NFC Based Mobile Single Sign-On Solution as a Chrome Extension

Ufuk Celikkan and Can Gelis
Department of Software Engineering, Izmir University of Economics,
Faculty of Engineering and Computer Sciences, Izmir,Turkey

Keywords: Single Sign-On, NFC, Authentication, Password, Security.

Abstract: We describe the design and implementation of Single Sign-On authentication solution that uses a Near Field Communication enabled mobile phone. Such a solution relieves the users from remembering multiple username and passwords when authenticating themselves to various services on the internet. Mobile phones are today's ubiquitous computing devices, used for a wide variety of purposes including authentication, tracking, medical care, entertainment and electronic payment. The primary advantage of NFC technology is that since it uses short range communication, it inherently provides another level of security, and being contactless, it is easy to use. Our solution is seamlessly integrated into the Chrome browser via a browser extension that allows users easy authentication and management personal information on the phone. The Google Chrome extension is written in JavaScript; However, this code (JavaScript) when running in a browser, cannot access the system resources of the computer due to browser security restrictions. Therefore a program written as a Java applet is implemented to run in the user's computer. This applet, injected into the current web page by the extension, provides access to NFC Reader and supplies the bridge between Java and JavaScript. The user does not need to enter any account information, because it is retrieved from the phone via NFC and automatically submitted to the web login page.

1 INTRODUCTION

Management of authentication information for multiple accounts is becoming an increasing problem as the internet growth continues. The oldest and most widely used mechanism for verifying the identity of a subject and for authenticating is the use of a username and password. Choosing the same username and password for all the accounts to circumvent the problem can be dangerous because if your authentication information is compromised then all your accounts are exposed. Moreover, it may not even be possible to choose the same username for different web sites. Maintaining synchronized passwords for multiple accounts can be difficult, since a change in one account requires all others to be changed which requires considerable effort and time. On the other hand, choosing different account names and passwords creates a challenge of remembering many different account names and passwords (Felten and Gaw, 2006, Florencio and Herley, 2007). Besides, a "good" password that contains numbers and special characters and is longer than 8 characters (US-CERT, 2009) increases the challenge of remembering passwords. There are several solutions offered to solve this problem, based on the notion of Single Sign-On (SSO). Browsers already provide a password vault to remember and auto-fill passwords, but this is not a secure solution as the security of the passwords depends on the security of the machine. Alternatively, the password managers may choose to store authentication information encrypted in the cloud that will prevent it to be read in clear. The drawback of this solution however, is that, you are no longer in the possession of your data. Therefore, you give up the integrity of your data which is vulnerable to processing of it. It can be seized, analyzed, distributed and attempted to be broken-in legally or illegally. Enterprise level single sign-on solutions some of which are based on Kerberos (Steiner et.al, 1988) address the same problem at an enterprise level which requires high assurances while other SSO solutions such as the OpenID are primarily targeted to Web environment to login to websites, blogs etc. (OpenID; Sovis et. al, 2010).

Storing usernames and passwords on an NFC enabled phone and seamlessly integrating it into a

Celikkan U. and Gelis C..
NFC Based Mobile Single Sign-On Solution as a Chrome Extension.
DOI: 10.5220/0005056703370343
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 337-343
ISBN: 978-989-758-045-1

browser is a secure and convenient solution. It provides two-factor authentication with the convenience of Single Sign-On solution. NFC based SSO provides much better security than password synchronization methods which maintains one single synchronized password for all applications (Chinitz 2000). In addition, some level of security is inherent in the nature of short range communication which is used by the NFC technology (Coskun et.al, 2013). Some RFID standards, Bluetooth and NFC readers exhibit similar security characteristics from an hacking perspective. They are both open to man-in-the-middle attacks and eavesdropping. However, sniffing communications and man-in-the-middle attacks are much harder to setup in NFC communications when compared others due to the close range between devices and the low power RF field of communication. However, the ultimate security is achieved by encrypting the channel communication between the reader and the phone using a key exchange algorithm such as Diffie-Hellman.The phone acts as an information depository to store web sites' URLs, usernames and passwords, and when a particular web site is accessed, the username and password fields are filled automatically from the phone. This allows users to login to web sites and access to their accounts by a single entry of the phone PIN. Once the user is authenticated to the phone using the PIN, no more manual username/password entry is required to access other accounts. This avoids the danger of over-the-shoulder-eavesdropping due to manual entry of username and password where the environment lacks sufficient privacy.

