2015 IEEE 16th International Symposium on High Assurance Systems Engineering

CamAuth: Securing Web Authentication with Camera

Mengjun Xie*, Yanyan Li*, Kenji Yoshigoe*, Remzi Seker†, Jiang Bian‡
* Department of Computer Science
University of Arkansas at Little Rock
Little Rock, AR 72204, USA
Email: {mxxie,yxli5,kxyoshigoe}@ualr.edu
† Department of Electrical, Computer, Software and Systems Engineering
Embry-Riddle Aeronautical University
Daytona Beach, FL 32114, USA
Email: sekerr@erau.edu
‡ Division of Biomedical Informatics
University of Arkansas for Medical Sciences
Little Rock, AR 72205, USA
Email: jbian@uams.edu

***Abstract*—Frequent outbreak of password database leaks and server breaches in recent years manifests the aggravated security problems of web authentication using only password. Two-factor authentication, despite being more secure and strongly promoted, has not been widely applied to web authentication. Leveraging the unprecedented popularity of both personal mobile devices (e.g., smartphones) and barcode scans through camera, we explore a new horizon in the design space of two-factor authentication. In this paper, we present CamAuth, a web authentication scheme that exploits pervasive mobile devices and digital cameras to counter various password attacks including man-in-the-middle and phishing attacks. In CamAuth, a mobile device is used as the second authentication factor to vouch for the identity of a use who is performing a web login from a PC. The device communicates directly with the PC through the secure visible light communication channels, which incurs no cellular cost and is immune to radio frequency attacks. CamAuth employs public-key cryptography to ensure the security of authentication process. We implemented a prototype system of CamAuth that consists of an Android application, a Chrome browser extension, and a Java-based web server. Our evaluation results indicate that CamAuth is a viable scheme for enhancing the security of web authentication.**

I. Introduction

Web has become the dominant interface for people to conduct their daily businesses on the Internet or a corporate network. People use their PCs to check email, access financial accounts, pay utilities bills, do online shopping, retrieve electronic health records and so on, all through a web browser. Web authentication stands as the first defense line to secure everyone's web accounts and online data. In general, a user authenticates herself to a web application hosted on a remote server by entering her username and password in the application's login page (either manually or automatically through a password manager). Password has been the *de facto* method for web authentication [1]. However, password only authentication cannot provide sufficient protection as the mechanism is prone to a variety of attacks including shoulder surfing attack [2], password guessing attack [3], [4], [5], man-in-the-middle (MITM) attack [6], phishing attack [7], [8], and so on.

To improve web authentication security and facilitate password management, mainstream web browsers (e.g., Chrome, Firefox, and Internet Explorer) have introduced built-in password managers. Standalone password managers [9] (e.g., 1Password and KeePass) and web-based password managers that run in a browser (e.g., LastPass, RoboForm, and PasswordBox) have also become quite popular. However, a password manager alone does not offer sufficient security assurance due to insecure computing environments at either local or remote. Zhao and Yue showed that none of the browser built-in password managers in mainstream web browsers could prevent malware from stealing passwords in a PC environment [10]. Recent studies on web password auto filling [11] and web-based password managers [12] reveal that there exist a number of serious vulnerabilities in popular password managers that can be exploited to launch password attacks and render disastrous consequences.

Recent years have witnessed frequent outbreak of data breaches and password database leaks that occurred on prominent websites such as LinkedIn [13], Yahoo! [14], and Gmail [15]. Those password leaks endanger millions of people's data security not only on those websites but also on other websites due to password reuse [16], [17]. To make matters worse, attackers often launch MITM attack and phishing attack to steal users' passwords. The recent MITM attack against Iranian Google users [6] demonstrates that even a user of well maintained and hardened website might be subjected to the MITM attack. According to [18], the total number of unique phishing sites detected worldwide in Q1 2014 is 125,215, a 10.7 percent increase over Q4 2013. Although TLS/SSL protocols can be applied to counter MITM and phishing attacks, the security offered by HTTP over TLS/SSL

978-1-4799-8111-3/15 $31.00 © 2015 IEEE
DOI 10.1109/HASE.2015.41

(HTTPS) is hinged on the validity of certificate [19] and the actual implementation [20], [21], which unfortunately often become the Achilles' heel. In addition, HTTPS is simply not available on many websites including governmental websites (e.g., `www.usa.gov` and `www.whitehouse.gov`).

As password-only authentication is evidently inadequate, two-factor authentication (TFA) has been strongly recommended and promoted to improve web authentication security. Special hardware based TFA solutions (e.g., SecurID and smartcard) were introduced long time ago but never reach general public. Aligned with the advancement of mobile computing technologies in the past ten years, many mobile device-assisted TFA schemes were proposed [22], [7], [23], [24], where an assumed trustworthy mobile device becomes the second factor in addition to the password. In practice, SMS based (e.g., [25], [26]) and soft token based (e.g., [27]) TFA schemes that leverage cellphones especially smartphones have been deployed. However, those schemes can incur costs on mobile communication (cellular or WiFi) and usually rely on mobile Internet, which may not always be available, for completing authentication.

