EAI Endorsed Transactions
on Security and Safety

Research Article ICST.ORG

UAuth: A Strong Authentication Method from Personal Devices to Multi-accounts

Yazhe Wang[1], Mingming Hu[1,*], Chen Li[1]

[1]State Key Laboratory of Information Security Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Abstract

In this paper we present UAuth, a two-layer authentication framework that provides more security assurances than two-factor authentication while offering a simpler authentication experience. When authenticating, users first verified their static credentials (such as password, fingerprint, etc.) on their devices to achieve the local-layer authentication, then submit the OTP-signed response generated by their device to the server to complete the server-layer authentication. We also propose the three-level account association mechanism, which establishes the association among devices, users and services, and then creates a mapping from user's devices to user's accounts. Users can gain access to different service via any device in the association easily. Our goal is to provide a quick and convenient SSO-like login process on the basis of security authentication. To meet the goal, we implement our UAuth, and evaluate our designs.

Received on 29 September 2014; accepted on 15 March 2015; published on 05 October 2015

Keywords: Authentication, Mobile terminal, Multi-accounts

doi:10.4108/eai.5-10-2015.150479

1. INTRODUCTION

With the development and popularity of the Internet, many people today have multiple accounts . Since more services move to the Internet, the number of accounts a user needs to remember is expected only to grow. If one uses different and unrelated passwords for each account, the coming up with secured passwords to remember is a very challenging task for him.

Single Sign-On (SSO) allows users to sign in numerous relying party (RP) websites using one single identity provider (IdP) account. Therefore, users are relieved from the huge burden of registering many online accounts and remembering multiple sets of password. However, it just reduces the problem of securely authenticating to relying parties to the one of securely authenticating to an identity provider. It does not, in fact, address the issue of securely authenticating. Now some popular SSO services (e.g., OpenID [16]) still use the traditional password authentication. An adversary who manages to steal the password in IdP from a legitimate user can impersonate that user to the trusted RPs, which leads to a chain reaction of resource misuse. Recently, some password leaks [12, 13] highlight the current traditional password authentication vulnerability. Though some additional encryption measures have been taken, users transmit the hash of password instead of plain text or transmit above SSL. The emergence of powerful password-cracking platforms [2, 18] or the use of vulnerabilities [3] has enabled attackers to recover the original passwords in an efficient manner, and then an attacker can still impersonate a legitimate user login into the website using the recovered password. We can find that all these vulnerabilities arise primarily due to the sensitive authentication credentials (e.g., password) are verified in the server layer, the credentials are not dynamic and are transmitted over the insecure Internet. Every time when user login into a website, they use the same credentials, so if an attacker steals the credentials, he can impersonate the user at any time without worrying about the password failure.

We propose Uniform Authentication (UAuth), a two-layer authentication framework. In addition to the server layer verification, a local layer verification is provided. The sensitive static authentication credentials are verified in local layer while the dynamic credential is verified after submitting to the server layer. The unpredictability of the credentials and multi-layer authentication makes the attacker have no approach to access the sensitive data, significantly improving the security of the authentication. The UAuth also provides a three-level account association about the mobile terminal, the account in UAuth and the account in SP(Server Provider. eg., Twitter, Facebook), which apparently reduces identity management and

*Corresponding author. Email: Humingming@iie.ac.cn

authentication infrastructure complexity. For example, a user who has an iPhone 5s may choose fingerprint verification as the local authentication method, then use the phone to login into Twitter since they are associated. He can also visit Facebook in the same way. When the phone is not available, he is able to visit these two websites by other device such as a pre-associated USB token which employs the password verification as the local authentication method. FIDO (Fast IDentity Online) Alliance [11] has proposed similar ideas. From its newly published specifications we can find that it concentrates little on Federated Login, and the discussion is only based on the high-level description . We make some improvements from it and develop a system that runs correctly. We also give the detailed implementation.

Multi-screen interaction is a hot topic in the industry nowadays. According to a recent Google study [1], "90% of people move between devices to accomplish a goal, whether that's on smartphones, PCs, tablets or TV", and smartphones are the backbone of our daily media interactions. The interaction and authentication between multiple screens are worth studying. Our UAuth is a proper solution to the interaction and authentication in this scenario.

Our Contributions. We present a two-layer authentication framework: UAuth. In the framework users can utilize a two-layer authentication method - the password authenticates in local layer and the OTP(One Time Password [5])-signed response authenticates in server layer - to login into UAuth and gain access to the SP account securely. In particular, our contribution is as follows:

- *Two-layer authentication.* We provide a two-layer authentication method. It verifies the credentials in the local layer as well as in the server layer, to avoid the problem in traditional authentication aroused by the method that only verifies the credentials in the server layer. Our framework considers the functional and security requirements and designs the system architecture of mobile phone based solution under realistic adversary models.

- *Three-level account association.* We present a three-level account association architecture, which establishes an association between devices, users and Internet services. Users act as the key intermediate connecting level.This method not only maintaining the devices, users and services associated, but also keeping their own independence. By hierarchical of the account association ensure both the integrity of the secure authentication process and the privacy of user data in each level.

- *Implementation.* We present an implementation of UAuth. We use an Android phone as the mobile terminal. After scanning a QR code in the UAuth login page and getting the Wi-Fi information, the phone will establish a Wi-Fi connection with the fixed terminal. Then the user will login into the UAuth after a two-layer authentication, and gain access to the SP account from UAuth at last.

- *Evaluation.* We rate our framework using Bonneau et al.'s framework of 25 different "benefits" that authentication mechanisms should provide [19]. We also compare our work with some other popular authentication mechanisms.

The rest of the paper is organized as follows. In Section 2 we illustrate and evaluate previous efforts at strengthening user authentication and establish the threat model in Section 3. We outline our framework and give some details in Section 4. We present the implementation in Section 5 and evaluate our solution in Section 6. Conclusions are in Section 7.

