ISSN (e): 2250 – 3005 || Volume, 08 || Issue, 03|| March – 2018 ||
International Journal of Computational Engineering Research (IJCER)

Cloud-Based Password Manager for Web Browser

Dr. Sasmita Pani[1], Jayarajan[2], Dr.Aurobindo Kar[3],
[1,3]*Associate Professor, Department of Computer Science&Engineering, Gandhi Institute For Technology (GIFT), Bhubaneswar*
[2] *Assistant Professor, Department of Computer Science&Engineering, Gandhi Engineering College, Bhubaneswar*

Abstract: *Web users are confronted with the daunting challenges of creating, remembering, andusing more and more strong passwords than ever before in order to protect their valuableassets on different websites. Password manager, particularly Browser-based PasswordManager (BPM), is one of the most popular approaches designed to address these challenges by saving users' passwords and later automatically filling the login forms on behalfof users. Fortunately, all the five most popular Web browsers have provided passwordmanagers as a useful built-in feature. In this paper, we uncover the vulnerabilities ofexisting BPMs and analyze how they can be exploited by attackers to crack users' savedpasswords. Moreover, we propose a Cloud-based Storage-Free BPM (CSF-BPM) design to achieve a high level of security with the desired confidentiality, integrity, and availabilityproperties. In this we stored the passwords in different location in the same server.so it's not attacked by the hacker's.We have implemented a CSF-BPM system into google chrome and evaluated its correctness, performance, and usability. Our evaluation results and analysis demonstrate thatCSF-BPM can be efficiently and conveniently used. We believe CSF-BPM is a rational designthat can also be integrated into other popular browsers to make the online experience ofWeb users more secure, convenient, andenjoyable.*

I. INTRODUCTION

Text-based passwords still occupy the dominant position in online user authentication.Password security heavily depends on creating strong passwords and protecting them from being stolen. However, researchers have demonstrated that strong passwords that are sufficiently long, random, and hard to crack by attackers are often difficult to remember by users.Therefore,users do not need to remember a large number of strong passwords; meanwhile, BPMs will only fill the passwords on the login forms of the corresponding websites and thus can potentially protect against phishing attacks. Fortunately, mainly to support the password auto fill and management capability, all the five most popular browsers Internet Explorer, Firefox, Google Chrome, Safari, and Opera have provided password managers as a useful built-in feature .In this paper, we uncover the vulnerabilities of existing BPMs and analyze how they can be exploited by attackers to crack users' saved passwords. Moreover, we propose a novel Cloud-based Storage-Free BPM (CSF-BPM) design to achieve a high level of security with the desired confidentiality, integrity, and availability properties.We have implemented a CSF-BPMsystem and seamlessly integrated it into the Firefox Web browser. We have evaluated the correctness, performance, and usability of this system.

We believe CSF-BPM is a rational design that can also be integrated into other popular browsers to make the online experience of Web users more secure, convenient, and enjoyable. We have followed standard responsible disclosure practices and reported those vulnerabilities to the respective browser vendors.Our vulnerability verification tools and the CSF-BPM system can be demonstrated and be shared with responsible researchers.

II. RELATED WORK ANDBACKGROUND

In this section, we briefly review the related password and password manager research, and provide the background information on the BPMs of the five most popular browsers.We do not advocate against any of these other approaches. We simply focus on the BPM security in this paper.

2.1. Related work

Morris and Thompson pointed out long ago in 1979 that weak passwords suffer from brute-force and dictionary attacks(Morris and Thompson, 1979). Later, Feldmeier and Karn further emphasized that increasing password entropy is critical to improving password security (Feldmeier and Karn,1989).Yan et al. (Yan et al., 2004) analyzed that strong password requirements often run contrary to the properties of human memory, and highlighted the challenges in choosing passwords that are both strong and mnemonic. Recently,Flor^ encio and

Herley performed a large-scale study of Web password habits and demonstrated the severity of the security problems such as sharing passwords across websites and using weak passwords (Florencio and Herley, 2007). A largescale user study recently performed by Komanduri et al.demonstrated that many Web users write down or other wise store their passwords, and especially those higher- entropy passwords (Komanduri et al., 2011).To help Web users better manage their online accounts and enhance their password security, researchers and vendors have provided a number of solutions such as password managers (Wu etal.,

2006; 1Password; RoboForm Password Manager), Web Single Sign-On (SSO) systems (Kormann and Rubin, 2000; Sun et al., 2010; OpenAuthentication 2.0; The OAuth 2.0 Authorization Framework), graphicalpasswords