Leveraging the unprecedented popularity of mobile devices (especially smartphones and tablets) equipped with built-in cameras and omnipresent practice of barcode scans through camera (e.g., snapping an ad in QR code, airport checking in with a mobile boarding pass in QR code), we propose CamAuth, a new TFA scheme for securing web authentication on PC. CamAuth uses a mobile device (smartphone or tablet) as the second factor to vouch for the user identity during login. The trusted device directly communicates with the PC through the visible light communication channels (established using camera-display links), which are secure as they are short-range, highly directional, fully observational, and immune to radio frequency interference. When applying CamAuth to web login, a user only needs to use her device to snap the barcode (e.g., a QR code) on the webpage and let the PC webcam capture the barcode generated by the mobile app for identity attestation. The use of barcode scan not only simplifies user action and secures information transfer but also makes the user be fully aware of the authentication process. Moreover, CamAuth has the following advantages:

1) The scheme requires no Internet connection for the mobile device during authentication. Therefore, it incurs no cost for the communications with the device.
2) The entire scheme is implemented at the application layer. There is no requirement to modify either the PC's operating system (OS) or the device's OS or firmware.
3) The scheme does not rely on SSL/TLS although the use of SSL/TLS can further enhance the security.

We have implemented a prototype system of CamAuth that consists of an Android application, a Chrome browser extension, and a Java-based web server. We evaluated our scheme against a few other popular authentication schemes and also conducted a small-scale usability study. Our evaluation results indicate that CamAuth is a viable scheme for enhancing the security of web authentication.

The rest of this paper is organized as follows: Section II briefly describes related work. Sections III and IV detail the design of CamAuth and its prototype implementation. Section V presents the evaluation for CamAuth. section VI concludes this paper.

II. Related Work

Two-factor authentication (TFA) requires the presentation of two or more authentication factors: something a user knows (e.g., a password), something a user has (e.g., a secure token), and something a user is (e.g., biometric characteristics). Using two factors as opposed to one factor generally delivers a higher level of authentication assurance. For example, passwords can be combined with security tokens such as RSA SecurID that implement one-time passwords or biometric characteristics such as fingerprint. With the popularity of mobile phone, a new category of TFA tools transforms a PC user's mobile phone into a token device using either SMS messaging [25], an interactive telephone call [26], or via a smartphone application [27]. A number of mobile device-assisted authentication schemes [28], [7], [23], [24] were proposed for protecting a user from either password stealing on an untrusted PC or phishing attacks. In those schemes, mobile devices are assumed to be trustworthy and able to perform certain computing operations such as hashing.

Phoolproof [7] is a public-key based scheme for strengthening bank transaction system. User is required to choose a bank site from the whitelist on the phone and then wait for information exchange between the phone and PC. MP-Auth [23] is a scheme that defends keylogger and phishing attacks with a cell phone by moving password input to mobile end and re-encrypting the username and password. Both Phoolproof and MP-Auth require wireless connection and well-implemented SSL/TLS. Czeskis *et al.* proposed PhoneAuth [24], a smartphone-based TFA scheme to strengthen user security in authentication. Despite that PhoneAuth and CamAuth share certain similarities, there also exist substantial differences. First, PhoneAuth is built upon the origin-bound certificate, which modifies TLS to realize strong client authentication. The deployment of PhoneAuth requires modification to current TLS, web browser, and smartphone firmware, which is not practical for average users. Second, PhoneAuth relies on Bluetooth for communications between the smartphone and PC. However, Bluetooth can be subjected to a variety of attacks. The Bluetooth module of smartphone has to stay active all the time, which is certainly not power efficient for mobile devices.

Recently camera-based communications have attracted much attention given the increasing popularity of mobile devices with one or more built-in cameras. Barcode scanning is the primary application domain of camera-based communications. A barcode is an optical machine-readable representation of information. There are two types of barcodes: one dimensional (1D) barcodes and two dimensional (2D) barcodes. Quick Response code (QR code) is a popular 2D

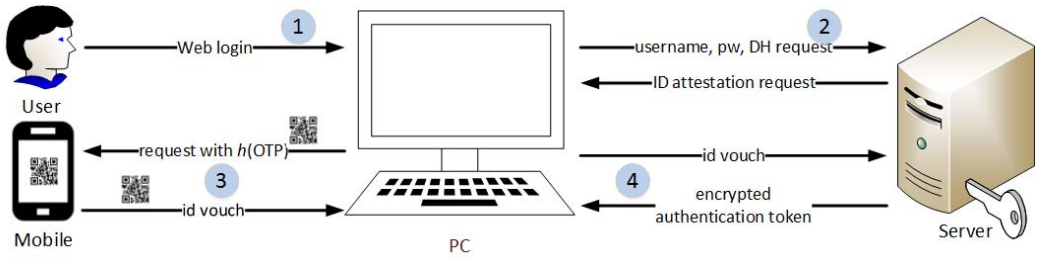


Fig. 1. Overview of CamAuth Authentication Process.

barcode. All major smartphone platforms support QR code scanning either natively or through third-party applications. As camera-based communications are short-range, highly directional, fully observational, and immune to electromagnetic interference, they have been applied to security applications.

