Opportunistic protected login: Next step in traditional password based user authentication

Rakesh Pandit
Department of Computer Science
University of Helsinki
Helsinki, Finland
rakesh.pandit@cs.helsinki.fi

Abstract—Password based authentication faces security related threats from phishing, server compromise and man-in-the-middle attack. Despite the poor security, it has been the primary method of user authentication on web since a decade now. This paper is a systematic review of a proposal, by Czeskis et al., which aims to provide opportunistic protected login for user authentication, for web services without changing user experience. This proposal involves using a personal device, other then the machine on which user is accessing the service, as a second factor authentication device. In addition, this solution involves using public key cryptography to provide secure channel, between client and server, to which authentication tokens can be attached. The solution aims not only to make first login secure but also prevent user from phishing and certain variants of man-in-the-middle attack. In addition to discussing the proposal, this paper is an effort at evaluating the assumptions, threat model and assembling open problems related to proposal.

Keywords—Communication system security; Information security; Authentication;

I. INTRODUCTION

Web services today use password based user authentication which involves using TLS protocol [7] to encrypt message exchange. Server issues a cookie after validating password from client, which in turn is used in subsequent requests from client to server. This method of user authentication is common, and provides best user experience, but has been target of many security attacks [11]. Two of the common attacks are phishing attacks or man-in-the-middle attacks. In both cases the attacker can get username and password or corresponding cookie from user and use that to authenticate itself to web services and get access to critical information. In 2011, there were cases of DigiNotar issuing certificates to attackers [10], for domains attackers didn’t own, and these certificates were then used by attackers to access confidential user information from different services. An attacker who can manage to issue a new certificates via compromised CA authority, can be a threat to users subscribed to services for which these certificates are issued. There are numerous cases of phishing happening daily against web services using old user authentication mechanism based on username and password [12]. It is common for same users to use same username and password for different web services. The level of security for not so important web services can lead to security breach and can compromise other web services being used by same users.

Since last decade there has been number of attempts to solve user authentication but none of them has been able to successfully replace current password based user authentication [5]. The proposal, by Czeskis et al. [1], aims to provide a replacement to current user authentication without changing the user experience but solving major concerns of user authentication information theft by attackers. The solution uses mobile devices as second factor authentication and binds cookie issued by service provider with a secure channel. It aims to keep user experience unchanged by avoiding adding any new user interface events. The proposal aims at providing 2-factor secure login, called opportunistic protected login. As the name suggests, authentication is opportunistic rather then forcing it every time and this depends upon whether all conditions to protected login are met or not. If mobile device is near machine via which user is trying to access the service, then more secure login takes place. In case machine via which user is accessing service cannot detect the mobile device, then user authentication reverts back to cookie based authentication. Based on the type of authentication service provider can limit available services or prevent user login. Establishment of secure channel with service provider and users prevents attacker from exploiting user even after stealing username and password from user via phishing or man-in-the-middle attack. The proposed solution uses public-key cryptography to create secure channel between user and service provider using TLS origin bound certificates [2]. These secure channels are created per origin i.e one for each service provider and are automatically selected at the login time. So even if the user is using same password for different web services and one of them has been compromised, the attacker cannot use same username and password without setting up a secure channel.

A few web service providers like Google and Facebook etc have already been using improved cookie based authentication [13] [14]. They register a user device on first login and subsequent logins from same device are considered
more secure. But first login when device is registered is still unprotected according to this proposal.

This paper provides a systematic review to a proposal by Czeskis et al. [1], which aims at solving some of the major issues with user authentication based on TLS without changing the user experience. By not changing the user experience, it means that user enters just username and password to authenticate itself to concerned service. In this paper we try to evaluate different assumptions, claims made by authors regarding their assessment of proposal against Bonneau et al.’s framework and list down open problems which are critical to success of this proposal.