2. RELATED WORK

Two-factor authentication. When users are authenticating, they have to enter, in addition to the password (something you know), some other authentication factor that they obtain from other sources (something you have or something you are) [21]. Many companies have employed it to strengthen their authentication method, such as Google 2-Step Verification [10]. It utilizes two factors from independent channel when authenticating, and provides sufficient protection against the threats the traditional password faced. But if users reuse passwords across different websites [23], at which point once the attacker gets the password in the site which employs two-factor authentication, they would be able to impersonate the user in other sites which don't employ two-factor authentication [9, 30]. It will also lead to poor user experience when copying the string of the OTP from a mobile phone to the login page. Czeskis et al. present PhoneAuth [20], an authentication method that does not require the operation of the phone. In its strict mode, there is no user interaction necessary during a login, other than typing the username and password. However, without user's operations, the automatic authentication can also lead to potential threats. At the same time, with the increase of number of accounts, the number of devices that authentication requires also increases. It is quite inconvenient either in portability or cost. All these limits enable users to prioritize other convenient methods rather than two-factor authentication.

Fast IDentity Online Alliance. The Fast IDentity Online (FIDO) Alliance was formed in 2012 by several companies (e.g., Lenovo, PayPal). The FIDO alliance

aims to bring different authentication schemes together by providing a set of standards that simplify their adoption and use in web authentication. The FIDO website [11] has published its specifications recently, and some details are provided from it. It aims to complete the verification of password in local device to avoid transmission of the password on the Internet, and use the dynamic data generated by the device as the authentication credentials to prevent attackers to compromise user accounts by eavesdropping, phishing or other means. Since there is no static sensitive data dissemination on the web, login authentication security can be greatly improved. However, it can be seen that users do not have specific methods to manage their accounts in the FIDO service. The over-reliance on the device makes the user helpless when the device is unavailable. They will also feel impotent to manage the devices when users have more than one device.

SAuth. Now Federation Login has developed many new ways, in addition to traditional authentication with the dependencies between IdP and SP. There are new ways that remove the IdP. The boundaries between the IdP and SP are increasingly blurred, website acts as IdP may be SP at the same time. Kontaxis et al. present SAuth [29], a protocol for synergy-based enhanced authentication. Users wishing to access their account on service S also have to authenticate for their account on service V, which acts as a vouching party. But it is obvious that it's a single factor authentication method, and the security has not greatly improved.

Several promising services are now available at various stages of polish, each with their own vision of user identification and authentication. The study by Bonneau et al. in 2012 lists some popular authentication mechanisms and critically analyzes them via a framework of 25 different "benefits" that authentication mechanisms should provide designed by the author. [20] also give an evaluation about their work using the framework. We agree with most of the analysis and rate our system under that framework.

3. THREAT MODEL

One goal of our work is to provide the strong authentication platform - UAuth. We assume the following threat model.

We allow adversaries to obtain the user's password - either through phishing or by social engineering attacks. The browser in the fixed terminal uses the certificate in AP to establish an SSL connection with the server. We assume that the data in the mobile terminal is stored in secure storage, only certain procedures can access their own resources. The attackers can perform software attacks against the terminal and install, modify or compromise all software components installed on the terminal. But it's obviously that they are unable to visit the data belonging to UAuth application in the security storage. The attacker is also able to deploy some malware on the user's machine, such as a keylogger. The malware has the access to the document in the user's machine and they can also visit the data in browser. However, the attacker is not able to simultaneously compromise the user's PC and user's mobile terminal.

Since there will be some sensitive data transmission among the mobile terminal, the user's PC and the UAuth server during the initial authentication step, the attacker may directly access the data easily. But the frequency of these cases is low, we choose to focus on the subsequent case after the initial step.

4. ARCHITECTURE

4.1. System Model

Our system model is depicted in Figure 1. The design consists of several categories of components: Authenticate Plug-in(AP), mobile terminal, Validate Server(VS), Validation Cache(VC),UAuth Web Server(WS), Server Provider(SP), wherein WS, VS, VC composed UAuth. They have comprised the three-level association system: the binding, authorization, management between the mobile terminal identification information (OID), the user account in WS (UID) and the user account in SP (SPID). VS provides registration functions. All of the mobile terminals need to be registered in it prior to use. It will negotiate the OID and the key that used to generate the OTP with mobile terminals, store these data and update it to the VC. VC is a caching server for the data in the VS, which is physical proximity to the WS. Each time after WS submits the OID and response, it verifies whether the OID and response correspond or not efficiently. WS is the core part of UAuth, with which user can manage their UID and account binding (UID and OID binding, UID and SPID binding). Users need to get the credential of SPID from WS when authenticating to SP. The mobile terminal provides a local layer authentication method. The terminal registers itself to the VS by negotiating the key used to generate the OTP and telling the server it's OID. It would not generate the OTP to achieve the server layer authentication unless the local layer authentication is successful. AP is a customized functional component installed in use's fixed terminal, and it helps to complete the authentication by establishing a communication between the WS and user's mobile terminal. It also informs the WS of the presence of a mobile terminal, and relays the encrypted authentication stream to the WS. SP is the entity that provides Internet services, it needs to establish a trust relation with WS and build a secure communication channel.