McCure *et al.* proposed an authentication scheme called Seeing-is-Believing (SiB) [29], which leverages the unidirectional visual channel between a 2D barcode and a camera phone for simple authentication and demonstrative identification of devices. Sexena *et al.* proposed a short-range device pairing protocol, VIC (Visual authentication based on Integrity Checking), which is also based on a unidirectional visual channel [30]. Another wireless communication channel (e.g., Bluetooth) has to be used to complete the pairing process. Neither SiB nor VIC is suitable for web authentication. Recently, Xie *et al.* proposed CamTalk, a light based communication framework for bidirectional secure information transfer between smartphones by leveraging smartphone's display-camera channel [31].

III. System Design

CamAuth is aimed to assure the security of web authentication through a PC web browser in a cost-effective and convenient manner. CamAuth uses a mobile device as the trustworthy second authentication factor. During the CamAuth authentication, the device is used to vouch for the user's identity. Neither cellular network nor radio frequency (RF) wireless network (e.g., WiFi and Bluetooth) is used for transferring the device's vouch, which avoids various RF attacks. Instead, CamAuth applies visible light communication (VLC) through camera to take advantage of the security and convenience offered by VLC.

Figure 1 depicts a normal authentication process through CamAuth. The process consists of four interactions between involved entities (i.e., user, PC, mobile device, and web server[1]).

1) A user performs a web login through a web browser by entering her username and password either manually or automatically through a password manager on the login webpage.
2) The web browser is activated to send the username and password (or its hash value; pw is used to represent either case for illustration purpose) along with a Diffie-Hellman (DH) exchange request including its dynamically generated DH public key to the remote server. After validating the password, the server runs the DH algorithm to derive the shared secret, denoted by OTP, and sends back its identity vouch request and its DH public key.
3) The browser first computes the shared secret based on the received message and then encodes the request with the hash value of the shared secret into a barcode and renders it on the webpage. After scanning the barcode, the CamAuth app on the mobile device verifies the request and vouches for the user's identity using public-key cryptography. The vouch message is encoded into another barcode, rendered on the screen, and captured by the PC's webcam.
4) The vouch message is transferred to the server. If the validation of the vouch succeeds, the server will generate an authentication token (usually a session cookie), encrypt the token with the shared secret, and send it back to the browser, which completes the authentication process.

In this section, we first describe the assumptions and threat model. Then, we detail the process of CamAuth registration and authentication, followed by security analysis. Fallback mechanism, which is always on and can be used when CamAuth authentication is not applicable, is discussed in the end.

A. Assumptions and Threat Model

As CamAuth relies on camera-display links for communications between the mobile device and the PC, we assume that the mobile device has a back-facing camera, which is universal for today's smartphones and tablets, and that the PC is equipped with a webcam, which is also quite common especially for laptops. We assume that there is an Internet connection between the PC and the web server. However, no network connection (either wired or wireless) is required for the mobile device during authentication. HTTPS is not required for the CamAuth authentication. We assume that the user can perform web login from different PCs (e.g., one at home and one at workplace) but always uses the same mobile device (her smartphone) for authentication.

We assign the adversary the following capabilities. The adversary can eavesdrop network traffic and launch a phishing

[1]For presentation purpose, web application and web server are used interchangeably in the paper unless otherwise noted.

attack or MITM attack between the user's PC and the remote server (no MITM attack between the mobile device and the PC due to short-range VLC). The adversary can compromise the CA that has issued a TLS/SSL certificate to the target web server or steal the server's TLS/SSL private key if HTTPS is used by the server. We assume that the operating system of the user's device is secure and the CamAuth mobile app is securely protected by either a password, a PIN, a draw patten, or a biometric. The adversary can either obtain the user's device or gain her web account password but not both. Similarly, we assume that the web server can be breached and the adversary can steal either the password database or the server's private keys for CamAuth but not both (e.g., they are stored separately). Malware is assumed to be able to steal user passwords from PC. Note that the denial-of-service (DoS) attack targeting either the network or the end systems and the session hijack attack are out of the scope of this work.

TABLE I
TABLE OF NOTATIONS

U, pw	User ID and password (or password hash value)
S	ID of web application
U_k, U_p	User U's public key and private key
S_k, S_p	Web application S's public key and private key
g, p	Public prime base and prime number for Diffie-Hellman (DH) key exchange
$DH_k^{\mathsf{U}\cdot\mathsf{S}}$, $DH_p^{\mathsf{U}\cdot\mathsf{S}}$	User U's DH public key and private key for DH exchange with S
$DH_k^{\mathsf{S}\cdot\mathsf{U}}$, $DH_p^{\mathsf{S}\cdot\mathsf{U}}$	Application S's DH public key and private key for DH exchange with U
$\|\|$	Concatenation
$h(\cdot)$	a cryptographic hash function
$\mathcal{E}_k()$, $\mathcal{D}_k()$	Symmetric encryption and decryption with key k
$\mathbb{E}_k()$, $\mathbb{D}_p()$	Pubic-key encryption with k and decryption with p

B. CamAuth Registration

A user has to register her mobile device with the web application prior to the use of the device. We assume that the user conducts the registration on the authentic website (i.e., no phishing attack) and there is no attack during the registration. To register the device, the user logs into the web application from the mobile device directly through the mobile app, and submits the public key U_k generated at the device onto the server. The server will store U_k along with the user's identity U. The device will also retrieve the application's public key S_k and securely store it with the application's identity S and user's identity U in the device. The user's public key pair (U_k, U_p) is generated during the registration while the application's public key is generated during its deployment (i.e., pre-generated) and shared with all its users.