(Davis et al., 2004; Thorpe and van Oorschot, 2007; Thorpe and van Oorschot, 2004), and password hashing systems(Halderman et al., 2005; Ross et al., 2005; Yee and Sitaker,2006). As analyzed in Section 1, password managers especially BPMs have the great potential to well address the challenges of using many strong passwords and protecting a gainst phishing attacks. The insecurity of third-party commercial password managers such as Last Pass (LastPassPasswordManager) and RoboForm (RoboForm Password Manager) areanalyzed by Zhao et al. in (Zhao et al., 2013). Web Wallet (Wuet al., 2006) is an anti-phishing solution and is also a password manager that can help users fill login forms using stored information; however, as pointed out by the authors, users have a strong tendency to use traditional Web forms for typing sensitive information instead of using a special browser sidebar user interface. In addition, Web Wallet is not cloud based. In terms of Web SSO systems, their security vulnerabilities such as insecure HTTP referrals and implementationsare analyzed in (Kormann and Rubin, 2000; Sun and Beznosov,2012; Wang et al., 2012b), their business model limitations such as insufficient adoption incentives are analyzed by Sunet al. in (Sun et al., 2010), and their vulnerabilities to phishing attacks against the identity provider (such as Google and Facebook) accounts are highlighted by Yue, 2013. Security limitations of graphical passwords are analyzed in Davis et al.,2004, Thorpe and van Oorschot, 2007, and Thorpe and van Oorschot,2004.

2.2 Password manager feature of browsers

Table 1 – Basic information of BPMs in five most popular browsers.

Browser	Configuration location	Enabled by default	Master password	Password Sync.
Internet Explorer (11.0)	Internet options → Content → AutoComplete Settings → Usernames and passwords on forms	Yes	No	No
Firefox (27.0)	Options → Security → Passwords	Yes	Yes	Yes
Google Chrome (33.0)	Settings → Show advanced settings ... → Passwords and forms	Yes	No	Yes
Safari (5.1.7)	Preferences → AutoFill → Usernames and passwords	No	No	No
Opera (20.0)	Settings → Privacy & security → Passwords	Yes	No	Yes

Table 1lists the basic information on the BPM feature of the recent versions of the five most popular Web browsers. The second column of the table provides the sequence of menu items that a user must click in order to finally access the BPM feature configuration interface. We can see that the BPM feature configuration locations are very different among browsers. Indeed, the feature configuration interfaces shown on those locations are also very different among browsers in terms of the configuration options and functions. The third column shows that the BPM feature is enabled by default in four browsers but not in Safari. The fourth column shows that only Firefox employs a master password mechanism, which is, however, not enabled by default and users may not be aware of its importance. Note that Opera employed a weak master password mechanism in its early versions such as version 12.02 (Zhao and Yue, 2013). The fifth column shows that Firefox, Google Chrome, and Opera provide a password synchronization mechanism that can allow users to access the saved passwords across different computers.

In terms of the dynamic behavior, the interfaces for trig-gering the remembering and autofill of passwords are incon-sistent among browsers. For one example, all the browsers display a dialog box to ask a user whether the entered pass- word for the current website should be remembered. The dialog boxes displayed by Firefox, Google Chrome, and Opera are associated with the address bar, thus technically hard to be spoofed. For another example, Internet Explorer, Firefox, and Opera require a user action before auto-filling the pass-word value on a website; however, Google Chrome and Safari autofill the username and password values once a user visits a login webpage, providing more opportunities for malicious JavaScript to manipulate the login form and information.

III. VULNERABILITYANALYSIS

In this section, we first define the threat model and assump- tions that we consider throughout this paper. We then use an analogy to justify the essential problem of existing BPMs.
Finally, we provide a detailed vulnerability analysis regarding without and with a master password mechanism.

3.1. Threat model andassumptions

"Where a threat intersects with a vulnerability, risk is present (Bowen et al., 2007)."For Browser-based Password Managers (BPMs), the threat sources are attackers who want to steal the sensitive login information stored by BPMs.

