SecuriCast: Zero-Touch Two-Factor Authentication using WebBluetooth

Thomas Dressel
Institut für Informationssysteme
Hof, Germany
thomas.dressel@iisys.de

Eik List and Florian Echtler
Bauhaus-Universität Weimar
Weimar, Germany
firstname.lastname@uni-weimar.de

ABSTRACT

Simple username/password logins are widely used on the web, but are susceptible to multiple security issues, such as database leaks, phishing, and password re-use. Two-factor authentication is one way to mitigate these issues, but suffers from low user acceptance due to (perceived) additional effort.

We introduce SecuriCast, a method to provide two-factor authentication using WebBluetooth as a secondary channel between an unmodified web browser and the user's smartphone. Depending on the usage scenario and the desired level of security, no device switch and only minimal additional interaction is required from the user. We analyse SecuriCast based on the framework by Bonneau et al., briefly report on results from a user study with 30 participants demonstrating performance and perceived usability of SecuriCast, and discuss possible attack scenarios and extensions.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**;

KEYWORDS

WebBluetooth; Bluetooth Low Energy; BTLE; two-factor authentication; TFA; smartphone; smartwatch

1 INTRODUCTION & RELATED WORK

Access to today's ubiquitous web services is mostly controlled by a single authentication factor, usually in the form of a username/password combination. Given that a single person can easily have dozens of different web-service accounts, average users are often overwhelmed by the complexity of managing these accounts. Consequently, they usually try to reduce the required effort, e.g. by recycling the same password for multiple services. In fact, about 55% of users confirm to re-use at least one of their passwords [6].

Figure 1: Using SecuriCast in notify mode to confirm two-factor authentication via a smartwatch.

While understandable, this behavior opens up several security risks. Account databases are regularly hacked and leaked online. For example, the recent Collection#1 breach contained around 2.7 billion records [13]. Low-complexity passwords or those stored in databases with insufficient protection can easily be brute-forced by adversaries, and re-used credentials then open the gates for abuse. Phishing attacks via spam mails are also widely used to collect account data from unsuspecting users, leading to similar issues.

Two-factor authentication (TFA) is one approach to counter these issues. In addition to regular username/password combination, a second authentication factor shall verify that the person trying to authenticate at the service is indeed the legitimate user. TFA generally tries to verify a physical quantity that can not easily be stolen or leaked online. It can use biometric measures as the second factor, but a more common approach is to verify possession of a physical token. The latter approach is prevalent today, as many people already carry a suitable, almost omnipresent token – namely, their smartphone. A dedicated app generates one-time passwords that the user has to enter within a short time window after the regular login process, thereby verifying that, indeed, the

EICS '19, June 18–21, 2019, Valencia, Spain

ACM ISBN 978-1-4503-6745-5/19/06…$15.00
https://doi.org/10.1145/3319499.3328225

same person has access to a) the account credentials, and b) the smartphone. Unfortunately, only few people seem prepared to shoulder the added effort for TFA, however small it may be. For example, in 2015, it was estimated that about 6.5% of Google accounts use TFA [19], and it is unlikely that this number has significantly increased since then.

We introduce SecuriCast, a TFA method that requires minimal user interaction beyond the normal login and automatically verifies the second factor via a parallel communications channel (WebBluetooth) between the browser and the user's smartphone. In particular, our approach does not require the user to switch devices during the login process. If desired, an additional level of security against co-located attackers can be introduced by requiring the user to verify a set of four keywords before the second factor is delivered. SecuriCast does not require installation of dedicated software or plugins in the browser, and can consequently be used in any environment such as an Internet café. *Related Work* In general, two-factor authentication was first established as a means to secure banking transactions. It received broader attention when Google [10], and other major Internet companies such as Apple or Facebook developed variants [2, 22]. Various research projects already have attempted to simplify the TFA process to foster adoption.

Scanning Visual Codes Quick-Response (QR) codes easen the transmission of challenges and responses between browser and smartphone. As two good examples, Dodson et al. proposed Snap2Pass [8] and van Rijswijk and van Dijk proposed Tiqr [24]. Their approach avoids sending data, which protects against nearby adversaries, and saves battery. However, the user must still start the app, focus, and scan the QR code, which becomes a dull task in the long run.