This registration is a one-time process per web application and device. The exchange can be done either remotely through the cellular network or locally through technical support. By doing so, the device becomes a trusted device by the web application. After the registration, the user needs to download and install the CamAuth web browser extension on the web browser in order to perform CamAuth authentication.

C. CamAuth Authentication

Figure 2 depicts the detail of the CamAuth authentication protocol. Once the CamAuth browser obtains the username, password and indication of using CamAuth (assume the user has a smartphone at hand that has been registered), the extension will start a Diffie-Hellman (DH) key exchange with the target server. The browser sends to the server the username (U) and password (pw) along with the DH public information (g, p) and dynamically generated public key ($DH_k^{\mathsf{U}\cdot\mathsf{S}}$).

When the server receives the request, it first verifies the username and password. After successful verification, the server S generates its DH public key based on g, p and its DH private key $DH_p^{\mathsf{S}\cdot\mathsf{U}}$ and computes the shared secret, denoted by OTP^{S}. Then, the server generates a vouch request VR_{S}, asking for the attestation by the device associated with that user account. The request VR_{S} contains the identity of the web application (S), the expiration time of this request (T)[2], and the hash value of $\mathsf{S}||\mathsf{U}||T||h(h(OPT^{\mathsf{S}}))$ encrypted by the application's private key S_p. After that, the server composes M_2 that consists of VR_{S}, the server's ID S, and its DH public key ($DH_k^{\mathsf{S}\cdot\mathsf{U}}$), and sends it back.

As soon as the browser receives M_2, it will strip $DH_k^{\mathsf{S}\cdot\mathsf{U}}$ off the message to compute the shared secret OTP^{U}. Next, the browser composes M_3 by concatenating S, VR_{S}, and the hash value of OTP^{U}, and encodes M_3 into a barcode. Then, the barcode is rendered on the webpage through HTML waiting for the user to snap it.

Upon capturing the code, the CamAuth mobile app decodes the message and decrypts VR_{S} using stored S's public key S_k. The device first verifies the origin (S) and checks freshness (T) based on the decrypted content, then computes $h(\mathsf{S}||\mathsf{U}||T||h(h(OTP^{\mathsf{U}})))$ (U can be obtained from stored information), and finally compares the hash value with $h(\mathsf{S}||\mathsf{U}||T||h(h(OTP^{\mathsf{S}})))$ decrypted from VR_{S}. If the origin is not correct, or the message is old, or the hash values do not match, the app will generate an error message notifying the user of the suspicious vouch request and abort the authentication. If the request passes those checks, the app will create the vouch M_4 for U by encrypting $h(\mathsf{S}||\mathsf{U}||T||h(OTP^{\mathsf{U}}))$ with its private key U_p. The result is encoded into a barcode and rendered on the device's screen.

The browser obtains M_4 through the PC's webcam and forwards it to the server. The server independently computes $h(\mathsf{S}||\mathsf{U}||T||h(OTP^{\mathsf{S}}))$ and compares the value with the decrypted M_4. If equal, the server will generate an authentication token (usually a session cookie that includes the information of both communication parties, unique number for the current session and time). Otherwise, an authentication failure notification will be generated and sent back. The server composes M_5 by encrypting the token with OTP^{S} and sends it to the browser. Once the browser successfully decrypts M_5 using OTP^{U}, the authentication is completed.

[2]We only assume a loose time synchronization here, which usually is not an issue for server and mobile devices.