3.1.1. Threatmodel

Our basic threat model is that attackers can temporarily install malware such as Trojan horses and bots on a user's computer using popular attacks such as drive-by downloads (Cova et al., 2010; Lu et al., 2010; Moshchuk et al., 2006; Provoset al., 2008; Wang et al., 2006). The installed malware can then steal the login information stored by BPMs. For example,Stone-Gross et al. inferred that 38% of the credentials stolen by the Torpig bot were obtained from the password managers of browsers, rather than by intercepting an actual login session (Stone-Gross et al., 2009). Note that the malware can run at the system-level or at the application-level, and can evenbemalicious browser extensions (Louw et al., 2008). Indeed, if the occurrences of such threats are rare or do not have high im-pacts, BPMs would not bother to encrypt their stored pass-words in the first place. Therefore, our focus will be on investigating the vulnerabilities of BPMs that could be exploited by potential threat sources to easily decrypt the passwords stored by BPMs.

3.1.2. Assumptions

We assume that it is very difficult for the installed malware to further compromise the operating system to directly identify cryptographic keys from a computer's memory (Halderman et al., 2008) because this identification often requires elevated privilege and is prone to false positives. We assume that the installedmalwarecanbe removed fromthesystemby security-conscious users in a timely manner, so that even though sensitive login information stored by BPMs can be stolen within a short period of time, it is very difficult for at- tackers to use tools such as keyloggers to further intercept users' passwords for a long period of time.

3.3. Without a master password mechanism

Through source code analysis, binary file analysis, and ex- periments, we found that chrome uses the three-key Triple- DES algorithm to encrypt a user's passwords for different websites.chrome saves each encrypted username, encrypted password, and plaintext login webpage URL address into the logintable of a SQLite (SQLite Home Page) database file name signons.sqlite. The Triple-DES keys are generated once by chrome and then saved into a binary file named key3.db starting from the byte offset location 02F90. Although the keys generated on different computers are different, they are not bound to a particular computer or protected by other mechanisms.

IV. CSF-BPMDESIGN

We now present the design of the Cloud-based Storage-Free BPM (CSF-BPM). It is cloud-based storage-free in the sense that the protected data will be completely stored in the cloud e nothing needs to be stored on a user's computer. We want to move the storage into the cloud for two key reasons.

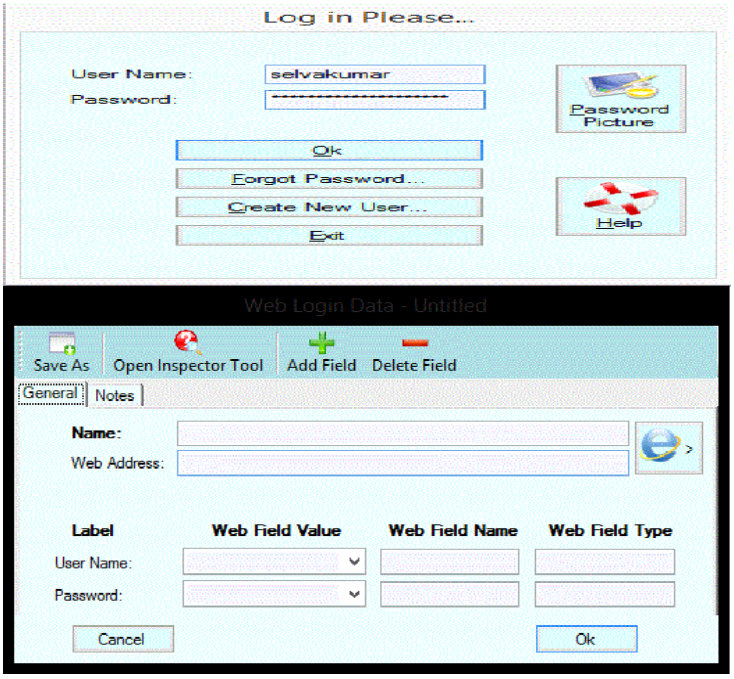

Figure1: CSF-BPM DESIGN

4.1. High-levelarchitecture

BPM itself does not include any persistent storage component such as afile or database; instead, it will generate Encrypted Login In-formation Records (ELIRs), save protected ELIRs to a Secure and Reliable Storage (SRS) service in the cloud, and retrieve pro-tectedELIRs in real-time whenever needed. Such a generic BPM design can be seamlessly integrated into different browsers,An SRS service simply needs to support user authentication (e.g., over HTTPS) and per-user storage so that its deployment in the cloud can be easily achieved. For example,thesynchronization service associated with Firefox or GoogleChrome.