Typing PINs The Google Authenticator (GA) app is representative for a wide range of existing one-time-password-based TFA protocols (further examples include, e.g., [1, 11]). All such apps require an initial setup step to exchange a cryptographic secret between the web service and the user's smartphone which is then used to generate the one-time passwords, usually via scanning a QR code. Later during login, the user has to start the GA app on their device and type the generated six-digit pin in a browser dialog, which is a small, but – in the long run, across web services – tedious process. Moreover, while phishing is an immanent risk to various protocols, OTP-based protocols open another potential attack angle: after obtaining the first factor, attackers "phish" the second one by simply asking their victims to forward the received SMS code [16]. This social risk is mitigated for WiFi-based solutions where no obvious notification occurs.

In 2016, Google [14] added push notifications for compatible smartphones to enhance user-friendliness; a similar approach had been introduced before in Duo Push [9]. After registration, a push notification with IP, homepage, and time is sent to the device on each login event, which can be either approved or denied by the user, both with one tap. The approach advances usability, but needs Internet or cellular access for the phone.

Interaction-free TFA Mechanisms. PhoneAuth [7], Knock x Knock [12], and Sound-Proof [15] are phone-based challenge-response mechanisms that try to avoid user interaction for the second factor. PhoneAuth needs a modified version of the GA plus a Chrome browser with a custom extension. The protocol offers a strict mode that enforces a cryptographically secure second factor, and an opportunistic mode that uses the second factor only if possible, but omits it otherwise and provides feedback to the user about login attempts. The opportunistic fallback to one factor is a clear drawback, and demands knowledgeable users.

Knock x Knock already uses Bluetooth Low Energy to transmit challenges. However, this approach requires both an application and a browser plugin on the local machine, and is consequently only suited for personally owned computers.

Sound-Proof compares the ambient sound of the user's environment on the computer with the browser and the smartphone for authentication. The approach clearly benefits the usability; however, recording environment sound introduces privacy risks. Moreover, this approach cannot protect against man-in-the-middle, phishing, or co-located adversaries; plus, it is difficult to deploy at scale.

Other Approaches Many more smartphone-based TFA schemes exist in the literature. Some concern transaction via untrusted browsers, e.g., PhoolProof by Parno et al. [20], Starnberger et al.'s QR-TAN [23], or Mannan and van Oorschot's MP-Auth [18]. Others include full protocol suites, such as that by Shirvanian et al. [21]; their variants with low interaction employ Bluetooth or WiFi, but require an extended version of the browser. Despite this large body of work, we are unaware of any two-factor protocols that enable authentication with little additional interaction in arbitrary environments (non-modifiable browser, lack of Internet connectivity, etc.). It is this gap that SecuriCast aims to address.

2 THREAT MODEL

The primary threat that SecuriCast is designed to defend against is password leakage. Thus, we assume that an adversary can obtain the user's login and plaintext password, either through a phishing scheme, a database leak, or a key logger. We optionally also consider the strong assumption that an adversary could identify the user personally and be in sufficient physical proximity so that Bluetooth communication can be passively observed, or even actively generated. We currently do not address scenarios wherein the adversary knows the user's credentials *and* simultaneously has full control over the user's smartphone (either through theft while unlocked, or via malware). Regarding the environment, we

