Extending the Scope of CardSpace

Haitham S. Al-Sinani*
Information Security Group
Royal Holloway, University of London
http://www.isg.rhul.ac.uk
Haitham.Al-Sinani.2009@rhul.ac.uk

Chris J. Mitchell
Information Security Group
Royal Holloway, University of London
http://www.isg.rhul.ac.uk
C.Mitchell@rhul.ac.uk

ABSTRACT

The recently-proposed PassCard scheme enables CardSpace to be used as a password manager, thereby both improving the usability and security of passwords as well as encouraging CardSpace adoption. However, this scheme does not work with sites using HTTPS, seriously limiting its practicality. In this paper we extend PassCard to support sites using both HTTP and HTTPS. Usernames and passwords are stored in CardSpace personal cards, and these cards can be used to sign on transparently to corresponding websites. PassCard does not require any changes to login servers, default browser security settings or to the CardSpace identity selector; in particular, it does not require websites to support CardSpace. PassCard operates with both the CardSpace and the Higgins identity selectors without any modification. We describe how this new version of PassCard operates, and give security and usability analyses.

General Terms

Security

Keywords

CardSpace, Password Manager, Browser Extension

1. INTRODUCTION

The most common means of user authentication remains the use of passwords, despite their well-known shortcomings. Moreover, as the number of on-line services requiring passwords continues to grow, users increasingly re-use passwords, write them down in insecure ways, and/or employ passwords which can be readily guessed. The result is an ever-increasing risk of exposure of passwords to malicious parties. Passwords could also be stolen [5] through key logging, phishing, sniffing, shoulder surfing, etc.

A solution that enables the use of site-unique strong passwords whilst maintaining user security and privacy is thus needed. Password managers of various types have been proposed to meet this need. A password manager stores usernames and passwords and makes them available when required. Users are not required to remember any passwords apart from a single master password which can be used to lock/un-lock the password manager.

CardSpace is a user-friendly tool supporting user authentication. To sign on to a site, a user selects a virtual card, known as an information card (InfoCard), from an interface provided by the CardSpace identity selector (or just the selector), instead of providing a username and password.

Despite the introduction of CardSpace (and other similar systems), the vast majority of websites still use username-password for authentication, and this is likely to continue for at least the next few years [5]. One major problem with CardSpace, and with other similar systems providing more secure means of user authentication, is that the transition from username-password is extremely difficult to achieve. Service providers will not wish to do the work necessary to support CardSpace if very few users employ it; equally, users are hardly likely to use CardSpace if it is only supported by a tiny minority of websites. PassCard is designed to help overcome this barrier to change by allowing an evolutionary deployment of CardSpace, initially as a password manager and subsequently, once users are familiar with its interface, as a more sophisticated means of user authentication.

In this paper we propose a new version of PassCard. The goal is to develop a simple and intuitive approach to password management, transparent to both the selector and relying parties (RPs). Unlike the previous version of PassCard [2] which only supports sites using HTTP, the new version supports both HTTP and HTTPS-enabled sites.

The remainder of the paper is organised as follows. Section 2 introduces CardSpace, and, in section 3, we present the proposed scheme. Section 4 highlights a number of PassCard features and limitations. Section 5 reviews related work, and, finally, section 6 concludes the paper.

2. CARDSPACE

CardSpace [7] is Microsoft's implementation of a digital identity metasystem, in which digital identities are represented to users as Information Cards (or InfoCards). There are two types of InfoCards: personal (self-issued) cards and managed cards, issued by remote IdPs. CardSpace is supported in Internet Explorer (IE) from version 7 onwards. Extensions to other browsers, e.g. Firefox and Safari, also exist.

*This author is sponsored by the Diwan of Royal Court, Sultanate of Oman.

SIN'11, November 14–19, 2011, Sydney, Australia.

An updated version, CardSpace 2.0 Beta 2, was released, although Microsoft announced in early 2011 that it will not ship; instead Microsoft released a technology preview of U-Prove (`http://blogs.msdn.com/b/card/archive/2011/02/15/beyond-windows-cardspace.aspx`). We refer throughout to the CardSpace version that is shipped by default as part of Windows Vista and Windows 7, that is available as a free download for XP and Server 2003, and which has been approved as an OASIS standard [6].

Personal cards are created by users themselves, and the claims listed in such cards are asserted by the self-issued identity provider (SIIP) that co-exists with the CardSpace identity selector. Personal cards can contain claims of 14 editable types, including *First Name*, *Last Name*, and *Web Page*. We use personal cards to enable CardSpace to function as a password manager. Hence, we next describe how the personal card protocol operates [3] (in the case where the RP does not employ a security token service).

