パスワードリスト攻撃への耐性を持つ公開鍵暗号 Web アプリケーション認証方式の検討

満永 拓邦　　小林　裕士　　松本　悦宜　　重森 友行

JPCERT コーディネーションセンター
101-0054　東京都千代田区神田錦町 3-17 廣瀬ビル 11F
ww-info@jpcert.or.jp

あらまし　現在，多くの Web アプリケーションではパスワード認証が利用されている．ユーザが複数のサービスに対し同一の ID とパスワードを設定している場合，一つのサービスから ID とパスワードが漏えいすると，攻撃者はそれらを利用する事で，ユーザになりすまして他のサービスにログインする事ができる．こうした攻撃はパスワードリスト攻撃と呼ばれており，多数の不正ログイン被害が確認されている．本稿では，HTML5 で導入された Web Storage という機能を用いて，パスワードリスト攻撃への耐性を持つ公開鍵認証方式に基づく Web アプリケーション認証方式を提案を行う．

Secure Authentication System for Web Application based on a Public Key Cryptosystem

Takuho Mitsunaga　　Hiroshi Kobayashi　　Yoshinori Matsumoto
Tomoyuki Shigemori

JPCERT Coordination Center
Hirose Building Floor 11, 3-17 Kandanishiki-cho, Chiyoda-ku, Tokyo, 101-0054, JAPAN
ww-info@jpcert.or.jp

Abstract　Password authentication is widely used for a login system of web applications. A user of web applications tends to set the same password for several web applications. Under the circumstances, if IDs and passwords are leaked from one web application, an attacker could use these IDs and passwords for incorrect login attempt against different web applications. This type of attack is called a "list-based" attack, and the damage caused by this is rapidly increasing. Therefore secure alternative authentication method replaced by password is required. In this paper, we introduce a new authentication system for web application based on a public key cryptosystem by a new HTML5 API, Web Storage.

1　Introduction

従来，Web サービスのアカウントに対する攻撃の代表例として，ブルートフォース攻撃や辞書型攻撃が挙げられることが多かった．これらの攻撃は，同一のアカウントに対して繰り返し試行が行われるため，ログインエラーが多発し，攻撃の発生や傾向を把握することが容易であった．また同一のアカウントに対し複数回のログイン試行の失敗すると，キャプチャーを表示し，機械的なログイン試行を回避したり，一

時的にそのアカウントを無効にしたりする対策が有効であったため，それらの対策に併せて十分な強度のパスワードを設定することで被害の発生を抑制が可能であった．2013 年から，どこからかのサービスから漏えいしたと考えられる ID とパスワードの組み合わせを用いて，他のサービスに対して不正にログインを行う攻撃，いわゆるパスワードリスト型攻撃が広く用いられ，多数のサービスにおいて被害が発生した．Web アプリケーション認証に用いる ID はメールアドレスが利用されることが多く，また利用者が複数のサービスに対して同一のパスワードを設定していることが多いため，パスワードリスト型攻撃はブルートフォース攻撃や辞書型攻撃と比較して，ログイン成功確率が高く，また同一のアカウントへのログイン試行が少ないため，攻撃の検知が困難である．調査によると，2013 年から 20 以上のサービスに対して攻撃が発生し，600,000 件以上のアカウントに対して不正なログインが行われたとのことである [5]．ebook Japan 社が公開しているレポートによると一回のログイン試行で成功した割合は半数以上である．今もなお，同様の攻撃の発生が確認されているため，対策は急務であると考えられる．本稿では，それらの攻撃に対して耐性を持つ公開鍵方式に基づく Web アプリケーションのユーザ認証を行う方式を紹介する．

2 関連研究

パスワード以外の Web アプリケーション認証方式としてクライアント証明書がある．これは公開鍵方式に基づいた認証方式であり，セキュリティベンダーによって有料でサービスが提供されている．クライアント証明書を利用するためには，証明書の発行と利用者端末への導入が必要であるため，利用者が多い Web サービスにおいては，サービス提供企業の配布コスト，利用者の導入コストは非常に高くなっている．そのためにクライアント証明書の利用が一般的に普及したとはいいがたい状況である．本稿では，それらの配布，導入コストを軽減した効率的な認証方式を提案する．

3 提案方式

登録および認証に関する基本的なプロトコルは以下の通りであり，既に提案されていたものである．本稿は，これらのプロトコルを HTML5 の機能を用いて実装し，安全性の比較を行う．

3.1 基本的なプロトコル

クライアント・サーバ間で行われる登録および認証に関わるプロトコルは以下のとおりである．$ENC_{PK}()$ および $DEC_{PK}()$ は，それぞれ公開鍵暗号の暗号化および復号を意味する．