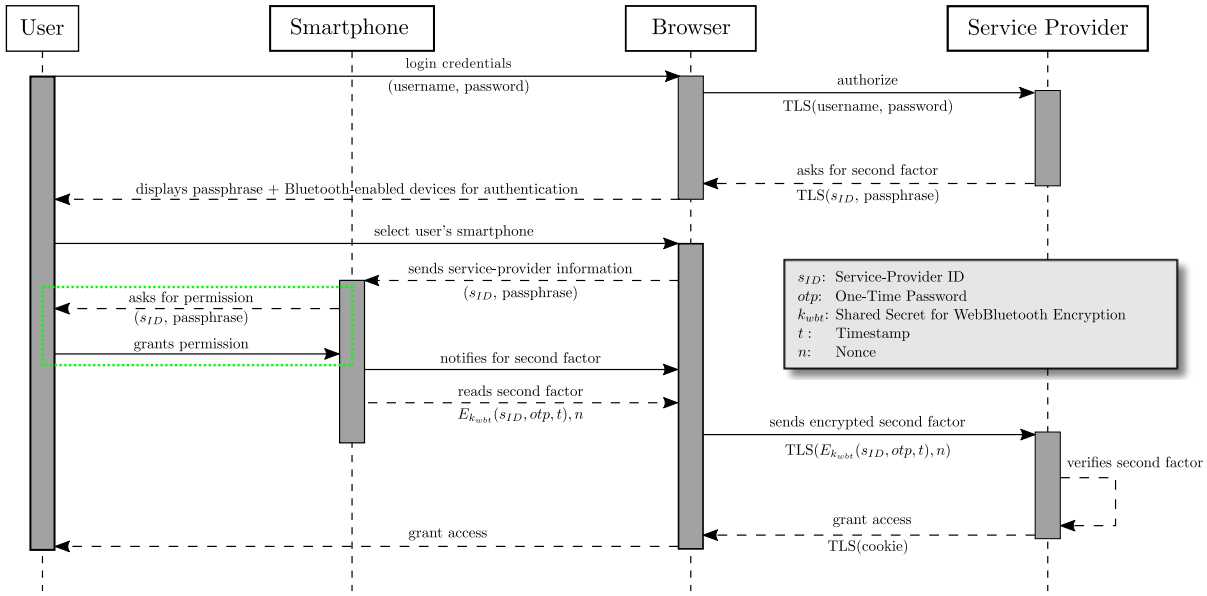

Figure 2: Sequence diagram of the TFA process with notification. Note that for zero-touch mode, the user interaction contained in the dashed green box is omitted and the second factor is immediately sent to the browser.

do not assume that the user is able to modify the computer they use (e.g. by installing browser plugins), and only assume that a recent version of the Chrome browser is used. We also consider the potential for a Denial-of-Service (DoS) attack in which the attacker disrupts Bluetooth communication.

3 SECURICAST IMPLEMENTATION

SecuriCast consists of three main components: the service provider who wants to authenticate the user, a browser with WebBluetooth support (Chrome 53+) for the login process, and the user's smartphone with the SecuriCast app that provides the second factor. We have implemented SecuriCast based on the open-source Google Authenticator framework [26]. Our implementation includes an app for Android 6.0+ with QR-code support for the setup, a login frontend for the browser with WebBluetooth support written in JavaScript, and an authentication backend based on Java servlets. Source code is available at https://github.com/mmbuw/securicast.

To use SecuriCast, the user first has to complete a one-time setup process to register their individual app instance with the service provider.We assume that this initial one-time setup process is conducted in a secure environment where the QR codes with symmetric encryption keys can be read only by the user. The setup has to be completed once for each service provider identified by a unique identifier s_{id} that concatenates its target domain name and username. All communication between the browser and the service provider is encrypted using the industry-standard TLS protocol v1.3. For the service provider, the server-side integration effort is the same as for the existing GA.

After the setup, SecuriCast can be used in any environment that provides a WebBluetooth-supporting browser. This includes personal computers, but also "borrowed" machines, e.g., in an Internet café. As Bluetooth Low Energy and, thus, WebBluetooth lack encryption by default for standard connections in the GATT protocol, a symmetric key k_{wbt} is exchanged during the setup process, and is used to encrypt the communication between the browser and the smartphone during authentication. SecuriCast uses AES-256-GCM with standard random 96-bit nonce and 128-bit tag sizes to prevent sniffing attacks. The second factor uses a custom BTLE service ID, that is also used by the browser to show only those devices for connection that offer this specific service.

SecuriCast supports two modes: a "zero-touch" mode that avoids any device switch or additional interaction from the user except initiating the WebBluetooth connection, and a "notify" mode wherein the user can compare four keywords to verify the authentication attempt (see Figure 2 for a detailed sequence diagram). As SecuriCast is based on the GA, the second factor is also displayed in the app as a six-digit PIN that can be entered manually as a fallback option, e.g., if no Bluetooth module is available or if a co-located attacker disrupts wireless communications. As with any TFA solution that uses a separate device, a loss of power on this device or a loss of it will prevent the user from authenticating. Mitigating this issue is beyond the scope of our work.