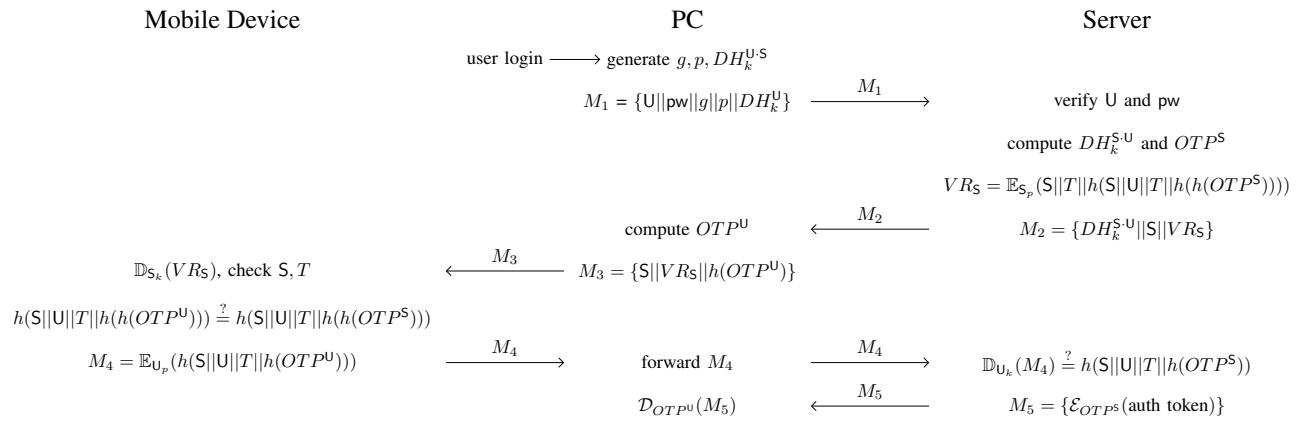


Fig. 2. CamAuth Authentication Protocol. The sequence diagram starts with "user login" at the top and proceeds downwards if there is no outgoing message.

D. Security Analysis

The CamAuth authentication protocol can effectively counter MITM and phishing attacks. Assume end systems are well secured and fully functional. First, as the communication channels between the PC and the mobile device are short-sight, directional and fully observational visible light channels, the threat of MITM attack between the PC and the device can be eliminated. Even if the adversary can eavesdrop the messages M_3 and M_4 by using a high-resolution camera, the shared secret (either OTP^{S} or OTP^{U}) is protected by the cryptographic hash function $h()$.

Second, if there exists an adversary between the PC and the server, the adversary can capture the password if the network connection is not secure (e.g., either no TLS or faulty TLS). To impersonate U, however, the adversary needs to have U's private key for vouch generation, which is securely stored in U's mobile device that is not in the possession of the adversary. The adversary can launch a MITM attack to intercept communications between the PC and the server. The middleman conducts two Diffie-Hellman key exchanges, one with the PC (resulting in OTP^{U}) and the other with the server (producing OTP^{S}). However, the middleman can be detected by the device and the server thanks to the protection provided by their private key. The vouch request VR_{S} (containing $h(h(OTP^{\mathsf{S}}))$) is signed by S's private key S_p. If the adversary passes the genuine VR_{S} but a forged DH public key (denoted by $DH_k^{\mathsf{E}\cdot\mathsf{U}}$ where E represents the adversary) to the PC, OTP^{U} computed by the PC based on $DH_k^{\mathsf{E}\cdot\mathsf{U}}$ will be different from OTP^{S}, which will be detected by the device in comparing hash values. Similarly, the vouch message M_4 (containing $h(OTP^{\mathsf{U}})$) is signed by U's private key U_p. If the adversary passes M_4 to S, the difference between OTP^{S} and OTP^{U} will also be detected by S.

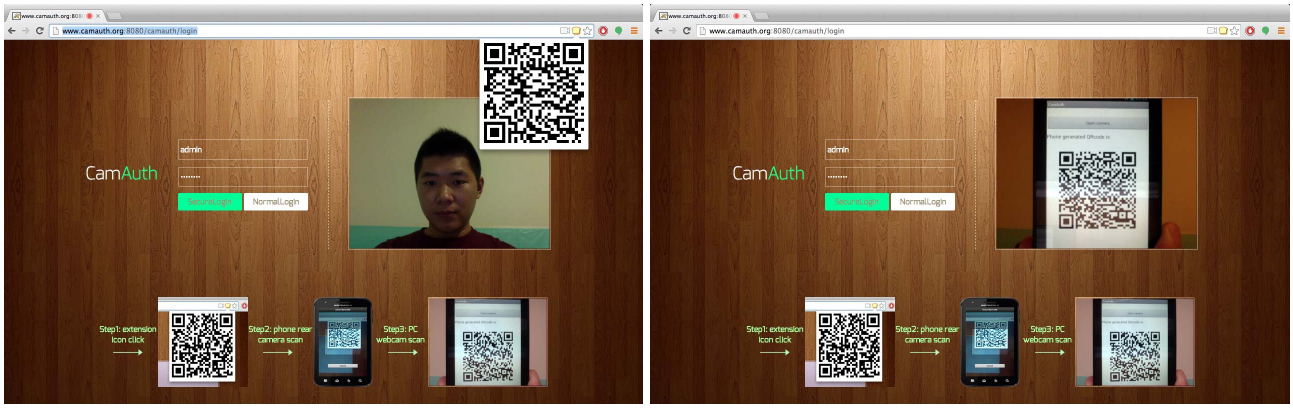
If the adversary steals the mobile device, he still cannot impersonate U without knowing U's username and password pairs. We assume that a user will request device revocation (or deregistration) as soon as the user discovers the loss of her authentication device, which is pretty standard practice when a user's smartphone gets lost. Even if the mobile device is compromised by malware, the malware actually cannot infer the shared secret OPT from either M_3 or M_4 as OPT is protected by the cryptographic hash function $h()$. Note that CamAuth does not address session hijack attacks in which the authentication token is stolen from the PC after the authentication completes but it can help curtail the attack by periodical re-authentication.