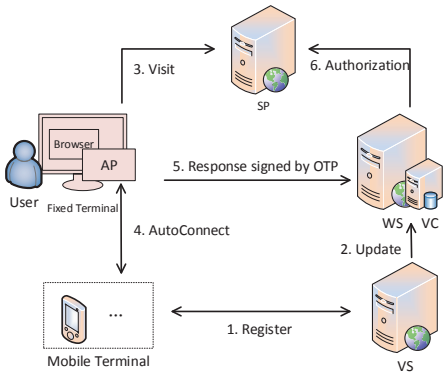

Figure 1. System model

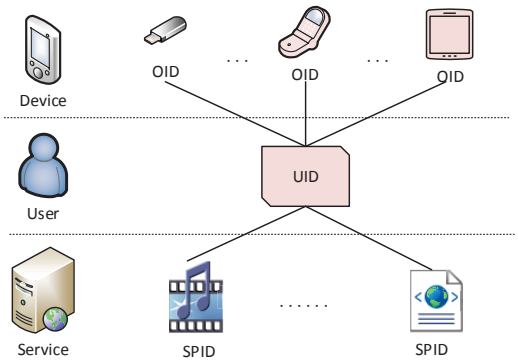

Figure 2. Three-level account association

4.2. Three-level account association

The three accounts correspond to the OID, UID and SPID. Our primary goal of the design is to protect the privacy in each level and achieve an easy management mechanism of the authentication in multiple identities.

OID is regarded as identification information of the mobile terminal. It is generated in the initial process when mobile terminal registers to the VS. Each terminal has a unique OID. When users are authenticating, the terminal will provide the OID to identify itself. UID is the user's account in the WS, at the same time we regard it as the user's identity. Users register their own UID on the WS server. UID information is the basis of the framework and is saved in the WS, while becoming the associate middleware of OID and SPID. SPID is the user's account in SP, such as a Gmail account. The registration of SPID is performed in the SP, while the information of SPID is stored in the SP, and controlled by SP.

Users can bind their OID with UID, and comlete the mapping from terminal to user. Then they can visit their account in the WS using their terminals. The security of the authentication is improved since the user's personal terminal is involved. Users can also bind their UID with SPID, so the mapping from user to Internet service is established, and they may access the binding service with UID easily. SPID is identifiable information to SP. It makes access control retained in SP, while the authentication is taken over by UAuth. Meanwhile, it ensures there is little change to the SP, which minimizes the cost when docking with SP. This mechanism increased the system's ease of deployment, and make the control party dispersed in various SP, to some extent, protecting the user's privacy. After the two bounds, the three-level account association is built, using UID as the medium, users can use their own terminal to get access to the Internet service. Moreover, all the association is controlled by WS, and WS is able to create a management module to complete the multi-binding, which means users are capable of binding more than one terminal or SP to their own UID. As can be seen from Figure 2, multiple OID as well as SPID are bound to UID. So when authenticating, user can choose the appropriate device to visit the Internet service he wants. Three-level account association has greatly increased the convenience and flexibility of authentication.

4.3. Two-layer authentication

A complete login process of UAuth consists of two-layer authentication, the local layer authentication of static authentication information verified in the mobile terminals, and the second layer of response server-side validation in WS. Because user's personal device is varied, they may select a smartphone, a USB token, a fingerprint reader or others as an authentication terminal. We design various local layer authentication methods. It enables users to choose an appropriate method depending on the device they are using. For example, user can use password verification when they use their phone or fingerprint to authentication with their fingerprint reader. A successful verification of user's password or other authentication information in local layer proves that user has the corresponding authentication device, and the device is not used by an impersonate user. Since the local authentication credential is only known or owned by the legitimate user, even if the attacker gets the device will failed to login because lack of local authentication credentials, and if the attacker gets the local authentication credentials by some means (e.g., social engineering), he will also fail because he is not able to fake the certain device. The attacker may simultaneously compromise the user's local authentication credentials and user's device, this situation is rare. Even it happens, the user can login through the fallback mechanism and revoke the device. The server layer authentication of OTP-signed response generated by the mobile device, indicates that the user who is authenticating and proves that the legitimate user is using the correct device to

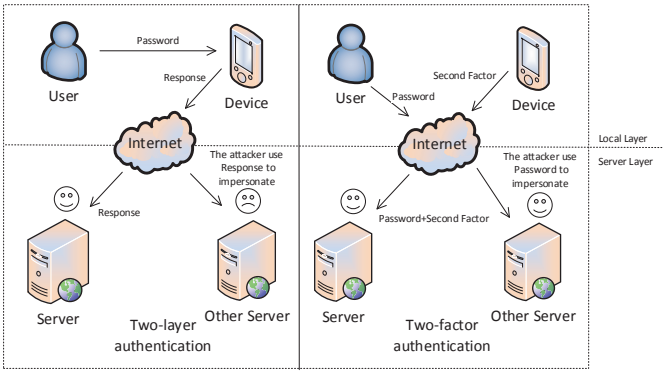

Figure 3. Two-layer authentication and two-factor authentication. In two-layer authentication, the attacker intercepts the response which successfully authenticated and impersonates the user to the server, due to the expiry of OTP, the login request is denied and the attack is failed. In two-factor authentication, the attacker intercepts the static password and impersonates the user to the server which do not employ two-factor authentication. Due to the reuse of password, the login request is accepted and the attack is successful.

login again. Since OTP is changing with time, even if an attacker cracks the encrypted channel and gets the response, OTP at this time is likely to have been in ineffective. The signature of the data in the transfer process also protects the integrity of the data, therefore an attacker is difficult to modify or forgery the data. The combination of the two layers, either prevents the attacker to steal the static password when transmitting in the insecure network, or protects the data in user's device from being read and used when the device is lost, at the same time the response validation ensures the security in authentication process.

Two-layer authentication is similar to two-factor authentication, but it has its own advantages. Two-factor authentication may encounter the problem discussed in Section 2, the reuse of a stolen password may lead to impersonate. What's worse, a study has shown that it can push users to weaker passwords [28], which leads to a potential dictionary and brute force attacks. In two-layer authentication, the verification of the static password is performed from the server to the local device, so that the static password is not transmitted in the complicated network, ensuring security while avoiding the problems above.