The two modes of SecuriCast support two levels of security, with a corresponding usability trade-off. The zero-touch mode requires no device switch and only three additional clicks (1. start Bluetooth discovery, 2. select a target device, and 3. initiate the connection), but cannot fully protect against a co-located adversary with prior knowledge of the user. In this specific targeted attack scenario, the adversary already has gained access to the first factor, e.g. through a phishing email or a database leak. If the adversary can then a) personally identify the targeted user and b) be in physical proximity to this user, they could login with the stolen first factor, and also remotely employ the user's SecuriCast app as second factor without the user's knowledge, as the browser does not have to identify itself to the smartphone.

If users also want protection against such advanced attacks, they can utilize the notify mode. Here, the smartphone (or connected smartwatch) shows a notification when the second factor is requested. The notification consists of four pictureable words (e.g. "cat" or "house" that can be associated with images and are therefore easier to remember than random words) which are also displayed in the browser while the login process waits for the second factor. We select 4 words from a pool of 1000 to avoid overloading the users' short-term memory while still providing reasonable security against guessing attacks with 10^{12} possible combinations. If the words match, the user can confirm the notification with a single tap (see also Figure 1) to continue. Otherwise, they can deny the login attempt, also with one tap.

Even if the adversary also installs the SecuriCast app, they will still lack the private key and thus cannot generate correct one-time passwords. On the other hand, if an adversary solely gains access to the user's smartphone, e.g. by theft, they will be unable to use SecuriCast without also having access to the first login factor (i.e., username and password).

4 USER STUDY

We have evaluated the usability of SecuriCast in a lab-based user study. Our study was conducted with 30 volunteer participants recruited from employees and students of our local university who did not receive any compensation for their participation. They had a mean age of 25.4 years (six female, 24 male) and 28 had a background in computer science. After completing a consent form and demographic questionnaire, the users were briefed about TFA and the general study content. Then, they had to complete three login tasks (counterbalanced by latin-square order) on a laptop with a second authentication factor: using Google Authenticator, SecuriCast (zero-touch), and SecuriCast (notify). First, the users had to complete a regular website login with a given username and password (identical for all tasks, provided on the instruction sheet). Next, they had to provide the second authentication factor, either by triggering the SecuriCast authentication or by entering the six-digit PIN from Google Authenticator. For SecuriCast, the users had to select and confirm the smartphone connection in a pop-up; in notify mode, they further had to compare and confirm the four picturable words shown on the login page and on the smartphone (see also Figure 1). The study concluded with a short semi-structured interview. All tests were conducted on a laptop with an unlocked Moto E (2nd gen.) smartphone placed besides the keyboard.

After each task, the participants were asked to complete an SUS questionnaire [4]. A Kolmogorov-Smirnov test showed that the data is not normally distributed. While a subsequent Friedman test showed no significant differences between the modes ($p = 0.228$, $z = 2.960$, $r = 0.54$), descriptively, both SecuriCast modes received higher ratings (median 87.5) than Google Authenticator (median 82.5). In the post-task interview, participants were asked which mode they prefer, and why. 15 users preferred SecuriCast (zero-touch), nine SecuriCast (notify), and six Google Authenticator. The informal feedback suggests that a) users felt positive about the usability of SecuriCast when compared to GA, b) it is important to leave users a choice of the mode depending on the situation, and c) a hybrid mode that shows a plain notification without keyword comparison could further increase adoption.

As the first factor was identical for all iterations of our study, we disregarded the time for entering the standard credentials. We logged the time after the first factor had been completed, and after the second factor had been received. In this time, users had to a) read and enter a PIN code (GA), or b) select the smartphone from the Bluetooth device list (SecuriCast zero-touch), or c) select the smartphone and compare the four words in the notification (SecuriCast notify).