1. User Agent (UA) → RP. Request: GET (a login page).
2. RP → UA. A login page is returned in which the RP security policy is embedded.
3. User → UA. The RP page offers the option to use CardSpace; selecting this option activates the selector, which is passed the RP policy. If this is the first time that this RP has been contacted, the selector will display the identity of the RP and give the user the option to either proceed or abort the protocol.
4. Selector → User. The selector highlights InfoCards matching the policy.
5. User → Selector. The user chooses (or creates) a personal card. The user can preview the card to ensure that they are willing to release the claim values.
6. Selector ⇌ SIIP. The selector sends a Request Security Token (RST) to the SIIP, which responds with a Request Security Token Response (RSTR).
7. UA → RP. The RSTR is passed to the UA, which forwards it to the RP.
8. RP → User. The RP verifies the token, and, if satisfied, grants access.

Note that if the visited site uses HTTPS, then the RSTR message in step 7 (above) will be encrypted using the public key of the visited site. This explains why the previous version of PassCard, which relies on a browser extension that parses the RSTR, only works with websites using HTTP. The extension does not have access to the key necessary to decrypt the RSTR, and hence cannot operate. This issue is addressed in the revised PassCard scheme described below.

3. PASSCARD

We now describe the operation of the new version of PassCard. The parties involved are an RP, a CardSpace-enabled UA (e.g. a suitable web browser), an HTTP Server (HS), and a browser extension implementing the protocol described in 3.1. The HS is user-selectable, can be any HTTP site, and is used in a passive manner. The introduction of the HS addresses the main limitation with the previous version of PassCard (described at the end of section 2), thus enabling PassCard to also operate with HTTPS-based websites.

Either prior to, or during, use of PassCard, the user must first create a special personal card, referred to here as a PassCard, containing a username and password in certain card fields, the choice of which is implementation-specific (e.g. the user could insert their username in the *First Name* field and password in the *Last Name* field). Basic protection against phishing can be provided if the URL of the target site is included in the PassCard (e.g. it could be inserted in the *Web Page* field). However, this is optional, as users may wish to use a single PassCard with multiple websites sharing the same user credentials. For ease of identification, the user can give the PassCard a meaningful name, e.g. of the target site. The user can also upload an image for the PassCard, e.g. containing the logo of the intended site.

3.1 PassCard Operation

We now describe the protocol steps for PassCard. Its operation differs depending on whether the RP uses HTTP or HTTPS. We therefore divide the protocol description into two cases. Protocol steps 1–3b are the same for both cases, and are described first. Note that the HTTP case is precisely the same as the previous version of PassCard [2]; we include the description here for completeness.

1. UA → RP. HTTP Request: GET (a login page).
2. RP → UA. (the login page is returned).
3. The browser extension performs the following processes.
   (a) It scans the RP page for a form containing a username and password field and a submit button. If all found, it continues; otherwise it terminates.
   (b) It determines the protocol (i.e. HTTP or HTTPS) in use with the RP.

Execution continues as described in sections 3.1.1 and 3.1.2, depending on whether the RP is using HTTP or HTTPS.

3.1.1 HTTP-based PassCard.

Steps 3c–8 (below) apply for HTTP-based RPs.

3. Following step 3b in section 3.1, the browser extension continues as follows.
   (c) It adds CardSpace-enabling tags to the login page, setting the embedded security policy to require a token asserting claims of the types in which the user credentials are stored.
   (d) It adds a function to the login page to intercept the RSTR that will be returned by the selector.
   (e) It embeds a PassCard icon to the page, causing it to appear above the submit button (see Fig. 1).
4. The user clicks on the icon and the selector lights up.
5. The user selects (or creates) and submits a PassCard. The selector creates and sends an RST to the SIIP, which responds with an RSTR.
6. The selector passes the RSTR to the browser.
7. The browser extension performs the following tasks.
   (a) It intercepts and parses the RSTR.

(b) If the token contains the URL of the target site, the extension compares it with the URL of the visited site, and only proceeds if they match.

(c) It extracts the username and password from the pre-specified fields.

(d) It auto-populates and submits the login form.

8. The RP verifies the credentials and, if satisfied, grants access.

3.1.2 *HTTPS-based PassCard.*

Steps 3c–8b (below) apply for HTTPS-enabled RPs.

3. Following step 3b in section 3.1, the browser extension continues as follows.

(c) It obtains the page's URL, referred to throughout as the 'target URL'.