4.4. Protocol Details

When users are authenticating with SP, they have to successfully authenticate to our UAuth via a two-layer authentication firstly: after the local layer verification on the mobile terminal, users will forward the response generated by the mobile terminal on-the-fly to the UAuth, then obtain the credential of SP account and login to SP. As shown in Figure 4, a complete authentication requires these steps. Since in our framework the mobile terminal is not limited to a certain type, which can be a variety of devices, in this paper we use the smartphone as an example to illustrate. The steps 1-8 comprise a UID login process while steps 8-12 are the authorization process for SPID.

1. User launches a browser B on the fixed terminal, and visits the website SP, choose to sign in with U. The page will be redirected to the login page on WS.

2. B and WS establish an SSL session; let K_{WB} be the established SSL secret key. WS will generate a QR code with a random string S in it and display it on the login page,so the content of the QR code is S. The string S will be transmitted to the authentication plug-in (AP) installed in the fixed terminal at the same time. AP parses S from WS with a certain function F designed by us and gets the results such as Wi-Fi SSID , password, then establishes a Wi-Fi access point using those results. The results are:

$$SSID, Password, Address, Port \tag{1}$$

   Wherein the $SSID$ is the name of the access point, $Password$ is the password for the access point. $Address$ is the IP information of the access point and $Port$ is the connect port that the mobile terminal should connect to.

3. The user first inputs his local password in his mobile terminal M to unlock his account, which is the local authentication procedure. Then open the application and scan the QR code displayed on the page. Get S, parse it using the same function F and retrieve the information in it, with the information M and fixed terminal establish an SSL session at last; let K_{BM} be the established SSL secret key.

4. M generates an OTP using the pre-negotiated key K. M signs and encrypts the data below and sends message (3) to B.

$$Sig = HMAC\{OID, S\}_{OTP} \tag{2}$$

$$\{OID, S, Sig\}_{K_{BM}} \tag{3}$$

   Wherein S is the string from the QR code.

5. From message (3), after SSL decryption, AP obtains the information. AP (via SSL) encrypts the information with K_{WB}, and forwards the result (5) to WS through the SSL session.

$$Sig = HMAC\{OID, S\}_{OTP} \tag{4}$$

$$\{OID, S, Sig\}_{K_{WB}} \tag{5}$$

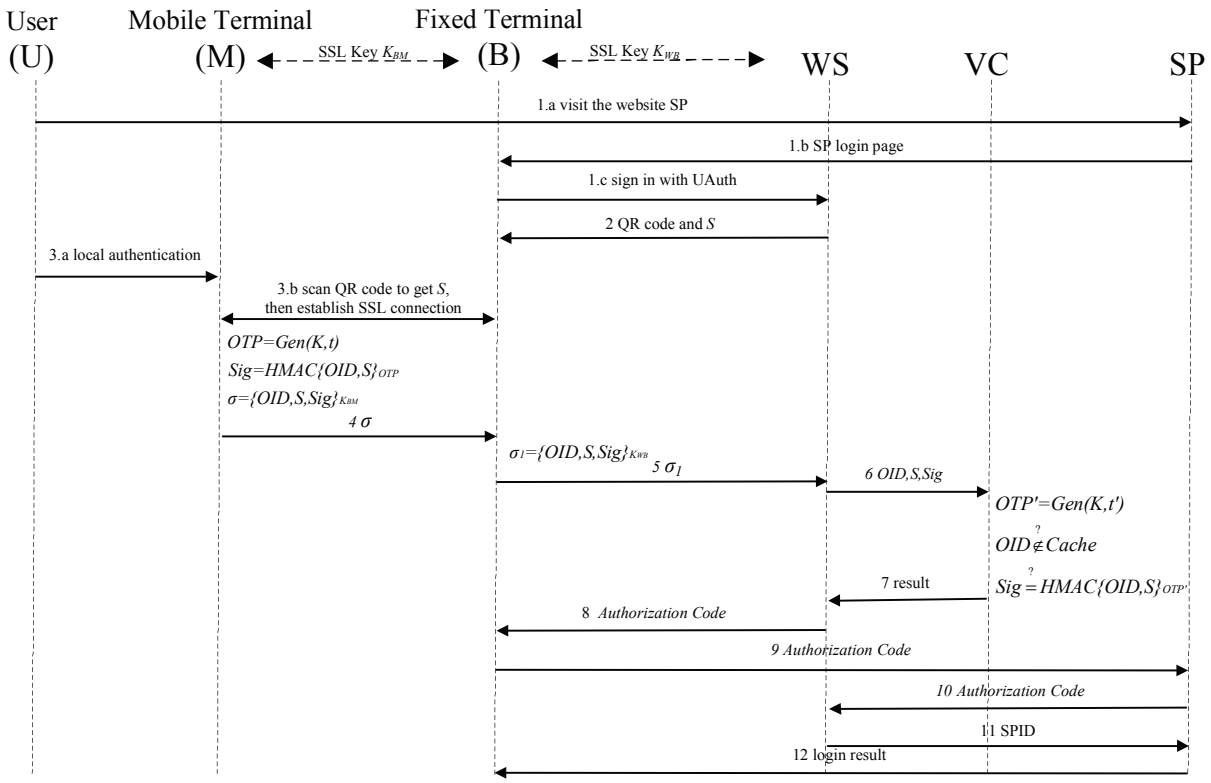

Figure 4. Protocol Details

6. WS receives the message from the AP, after SSL decryption, and verifies S that it is generated by WS and it is not expired. Upon successful verification, WS forwards the OID, S and Sig to VC and submits a request to verify the validity of these data.

7. VC finds the secret key corresponding to OID in the database and use the secret key K to generate OTP', then signs the S, OID using OTP' in the same HMAC algorithm and compare the result with Sig, responds the comparison result to WS.