A phone based SSO is called local pseudo SSO, because a separate authentication occurs every time the user logs into a web site using authentication information stored locally at the web site (Pashalidis and Mitchel, 2003, De Clercq, 2002). This is in contrast to a true SSO, such as the Kerberos protocol in which a trusted third party manages your credentials. Kerberos is available in modern operating systems. A commercial example of a local pseudo-SSO solution is Password Director (Password Director, 2014) which stores the passwords in an encrypted password database on local disk and has support for USB flash drives.

Our main objective is to create a SSO solution with a two-factor authentication. From an SSO perspective, a user who knows the phone PIN can be authenticated on other sites without entering another password. In addition to knowing the PIN ("something you know"), the user should have the possession of the phone ("something you have") to activate the system. The second objective is to create an easily installed solution with minimal foot-print and which is convenient to use. The solution described in this paper requires the installation of the Google Chrome extension (Chrome, 2014) and the Android application.

Using an NFC enabled phone for login helps to address the problem of misspelled and fake URLs. For instance, if one enters mail.*yohoo*.com (i.e. a fake address with malicious intent) erroneously in the browser, an attempt to match this URL with the one on the card will fail, and login will not occur. Due to the storage capability of a phone, the login history of a user can be stored on the card, which later can be used to verify suspicious logins, login attempts and also for auditing purposes.

The rest of the paper is organized as follows. Section 2 includes an overview of the system architecture of the solution. In Section 3, the authors describe the implementation details of the system. Finally, Section 4 discusses the advantages of using the system, the remaining issues and future work and enhancements.

2 SYSTEM ARCHITECTURE

The phone is integrated into the browser through a browser extension. The solution has two components: an Android application that runs on the mobile phone, and a browser extension that runs on the host machine. An NFC reader attached to the computer facilitates the communication between the mobile phone and the computer. The overall high level view of the system architecture is shown in Figure 1.

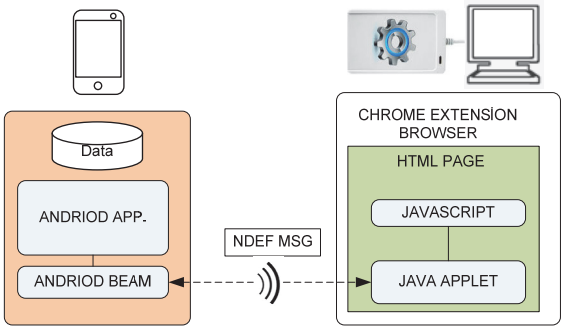

Figure 1: System overview.

After the extension is installed, it is activated only when a user attempts to login to a web site, which results in the injection of an applet into the web page. The applet then starts polling the NFC

device. Having received the needed data from the mobile device to login to a website, namely a username and password, the extension initiates the authentication sequence by automatically filling the login fields on the form and submitting the request. The actions performed by the extension during the login process are shown in Figure 2.

2.1 Android Application

The Android application on the phone stores user account information and transmits this information to the browser on the host machine via NFC. NFC Communication is provided by high-level Android NFC libraries that use Simple NDEF Exchange Protocol (SNEP, 2013) to transmit data to the peer device (SNEP is explained in the Implementation Section). SNEP protocol messages are in NFC Data Exchange Format (NDEF, 2006) where user account information is sent in the payload field.

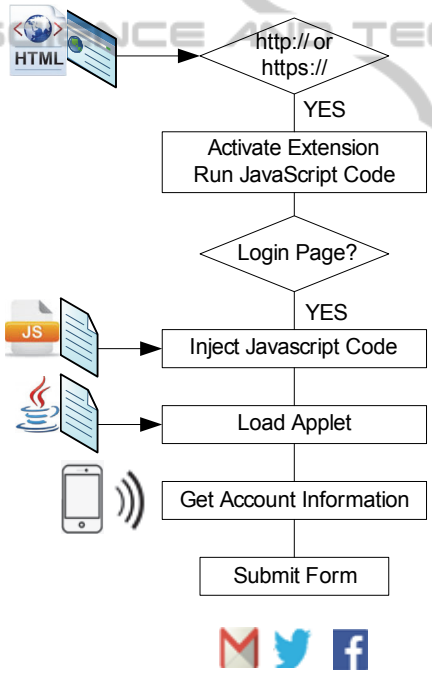

Figure 2: Extension's authentication flow.