登録

1. クライアントは秘密鍵・公開鍵ペア PK,SK を生成する．
2. クライアントは ID と PK を認証サーバに送信する．
3. 認証サーバは ID と PK を保存する．

認証

1. クライアントから認証サーバに ID を送る．
2. 認証サーバはセッション鍵 K を生成し，$C = Enc_{PK}(K, PK)$ をクライアントに送信する．
3. クライアントは C を復号して セッション鍵 K' を得る $(K' = DEC_{PK}(C, SK))$．
4. $K = K'$ が成立すれば認証成功とする．

3.2 Techniques

本章では，実装に用いる Web Storage および暗号ライブラリについて述べる．

3.2.1 Web Storage

Web Storage は，HTML 5 で導入された，JavaScript からアクセス可能なデータをクライアント側に

保存するための機能である．データ保存量の上限はブラウザへの推奨値が 1 オリジン当たり 5MB であり，データがセッション終了時に破棄される sessionStorage と，ブラウザ終了後も永続的にデータを保持し続ける localStorage ある．いずれの場合も，読み込みおよび書き込みは同一のオリジンに制限されているため，他のオリジンの JavaScript からアクセスされることはない．本稿では，Web Storage に格納するデータはブラウザ終了後も保持する必要があるため，localStorage を利用する．なお，ブラウザ内に保管したデータは，複数の利用者が一台の端末を操作する環境にあっては，他のユーザからアクセスできる可能性があるため注意が必要である．

3.2.2 Javascript Libraries

ユーザのブラウザで暗号処理を行うために，JavaScript ライブラリを利用する．jsbn.js[4] をベースとした JSEncrypt[2] は RSA 暗号に関わる処理に，CryptoJS[1] は AES 暗号に関わる処理に利用する．

3.3 実装方式

次の二種類の方式の実装を行った．提案方式 1 では生成した秘密鍵をそのままブラウザ内の Web Storage に保管する．提案方式 2 においては，秘密鍵を AES を用いて暗号化したうえで Web Storage に保管する．Web Storage 内のデータは同一の端末であればアクセスが可能であるため，安全性が高い提案方式 2 は図書館などの共有端末に適している．実装環境はブラウザを利用できるクライアント端末と認証方式を持つ Web サーバを構築した．$ENC_{RSA}()$ と $DEC_{RSA}()$ は，それぞれ RSA 暗号方式の暗号化と復号を意味し，$ENC_{AES}()$ と $DEC_{AES}()$ は，それぞれ AES 暗号方式の暗号化と復号を意味する．提案方式のフローは以下の通りである．

3.3.1 提案方式 1

登録

1. 利用者 U_i は ID_i を入力し，秘密鍵・公開鍵ペア PK_i,SK_i を生成する．SK_i をブラウザの Web Storage に保管する．
2. U_i は ID_i と PK_i をサーバに送信する．
3. サーバはデータベースに ID_i と PK_i を保管する．

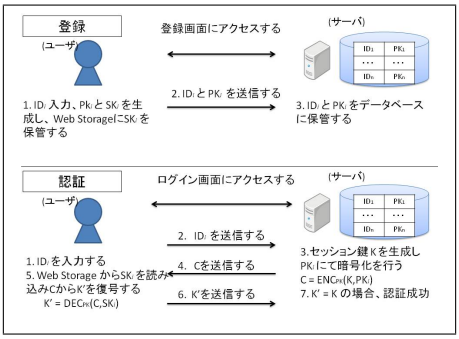
認証

1. U_i は ID_i を入力する．
2. U_i はサーバに ID_i を送信する．
3. サーバは ID_i に対応する PK_i をデータベースから得る．セッション鍵 K を生成し，PK_i を用いて暗号化を行う．$(C = ENC_{RSA}(K, PK_i))$.
4. サーバは C を U_i へ送信する．
5. U_i は SK_i をブラウザの Web Storage から得て, C を復号し，K' を得る $(K' = DEC_{RSA}(C, SK_i))$.
6. U_i は K' をサーバに送信する．
7. サーバにおいて，$K' = K$ が成立すれば認証成功とする．