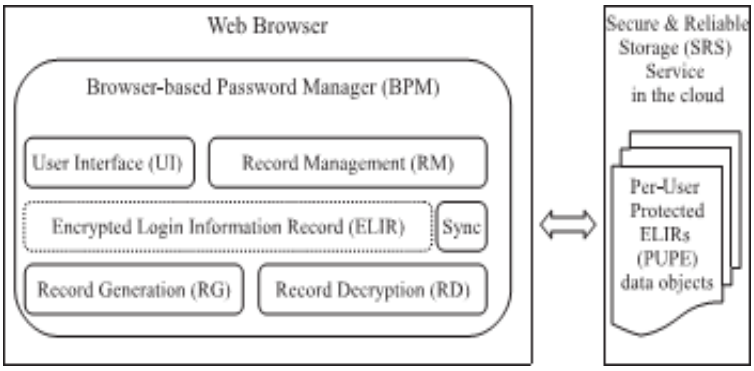

Figure2- High-level architecture of the Cloud-based Storage-Free BPM (CSF- BPM).

4.2. Designdetails

In the following subsections, we detail the basic usage of CSF- BPM, the ELIR record, the key derivation and password encryption process, the PUPE data object, and the password decryption process.

4.2.1. Basicusage

To use CSF-BPM, a user needs to remember a Single Strong Master Password (SSMP) with the strength (Burr et al., 2011; Clair et al., 2006) assured by the traditional proactive pass- word checking Using a master password is also advocated in other proposed systems such as Nigori (Laurie). The user also needs to set up an account (srsUsername, srsPass- word) on an SRS service and configure this service once through the UI component of BPM.At the beginning of each browsing session, the user needs

to authenticate to the SRS service and provide the SSMP to BPM. After that, BPM will take care of everything else such as triggering the remembering of website passwords, encrypting and decrypting ELIRs, and triggering the autofillof passwords.Both the srsUsername and srsPassword pair and SSMP need be provided only once in a session through the specialUI.

4.2.2. ELIR record

The basic format of an ELIR record is shown inFig. 3. Here, RecordSalt is a large and sufficiently random per-record salt generatedby BPM. It is used to calculate the symmetric record key (denoted record Key) for encrypting a user's plaintext password (denotedsitePassword) for an account (denotedsiteUsername) on a website (with siteURL as the login web-page URL address). The recordKey can be deterministically generated by using a password-based key derivation function such as PBKDF2 specified in the PKCS5 specification version
2.0 (Kaliski, 1999). The basic format of an ELIR record can also include the IDs (or names) of the username and password fields in the login webpage, and it can be further extended if necessary.

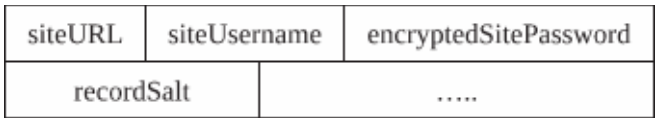

Figure3- The basic format of an Encrypted Login Information Record (ELIR).

4.2.3. Key derivation and password encryption

Using PBKDF2 (Kaliski, 1999), our SSMP-based key derivation and password encryption process consists of five steps illustrated in Formulas 1, 2, 3, 4 and 5 inFig. 4. The input parameters mainSalt and aeSalt in Formulas 1 and 2 are large and sufficiently random per-user salts generated by BPM at the first time when a user authenticates to the SRS service through the UI component of BPM. In Formulas 1, 2, and 3, the input parameters c1, c2, and c3 represent iteration counts for key stretching; the input parameters dkLen1, dkLen2, and dkLen3 represent lengths of the derived keys, and they are related to the underlying pseudorandom function used in the PBKDF2 implementation.The salts and iteration counts in PBKDF2 are used to secure against dictionary and brute-force attacks, and they need not be kept secret (Kaliski,1999).

The strength of SSMP also helps secure against these two types of attacks. In Formula 1, a mainKey is calculated and will be used in each browsing session.

SSMP is typed only once and will be erased from Memory after mainKey is calculated. In Formula 3, a unique RecordKey is generated (using the per-record recordSalt) for each website account of the user. In Formula 4, a NIST- approved symmetric encryption algorithm E such as AES (Advanced encryption standard (AES), 2001) (Together with a block cipher mode of operation if the site Password is long) can be used to encrypt the sitePassword. In Formula 5, a NIST-approved Authenticated Encryption block cipher mode AE such as CCM (Counter with CBC-MAC) (Dworkin, 2004) can be used to simultaneously protect confidentiality and authenticity (integrity) of theconcatenatedELIRs(i.e., the concatenated string of all the ELIR records) of an SRS user. The aeKey used here is generated by Formula 2.

