no way that everywhere a visual code is displayed on-screen, or printed, can be trusted.

Secure bookmarks As the session phishing attacks presented in this paper rely on the user scanning a relayed visual code, they can, of course, be prevented using a different sort of authentication scheme which doesn't use visual codes. In

[12] For example the qrcode.js library (`https://github.com/davidshimjs/qrcodejs` uses the new HTML5 canvas drawing element.

a secure bookmarks scheme, such as Phoolproof [16], a hardware authenticator device, analogous to our scanner, holds all the keys and secrets and authenticates to websites on behalf of the user. When using a secure bookmark system, the user is responsible for manually selecting the website they wish to authenticate to.

While such a system is resilient to session phishing because it doesn't use any kind of browser nonce, it does require a channel from the authenticator device to the web browser, just like a scanner using our session delegation protocol, and thus faces the same deployability issues. The secure devices have a similar role in both types of scheme, namely that of brokering a session between the browser and the website. One usability benefit of visual code authentication schemes over secure bookmark schemes is that they do not require the user to select the website and browser they want to broker a session between because both are identified in the visual code.

5 Related work

Desmedt *et al.* [17] introduces the term mafia fraud in the context of a mafia owned restaurant (cfr. Fig 3). In this paper we appropriate the term mafia fraud to describe the similar attack in which the visual code is relayed unchanged by an active man-in-the-middle. It is clearly impossible to relay the visual channel undetectably between the web browser and the scanner. And whilst it is possible to relay the channel between the scanner and website, the use of SIGMA-I ensures that the attacker gains no benefit for doing so.

Beth and Desmedt [18] seek to mitigate relay attacks by enforcing a maximum round-trip time of a challenge-response protocol. Brands and Chaum [19] refine the technique and make it robust, introducing the first distance-bounding protocol. But distance-bounding does not relay attacks; rather, it solves the simpler problem of ensuring that the prover and verifier are located within a specified distance bound. However, when authenticating to web services, physical proximity is irrelevant because the honest prover could be in a different country or continent from the verifier. Furthermore, an honest user may wait an indeterminate amount of time before scanning a visual code even after it has loaded.

Parno, Kuo and Perrig's "Phoolproof Phishing Prevention" [16] uses a trusted mobile device to mutually authenticate with remote services from an untrusted terminal, the main objective being to prevent or limit the efficacy of phishing attacks. In this scheme the user selects a web service to authenticate to from a secure bookmark on the trusted authenticator device. A secure session is then brokered between the web service and the untrusted terminal by the trusted authenticator device. While Phoolproof is not a visual code authentication scheme, it does require a channel from the authentication device to the browser, like the scanner does when using our session delegation protocol; the comparison between the two is therefore instructive.

Mannan and van Oorschot [20] define a protocol, MP-Auth, for user authentication and secure financial transactions from an untrusted device with assistance of a trusted mobile device. Although not a visual code authentication scheme, MP-Auth shares many architectural similarities with such schemes. MP-Auth does not seek to address the fundamental security weakness of passwords, nor does it reflect the realities of modern mobile phone platforms where malware is common. And so, whilst MP-Auth is not vulnerable to the relay attacks presented in this paper, it does not offer the security and usability benefits of visual code authention schemes.

Laurie and Singer [21] argue that it is impossible to have a system which is both general-purpose and trustworthy. Furthermore, they define the requirements for a trusted device, referred to as the Neb, that may be used to authenticate online transactions. This is relevant to the decision as to whether the scanner should be a dedicated device or a smartphone application.

6 Conclusion

We presented the first comprehensive analysis of visual code relay attacks on the emerging class of authentication schemes in which users login to websites by scanning a visual code. We identified a variety of schemes in this class and highlighted their common features. In particular we generalised the protocol they all use to authorise a web browser running on a separate host. We found that all of the currently proposed schemes that we reviewed are vulnerable to attacks in which the visual code is relayed.

We have presented examples of such attacks and discussed the architectural reasons for the vulnerability. Such attacks are worrying because the attacker does not have to modify any visual code and the user's scanner authenticates to a trusted website. In particular, if the user's scanner acted according to specification there may be a burden of proof on the user to prove that any resulting transactions were fraudulent[13].

We reject claims that these attacks can be mitigated by requiring the user to carry out manual checks or using a trusted display. The root cause of this vulnerability is the use of a browser nonce which the browser obtains at the start of the protocol and then "trades in" for an authorisation cookie, c_U, once the scanner has, independently, authenticated user U. Our proposed solution, the session delegation protocol, allows authorisation of the web browser without the use of a browser nonce. Instead, the website sends the session cookie to the browser, via the trusted scanner, only *after* the authentication has taken place.

The cost of that solution is the need for an authenticated and encryped communication channel from the scanner to the browser. We have explored the use of a local radio channel (Bluetooth) and a connection via the Internet, assisted by a rendezvous point. In either case, modification of the browser is required which harms the deployability of scheme, which is perhaps why none of the

[13] The real problem instead being that the specification was wrong, in so far as the scheme is vulnerable to relay.

schemes surveyed adopted a similar approach. However, we have provided a fall-back mechanism for the occasions when browser modification is not possible.