8. If validation passes, it means that the UID account which OID bound to has login successful. WS page displays the SP accounts that the UID has bound with. The user is required to select the SP account which he wants to ues. WS will randomly generate an *Authorization Code* to map to the select SP account. Finally WS returns a link that will redirect to SP with parameters, which contains *Authorization Code*.

9. The B redirects the link to SP.

10. SP verifies the validity of the request to ensure that the redirection and the login in step 1 are from the same session. After that, the SP will submit the *Authorization Code* to WS through a pre-established secure channel to request authorization of SP account.

11. WS returns the selected SP account that corresponding to the *Authorization Code*.

12. SP checks the result from WS and displays success or failure on B.

Initialization. Before login with this method, users have to initialize the mobile phone and the fixed terminal.

The user installs the authentication application in the mobile phone and initializes it. It will connect to the VS server to register itself, and negotiate to get the user's mobile phone private certificate. The certificate is used to identify itself, and establish a SSL session with fixed terminal. It also negotiates to get the OID, and get key K which is used to generate OTP. These data are stored in the secure storage of the mobile phone, so only the application can access the data. After successful registration, VS will update the registration information in VC.

The user has to install the AP in fixed terminal, which assists to achieve a complete login process. A X.509 certificate will be installed to the fixed terminal that used to establish the SSL session between WS and AP as well as the mobile phone and AP. The AP also has other functions such as establishing a Wi-Fi access point based on the string from WS.

Binding and unbinding. The binding process contains two parts: UID and OID binding, UID and SPID binding. We assume that in the UID registration, the UID must bind an OID. If the OID and UID are not bound, then the UID registration also failed. In subsequent process user can bind multiple OID to an UID in the manage page of WS. Once there is one mobile phone whose OID is bound to user's UID, the user can login successful. UID and SPID binding can be carried out at any time, as long as the user has an UID and a SPID. The UID and SPID will be bound when the SP proofs that the current user has managed to successfully authenticate with the SPID. The association relationship will be recorded in WS server.

In the bind step of UID and OID, the user first authenticates the UID with the pre-bound mobile phone, and chooses to bind a new OID in the manage page in WS server. The WS will call the function in AP to create a Wi-Fi access point and generate a QR code that contains the Wi-Fi link information. By scanning the QR code, the mobile phone gets the information and establishes a SSL session with AP, then communicates with WS through AP. It also sends its specific device info generated by the authentication application to WS. The step will be complete after the user confirms that the device info of the mobile phone displayed on the WS page is the same as those displayed on the mobile phone screen. The whole process is similar with the steps 1-8 in Figure 4.

In the bind step of UID and SPID, user can choose to bind a SPID in the manage page in WS while the UID has logged on, then user has to login his SPID to proof ownership of the SPID. WS will check it and complete the process.

The process of unbinding is similar to the process of binding. It can be accomplished through the manage page when the UID has logged on, but there must be at least one OID left which is bound with UID when UID and OID unbind.

Fallback mechanism. It may happen that user's mobile phone becomes unavailable for some reasons (e.g., loss or out of power) and can not be used to authenticate. Our fallback mechanism allows user to log in just with something that "they know". We will send an e-mail to a backup address, and then the user operates under the direction of email, or ask questions about the message user left before. These mechanisms ensure that the user is able to access to their accounts while the mobile phone is invalid or missing.

Deploy of server. From Section 4.3 we can find the mobile terminal that UAuth support is varied. Taking into account these devices may come from a variety of vendors, we set up a VS server as the uniform registration server.VS receives the mobile terminal registration information ,which is synchronized to VC. VC is physical proximity to WS that WS can access to VC security, as well as to ensure a rapid response. VS updates its data to VC once there is a new registration or other change to ensure the consistency of information.

VC is mainly established to be able to respond the authentication request from WS quickly and accurately. This is also the reason why it is designed to be physical proximity to WS. Since OTP is time-sensitive, the verification must be completed in a short time. The time of verification will be less when VC is involved. WS will first issue the verification request to VC. If the cache hits, VC can directly determine and respond the result. If not, it will issue the request to VS and update VC at the same time. This allows to reduce the pressure on VS of responding the authentication request, while increasing the authentication accuracy. We recommend the adoption and implementation of the VC in the increasingly complex Internet environment now.

Mobile phone and fixed terminal communication. The mobile phone establishes a connection with the fixed terminal using the information from the QR code. If the attacker uses his device to connect to the fixed terminal before the legitimate user, obviously, the current login account is the attacker's account, and the attack is meaningless. But when the user is in the binding process, its SPID may bind to the attacker's UID under the attack, so we design a binding mechanism that users participate in: before binding, user is required to confirm that the message displayed on the browser is same to these displayed on the mobile phone's screen. If not, then it indicates that the binding process encounters certain problems, and the process is aborted. However, in practice, if we adjust the QR code being displayed in a proper size, the attacker will have trouble in scanning the QR code due to the factors of light and focus, which also increases the difficulty of the attack.

The mobile terminal may be other devices such as USB tokens, so the connect method between the mobile terminal and the fixed terminal is varied. Generally, direct physical connection like USB tokens links to the fixed terminal is secure. With the protection of AP, the attacker will have no chance to attack the channel. But the wireless means (such as Wi-Fi, Bluetooth, etc.) are vulnerable. We can increase the security by the way of user participation.

Security discussion. In the protocol, the VS sends a fresh random string S to the browser, and the browser forwards S to M. When the user starts a session with the VS, an attacker may start a parallel session from a different fixed terminal and grab the message (5) to login as the user. However, as the VS generates a new S for each login session, sending message (5) to the VS by any entity other than the corresponding browser would cause a login failure. When the VS

receives message and verifies the signature, it believes that the communication involves the M and the user, and the current protocol run is fresh. All the messages are cryptographically linked by S and the signature. This chaining prevents replay attacks. Since there is no static password in the messages, the protocol address risks such as password reuse, server database leak and phishing.