The phone sends messages to the NFC Reader via Android Beam and the applet injected into the web page receives those messages. User account information is stored on the phone in a SQLite database as triplets that contain domain, username and password information. This information in the database is secured by the operating system. By default, Android OS security architecture does not allow one application to access to another application's private data, such as files, without acquiring an explicit permission. However, if the phone is compromised and root privileges are acquired then the data in the database is exposed together with other data and applications on the phone. Encrypting the data on the database partially addresses the problem, since, now the user must enter the encryption key every time authentication to a web site is needed. This is of course in contrary to the idea of SSO solution that the paper proposes. A process that makes an attack harder could be employed as flows: Encrypt data in the database using the secret key generated from the PIN using PBKDF2WithHmacSHA1 algorithm. The user enters the PIN when the application starts a "session" after which passwords are unlocked and stored in clear in the memory as long as the user session is active. When the user ends the session due to the termination of the application, the clear passwords in the memory is deleted. However, note that this provides no extra security and will not eliminate an attack to recover clear passwords due to a compromised and rooted phone. This simply makes the attack little harder and when the application is not running the passwords remain encrypted in the database. Figure 3 shows user interface of the Android application.

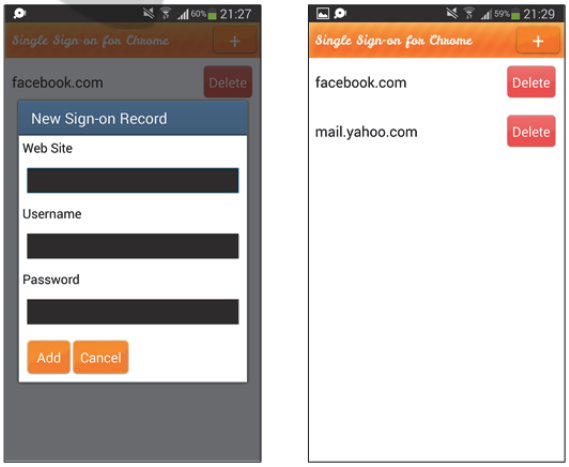

Figure 3: Android application's user interface.

The android application has the following account management functions:

- Add/Remove account: User can add and remove accounts;
- Update Account: User can update account;
- Backup Data: The information on the telephone will be backed up on an NFC card or tag. The user will be able to use this card for login purposes as well.

2.2 The Brower Extension

The extension is implemented in JavaScript. The *contents_script* section of the extension's manifest file indicates when to activate the extension and which JavaScript file to run. When a login page is encountered, the JavaScript code dynamically adds a second JavaScript file to the html document, and also loads the Applet. The extension is logically divided into two as Applet and Browser layers, each performing a phase of the authentication process.

2.2.1 Applet Layer

This layer is a bridge between the mobile phone and the browser. It polls the NFC reader for the presence of a phone, retrieves the data using NFC hardware and passes the data to the browser. This layer provides peer-to-peer connection between the mobile device and the applet in which data is carried in NDEF format and transmitted using Simple NDEF Exchange Protocol. The payload of the NDEF message carries JavaScript Object Notation formatted account information. JSON is a lightweight data-interchange format which can be easily parsed by the machines and read by the humans.

2.2.2 Browser Layer

The browser layer is written in JavaScript and HTML. The corresponding JavaScript code and the HTML applet tag code to load Java Applet are injected into the web site for every web page that contains a login form. Browser layer parses the JSON formatted data received from the applet to extract username and password and completes the authentication by auto-filling the login fields of the page with account information, followed by the submission of the page.