E. Fallback Mechanism

CamAuth can fall back to current password only scheme if a user does not have necessary hardware (e.g., smartphone or webcam) or software (e.g., browser extension) support. However, extra caution must be taken to prevent attackers from exploiting the fallback. One fallback mechanism is to use one-time password through email. If a user chooses to use the traditional login via username and password (e.g., clicking the special NormalLogin button provided by server), she will be informed that a short-lived one-time password (OTP) has been sent to her preset email address (the email server must be different from current login web server). For login, the user has to enter the password again with the OTP before it expires. Although this mechanism is not as secure as CamAuth, it is still better than password only thanks to the wide deployment of TLS for webmail (e.g., Gmail and Outlook). The server could also treat login sessions through the fallback mechanism differently from CamAuth login sessions. For example, the server could limit the functions that are accessible to sessions made by the fallback.

IV. Prototype Implementation

We have implemented a prototype system of CamAuth that consists of a web browser extension, a smartphone application, and a simple web server that understands CamAuth protocol. The browser extension is developed under the Google Chrome (version 32) environment due to its popularity and security design. Strong isolation is enforced in Chrome to achieve secure communications. For example, extensions are isolated from the JavaScript code embedded in the webpage and also from other extensions (except for the extensions using



(a) (b)

Fig. 3. Screenshots of CamAuth authentication using the prototype system. (a) The screenshot shows that a QR code is generated from the identity vouch request message and rendered in a pop-up webpage, waiting to be snapped by the mobile device. (b) The screenshot shows that the PC webcam is capturing the QR code of the vouch generated by the device and rendered on its screen.

message passing API). The browser extension for CamAuth is responsible for obtaining input data, conducting Diffie-Hellman key exchange, encoding and rendering barcode, and so on. We use QR code for the barcode in the prototype and generate QR code through a JQuery plugin. The extension has over 200 lines of JavaScript code, excluding the libraries.

The CamAuth mobile application is developed on Android 4.1 and compatible with Android 4.x and upward platforms. The application is responsible for device registration and CamAuth authentication. The application includes all the necessary QR code functions (e.g., decoding and encoding) using ZXing 2.0 library, and cryptographic functions (e.g., cryptographic hashing, encryption and decryption) using Bouncy Castle Java API. In total, this application has over 300 lines of Java code.

We also developed a CamAuth-aware Java-based web server. A number of Java Servlets were implemented to realize the core functionality of CamAuth. Once the servlet that handles OTP generation receives the DH exchange message, it will generate its own DH keys and compute the OTP, which is saved into the database temporarily in order to be compared to the OTP derived by the extension afterwards. A few pieces of JavaScript code are generated and injected into the login page by the Servlets to capture the QR code on the device screen using PC webcam and to decode the code and send the decoded message to the server. The implementation for the server contains over 200 lines of Java code.

Figure 3 shows two screenshots of the Chrome browser with the CamAuth extension when a user is logging into the CamAuth-aware web server. The screenshots were captured on a laptop that has a built-in webcam. If the intended website is CamAuth aware, a yellow block at the right end of the address bar (next to the bookmark star) will show up and the SecureLogin button will be highlighted. After the user inputs her username and password (either manually or through a password manager) and clicks the SecureLogin button, the webcam viewfinder (assume the webcam is allowed to be accessed) will appear along with a three-step graphical CamAuth usage guideline rendered at the lower part of the page to aid user's login. In the first step, the user clicks the yellow block (the CamAuth extension icon), which triggers the QR code of the vouch request to pop up, as shown in Fig. 3(a). In the second step, the user scans the QR code via the device's back facing camera. After the CamAuth mobile app completes the vouch, a QR code will be rendered on the device's display with a beep. In the last step, the user flips the device to let the PC webcam scan the QR code, as illustrated in Fig. 3(b). The webcam viewfinder helps a user position her device for barcode scan. If both password and identity vouch are accepted, the user is logged in and directed to the homepage; otherwise a failure page will be presented.

V. Evaluation

We first evaluate CamAuth using the web authentication assessment framework proposed by Bonneau *et al.* [1]. We compare CamAuth with passwords, a most popular TFA scheme—Google 2-step verification (2SV) [27], and a relevant mobile device based TFA scheme—PhoneAuth (in strict mode) [24]. The comparison results are shown in Table II.

In terms of usability CamAuth is highly comparable to 2SV and PhoneAuth. As barcode scanning using smartphone is already widely used, we grant *Easy-to-Learn* and quasi *Easy-to-Use* to CamAuth according to the definition of those benefits [1]. We believe that CamAuth (barcode scanning as user action) is easier to use than 2SV (PIN typing as user action) even though both have the same rating. We grant quasi *Infrequent-Errors* to CamAuth, the same as all others, as barcode scanning from either smartphone or PC is fairly accurate and the camera performance keeps improving. We believe that CamAuth and 2SV (and PhoneAuth) are at the same level ('s') for *Easy-Recovery-from-Loss* as their recovery mechanisms are much similar: users need to revoke the old device, install the app on the new device and register the new device.

TABLE II
COMPARISON OF CAMAUTH, PASSWORDS, GOOGLE 2-STEP VERIFICATION (2SV), AND PHONEAUTH (IN STRICT MODE).