For the authentication phase in UAuth, the browser simply forwards messages between the mobile terminal and server. All the credential that the malware can get is time-sensitive, the attacker cannot make use of it in the next login process. But the malware in the fixed terminal may freely create the Wi-Fi access point. This may expose DOS attacks, and make the protocol unavailable.

The malware in the mobile terminal may be a keylogger, so it can grab the static credentials. But it is meaningless because the attacker cannot get the phone. The authentication data such as K is encrypted and stored in a secure storage so normal malware could not obtain it easily. However, the attacker may gain access to sensitive data file and perform an offline dictionary attack on it. What is worse, unconscious user may give the malware root privilege, which makes it easier to attack. The Trusted Execution Environments (TEEs)[17], such as ARM TrustZone, which provide isolated execution of applications and secure storage of credentials helps to prevent these attacks[31, 32]. We are currently in the beginning stages of building a system to handle this case and leave further discussion as future work.

In the process of authentication, the protection of the secure channel and OTP signatures can prevent the data from being modified or stolen in the transfer process. Response validation ensures the timeliness of the authentication and the authentication data cannot be replayed. These considerations have already discussed above. Now we discuss some other considerations in the authorization process. In the authorization process, the *Authorization Code* is the key element, so it's essential to protect it. Think that the *Authorization Code* may be tampered by an attacker in the step that WS page redirects to SP page, the attacker could replace the *Authorization Code* into their own, or intercept the *Authorization Code*. To deal with it we add a *State* parameter in each redirect link, which is an unpredictable value stored in the session, and the server will validate it when callback using. It can guarantee that the authorization operations are completed in one session. When the *Authorization Code* and *State* are validated, it can be sure that: (1) the *Authorization Code* came over the same session as the user request, not from other users. (2) *Authorization Code* has not been replaced or modified by attacker. (3) the user has been authorized for SPID, SP is required to query the specific SPID from WS.

User experience. The implementation is designed under the guide that "Minimizing the user's operations while ensuring the safety". We carefully designed our system, and try our best to change the user's keyboard stroke to mouse click. Users only need to enter the password and use the mobile to scan the QR code once, which will have a better user experience than the traditional two-factor login service that includes typing a username and password, and copying a string from a smart phone app to the login page. In our UAuth, users only need to enter the password and scan the QR code, the rest will automatically be done by interact of mobile phone and fixed terminals. Through reducing the operations of user recognition and input, it is sufficient to provide positive user experience.

5. IMPLEMENTATIONS

In order to demonstrate the feasibility of UAuth, we implemented the system discussed in Section 4. We built an authentication framework prototype and simulated the service requests. The prototype includes Authenticate Plug-in(AP), Validate Server(VS), Validation Cache(VC),UAuth Web Server(WS) and mobile terminal application.

Authenticate Plug-in. The idea of designing the AP comes from the definition of the FIDO alliance framework. We inherited the idea and implemented it. We designed a chrome extension, mounted on the Chrome browser in fixed terminal, which acted as the role of authentication plug-in. The extension function mainly includes two aspects: establishing the Wi-Fi access point and receiving data from a mobile phone. Since regular Chrome extensions do not have the ability to build a Wi-Fi access point and communicate with the mobile phone, we write a NPAPI plug-in to complete the function above. We employ the functions provided by the Wlanapi library which can operate the Wi-Fi to complete the establishment of the Wi-Fi access point. It will create a Wi-Fi access point using the results parsed from the random string from WS. Users can use their mobile phone to scan the QR code and get the information, with which the mobile phone could establish a connection with the AP. AP will listen to the connection request from the mobile phone, in order to prevent the plug-in be blocked for listening, which may lead the browser to crash. This function will be placed in a new thread. The AP carries a preset UAuth X.509 certificate. When building a SSL connection with a mobile phone or WS, it will use the certificate as an identity certificate. When the AP receives the login data from the mobile terminal, it will call the NPN_PluginThreadAsyncCall() function from NPAPI

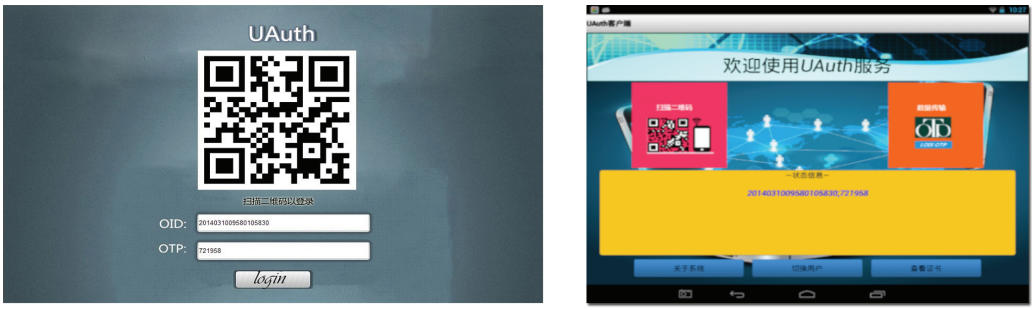

Figure 5. Our Implementation of the UAuth login page and terminal application.

to call the JavaScript function in the web page, the data will be displayed in the login page and then login.

In total, the AP consisted of 1000 lines of C++ and supported the Chrome browser on Windows platform.

Mobile Phone. Our application is developed in the Android 4.2.2 development environment at first, and then we upgrade it to the latest Android 4.4. The UI component of the app is represented by the keyboard and display drivers already present in the Android. We also employ the Wi-Fi connection API that the Android provided; it will establish a SSL2.0 session with the fixed terminal when the link parameters are ready. The QR code scanning function is derived from the API provided by ZXing [7]. The certificate in our app is stored in the BKS certificate repository, and the K used to generate the OTP is encrypted with the private key of app and stored in the secure storage area.