We observed that SecuriCast (zero-touch) offers slightly better performance ($\bar{x} = 15.46s$, $SD = 6.60$) than Google Authenticator ($\bar{x} = 18.52s$, $SD = 8.44$), although not on a statistically significant level. The small difference is likely due to the additional interaction currently needed for setting up the Bluetooth connection, which may disappear in future versions of WebBluetooth. Moreover, participants tended to re-check the instruction sheet more often for SecuriCast, which may explain an additional small delay. Using SecuriCast (notify) takes longer than both other modes ($p < 0.001, \bar{x} = 25.85s$, $SD = 14.33$), however, this difference is likely due to the relatively unfamiliar process of comparing four keywords and would decrease with practice.

5 THEORETICAL ANALYSIS

We analyze SecuriCast in the framework for web-authentication methods by Bonneau et al. [3]. It introduces 25 features grouped into *Usability*, *Deployability*, and *Security*. For each feature, a scheme can receive up to two points, which yields a hypothetical maximum of 50 points. For the existing systems

Scheme		Usability								Deployability						Security												Total		
		Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically-Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	R.-t.-Physical-Observation	R.-t.-Targeted-Impersonation	R.-t.-Throttled-Guessing	R.-t.-Unthrottled-Guessing	R.-t.-Internal-Observation	R.-t.-Leaks-from-Other-Verifiers	R.-t.-Phishing	R.-t.-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable	#●	#○	Score
Google Authenticator	[3]	-	-	○	-	●	○	○	○	○	-	-	●	●	-	○	○	●	●	-	●	●	●	●	●	●	11	7	29
YubiKey	[17]	-	-	-	-	●	○	○	-	●	-	-	●	●	-	●	●	●	●	●	●	-	●	-	●	●	13	2	28
PhoneAuth (strict)	[7]	-	-	○	-	●	●	○	○	●	○	-	-	-	●	●	●	●	●	○	●	●	●	●	●	○	13	6	32
PhoneAuth (opportunistic)	[7]	-	-	○	-	●	●	○	○	●	●	-	●	-	●	○	○	○	○	○	○	○	●	●	●	○	9	10	28
Sound-Proof	[15]	-	-	○	-	●	●	○	○	●	●	-	●	-	●	○	-	●	●	-	●	●	●	●	●	-	13	4	30
SecuriCast (zero-touch)		-	-	○	-	●	●	○	○	●	●	-	●	-	●	●	-	●	●	○	●	○	●	●	●	●	**14**	**5**	**33**
SecuriCast (notify)		-	-	○	-	●	○	○	○	○	●	-	●	-	●	●	●	●	●	○	●	○	●	●	●	●	**13**	**7**	**33**

Table 1: Comparison of TFA methods in the framework by *Bonneau et al.*; each column represents a feature that can be rated as given (●, 2 pt.), quasi-given (○, 1 pt.), or as absent (-). Ratings from [7] (with shaded background) have been adjusted downwards from their original source to be consistent with [3,15,17]. R.-t. = Resilient-to.

(YubiKey, GA, SoundProof, and PhoneAuth), we took their self-assigned ratings from the respective publications (see Table 1). For the feature *Scalable-for-Users*, we have adjusted the existing ratings to zero to all schemes, as users are still required to remember individual credentials for each service; none of the present schemes scales to hundreds of services – as has also been discussed by Bonneau et al.; similarly, all discussed schemes need at least minor modifications to the backend, and can therefore not be rated as *Server-Compatible*. For detailed descriptions of the features and criteria for gradings, we refer to [3]. Hereupon, we focus on the aspects where we have rated SecuriCast differently than GA.

Efficient-to-Use. As the user has to neither switch to a different device, nor to type TOTP numbers, SecuriCast (zero-touch) is more efficient to use than GA. Since SecuriCast (notify) still requires at least an attention switch to the device, we rate it approximately equivalent to GA here.

Accessible. We rated SecuriCast (zero-touch) as fully *Accessible*, as it does not add interaction modalities. Users who can interact with the browser can also use SecuriCast (zero-touch), regardless of physical limitations; GA and SecuriCast (notify), however, need a switch to another device.

Mature. Since neither SecuriCast nor PhoneAuth or SoundProof have been used in large-scale deployments, they have not been rated as *Mature*, in contrast to GA and YubiKey.

Non-Proprietary. GA is considered proprietary, as only an older version is available as open-source. The other schemes could be re-implemented based on their publications (except for YubiKey); SecuriCast is fully open-source.