The paper has been divided into number of sections. Section I discusses goals of the proposal, threat model and assumptions. Section II explains the architecture of proposal and explains in detail different aspects of design. Section III explains already existing experimental results which support some of aspects of solution. Section IV lists down possible issues, open problems and more experimental requirements.

II. THREAT MODEL, GOALS AND ASSUMPTIONS

In this section we discuss the threat model this proposal [1] aims to counter, goals which aims to achieve and certain assumptions taken. The proposal assumes that attacker can steal credentials via phishing, compromised web service or other means. Attackers can create a server which acts as a web service used by different users and provide an interface which replicates the interface provided by original web service. It is common for users to use same credentials for more then one service. Attackers can get hold of credentials from one of those compromised services. In addition to credentials, attackers can also have a valid TLS certificate via compromised CA authority or other means, and can use those certificates to start man-in-the-middle attack between users and related web services. Attackers can get private key from compromised CA servers and start making certificates for different domains and use them to attack corresponding services provided by domain. Threat model also includes attackers who can steal cookies and try to use them later for authentication and act as if original user is using them. This attack is possible in TLS based password authentication. Threat model includes some kind of malware e.g. keyloggers.

The solution aims to achieve a superior level of user authentication solution which solves above mentioned threat model. Proposal aims to achieve following goals in addition to solving threat model.

- Don’t change user experience at all. That is, user just enters credentials via user authentication interface and doesn’t deal with any other interface. The proposal aims at using mobile phone as second factor authentication device on first login. But this doesn’t involve user involvement and browser automatically interacts with mobile device at backend.

- Use public key cryptography to solve man-in-the-middle (MITM) attack [2]. MITM attack has been solved by public-key cryptography long ago but strangely current password based user authentication solution doesn’t use it at all. This is achieved by using TLS origin bound certificates.

- Use second factor authentication when user mobile device is in vicinity of machine via which user is trying to authenticate and do this without any user intervention. If device is not available then fall back to unprotected login.

- During the first login at each new device usual services using 2-factor authentication use an unprotected login [3]. This proposal aims to solve that and instead use protected login [3] based on availability of mobile device near machine via which user is interacting with web service.

- TLS origin bound certificates used to solve MITM attacks use client signed automatically generated certificates and don’t need certificate provisioning. For every web service, browser has a unique certificate [2] and can be reused later to create a secure channel again. These certificates don’t identify user but they identify the channel via which user communicates with the web server.

There are certain assumptions in above discussed threat model. Attacker cannot simultaneously do MITM on both user browser to web server link as well as on communication between browser and mobile device. This kind of attack is rare and doesn’t come under this proposal. Proposal doesn’t use any other second factor authentication device or token system so as not to change user experience. Proposal assumes that users carry mobile device every time with them and keep it close to machines they are using to access different web services. Even though this proposal makes it possible to do protected login even first time from a new machine using mobile device as second factor authentication, it assumes there is no attack at the time of registration of this device with web service. This registration happens once and needn’t be done again unless mobile device has been lost or changed or user wants to update it for other reasons.

III. DETAILS OF PROPOSED SOLUTION

During the last decade user authentication in web services was implemented using simple username and password via TLS protocol [6]. After initial TLS handshake, server supplies a cookie to user and all subsequent requests from client use that cookie for authentication. Cookie is a form of username and password. Proposed solution categorizes this as unprotected authentication because it can be targeted using phishing which can result in loss of credentials, MITM attacks which can lead to credential or cookie loss. MITM attack can be done by attacker who has a TLS certificate obtained either via compromised CA. Czeskis et
al., have come up with a solution to MITM attack after TLS handshake has exchanged username and password. This is based on an extension to TLS protocol [2], which involves establishing a channel between client and server based on public-key cryptography. The current proposal uses this solution and involves steps to prevent MITM before username and password exchange in TLS handshake. The proposal binds cookie issued by server with the channel established during TLS extended OBC handshake. This results in protected login.