The application required 1000 lines of Java code.

Server. We completed the WS server, VS server, VC server and a SP server for evaluation.

All the servers are mostly implemented in Java Server Page (JSP), they have achieved all the functions detailed above. we use Bouncy Castle [6]for the server-side cryptographic operations.

The complete implementation of those servers consisted of 5000 lines of Code.

6. EVALUATION

We evaluate our system using Bonneau et al.'s framework of 25 different "benefits" that authentication mechanisms should provide. We analyzed our work from three aspects: the usability, deployability and security. We also include the incumbent passwords, Google 2-step verification, PhoneAuth and SAuth as a baseline. The results of our evaluation are shown in Table 1.

Usability. In the usability arena, since there is little user interaction necessary during a login, and existence of private mobile terminal certificate enabling that multiple mobile terminal can be bound to one UID, at the same time the UID can bind a plurality of SPID, so it is *Scalable-for-Users*. It is *Easy-to-Learn* and *Easy-to-Use* because users only need to enter a password and use the mobile to scan the QR code once, which also makes it does not have *Memorywise-Effortless*, *Nothing-to-Carry* and *Physically-Effortless*. We rated it as somewhat providing the *Infrequent-Errors* benefit since they will cause an error if the wireless connection does not work or the mobile terminal is not available. Similar to Google 2-step Verification, it somewhat provides the *Easy-Recovery-from-Loss* because of the inconvenience of having to replace the phone is then compounded by the fact that the lost phone also holds the secrets.

Deployability. Accessing the deployability benefits comes down to evaluating how much change would be required in current systems in order to get our proposed system adopted. In UAuth system, since the authentication is completed by the phone in conjunction with local password, and requires AP to play a role in the process, *Accessible*, *Negligible-Cost-Per-User* and *Browser-Compatible* is somewhat provided. For general SP server, it only needs to be modified to be compatible with UAuth, the change is little so it is somewhat *Server-Compatible*.

Security. Since the UAuth employs both local password and OTP, it can meet most of the security properties. When faced with phishing and eavesdropping, it would have a good security performance. However, some real-time attack makes it vulnerable. The adversary may steal the sensitive data in this attack, so it somewhat provides *Resilient-to-Internal-Observation* benefits.

Conclusion. In Table 1, it can be seen that UAuth owns a good performance in the evaluation and get a high score in the ease of Usability, Deployability and Security.

Performance. We evaluated the performance of our implementation of UAuth. Measuring the performance of our login scheme is a complex task. So we just select the more representative argument - time - to be evaluated. We made the following measurements: (1) the time required to establish the Wi-Fi access point, (2) the time required to establish the connection between

Table 1. Comparison of UAuth against password, Google 2-step Verification, PhoneAuth and SAuth using Bonneau et al.'s evaluation framework. 'y' means the benefit is provided, 's' means the benefit is somewhat provided, while blank means the benefit is not provided

	Usability								Deployability						Security										
Scheme	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically-Effortless	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	s	y	y	y	y	y	y	y		s						y	y	y	y
Google 2-Step Verification			s		y	s	s	s	s			y	y		s	s	y	y		y	y	y	y	y	y
PhoneAuth-strict		s			y	y	s	y	y	s	s	s	s	y	y	y	y	y	s	y	y	y	y	y	s
SAuth			y		y	y	s	y	y	y		y	s	y		s						y	y	y	y
UAuth		y			y	y	s	s	s	s	s	s	s	y	y	y	y	y	s	y	y	y	y	y	y

the fixed terminal and the mobile terminal, and (3) the time required to complete the UID login process, that is to say we measure the time between the step 1 to step 8. This can reflect the total performance since the step 9 to step 12 spends little time that can be ignored (usually less than 0.2s). The result of our measurements averaged over 50 protocol runs are shown in Table 2. Below we give a discussion about these data.

Table 2. Performance of UAuth

	Wi-Fi AP	Connection	Total
Avg. Time(s)	1.1	4.2	11.4
[Min,Max](s)	[0.7,1.8]	[2.5,9.2]	[6.8,19.3]

Removing the time of user operations (such as taking their phone out of their pocket and unlock it), the total time of a complete login process mostly spent on establishing a Wi-Fi access point and the connection between the mobile terminal and fixed terminal. Building a Wi-Fi access point is relatively simple: call a series function and wait for the access point established. The time of this procedure most depends on the hardware and the operating system on the fixed terminal, but it will not take too long. In our evaluation, the time is about 1 second which is acceptable to users.

This is perhaps the most important type of overhead incurred: the fixed terminal has to establish a SSL connection to the phone. We believe that the process is started from the time user begin to scan QR code to time the SSL connection establishes successfully. As a baseline comparison, we measured how long the mobile phone took to obtain the information in the QR code (open the scanning function and scan the QR code) - an average of 1.2 seconds. Repeating the same login while also obtaining the information increased the average time to 4.2 seconds. The additional 3.0 seconds are mostly spent in establishing the SSL connection, including connecting to the access point and establishing the SSL connection. Connecting to the access point costs a large amount of time. The app on the mobile phone has to invoke the Wi-Fi connection service in the Android. The response of the Android to this call will greatly affect the final performance. The system will find the corresponding access point which has the same SSID from the broadcast information, and connect to it with a proper configuration. The process is slow, what is worse, if DHCP is used to allocate IP address, it will take longer time. Since we have included the address of the fixed terminal in the QR code, if we set a static address to the mobile terminal which is in the same subnet with the fixed terminal, the speed will be significantly improved, finally we get the average time - 11.4 seconds.

Note that for a user that uses 2-factor login service, he will type a username and password, and copy the OTP from the mobile phone to the page, which will take an average login time of 24.5 seconds [20]. Login has sped

up with our system, while at the same time improving the login experience to a simple input and scan and enhance security.