3 IMPLEMENTATION

NFC-SSO implementation has two main components, as shown in Figure 4. The mobile application is a Java program running on Android operating system, and host part is implemented as a Chrome browser extension using JavaScript, HTML, CSS and includes a Java applet. An NFC reader provides the contactless communication between the NFC enabled phone and the host. ACR122U NFC contactless Smart Card Reader is used as the NFC Reader (ACR122U). The implementation is tested on an NFC enabled phone running Android 4.1.2 and Chrome browser running on Windows operating system.

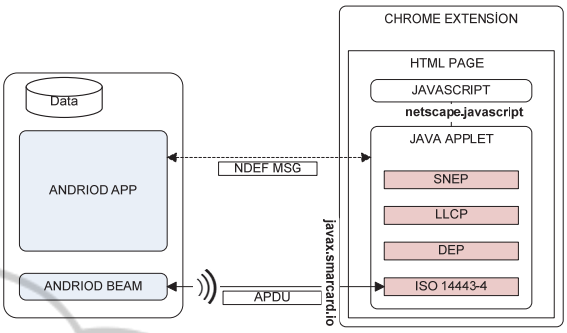

Figure 4: Detailed view of the system.

Peer to peer mode is used between the NFC phone and the NFC reader, in which devices send data to each other using the Simple NDEF Exchange Protocol. The Android application uses Android Beam, which in essence is the Link Layer Control Protocol (LLCP, 2011) and handler system that enables an application to respond automatically when the phone is in range of another device that supports NDEF push. LLCP is similar to TCP or UDP for HTTP requests, handling the exchange of NDEF messages between two active devices.

3.1 Android Application

The Android Application stores the user account information and pushes this data, encapsulated in an NDEF message, to the Java Applet residing on the host via Android Beam. Android Beam is part of Android's native NFC libraries (android.nfc) (Android, 2014). It uses SNEP protocol as the default transmission mechanism, falling back to a proprietary mechanism called NDEF Push Protocol (NPP). The application implements the *CreateNdefMessageCallback* interface which has the *createNdefMessage* method to create an NDEF message to push. NDEF is the container format to store NFC data in NFC tags. NDEF is a lightweight binary message format designed to encapsulate one or more application-defined payloads into a single message construct. An NDEF message contains one or more NDEF records, each carrying a payload of arbitrary type, and can be as large as 2^{32}-1 octets in size (NDEF, 2006). *NdefMessage* class provides the abstraction for the NDEF messages. *createNdefMessage* method is called when the phone is in range of another device that might

support NDEF push, thus allowing the application to create the NDEF message only when it is required.

The account information to be sent in the message payload is stored in a SQLite database on the mobile device, as shown in Figure 5. The information is stored as triplets that contain domain, username and password information and converted into JSON format for easy parsing by the JavaScript before it is sent to the host machine.

Auth		
PK	id	INTEGER
	domain username password	VARCHAR(255) VARCHAR(255) VARCHAR(255)

Figure 5: Authentication information database table.

3.2 Android Manifest File

Android manifest.xml file needs the ***uses-permission*** element i.e., ***<uses-permission android:name = "android.permission.NFC"/>*** to get permission to access to the NFC hardware. Since the application depends on the NFC hardware, the following ***uses-feature*** element is included in the manifest file: ***<uses-feature android:name= "android.hardware.nfc" android:required="true "/>***. In order for the single sign-on application to be dispatched in response to an NDEF payload (i.e. when the phone is near to the NFC reader), the following intent-filter element is added: ***<action android:name = "android.nfc.action. NDEF_DISCOVERED"/>***

3.3 Host Application: Chrome Extension

The host application is simply a Chrome extension. The extension achieves sign-on by injecting JavaScript and applet code into the page containing the login form. In order to do this, *content_scripts* section in the manifest file is needed to access to the current page's Document Object Model tree. *content_scripts* section is also used to activate the extension and specify which JavaScript files to run. The JavaScript code checks the existence of the login form on the current web page by searching **<input type="text">** and **<input type="password">** tags in the HTML form. The presence of these tags indicates that the current web page is indeed a login page. Consequently, a second JavaScript code and the Java applet are programmatically injected into the web page. The JavaScript function *login()* called by the Java Applet is responsible for passing the data received from the phone to the login form and submitting the form. *netscape.javascript* library is used to implement the bridge between the applet and the browser's java script code. The extension obtains the URL from the browser and auto-completes the username and password fields on the page.