(d) It causes the PassCard icon to appear above the submit field, in such a way that clicking it results in an HTTP redirect.

4. If the user clicks the icon, the browser is redirected to the pre-specified HS, and the target URL is also transmitted as a URL query parameter.

5. While interacting with the HS, the browser extension:

(a) adds a 'hidden' password login form to the returned HS page, if it does not have one, at which point steps 3c–3e of section 3.1.1 are executed;

(b) recovers and stores the target URL; and

(c) transparently invokes the selector, at which point steps 5–7c of section 3.1.1 are executed.

7. The browser extension continues as follows.

(d) It encrypts the username and password values with a secret key known only to the browser extension (see section 3.2).

(e) It transparently redirects the user to the target URL, and the 'encrypted' username-password values are transmitted as URL query parameters.

8. While interacting with the target URL site, the extension:

(a) recovers and decrypts the username and password values; and

(b) auto-populates and auto-submits the login form, after which step 8 of section 3.1.1 is executed.

3.2 Discussion

The PassCard user experience when operating in HTTP mode is the same as with 'conventional' password-based authentication except that, instead of manually entering and submitting a username and password, the PassCard user selects and submits a virtual card. The user experience in HTTPS mode is similar to that of HTTP mode, except that users (depending on their Internet speed, machine speed, etc.) may or may not experience a redirect from the target HTTPS site to the HS and vice versa, resulting in the temporary display of the HS page. Note that, in both PassCard modes, users are not required to click the PassCard icon more than once or to remember any passwords.

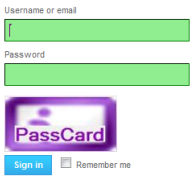

Figure 1: PassCard Icon

Regardless of whether or not an RP already supports CardSpace, the browser extension will always add the PassCard icon to the RP page, as long as it detects username-password prompts on the page. Informal tests on the prototype implementation [1] suggest that this will not disrupt the normal operation of CardSpace. Thus, if an RP supports CardSpace and simultaneously supports username-password authentication, the browser extension will still insert the PassCard icon above the submit button of the password login form. In such a case, users can sign on using any of PassCard, CardSpace or a manually entered password.

The encryption of the username and password in step 7d of section 3.1.2 is not necessary to prevent channel eavesdropping, because an SST/TLS channel is already established between the browser and the target HTTPS site. However, if the username and password are sent as part of the URL (as is the case in the PassCard prototype [1]), then, if sent in plaintext, these values will be vulnerable to shoulder-surfing attacks, since they will be shown in the browser address bar (and possibly also in the browser status bar). The prototype implementation uses a simple symmetric encryption scheme for this purpose to minimise the overhead.

The HS can be any HTTP site, and is not actively involved in the protocol except to serve a web page when requested. It simply acts as a convenient way to avoid the problems associated with use of CardSpace with an HTTPS-enabled site. The choice of HS is not particularly security-sensitive in that it does not learn any sensitive data; it simply learns that a client at a particular IP address is using PassCard. The choice of HS is a configuration option in the prototype, so if the default site is regarded as privacy-threatening then a user can change it to any trusted site address, e.g. that of a personal web page.

4. PASSCARD PROPERTIES

4.1 Security and Usability

PassCard takes advantage of the selector, and is supported by its built-in security features. For example, when invoked, the selector runs in a separate private desktop session, mitigating the risk of other applications, e.g. malware, from interacting or interfering with it. In addition, all values inserted in the fields of a PassCard are stored in encrypted form on the user machine.

The selector identifies the RP to the user and indicates whether or not they have visited that RP before; if the user is visiting this RP for the first time, CardSpace requests the user's permission to proceed. This helps to support mutual authentication since the user and the RP are identified to each other (a security gain over 'conventional' password authentication, where an HTTP-based RP is not identified).

As with any local password manager, PassCard (when running in HTTP mode) avoids the need for trusted third

parties. In addition, the automatic form-filling feature reduces exposure to shoulder-surfing attacks and also helps to thwart key loggers. PassCard reduces the threat of phishing attacks involving impersonation of legitimate sites by comparing the URL in the PassCard with that of the visited website. PassCard also supports the use of strong per-site passwords, since users no longer need to memorise or write down passwords. PassCard does not require any changes to the default IE configuration, thereby avoiding potential vulnerabilities resulting from lowered browser security settings.

PassCard provides a simple, intuitive user experience via its use of the selector interface. At the same time, it familiarises users with CardSpace, thereby potentially facilitating future adoption of more secure means of authentication. Unlike other password managers which represent credentials in text form, PassCard credentials are stored in PassCards which can be equipped with a readily recognisable image.