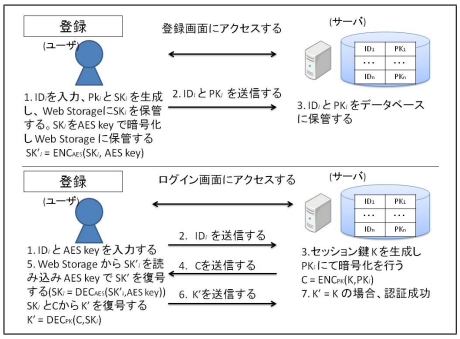
3.3.2 提案方式 2

登録

1. 利用者 U_i は ID_i と AES 用の鍵 $(AES\ key)$ を入力し，秘密鍵・公開鍵ペア PK_i,SK_i を生成する．$AES\ key$ を用いて SK_i を暗号化 $(SK_i' = ENC_{AES}(SK_i, AES\ key))$ し，SK_i をブラウザの Web Storage に保管する．
2. U_i は ID_i と PK_i をサーバに送信する．
3. サーバはデータベースに ID_i と PK_i を保管する．



提案方式 1



提案方式 2

図 1: 提案方式のフロー

認証

1. U_i は ID_i と $AES\ key$ を入力する．
2. U_i はサーバに ID_i を送信する．
3. サーバは ID_i に対応する PK_i をデータベースから得る．セッション鍵 K を生成し，PK_i を用いて暗号化を行う．$(C = ENC_{RSA}(K, PK_i))$.
4. サーバは C を U_i へ送信する．
5. U_i はブラウザの Web Storage 内の SK'_i と $AES\ key$ から SK_i を得て $(SK_i = DEC_{AES}(SK'_i, AES\ key))$，$C$ を復号し，K' を得る $(K' = DEC_{RSA}(C, SK_i))$.
6. U_i は K' をサーバに送信する．
7. サーバにおいて，$K' = K$ が成立すれば認証成功とする．

4 評価

両提案方式およびパスワード認証について，攻撃手法への耐性を表 1 に示す．提案方式は各サービスごとに認証情報である鍵のペアが異なるため，あるサービスから認証情報が漏えいした場合でも，それを用いて攻撃者が他のサービスに不正なログインを行うことはできない．したがって，両提案方式はパスワードリスト型攻撃に対して安全である．提案方式1において，攻撃者が利用者の PC に侵入してブラウザの Web Storage から秘密鍵を窃取した場合，攻撃者が利用者になりすまして当該サービスにログインすることは可能である．一方で，提案方式 2 は，認証が成功するためには，秘密鍵に加えて AES の鍵が必要となるため，PC 内の情報を窃取するだけでは不十分であるため，安全は確保されている．もし端末ののキーボード入力情報を摂取するキーロガーが仕込まれてしまっていた場合，パスワード認証においては入力したパスワードが摂取されてしまい，攻撃者が利用者になりすます不正ログインが可能であるが，提案方式はともにキーボードの入力だけでは認証情報を窃取されないため安全である．盗聴に対しては，SSL の利用により提案方式およびパスワード認証ともに安全である．

5 Conclusion

本稿では，Web Strorage を利用して，パスワードリスト型攻撃への耐性を持つ公開鍵に基づく認証方式を二種類提案した．提案方式およびパスワード認証の安全性の比較を行い，提案方式が比較的安全であることを示した．本方式は，将来的には安全性の高い認証プラットフォームとして利用も可能であると考えられる．

参考文献

[1] ”CryptoJS”. Retrieved March 10, 2014, from ”https://code.google.com/p/crypto-

表 1: 各方式における攻撃耐性の比較

攻撃手法	提案手法 1	提案手法 2	パスワード認証
パスワードリスト型攻撃	◯	◯	×
端末からの情報漏えい	×	◯	◯
キーロガー	◯	◯	×
盗聴	◯	◯	◯

(◯:攻撃手法に対して安全である, × :十分に安全ではない)

js/”

[2] ”JSEncrypt - Travis Tidwell”. Retrieved March 10, 2014, from ”http://travistidwell.com/jsencrypt/”

[3] T. Kobayashi, R. Yoshida and T. Yamamoto ”Fine-grained access control system with public key(Japanese)”, The 31st Symposium on Cryptography and Information Security, SCIS 2014, 3C2-4, 2014.

[4] ”RSA and ECC in JavaScript”. Retrieved March 10, 2014, from ”http://www-cs-students.stanford.edu/ tjw/jsbn/”

[5] ”TrendLabs Annual Security Roundup(Japanese)”. Retrieved March 10, 2014, from ”http://www.trendmicro.co.jp/cloud-content/jp/pdfs/security-intelligence/threat-report/pdf-2013asr-20140217.pdf?cm_sp=Corp-_-sr-_-2013asr”

[6] ”Web Storage - World Wide Web Consortium”. Retrieved March 10, 2014, from ”http://www.w3.org/TR/webstorage/”