There are three main parts in user authentication according to proposed solution. Initial registration of mobile device and creation of origin bound certificates at first login from any machine is first step [1]. This initial registration of mobile device is unprotected login because proposed solution isn’t active before that and solution needs a registered mobile device. During device registration server and client mobile device agree on master key for future communication. Second part involves setting up of TLS-OBC based channel between web server and client [2]. Clients automatically generates per origin certificates first time and pass this certificate to server at the time of registration. These certificates are then used to setup channels between client and server on subsequent logins. Third part of proposal in executed every time user wants to authenticate. Based on whether client has mobile device near the machine via which user is trying to authenticate, this authentication will be protected or unprotected. That is why this authentication is called opportunistic protected login. Following is the algorithm followed by client and server to authenticate each other first time. In subsequent authentication attempts from same machine, only credentials and already existing OBC based cookie are used. Identify assertion via mobile phone isn’t required. It involves second and third part of proposal discussed above, because first part of proposal just happens once for a given mobile device.

- In TLS handshake, client uses TLS-OBC certificate to establish a TLS channel. Server already knows about TLS-OBC that client used last time. So, during TLS handshake it does verify whether client certificate is valid or invalid. If it is valid a channel is setup. Note that TLS-OBC is per origin per browser i.e if user uses different browser on same machine, then that will generate a separate TLS-OBC certificate.
- User supplies username and password and server replies with a login ticket to the browser which initiated the request only after checking whether credentials are correct. Login ticket can only be decrypted by client because master key is only shared between mobile device and server [2].
- In case browser is able to detect mobile device indicated by server in vicinity of machine, then it sends login ticket and some additional information which includes TLS channel id to mobile device. Device decrypts the information inside login ticket and performs few tests. It also verifies whether TLS channel id supplied by browser and the one supplied by server directly via login ticket are same. This prevents TLS MITM attack and assures that the channel is the expected channel.
- If everything is fine then mobile signs the identity assertion and sends it back to browser. Browser in turn forwards it to server which performs few more test cases. If everything is fine, server sends a cookie bound with TLS channel id and marks current login as protected login.
- If TLS channel id doesn’t match or device test checks fail or server checks on identity assertion returned from browser fail then login is reverted to protected login.

Cookies issued in TLS-OBC handshake has two parts [2]. First part is the hash value obtained by hashing OBC fingerprint and cookie value with a server secret key. Second part is the actual cookie value. So, the cookie is a tuple:

\[ < v, HMAC_k(v + f) > \]  

where \( f \) is fingerprint, \( v \) is value and \( k \) is server key. Every time a new origin is being authenticated a new cookie is generated during handshake and server keeps track of that cookie for future authentication attempts by user.

This information is used by server to verify in future exchanges whether request came from authentic user or not. In case an attacker tries to do a MITM attack, then attacker needs to supply a cookie which is impossible to generate because attacker doesn’t know OBC fingerprint nor does attacker has server secret key to generate hash value. If a web service considers not using protected login, then an attacker can do a MITM or phishing attack. MITM attacks are easy if user has a forged CA certificate for concerned server. An attacker would be able to create a TLS connection between itself and client. Attacker will also be able to create an account between attacker and web service provider. This means without cookie binding to TLS-OBC channel based authentication, TLS authentication is susceptible to attacks.

There are two tasks related to TLS-OBC that browser needs to do to communicate with an origin. First, generate automatic certificates and then sign them. It can use spare CPU cycles in free time to regenerate certificates, but it has to wait for origin to sign them. Experimental results have shown that generating TLS origin bound certificates increases connection latency first time but on subsequent logins which involve reusing same certificate, it doesn’t add too much latency. Generation time certificates using 2048 bit RSA is maximum were as ECDSA takes more CPU cycles to sign but the amount of latency introduced in signing is minimal.