The iteration count c1 used in Formula 1 should be large so that the mainKey calculation will take a few seconds; there- fore, brute force attacks against SSMP become computation- ally infeasible. But c1 should not be too large to make a user wait for a long period of time at the beginning of a session.

$$mainKey = PBKDF2(SSMP, mainSalt, c1, dkLen1) \quad (1)$$

$$aeKey = PBKDF2(mainKey, aeSalt, c2, dkLen2) \quad (2)$$

$$recordKey = PBKDF2(mainKey, recordSalt, c3, dkLen3) \quad (3)$$

$$encryptedSitePassword = E(recordKey, sitePassword) \quad (4)$$

$$protectedELIRs = AE(aeKey, concatenatedELIRs) \quad (5)$$

ig. 4 – Formulas in the SSMP-based key derivation and assword encryption process.

4.2.5. Password decryption

To decrypt the saved website passwords for autofill, BPM will perform five steps: (1) retrieve the PUPE data object saved for the SRS user; (2) generate the mainKey and aeKey using For- mulas 1 and 2; (3) decrypt and verify the protectedELIRs using the reverse process of Formula 5 such as the CCM Decryption-Verification process (Dworkin, 2004); (4) obtain the recordSalt of each ELIR and generate the recordKey using Formula 3; (5)finally, decrypt the encrypted SitePassword using the reverse process of Formula 4. Note that at step (3), both the integrity of the protectedELIRs and the authenticity of the BPM user are verified because the success of this step relies on using the correct SSMP. Also at this step, siteURL and siteUsername of all the ELIRs can be obtained by BPM to determine whether this user has previously saved login information for the currently visited website. Normally, the first three steps will be performed once for the entire browsing session,

and the last two steps will be performed once for each website that is either currently visited by the user, or its domain name is queried by the user to simply look up the corresponding username andpassword.

4.3. Design rationales and securityanalysis

We nowfurther justify the important design rationales of CSF- BPM by focusing onanalyzing its confidentiality, integrity, and availability security properties, and by comparing its design with other design alternatives especially the BPM design of Firefox that also provides a master password mechanism.In terms of the confidentiality, first, by having a unique cloud-based storage-free architecture, CSF-BPM can in the long run effectively reduce the opportunities for attackers tosteal and further crack regular users' saved website pass- words. Second, even if attackers (including insiders of an SRS service) can steal the saved data, it is computationally infea- sible for attackers to decrypt the stolen data to obtain users' login information for different websites. CSF-BPM provides this security guarantee by mandating a strong SSMP that satisfies certain strength requirements (Bishop and Klein, 1995; Kelley et al., 2012; Yan, 2001), by using the PBKDF2 keyderivation function (Kaliski, 1999) with randomly generated salts and adaptively computed large iteration counts, and by following NIST-approved symmetric encryption (Advanced encryption standard (AES), 2001) and authenticated encryp- tion (Dworkin, 2004) algorithms. Basically, even if attackers can steal the saved data, they have to guess (albeit stealing attacks are still possible as discussed in Section7) a user's strong SSMP in a very large space determined mainly by the length and character set requirements of SSMP with each try taking seconds of computation time.

We can estimate the effort of brute force attacks based on the computational power exemplified in a very popular cryptography textbook (Stallings, 2013) authored by William Stal-lings. The Table 3.5 (chapter 3, page 78, and 6th edition) of this textbook shows that a rate of 1 billion (109) decryptions per second can be achieved by today's multicore computers. For simplicity but without loss of generality, we consider that

SHA-1/SHA-2 (NIST) hash operations can be performed at the similar rate by multicore computers, although this is a con- servative consideration because a hash operation is normally more efficient than a decryption operation. If each master password character can be an upper case letter, a lower case letter, or a decimal digit, then it could be one of the 62 (26þ26þ10) possibilities. The search space for an 8-character master password will be 628. Moreover,this detection is securely performed in the sense that attackers cannot take advantage of it to effectively conduct brute force attacks against the SSMP.

V. EVALUATION

We built the Firefox version CSF-BPM on a Ubuntu Linux system. We tested the correctness of our implementation and Fig. 4eDetailed implementation of CSF-BPM in Firefox. computrs& security 46 (2014) 32e47 40 its integration with the Firefox Web browser, we intensively evaluated its performance, and we also evaluated its usability through a user study.