	Usability						Deployability						Security									
Scheme	Scalable-for-Users	Nothing-to-Carry	Quasi-Nothing-to-Carry	Easy-to-Learn	Easy-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-Per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Passwords		y	y	y	y	s	y	y	y	y	y	y	y		s						y	y	y	y
Google 2-step verification			y	y	s	s	s	s			y	y			s	y			y	y	y	y	y	y
PhoneAuth – strict			y	y	y	s	s	y	s	s	s		y	y	y	y	y	s	y	y	y	y	y	s
CamAuth			y	y	s	s	s	s	s	s	s		y	y	y	y	y	s	y	y	y	y	y	y

Note: 'y' means the benefit is offered and 's' means the benefit is somewhat offered. The scores for passwords, 2SV, and PhoneAuth are from [24]. However, we change the scores for *Scalable-for-Users*, *Mature* and *Easy-Recovery-from-Loss* for PhoneAuth as we disagree with the authors' rating.

Evaluation on the deployability of CamAuth is mainly based on what changes to current systems would be required for deploying CamAuth. As CamAuth can be implemented at userland and application layer and requires no modification to OS kernel, device driver, or lower-layer protocols, the deployability of CamAuth is closely comparable to PhoneAuth. We rate CamAuth quasi *Accessible* as senior or disabled people might feel inconvenient when performing barcode scanning.

On security, CamAuth is resilient to physical observation, targeted impersonation, throttled/unthrottled guessing as the adversary still cannot log in without the user's device even if he obtains the password. It is quasi resilient to internal observation because both the device and the PC have to be compromised by the malware (i.e., internal observer). As the device has a separate key pair for each web application (i.e., verifier), CamAuth is resilient to leaks from other verifiers and is also unlinkable. It is certainly resilient to phishing and theft due to two-factor authentication. At last, CamAuth requires no third party (even no need of TLS/SSL) but explicit user consent (barcode scanning).

Performance of our scheme, i.e., time spent on login process, certainly affects user experience. As CamAuth involves two barcode scanning in a login, we are interested in its performance. We conducted an experiment to measure the average time of a CamAuth login performed by an average user who is familiar with CamAuth. We used a laptop that was bought in late 2011 and has an embedded 2-megapixel webcam and a Samsung S3 smartphone with a 8-megapixel rear camera for the experiment. Five users were involved in the experiment and each performed CamAuth login ten times. On average, it takes 172 milliseconds (ms) for the browser extension to generate a QR code. The smartphone spent 3.4 seconds on average from launching the app, scanning the QR code, to generating and rendering the vouch QR code. The average time from PC webcam scanning the code to login success is 2.9 seconds. We believe that fast-paced advancement of camera (e.g., higher resolution and faster auto-focus) and smartphone technologies will significantly improve the performance and user experience in near future.

To further understand the usability of CamAuth and how it is perceived by average people, we also conducted a small-scale usability study. We recruited 12 volunteers (eight males and four females) from the campus and asked each of them to test our prototype and complete a short survey. We readily acknowledge that our usability study is limited and bias exists. However, this preliminary assessment still provides useful information for us to further improve our work.

Among 12 testers, ten are in age range 20 and 30 and the rest two are in 30 and 40. Most of the testers are college students. One third of them received some level of computer security related training. And three quarters of them uses smartphone every day. We believe this group of people represents the user base that is most likely to try and adopt new technologies such as CamAuth. When asked if they are concerned about password authentication, two thirds of the testers are either very concerned or concerned.

Two questions regarding user technical background are: "Are you familiar with QR code?" and "Are you familiar with web browser extension?" For the first question, four testers answered that they like it and use it frequently and another four responded that they know the technology well and use it sometimes. The rest four indicated that they are not familiar with QR code. The answers to the second question follow a similar distribution.

Three questions regarding CamAuth usability were asked:

"Do you think CamAuth is easy to learn and use?", "Do you have the perception of the increase of security brought by CamAuth?", and "Will you use CamAuth to log into email or social network account if recommended?" For the first two questions, all the testers answered *yes* while all but one answered *yes* for the last question. Among all the testers, seven expressed that they need little time to learn and the rest five responded that learning does take some time. Although all but one showed willingness of using CamAuth, seven of them indicated that they will use CamAuth when they feel the environment is not secure, which suggests that more study on human cognition of cyber "secure" environment is necessary in order to design usable authentication schemes.

VI. Conclusion

In this paper we presented CamAuth, a camera based TFA scheme that augments the security of web login from PC. Leveraging the high market penetration of mobile devices and pervasive barcode scanning through camera, CamAuth realizes two-factor authentication through passwords plus barcode scanning using user's mobile device. The public-key cryptography and secure visible light communications ensure that CamAuth can effectively defeat password stealing attacks including man-in-the-middle and phishing attacks. CamAuth requires no modification to existing network protocols and operating system of PC and mobile device. Our prototype system and preliminary user study demonstrate the viability of the scheme. In future, we plan to conduct an extensive usability study to better understand the impact of using barcode scanning for web login on average users physically and psychologically.

Acknowledgment

We thank Liang Hao for his contribution in the early development of the prototype.

References

[1] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano, "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*, 2012, pp. 553–567.