During authentication if mobile device based identity assertion fails [1], but user authentication credentials succeed
then login can still happen. Depending upon web service provider, login can have two modes: opportunistic and strict. In former mode, identity assertion failure doesn’t mean user cannot successfully authenticate and get access to services. Depending upon usecase service provider may restrict services if user login is unprotected i.e when credentials are correct but identity assertion by mobile device failed. User can be informed via email, SMS or by other means, every time this happens. Service provider can give full access to user only if login is protected, that is if credentials as well as identity assertion via mobile device both succeed.

After initial TLS handshake and once user authentication credentials are verified, server sends login ticket and certauth id to browser. Certauth id is used by browser to recognize mobile device. Certauth id also has origin protection key, which can be used by browser to communicate with device. Login ticket has lot of information inside it but everything is encrypted via master key shared between server and mobile device. This information includes copy of origin protection key. Certauth id has device type and address. This information is used by browser to locate mobile device. Browser then creates an assertion request for mobile device. Assertion request has login ticket and some metadata encrypted using origin protection key. This metadata has origin, TLS channel id and TLS OBC support information in it. Mobile device decrypts the login ticket and runs some tests. It uses the origin protection key obtained from login ticket to decrypt metadata sent by browser and extracts TLS channel id with the one sent in ticket to prevent MITM attack.

Proposal suggests use of a single mobile device to act as a second factor authentication [1]. If user tries to register more than one mobile device, then proposal directs server to update the old one with new one, but maintain just one. This makes it simple for user, otherwise users can get confused. In case user wants to update registered mobile device and has an active user session available on browser, then user can use that session to update device. Server can send QR code or SMS to validate device. In case there is no active session, then proposal doesn’t give details about steps to recover, but expects service provider to use best recovery method depending upon services being offered.

The mode of communication between machine via which user is authenticating to web service and mobile device is bluetooth. Most of mobile device and laptops have bluetooth available in them and this mode of communication doesn’t require any other form of connectivity and internet access. Because this mode of communication is different then the one used by machine with browser and web server, it provides safety against MITM attacks. There are two challenges here [1]

- User’s browser and mobile device need to communicate without any input from user.
- User’s browser and mobile device should communicate even if machine having browser has never seen mobile device before.

Second goal is difficult to achieve. It proposes to use unauthenticated RFCOMM connection to communicate with bluetooth device. Ironically, this feature hasn’t been of much use until now. But even then user needs to enter MAC address of mobile device into machine having browser running for user authentication. Proposal claims no user interaction between mobile device and machine running browser, but if every time user needs to add MAC address of mobile device to a new machine via which it wants to authenticate with web service, then it does seem to have a negative impact on usability assumptions made in proposal.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

This section discusses the experimental work carried out by Czeskis et al., and corresponding results. Proposal also tries to evaluate solution against Bonneau et al.’s password evaluation framework [5]. Bonneau et al.’s framework has 25 authentication related factors against which a particular user authentication scheme can be evaluated.

A. Comparison with passwords and Google 2-Step verification

Czeskis et al., has used Bonneau framework to compare their proposal with Google’s 2-Step verification (2SV) and have also compared proposal with basic password based authentication [5]. Basic password based authentication is the one being used by different web services for decades. Based on previous discussion, we know that proposal does speak about two modes in which opportunistic protected login can happen. They are opportunistic and strict modes. Both modes have different usecases and properties, and both of them have been used separately in comparison. There are three aspects of user authentication benefits: Usability, Deployability and Security.

In usability benefits [5], easy-to-use means how much time does user spend on authenticating itself to web service every time authentication is required were as easy-to-learn means how easy it is for a new user to learn the scheme. Proposal claims to have easy-to-use and easy-to-learn advantage as it doesn’t aim to change usability experience so it does provide those benefits. It also claims to somewhat provide infrequent-errors benefit similar to passwords and 2SV. Mobile device bluetooth communication troubles and users forgetting their passwords degrade this benefit from full to somewhat. Opportunistic protected login in opportunistic mode does provide some level of nothing-to-carry benefit because these days user do carry mobile device with them by default. Strict mode makes it mandatory for a user to have a mobile device, so nothing-to-carry benefit isn’t provided at all. However both modes do provide quasi-nothing-to-carry benefit.