Resilient-to-Physical-Observation. Shoulder-surfing can reveal the user's PIN in the case of GA. With SecuriCast, however, the password is never visible to the adversary.

Resilient-to-Targeted-Impersonation. For SecuriCast (zero-touch), a co-located attacker could surreptitiously access the user's smartphone with their own browser and thereby gain access to the second authentication factor. However, SecuriCast (notify) is able to defend against all types of co-located attacks, and is therefore rated as being fully resilient.

Resilient-to-Internal-Observation. Adversaries that are able to capture the user's keyboard input can gain access to a service using GA if the second factor is used in a short time frame. Since both SecuriCast modes avoid keyboard input for the second authentication factor, they resist such attacks.

Resilient-to-Phishing. Both SecuriCast modes can only partially resist sophisticated phishing attacks. If an adversary gained control over the channel between the user's browser and the service provider (including TLS), they could intercept one-time passwords to authenticate in a parallel session.

Both SecuriCast modes arrive at a final rating of 33 points. The ratings reflect the slightly higher usability and deployability for the zero-touch, and higher security (particularly against co-located adversaries) for the notify mode. Their different interactions cannot be represented in this framework currently, as both modes need at least some user interaction (i.e., for the first factor). Both methods are on par or better to widely used commercial offerings, and can therefore be considered suitable for usage in real-world environments.

6 DISCUSSION & FUTURE WORK

One limitation of our approach is that WebBluetooth currently prohibits connections without user interaction. This requirement from the current standard draft is designed to prevent unauthorized websites from trying to connect to devices without user consent [27]. Since WebBluetooth is still

under development, future versions may allow unsupervised connections from trusted websites to known devices.

While WebBluetooth is currently supported only by Google Chrome 53+, its market share and auto-update feature provide a broad support base. Moreover, WebBluetooth is to become a W3C standard and is likely to be integrated into all major browsers. As a future alternative, the recently finished WebAuthentication standard [25] also allows for BTLE as a transport protocol. As both follow a similar control flow, it should be easily possible to adapt SecuriCast to the standard.

Many devices use their model designation as default Bluetooth name. If SecuriCast should gain wider adoption, choosing the correct device might become a source of errors. This can be partly mitigated when users select unique device names, but cannot fully prevent misconnections. As all Bluetooth communication is encrypted with AES-256-GCM under a secret key, connecting to the wrong smartphone would not compromise security, as the transmitted data from an erroneously selected device could not be decrypted properly.

The user feedback suggested a third, hybrid, mode that only requests a user confirmation before authentication, but avoids the keyword comparison. This would assuage users who felt insecure with the zero-touch mode, while still saving them time. This mode would also fit well in a smartwatch-based scenario, where a quick confirmation can be issued directly from the watch. Finally, we plan to conduct an additional study that will focus on participants with less security experience and examine their mental models of the TFA process, similar to the study in [5]. In parallel, a longer-term study in an everyday context could yield further insights.

REFERENCES

[1] Fadi A. Aloul, Syed Zahidi, and Wassim El-Hajj. [n. d.]. Two Factor Authentication Using Mobile Phones. In *AICCSA'09*, E. M. Aboulhamid and J. L. Sevillano (Eds.). IEEE Computer Society, 641–644.

[2] Apple. 2016. Two-factor authentication for Apple ID. https://support.apple.com/en-us/HT204915 [last accessed 2018-02-09].

[3] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. *The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes.* Technical Report UCAM-CL-TR-817. University of Cambridge. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf [last accessed 2018-02-09].

[4] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[5] Sonia Chiasson, P. C. van Oorschot, and Robert Biddle. 2006. A Usability Study and Critique of Two Password Managers. In *USENIX SS (USENIX-SS'06)*. USENIX Association, Article 1. http://dl.acm.org/citation.cfm?id=1267336.1267337

[6] Graham Cluley. 2013. 55% of net users use the same password for most, if not all, websites. When will they learn? https://nakedsecurity.sophos.com/2013/04/23/users-same-password-most-websites/ [last accessed 2018-02-09].