PassCard operates completely transparently to external parties, and hence does not require any changes to RPs, identity selectors or to browser settings. PassCard is also flexible, since users can choose whether or not to use it simply by electing to click the PassCard icon (or not). In addition, it gives flexibility in the choice of the HS when running in HTTPS mode.

By making use of CardSpace features, PassCard supports a degree of roaming. A user can transfer PassCards from one PC to another using the CardSpace backup facilities. If the CardSpace backup file (which holds data in encrypted form) is stored on a portable storage medium (e.g. a USB drive) then full mobility is provided, as well as robustness in the form of protection against loss of credential data.

4.2 Limitations

An obvious limitation is that anyone with access to a Windows user account can access the PassCards and use the stored credentials. This is a fundamental limitation of CardSpace which, by default, does not impose any additional protection on the use of the selector. To address this issue, we observe that CardSpace allows individual InfoCards to be PIN-protected, which should be considered for PassCards stored on machines accessible to other users. It may also be possible to cause CardSpace to run under UAC (User Account Control), so that running CardSpace causes Windows to prompt the user for an administrator password. In addition, it could be required that, during the process of user authentication on a PC using CardSpace, a random and short-lived one-time password is sent to a user's mobile device; this must then be entered into the PC by the user when prompted [4].

The plug-in must scan every browser-rendered web page to detect whether it supports password authentication, and this may affect system performance. However, informal tests on the PassCard prototype [1] suggest that this is not a serious issue. Also, the plug-in can be configured so that it only operates with certain sites.

5. RELATED WORK

Password managers, which store passwords in a (secure) location either on the user PC or remotely, are now widely available. They typically store passwords in encrypted form and, unlike PassCard, require users to use a single master password to access the password store. Some are also capable of masking passwords, and others, much like PassCard, provide automatic password entry. Examples of password managers include open source schemes such as Password Safe (`passwordsafe.sourceforge.net`), KeePass (`KeePass.info`), and PINs (`http://mirekw.com/winfreeware/pins.html`) as well as commercial products such as RoboForm (`roboform.com`), and Any Password (`anypassword.com`).

Perhaps the most distinctive feature of PassCard is its dependence on CardSpace, whereas most other password managers are independent applications. PassCard can thus benefit from the CardSpace security features, which may give users greater confidence in its use.

6. CONCLUSIONS

We have proposed a new version of PassCard, which (unlike its predecessor) supports almost all websites, including those using HTTPS. Users store their usernames and passwords in CardSpace personal cards, and use such cards to transparently sign on to corresponding websites. PassCard is based on a browser extension, and is transparent to login servers, identity selectors, and browser security settings. It does not require websites to support CardSpace.

The scheme uses the CardSpace identity selector interface to seamlessly authenticate users to websites. It extends the use of personal cards to allow for transparent password management. Such an approach could help to extend the applicability of CardSpace. A full version of this paper, including a description of a prototype implementation, is available [1].

7. REFERENCES

[1] H. S. Al-Sinani and C. J. Mitchell. *Implementing PassCard — a CardSpace-based Password Manager*. Technical Report: RHUL–MA–2010–15 (Department of Mathematics, Royal Holloway, University of London), 2010. `http://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-15.pdf`.

[2] H. S. Al-Sinani and C. J. Mitchell. Using CardSpace as a password manager. In E. de Leeuw, S. Fischer-Hübner, and L. Fritsch, editors, *Proceedings of IFIP IDMAN'10*, volume 343 of *IFIP Advances in Information and Communication Technology*, pages 18–30. Springer, Boston, 2010.

[3] H. S. Al-Sinani and C. J. Mitchell. Client-based CardSpace-OpenID interoperation. In *Proceedings of ISCIS'11*. Springer [LNEE], (to appear), 2011.

[4] H. S. Al-Sinani and C. J. Mitchell. Enhancing CardSpace authentication using a mobile device. In Y. Li, editor, *Proceedings of DBSEC'11*, volume 6818, pages 201–216. Springer (LNCS), 2011.

[5] C. Herley, P. C. van Oorschot, and A. S. Patrick. Passwords: If we're so smart, why are we still using them? In R. Dingledine and P. Golle, editors, *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, 230–237, 2009.

[6] M. B. Jones and M. M. (editors). *Identity Metasystem Interoperability Version 1.0 (IMI 1.0)*. OASIS Standard, 2009. `http://docs.oasis-open.org/imi/identity/v1.0/identity.html`.

[7] M. Mercuri. *Beginning Information Cards and CardSpace: From Novice to Professional*. Apress, New York, 2007.