For deployability benefits [5], opportunistic mode is easy to deploy and does provide most of the benefits provided
by password based authentication. Strict mode does require some changes on server side depending upon application of usage and hence somewhat provides benefits but because solution is open and can be implemented using the documentation available, it doesn’t hinder developers. Opportunistic mode provides accessible, negligible-cost-per-user, browser-compatibility, mature and non-proprietary benefits. This mode somewhat provides server compatibility benefit as well. Strict mode provides accessible and non-proprietary benefits in full but only provides some part of rest of deployability benefits. Both of the modes are open and have already started to get integrated into browsers. TLS-OBC related changes are being evaluated by IETF as well.

The proposal outshines other method of user authentication when it comes to security benefits [5]. Strict mode provides resilient-to-physical observation and opportunistic mode somewhat provides this benefit depending upon whether assertion request is successfully met. According to this benefit attacker cannot impersonate the user even after observing users for long duration of time. Similarly it does provide resilient-to-unthrottled-guessing benefit in strict mode and somewhat in opportunistic mode depending whether protected login happens successfully or not. Unthrottled guessing involves attacker using his computational resources to its best and trying to crack passwords. In addition strict mode provides resilient-to-targeted-impersonation, resilient-to-throttled-guessing, resilient-to-leaks-from-other-verifiers, resilient-to-phishing, resilient-to-theft, no-trust-third-party and requiring-explicit-consent. Opportunistic mode somewhat provides resilient-to-targeted-impersonation, resilient-to-throttled-guessing, resilient-to-leaks-from-other-verifiers and resilient-to-phishing depending upon whether it successfully authenticates and achieves protected login. There is one benefit which passwords and 2SV provides but is only somewhat provided by protected login. It is unlinkable benefit and because user would be exposing MAC address of mobile device to multiple verifiers it cannot be fully provided even if communication between browsers and device is encrypted.

Overall based on Bonneau et al.’s framework this proposal scores better compared with traditional password and 2SV user authenticates methods.

B. Performance measurements

There are number of issues and performance measurements which will be important to access success of this proposal. Some measurements would be related to cryptographic overhead and they are creation of keys, signing or keys, encryption and decryption etc. Cryptographic measurements include both TLS origin bound certificate generation by browser as well as key generation by client. Others are related to round trip time (RTT) measurements between mobile device and browser and between browser and web server. Proposal also needs to address failures which can hinder users if failures result in prolonged wait and increased latency of fallback login (unprotected login), in case assertion request fails.

As discussed earlier the overhead related TLS-OBC key generation is low for browser [2] because of machines user use to authenticate with server are far better in CPU power then mobile device. One keys are generated subsequent contacts result in negligible latency increase. Moreover, to decrease latency in generating keys first time, browser can use spare CPU cycles to pre-generate them for expected new origins before hand. This step will slow down latency in many cases.

The proposal claims that the overhead related to ticket checks performed by mobile device [1], and signing the final identity assertion have been found to be negligible and don’t have any end user impact because typing username and password does take a substantial time and this minor increase wouldn’t hurt user experience. Even though login ticket signing and checking takes negligible time, the overhead related to additional RTTs sent from browser to server varies from milliseconds to few seconds depending upon network availability. This addition to overall login timings is low if compared with different web services provided by email server providers e.g. gmail by Google or social networking sites e.g. Facebook.

The problematic overhead is when browser tries to locate bluetooth device but fails [1]. Different measurements carried out by Czeskis et al. has shown that it can take maximum of 7 seconds to establish a connection, so maximum login time could go up to 17 seconds. But this was one of the worst scenario encountered by experiments. Proposal suggests to use 7 seconds as upper time till browser can keep looking for mobile device and give up if it cannot establish connection with mobile device. Certain techniques e.g. lazy verification or laptop looking for bluetooth device around it before user opens up a particular origin can be used to speed up this processes, but proposal doesn’t discuss or take it further. The average login time using 2-factor authentication for gmail is 25 seconds which is far greater then worst time of 17 seconds for proposal in discussion.