3.4 Java Applet

The Java Applet is responsible for retrieving the data from the NFC Reader and passing the data to the browser. When the extension recognizes a web page as a login page the applet is injected into the Web page. On Android API level 14, Android Beam performs NFC peer-to-peer communication using LLCP with SNEP being the default data transfer protocol. Therefore, to match the protocol used by the Android Beam, SNEP over LLCP is implemented in the applet code to transfer data from the Android NFC phone.

The applet uses *javax.smartcard.io* library for the data transfer between the reader and the applet. This library provides an abstraction between the physical device and the Java code where the Java methods are bound to the underlying PC/SC (Personal Computer/Smart Card) subsystem (PCSC, 2014). The *Card.transmitControlCommand* method in *javax.smartcard.io* library is the wrapper for PC/SC's *SCardControl* function. In PC/SC architecture, the *SCardControl* function implements the dialogue between an application and the reader, regardless of whether there is a card in the slot. This function gives you direct control of the reader. **Direct Transmit** is used to send the data to the NFC reader providing data transmission to the reader when no NFC device or card is presented on the reader. In the other words, the data is transmitted to the reader directly not to the card or the device.

A polling mechanism is implemented because most NFC readers currently on the market do not support this mechanism for peer-to-peer connections. Applet's polling mechanism samples the status of the reader every 500 milliseconds to detect the presence of a device. If a device is detected by the reader, polling stops and the data transmission starts. Figure 6 shows the calls made to detect the presence of an NFC Reader and polling of an NFC enabled phone. Figure 7 shows the DirectTransmit mechanism used to send Application Protocol Data Units to the NFC reader.

```
import javax.smartcardio.*;

private static final byte[] FELICA_POLLING = { (byte) 0x4A, (byte) 0x01, (byte) 0x02,
  (byte) 0x00, (byte) 0xFF, (byte) 0xFF, (byte) 0x00, (byte) 0x00 };

TerminalFactory factory = TerminalFactory.getDefault();
CardTerminal terminal = factory.terminals().list().get(0);
Card card = terminal.connect("direct");
// Start Polling
while (true) {
   response = sendCommand(FELICA_POLLING);
   if (response[2] == (byte) 0x01)
    break;
   Thread.sleep(S_TIME);
}
```

Figure 6: NFC reader discovery.


```
private static final byte[] ACR122_HEADER = {
 (byte) 0xFF, (byte) 0x00, (byte) 0x00, (byte) 0x00 };
private static final int ACR122_CONTROL_CODE = 0x310000 + 3500 * 4;

private byte[] sendCommand(byte[] cmdIn) throws CardException {
  /* command = command length + 0xD4 + ACR122 HEADER + command */
  byte[] cmd = Util.catBytes((byte) (cmdIn.length + 1),(byte) 0xD4);
  cmd = Util.catByteArrays(ACR122_HEADER, cmd);
  cmd = Util.cat ByteArrays(cmd, cmdIn);

  byte[] response = card.transmitControlCommand(ACR122_CONTROL_CODE, cmd);
  return response;
}
```

Figure 7: Direct Transmit of APDU's.

After data is retrieved from the NFC reader, it is passed to the browser using *netscape.javascript* library. This library provides a bridge between the Java Applet and the JavaScript code. The library has the ability to call JavaScript functions from Java or alternatively Java methods can be called through JavaScript.

4 CONCLUSION

The ubiquitous mobile phone has many uses, from entertainment to mobile payments. In this paper we demonstrate a mobile phone based SSO solution that is seamlessly integrated into Chrome as a browser extension to address the problem of remembering multiple passwords and account names. Authenticating to multiple web sites using different account names and passwords causes username and password management difficulties and in some cases, security issues. The solution provided in this study consists of an application for the Android phone and an extension for the Chrome browser.

The solution presented in the paper is effective against key loggers since authentication information is auto-filled without user typing any information. It can also be used to address the phishing problems because the URL of the imitated or look-alike web site will not match the one stored on the phone. The solution however, cannot handle man-in-the-browser attacks. The man-in-the-browser attacks targeting a browser must be handled in a more general setting by employing several mitigation factors, of which awareness is the starting point. Number one priority is protecting the computer. Enhancing the browser security against malware and trojans by hardening

the browser is also helpful against man-in-the-browser attacks.