[2] F. Tari, A. A. Ozok, and S. H. Holden, "A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords," in *Proceedings of the Second Symposium on Usable Privacy and Security*, ser. SOUPS '06, 2006, pp. 56–66.

[3] D. C. Feldmeier and P. R. Karn, "Unix password security - ten years later," in *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, 1990, pp. 44–63.

[4] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, pp. 364–372.

[5] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, ser. SP '12, 2012, pp. 538–552.

[6] S. Sengupta, "In latest breach, hackers impersonate google to snoop on users in iran," http://www.nytimes.com/2011/08/31/technology/internet/hackers-impersonate-google-to-snoop-on-users-in-iran.html, August 2011.

[7] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," in *Proceedings of the 10th International Conference on Financial Cryptography and Data Security (FC 2006)*, 2006, pp. 1–19.

[8] C. Yue and H. Wang, "Bogusbiter: A transparent protection against phishing attacks," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 6:1–6:31, 2010.

[9] K.-P. Yee and K. Sitaker, "Passpet: convenient password management and phishing protection," in *Proceedings of the second symposium on usable privacy and security (SOUPS '06)*, 2006, pp. 32–43.

[10] R. Zhao and C. Yue, "All your browser-saved passwords could belong to us: a security analysis and a cloud-based new design," in *Proceedings of the third ACM conference on Data and application security and privacy*, ser. CODASPY '13, 2013, pp. 333–340.

[11] D. Silver, S. Jana, D. Boneh, E. Chen, and C. Jackson, "Password managers: Attacks and defenses," in *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14)*, Aug. 2014, pp. 449–464.

[12] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14)*, Aug. 2014, pp. 465–479.

[13] Z. Whittaker, "6.46 million linkedin passwords leaked online," http://www.zdnet.com/blog/btl/6-46-million-linkedin-passwords-leaked-online/79290, June 2012.

[14] D. Hamilton, "Yahoo's password leak: What you need to know (faq)," http://www.cnet.com/news/yahoos-password-leak-what-you-need-to-know-faq/, July 2012.

[15] J. Leyden, "Leak of '5 meelllion gmail passwords' creates security flap," http://www.theregister.co.uk/2014/09/11/gmail_password_leak_flap/, Sept. 2014.

[16] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07, 2007, pp. 657–666.

[17] J. Bonneau, "Measuring password re-use empirically," http://www.lightbluetouchpaper.org/2011/02/09/measuring-password-re-use-empirically/, Feb. 2011.

[18] Anti-Phishing Working Group (APWG), "Phishing attack trends report: Q1 2014," http://docs.apwg.org/reports/apwg_trends_report_q1_2014.pdf, Anti-Phishing Working Group (APWG), June 2014.

[19] S. Schoen and E. Galperin, "Iranian man-in-the-middle attack against google demonstrates dangerous weakness of certificate authorities," https://www.eff.org/deeplinks/2011/08/iranian-man-middle-attack-against-google, August 2011.

[20] L. Bershidsky, "Heartbleed's password heartbreak," http://www.bloombergview.com/articles/2014-04-11/heartbleed-shows-open-source-needs-your-cash, Apr. 2014.

[21] M. Riley, "Nsa said to exploit heartbleed bug for intelligence for years," http://www.bloomberg.com/news/2014-04-11/nsa-said-to-have-used-heartbleed-bug-exposing-consumers.html, Apr. 2014.

[22] M. Wu, S. Garfinkel, and R. Miller, "Secure web authentication with mobile phones," in *DIMACS Workshop on Usable Privacy and Security Software*, 2004.

[23] M. Mannan and P. van Oorschot, "Leveraging personal devices for stronger password authentication from untrusted computers," *Journal of Computer Security*, vol. 19, no. 4, pp. 703–750, 2011.

[24] A. Czeskis, M. Dietz, T. Kohno, D. Wallach, and D. Balfanz, "Strengthening user authentication through opportunistic cryptographic identity assertions," in *Proceedings of the 2012 ACM conference on Computer and communications security*, ser. CCS '12, 2012, pp. 404–414.

[25] "Mobile-otp: Mobile one time passwords," http://motp.sourceforge.net/.

[26] I. Duo Security, "Duo security: Two-factor authentication made easy," https://www.duosecurity.com/.

[27] Google, "Google 2-step verification," http://www.google.com/landing/2step/.

[28] D. Balfanz and E. W. Felten, "Hand-held computers can be better smart cards," in *Proceedings of the 8th USENIX Security Symposium*, August 1999, pp. 15–24.

[29] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-is-believing: Using camera phones for human-verifiable authentication," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005, pp. 110–124.

[30] N. Saxena, J. E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel: Design and usability study," *IEEE Trans. Info. For. Sec.*, vol. 6, no. 1, pp. 28–38, Mar. 2011.

[31] M. Xie, L. Hao, K. Yoshigoe, and J. Bian, "Camtalk: A bidirectional light communications framework for secure communications on smartphones," in *Proceedings of the 9th International Conference on Security and Privacy in Communication Networks (SecureComm'13)*, 2013, pp. 35–52.