V. OPEN PROBLEMS

This section discusses the open problems and different claims made in proposal by Czeskis et al. Proposed solution threat model claims to solve every form of man-in-the-middle attack by using TLS origin bound certificates. The assumption also claims that attack has no access to user’s machine via which user is trying to authenticate with web service.

Proposal depends upon TLS-OBC implementation and current implementation in Chromium browser is in flux and the corresponding draft to IETF has expired and taken a
reshape. It is now called TLS-ChannelID proposal. The TLS-OBC proposal had a flaw that client certificates are too big in size and they make the session state too big. This means lot of communication and processing overhead [9]. Moreover, if attacker gets access to one of these client certificates, using malware, then attacker can act as if it is the owner of private keys for that user and setup new connection to different web services with user getting to know about it. New proposal doesn’t involve certificates but rather has channel IDs and these are are not bound to single origin. Same top level domain can be used to provide many services including a web service. Different layers use different client keys for different origins inside same top level domain, hence binding a domain cookie to TLS channel isn’t possible.

In Bonneau et al.’s framework the proposal claims to achieve quasi-nothing-to-carry benefit. But this claim doesn’t stand in current situation when tablets are very common among users. Tablets are higher version of smartphones and user authenticating with tablets may not have a second factor mobile device around if their primary mobile device is tablet itself. In this case if user needs to carry a smartphone with a tablet then it may not make sense. In summary the assumption that smartphones are always with user, may not be true. The proposal doesn’t discuss user who login to a web service from smartphones. Proposal also claims to have easy-to-learn and easy-to-use benefits, even though one still manually needs to enter MAC address of bluetooth device into machine via which user is trying to authenticate with web service. Find MAC address and adding it wouldn’t be a user friendly and easy step. Czeskis et al.’s proposal also claims to have an easy-recovery-from-loss benefits, and it claims that 2SV somewhat provides it but not fully. This claim cannot be true unless proposal explicitly mentions details of how it can be done easily if there is no active session between browser and server. If there is an active session between browser and server the recovery process is similar for 2SV and current proposal. If there is no-active proposal an alternate email address, or alternate phone number, or back up codes are used by 2SV but current proposal doesn’t discuss recovery process for no active session. but still claims it to provide easy recovery benefit.

Proposal claims to work with bluetooth mobile device flawlessly without device introducing much delay, but no comparison between distance between device and machine via which user tries to authenticate has been done. Connection oriented communication between two bluetooth enabled devices varies inversely with distance and in a home or office space users don’t always carry mobile device around all the time. This is more true with new affordable tablets around them. The latency introduced with different distances and mobile devices in different parts of same house will make it possible to analyse the solution better. RFCOMM feature of mobile device [1] only works if machine via which user is authenticating already knows the address of mobile device. Locating MAC address of different mobile device and manually adding that to machine may be a difficult for normal user. The claim that user does no interaction for login doesn’t seem to be true at least first time.

VI. Conclusion

The current proposal to enhance age old password based authentication to prevent common phishing and MITM attacks seems promising, but there are lots of issues discussed in open problems section, which need more work. Claims that there is no user interaction required for providing an opportunistic second factor authentication at the backend doesn’t seem to work with present proposal everytime a mobile is moved to a new machine. Proposal doesn’t talk about tablets replacing smartphones or smartphones being used as browsing devices to access web services. Mobile devices and tablets don’t go hand in hand and taking them together every places doesn’t make sense either. The measurements related to bluetooth don’t take distance between machine and mobile distance into account.

Proving support for TLS-OBC extension to browsers and different security libraries have resulted in practical problems and changed the draft and implementation over the period of time. So, the base over which this proposal depends is still under flux. Still this proposal seems promising and future work will hopefully solve all open issues and bring TLS extension into current TLS standard.

REFERENCES