The only hardware requirement is a USB NFC reader and its associated device driver. Even though not wide spread, there are commercial desktops, laptops, tablet PCs, keyboards and monitors that are equipped with built-in NFC readers (antenna). This is very promising and gives the solution presented in the paper a better chance to be accepted by the end user. *javax.smartcard.io* provides the low level communication with the NFC reader. The software is architected following design pattern guidelines, and therefore easily extendible to support other hardware and software tokens. The content of the card is backed up encrypted for recovery purposes.

Currently, the solution works only on Chrome browser and only for web forms. Supporting other browsers, while maximizing the code reuse in the mean time, requires a number of changes, some of which are architectural changes in our solution. In order to be able to automatically provide username and password data to a web site, the web site's login page is parsed for the field names. Although for well established web sites field names are not subject to frequent change, any changes can adversely affect the functioning of the solution because automatic login relies on these field names. One approach to this problem is to store the field names on the phone and update them when a web site changes the field names. The extension then reads the field names from the phone in addition to the username and passwords.

The solution presented in the paper can be enriched by adding features such as automatic password capture, password generator and, password strength analyzer. Currently, the implementation does not support NFC tags and cards; therefore functionality needs to be added to manage tags and cards through the browser extension. However, the most important priority is adding support to enable the solution to be used with different browsers.

REFERENCES

Felten, E.W. and Gaw, S., 2006. Password management strategies for online accounts, In *Proceedings of the second symposium on Usable privacy and security*, 2006, pp. 44-55.

Florencio, D. and Herley, C., 2007. A large-scale study of web password habits, In *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 657 – 666.

US-CERT, 2009. Choosing and Protecting Passwords, http://www.us-cert.gov/cas/tips/ST04-002.html, [Accessed 8 April 2014].

OpenID, http://www.openid.net, [Accessed 8 April 2014].

Steiner, J.G., Neuman, C. and Schiller, J.I., 1988. Kerberos: An Authentication Service for Open Network Systems, In *Proceedings of Winter USENIX Conference, 1988*.

Sovis, P., Kohlar, F. and Schwenk, J., 2010. Security analysis of OpenID, In *Proceedings of the Securing Electronic Business Processes–Highlights of the Information Security Solutions Europe 2010 Conference*.

Chinitz, J., 2000. Single Sign-On: Is It Really Possible? *Information Systems Security*, 9(1), pp 1-14.

Coskun, V., Ozdenizci, B. and Ok, K., 2013. A Survey on Near Field Communication (NFC) Technology, *Wireless Personal Communications*, August 2013, 71 (3), pp. 2259-2294.

Pashalidis, A. and Mitchel, C.J., 2003. A taxonomy of single sign on systems, In *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, July 9-11, 2003.

De Clercq, J., 2002. Single Sign-On Architectures, In *Proceedings of the International Conference on Infrastructure Security InfraSec '02,* pp 40-58.

Password Director, Last Bit software. http://www.passworddirector.com, [Accessed 8 April 2014].

Chome extension development http://developer.chrome.com/extensions/getstarted, [Accessed 8 April 2014].

SNEP, 2013. Simple NDEF Exchange Protocol. Technical Specification, version 1.0, 2013. NFC Forum. http://members.nfc-forum.org/specs/spec_license, [Accessed 8 April 2014].

NDEF, 2006. NFC data exchange format Technical specification, version 1.0, 2006. NFC Forum. http://members.nfc-forum.org/specs/spec_license, [Accessed 8 April 2014].

ACR122U USB NFC Reader Aplication Programming Interface V2.02. http://downloads.acs.com.hk/drivers/en/API-ACR122U-2.02.pdf, [Accessed 8 April 2014].

LLCP, 2011.Logical link control protocol . Technical specification, version 1.1 2011. NFC Forum. http://members.nfc-forum.org/specs/spec_license, [Accessed 8 April 2014].

Android NFC development. http://developer.android.com/guide/topics/connectivity/nfc/nfc.html, [Accessed 8 April 2014].

PCSC, 2014. http://www.pcscworkgroup.com, [Accessed April 8 2014].