7. CONCLUSION

We have presented UAuth, a two-layer authentication framework. It enables users to enjoy the security benefits of using the physic device to authenticate: use the OTP generated by the device in the two-layer authentication. At the same time, users receive the convenience of the SSO-like login: user can visit more than one SPID with their UID in the three-level account association. Specifically, in local authentication, users can choose a correct way to authenticate depending on their own devices. For example: if the user has a USB token, he can insert the token in the fixed terminal and use the password to unlock the token with the help of AP, this would be a proper local authentication method. And if the user has an iPhone 5s that has the fingerprint-scanner, he can choose the fingerprint to be verified in the app as the local authentication method, such approaches are feasible. The server and user are free to choose the local authentication method. The server even can employ the UAuth as a validation factor in their own two-step verification, combining with other authentication methods and get a more secure process. The variety of personal devices and its flexibility also make our UAuth have a good performance in the multi-screen interaction and authentication.

We implemented and evaluated UAuth, and we get a conclusion that the UAuth has a relatively good performance in safety and user experience, it will enhance the current authentication technology on the web today.

Acknowledgments. This work has been supported by National Natural Science Foundation of China (Grant No.61202476); the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06010701, XDA06040502).

References

[1] The New Multi-screen World: Understanding Cross-platform Consumer Behavior,http://services.google.com/fh/files/misc/multiscreenworld_final.pdf
[2] Update: New 25 GPU Monster Devours Passwords In Seconds,https://securityledger.com/2012/12/new-25-gpu-monster-devours-passwords-in-seconds/
[3] SSL, GONE IN 30 SECONDS, https://media.blackhat.com/us-13/US-13-Prado-SSL-Gone-in-30-seconds-A-BREACH-beyond-CRIME-Slides.pdf
[4] HOTP: An HMAC-Based One-Time Password Algorithm,http://tools.ietf.org/search/rfc4226
[5] TOTP: Time-Based One-Time Password Algorithm,http://tools.ietf.org/html/rfc6238
[6] The Legion of the Bouncy Castle,http://www.bouncycastle.org/
[7] ZXing,https://github.com/zxing/zxing
[8] Multi-factor Authentication,http://en.wikipedia.org/wiki/Multi-factor_authentication
[9] The Domino Effect of the Password Leak at Gawker,http://voices.yahoo.com/the-domino-effectpassword-leak-gawker-10566853.html
[10] Google 2-Step Verification,http://www.google.com/landing/2step/
[11] FIDO Alliance,http://fidoalliance.org/
[12] Twitter detects and shuts down password data hack in progress,http://arstechnica.com/security/2013/02/twitter-detects-and-shuts-down-password-data-hack-in-progress/
[13] 123456: Millions of Adobe hack victims used horrible passwords,http://www.pcworld.com/article/2060825/123456-millions-of-adobe-hack-victims-used-horrible-passwords.html
[14] The OAuth 2.0 Authorization Framework,http://tools.ietf.org/html/rfc6749
[15] What is OpenID,http://openid.net/get-an-openid/what-is-openid/
[16] OpenID Authentication 2.0,http://openid.net/specs/openid-authentication-2_0.html
[17] Building a Secure System using TrustZone Technology,http://www.arm.com
[18] Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J.: Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In: IEEE Symposium on Security and Privacy, pp. 523-537 (2012)
[19] Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes. Technical Report UCAM-CL-TR-817, University of Cambridge, Computer Laboratory (March 2012); Shorter version appears in Proceedings of IEEE Symposium on Security and Privacy (2012)
[20] Czeskis, A., Dietz, M., Kohno, T., Wallach, D., Balfanz, D.: Strengthening user authentication through opportunistic cryptographic identity assertions. In: Proceedings of the 2012 ACM CCS, pp. 404-414 (2012)
[21] Adida, B.: BeamAuth: two-factor Web authentication with a bookmark. In: ACM Computer and Communications Security, pp. 48-57 (2007)
[22] Florencio, D., Herley, C.: A large-scale study of web password habits. In: 16th International Conference on World Wide Web (WWW), New York, NY, USA (2007)
[23] Ives, B., Walsh, K.R., Schneider, H.: The domino effect of password reuse. Commun. ACM 47(4), 75-78 (2004)
[24] Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In Proceedings of the second ACM workshop on Digital identity management, pp. 11-16 (2006)
[25] Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J. C.: Stronger Password Authentication Using Browser Extensions. In Usenix security, pp. 17-32 (2005)
[26] Marforio, Claudio., Karapanos, Nikolaos., Soriente, Claudio.: Smartphones as Practical and Secure Location Verification Tokens for Payments. In: NDSS '14, pp. (2014)

[27] Florêncio, D., Herley, C.: One-time password access to any server without changing the server. In Information Security, pp. 401-420 . Springer Berlin Heidelberg. (2008)
[28] Wimberly, Hugh., Liebrock, Lorie M.: Using fingerprint authentication to reduce system security: An empirical study. In:Security and Privacy (SP), 2011 IEEE Symposium on, pp. 32-46 (2011)
[29] Kontaxis, Georgios., Athanasopoulos, Elias., Portokalidis, Georgios., Keromytis, Angelos D.: SAuth: protecting user accounts from password database leaks. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 187-198 (2013)
[30] Cheswick, William.: Rethinking passwords. In: Communications of the ACM, pp. 40-44 (2013)
[31] Saroiu, S., Wolman, A.:I am a sensor, and I approve this message. In Proceedings of the Eleventh Workshop on Mobile Computing Systems and Applications, pp. 37-42 (2010)
[32] Liu, H., Saroiu, S., Wolman, A., Raj, H.: Software abstractions for trusted sensors. In Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. 365-378 (2012)