The class of visual code authentication systems seemed a promising contender for replacing passwords but that critical and seemingly inherent vulnerability present in all past implementations made it not credible. By allowing ourselves to consider more fundamental changes to the architecture (such as the inclusion of another channel from scanner to web browser) rather than being constrained by backwards compatibility, we have found a way to stop an array of session phishing attacks. Now we can move forward by selecting the most appropriate trade-off between usability, deployability and security.

7 Acknowledgments

We gratefully acknowledge the European Research Council for funding this research under grant 307224.

We also thank Olgierd Pieczul for pointing out the login gifting attack during the workshop.

References

1. ISO: Information technology—Automatic identification and data capture techniques—QR Code 2005 bar code symbology specification. ISO 18004:2006, International Organization for Standardization, Geneva, Switzerland (2006)
2. Stajano, F.: Pico: no more passwords! In: Proceedings of the 19th international conference on Security Protocols. SPÍ1, Berlin, Heidelberg, Springer-Verlag (2011) 49–81
3. Bonneau, J., Herley, C., Oorschot, P.C.v., Stajano, F.: The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In: Proceedings of the 2012 IEEE Symposium on Security and Privacy. SP '12, Washington, DC, USA, IEEE Computer Society (2012) 553–567
4. Cobos, J.J.L., Hoz, P.C.D.L.: Method and system for authenticating a user my means of a mobile device. Patent filed 2009-09-17, published 2012-09-04.
5. DeSoto, D.B., Peskin, M.A.: Login using qr code. Patent filed 2013-02-15, published 2013-08-22.
6. Van Rijswijk, R.M., Van Dijk, J.: Tiqr: A novel take on two-factor authentication. In: Proceedings of the 25th International Conference on Large Installation System Administration. LISA'11, Berkeley, CA, USA, USENIX Association (2011) 7–7
7. Howard, A.: Qrauth. Bsc. thesis, Bournemouth University, Bournemouth, UK (2012)
8. Dodson, B., Sengupta, D., Boneh, D., Lam, M.S.: Secure, consumer-friendly web authentication and payments with a phone. In: In Conference on Mobile Computing, Applications, and Services (MobiCASE'10). (2010)
9. Fu, H.P.: Pico: No more passwords! Msc. thesis, University of Leuven, Flanders, Belgium (2013)
10. M'Raihi, D., Rydell, J., Bajaj, S., Machani, S., Naccache, D.: OCRA: OATH Challenge-Response Algorithm. RFC 6287 (Informational) (June 2011)

11. Inc., C.O.: QRAuth. `http://www.computingobjects.com/qrauthinfo` (2012) Accessed: 2013-11-13.
12. Gibson, S.: Secure Quick Reliable Login. `https://www.grc.com/sqrl/sqrl.htm` (October 2013) Accessed: 2013-11-6.
13. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Proceedings of the 5th International Workshop on Security Protocols, London, UK, UK, Springer-Verlag (1998) 91–104
14. Desmedt, Y., Goutier, C., Bengio, S.: Special uses and abuses of the fiat-shamir passport protocol. In: Advances in CryptologyCRYPTO87, Springer (2006) 21–39
15. Krawczyk, H.: Sigma: The sign-and-mac approach to authenticated diffie-hellman and its use in the ike protocols. In Boneh, D., ed.: Advances in Cryptology - CRYPTO 2003. Volume 2729 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 400–425
16. Parno, B., Kuo, C., Perrig, A.: Phoolproof phishing prevention. In Crescenzo, G.D., Rubin, A.D., eds.: Financial Cryptography and Data Security, 10th International Conference, FC 2006, Anguilla, British West Indies, February 27-March 2, 2006, Revised Selected Papers. Volume 4107 of Lecture Notes in Computer Science., Springer (2006) 1–19
17. Desmedt, Y., Goutier, C., Bengio, S.: Special uses and abuses of the fiat-shamir passport protocol. In: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. CRYPTO '87, London, UK, UK, Springer-Verlag (1988) 21–39
18. Beth, T., Desmedt, Y.: Identification tokens or: Solving the chess grandmaster problem. In: Proc CRYPTO 90. Number 53 in LNCS (1990)
19. Brands, S., Chaum, D.: Distance-bounding protocols (extended abstract). In: EUROCRYPT93, Lecture Notes in Computer Science 765, Springer-Verlag (1993) 344–359
20. Mannan, M., Van Oorschot, P.C.: Using a personal device to strengthen password authentication from an untrusted computer. In: Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security. FC'07/USEC'07, Berlin, Heidelberg, Springer-Verlag (2007) 88–103
21. Laurie, B., Singer, A.: Choose the red pill and the blue pill: A position paper. In: Proceedings of the 2008 Workshop on New Security Paradigms. NSPW '08, New York, NY, USA, ACM (2008) 127–133