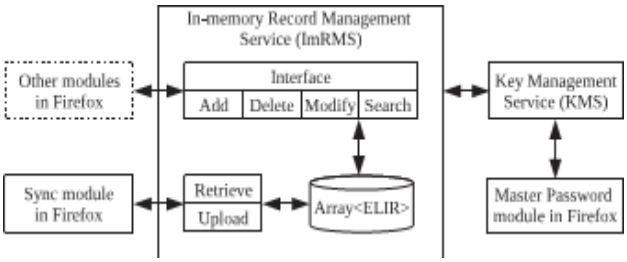

Figure4: DETAILED IMPLEMENTION OF CSF-BPM IN CHROME

5.1. Correctness

We selected 10websites perform the correctness verification. Most of the websites were selected from the top 25 websites listed by Alexa.com; however, weremoved non-English websites, gray content websites, and the websites that did not allow us to create an account.

We also selected some of our frequently used websites. On each website, we went through four main steps. First, we opened Firefox and typed an SRS account (i.e., a chrome Sync account) and SSMP. Second, we logged into the website and confirmed to save the website password. Third, we logged out the website and logged into it again with the auto-filled password. Finally, we closed Firefox, re-opened Firefox, typed the SRS account and SSMP, and logged into the website again with the auto-filled password.

Through the execution of those steps, we verified that our implementation works precisely as designed; meanwhile, it in- targets smoothly with Firefox and does not cause any logic or runtime error. In more details, we observed that CSF-BPM can correctly save and auto-fill passwords on all those websites. It alsoworks correctly in the situationwhenwoor more accounts on a website are used. In addition, it does not affect the func-

tionality of other features in Firefox such as the form autocom-plete feature and the Sync feature.We also verified that nothing is saved to the original persistent password storage of chrome.We have two other observations in our experiments. One is that some other websites share the same siteURL (i.e., the login webpage URL) values with the websiteslisted. For exampleyoutube,gmail,Hotmail,ymail.

REFERENCES

[1]. Advanced encryption standard (AES). NIST FIPS 197;2001.http://csrc.nist.gov/publications/fips/fips197/ fips-97.pdf.
[2]. Anti-Phishing Working Group,http:// www.antiphishing.org.
[3]. Bessani A, Correia M, Quaresma B, Andr e F, Sousa P. Depsky:dependable and secure storage in a cloud-of-clouds. In: Proceedings of the European Conference on Computer Systems (EuroSys); 2011.
[4]. Bowen P, Hash J, Wilson M. Information security handbook: a guide for managers.Special Publication 800e100. NIST; 2007., http://csrc.nist.gov/publications/nistpubs/800-100/SP800-100-Mar07- 2007.pdf.
[5]. Burr WE, Dodson DF, Newton EM, Perlner RA, Polk WT, Gupta S, et al. Electronic authentication guideline. Special Publication 800-63-1. NIST; 2011.,http://csrc.nist.gov/publications/ nistpubs/800-63-1/SP-800- 63-1.pdf.
[6]. RoboForm Password Manager,http:// www.roboform.com/.
[7]. Ross B,JacksonC,MiyakeN,BonehD,Mitchell JC. Strongerpasswordauthentication using browser extensions. In: Proceedings of theUSENIX Security Symposium; 2005. pp. 17e32.Shamir A. How to share a secret. Commun ACM 1979;22(11):612e3.
[8]. Kaliski B.RFC 2898, PKCS5: password-based cryptography specificationversion 2.0;1999.http:/www.ietf.org/rfc/rfc2898
[9]. RoboForm Password Manager, http:// www.roboform.com/.RossB, JacksonC, Miyake N,BonehD,MitchellJC.Strongerpassword authentication using browser extensions. In: Proceedings of theUSENIX Security Symposium; 2005. pp. 17e32.Shamir A. How to share a secret. Commun ACM1979;22(11):612e3.
[10]. SQLite HomePage,http://www.sqlite.org.
[11]. Stallings W. Cryptography and network security: principles andpractice. 6th ed. Prentice Hall Press;2013.
[12]. Stark E, Hamburg M, Boneh D. Symmetric cryptography in javascript. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC); 2009. pp.373e81.
[13]. Stone-Gross B, Cova M, CavallaroL, Gilbert B, SzydlowskiM,Kemmerer RA, et al. Your botnet is my botnet: analysis of abotnet takeover. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS); 2009.pp.635e47.
[14]. Sun S-T, Beznosov K. The devil is in the (implementation) details:an empirical analysis of oauthsso systems. In: Proceedings of the ACM Conference on Computer andCommunications
[15]. Security (CCS); 2012.Sun S-T, Boshmaf Y, Hawkey K, Beznosov K. A billion keys, but few locks: thecrisisofwebsingle sign-on. In: Proceedings of the Newsecurity Paradigms Workshop (NSPW); 2010. pp.61e72.