[7] Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan S. Wallach, and Dirk Balfanz. 2012. Strengthening user authentication through opportunistic cryptographic identity assertions. In *ACM CCS*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 404–414. https://doi.org/10.1145/2382196.2382240

[8] Ben Dodson, Debangsu Sengupta, Dan Boneh, and Monica S. Lam. 2010. Secure, Consumer-Friendly Web Authentication and Payments with a Phone. In *MobiCASE (LNICST)*, M. L. Gris and G. Yang (Eds.), Vol. 76. 17–38.

[9] Duo Security Inc. 2015. Duo Mobile. https://play.google.com/store/apps/details?id=com.duosecurity.duomobile [last accessed 2018-02-09].

[10] Google Inc. 2013. Google Authenticator. https://github.com/google/google-authenticator-android/ [last accessed 2018-02-09].

[11] Steffen Hallsteinsen, Ivar Jorstad, and Do Van Thanh. 2007. Using the Mobile Phone as a Security Token for Unified Authentication. In *ICSNC*. 68.

[12] Eiji Hayashi and Jason I. Hong. 2015. Knock x Knock: The Design and Evaluation of a Unified Authentication Management System. In *UbiComp*. ACM, 379–389. https://doi.org/10.1145/2750858.2804279

[13] Troy Hunt. 2019. The 773 Million Record Collection #1 Data Breach. https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/ [last accessed 2019-03-16].

[14] Google Inc. 2016. Sign in faster with 2-Step Verification phone prompts. https://support.google.com/accounts/answer/7026266?hl=en&ref_topic=7189145 [last accessed 2018-02-09].

[15] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound.. In *USENIX Security*. 483–498.

[16] Brian Krebs. 2016. The Limits of SMS for 2-Factor Authentication. https://krebsonsecurity.com/2016/09/the-limits-of-sms-for-2-factor-authentication/ [last accessed 2018-02-09].

[17] Juan Lang, Alexei Czeskis, Dirk Balfanz, Marius Schilder, and Sampath Srinivas. 2017. Security Keys: Practical Cryptographic Second Factors for the Modern Web. In *FC*, Jens Grossklags and Bart Preneel (Eds.).

[18] Mohammad Mannan and P. C. Van Oorschot. 2007. Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer. In *FC (LNCS)*, S. Dietrich and R. Dhamija (Eds.), Vol. 4886. 88–103.

[19] Jon Oberheide. 2015. Estimating Google's Two-Factor (2SV) Adoption with Pen, Paper, and Poor Math. https://duo.com/blog/estimating-googles-two-factor-2sv-adoption [last accessed 2018-02-09].

[20] Bryan Parno, Cynthia Kuo, and Adrian Perrig. 2006. Phoolproof Phishing Prevention. In *FC (LNCS)*, G. Di Crescenzo and A. D. Rubin (Eds.), Vol. 4107. 1–19.

[21] Maliheh Shirvanian, Stanislaw Jarecki, Nitesh Saxena, and Naveen Nathan. 2014. Two-Factor Authentication Resilient to Server Compromise Using Mix-Bandwidth Devices. In *NDSS*. The Internet Society.

[22] Andrew Song. 2011. Introducing Login Approvals. https://www.facebook.com/notes/facebook-engineering/introducing-login-approvals/10150172618258920/ [last accessed 2018-02-09].

[23] Guenther Starnberger, Lorenz Froihofer, and Karl M. Göschka. 2009. QR-TAN: Secure Mobile Transaction Authentication. In *ARES*. IEEE Computer Society, 578–583.

[24] Roland M. Van Rijswijk and Joost Van Dijk. 2011. Tiqr: A Novel Take on Two-factor Authentication. In *LISA (LISA'11)*. USENIX Association.

[25] W3C. 2018. Web Authentication: An API for accessing Public Key Credentials. https://www.w3.org/TR/webauthn/ [06.07.2018].

[26] Nathan Willis. 2014. FreeOTP multi-factor authentication. https://lwn.net/Articles/581086/ [last accessed 2018-02-09].

[27] Jeffrey Yasskin. 2016. The Web Bluetooth Security Model. https://medium.com/@jyasskin/the-web-bluetooth-security-model-666b4e7eed2.