2008

Enhanced Security for Preventing Man-in-the-Middle Attacks in Authentication, DataEntry and Transaction Verification

Jason Wells
Deakin University

Damien Hutchinson
Deakin University

Justin Pierce
Deakin University

DOI: 10.4225/75/57b56646b8774
Originally published in the Proceedings of the 6th Australian Information Security Management Conference, Edith Cowan University, Perth, Western Australia, 1st to 3rd December 2006.

Enhanced Security for Preventing Man-in-the-Middle Attacks in Authentication, Data Entry and Transaction Verification

Jason Wells[1]
Damien Hutchinson[2]
Justin Pierce[3]

School of Engineering and Information Technology[1 2] School of Information Systems[3]
Deakin University
Victoria, Australia
Email: wells@deakin.edu.au

Abstract

There is increasing coverage in the literature highlighting threats to online financial systems. Attacks range from the prevalent reverse social engineering technique known as phishing; where spam emails are sent to customers with links to fake websites, to Trojans that monitor a customer's account log on process that captures authentication details that are later replayed for financial gain. This ultimately results in loss of monetary funds for affected victims. As technological advances continue to influence the way society makes payment for goods and services, the requirement for more advanced security approaches for transaction verification in the online environment increases. This paper has three main purposes. The first is to detail the current threats and vulnerabilities to online financial systems and in particular online banking, from the selected literature. The second is to present the known prevention techniques for protecting against these attacks. The third is to present a conceptual model for authentication, data entry and transaction verification. It is suggested that the design adds another layer of security to existing methods to either prevent a MitM attack or to make the procedure of capturing and reassembling customer log on and transaction details more computationally and time intensive than what it is worth to an attacker. The model is based on a graphical authentication application previously developed called Authentigraph.

Keywords

Internet banking, security, authentication, Man-in-the-Middle attacks

INTRODUCTION

As the use of the Internet by businesses and consumers for performing online services including monetary transactions continues to increase, the requirement for more effective transaction verification techniques to protect customers' credentials remains an issue of paramount importance. In fact it has been stated that the burden imposed by the authentication requirements of multiple systems is arguably one of the main remaining obstacles to wide-spread e-commerce use (Rash 2002). The password remains the most popular authentication mechanism in use today. They have been in use for centuries and are therefore well understood and widely accepted (Renaud 2007). In order to complete any web-based transaction exchange the user will be required to remember and enter their password into an online system. There are two main problems with this form of authentication that our proposed model aims to resolve. The first is the abundance of different passwords that a user must remember when undertaking web-based transactions at multiple sites. The potential for fraud is real and users' will have to deal with this unpleasantness if they fail to maintain security best practice by using different strong passwords for all web accounts (Renaud 2007). The second is that the password used in its traditional form is incapable of protecting users against a Man-in-the-Middle attack. Although there have been several schemes deployed for protection and prevention from the latest attacks including phishing attacks (Zishuang and Smith 2002, Chou et al. 2004, Waterken 2004, Ross et al. 2005, Dhamija and Tygar 2005), several groups have already adapted to counter them (Jackobsson 2005, Jackobsson and Young 2005, Leyden 2005, Parno et al. 2006) and continue to attack the banking system (Johnson and Moore 2007). None of these schemes protect users against a full MitM attack using a compromised terminal (Johnson and Moore 2007). This involves an attacker, known as a middle-person, redirecting the user's traffic, to their e-banking website via the attacker's site. When the user who is a customer of the bank attempts to perform a legitimate transaction, details such as the destination account number and value of the transaction are rewritten before forwarding them to the bank. Responses which show the rewritten destination are also changed before the user sees them and nothing appears amiss (Johnson and Moore 2007). In order to address these two main problems this paper presents a practical implementation of a previously developed graphical authentication application called Authentigraph for preventing MitM attacks in authentication, data entry and transaction verification.

TRADITIONAL AUTHENTICATION TECHNIQUES

Authentication has been defined in various ways (Sandhu and Samarati 1996, p. 241, Kambil and van Heck 1998, p. 5, Renaud and Smith 2001, p.1, Basu and Muylle 2003) converging on a means to verify the identity of a person or a process. This paper defines authentication as the process of determining whether someone or something is who they claim to be, and establishing the identity of one party to another. Authentication has traditionally been centred on what you know, a concept typically linked to Personal Identification Numbers (PINs) and passwords. Pierce et al. (2004b) suggest the fallibility of passwords and PINs is exemplified in several well-known shortcomings implicit in their use. Shortcomings which include the difficulty users experience in remembering strong passwords, the susceptibility to a broad range of electronic attacks, combined with the tendency of users to share passwords.

As a consequence of the relative insecurity of passwords and PINs, differing authentication techniques have been used including token based, the notion of *what you have*, along with biometric authentication, the notion of *what you are*. Both these approaches suffer from vulnerabilities and shortcomings, including the significant investment in extra hardware required to interface with the users, along with privacy and impersonation issues and the susceptibility to attack (Jain *et al.* 2000, Pierce *et al.*, 2004a). This exposes individuals and organisations to potential security threats, and does not remedy the main cause of authentication and password insecurity. There is a void left between the traditional authentication information security authors Warren and Hutchinson (2003) suggest needs to be filled. Many authors in the area of authentication, including Pierce *et al.* (2004a), Furnell *et al.* (2004), Dhamija (2000), Thorpe (2004), Perrig and Song (1999) and Birget *et al.* (2003) have suggested that all of the traditional authentication techniques do not remedy the main cause of authentication and password insecurity, which is the human limitation of memory for secure passwords. This has led to an alternative method for authentication, namely graphical authentication.

GRAPHICAL AUTHENTICATION

Many forms of graphical authentication have been proposed, based on psychological studies showing that humans can remember pictorial representations more readily than textual or verbal representations. These include Déjà vu (Dhamija & Perrig, 2000), PassImages (Furnell et al., 2005), Graphical Passwords (Blonder, 1996), Robust Discretisation (Birget et al., 2003), Draw-A-Secret (DAS) (Jermyn et al., 1999), Passdoodles (Varenhorst, 2004) and Inkblot Authentication (Stubblefield, 2004). These techniques have the potential to fill the gaps left between traditional authentication techniques, including trade-offs between security levels, expense and error tolerance.

Graphical authentication authors have also stated that the developed techniques display extremely low authentication failure rates, suggesting that by using the graphical images and pictures to make up passwords, recall is aided compared to remembering strong textual passwords. The graphical images also allow for larger possible password spaces, where Dhamija and Perrig (2000) suggest that selecting five images from a challenge set of 25 images results in 53,130 possible combinations.

Jermyn *et al.* (1999) suggest that intrinsically the graphical authentication techniques deal with the heightened vulnerabilities introduced to most systems, resulting directly from bad end user behaviour, where users often write down, or share passwords. The graphical authentication techniques make it exceedingly hard for this end user behaviour to occur.

With little research and no reports on real cases of breaking graphical passwords, it is somewhat difficult to quantify their effectiveness. In their survey of graphical passwords, Suo et al. (2006) provide some insight by presenting a subjective comparison of the security offered by a graphical password with its text based counterpart. Threats including a brute force search, dictionary attacks, guessing, spyware and shoulder surfing are all considered. The belief put forward indicated that it is more difficult to break graphical passwords using the traditional attack methods like brute force search, dictionary attack, and spyware which have been plaguing knowledge-based and token-based authentication (Suo et al. 2006). However the latest attacks indicate that breaking graphical passwords is not the objective for the attacker but rather hijacking the session to capture the user's authentication credentials to later replay to the system. Therefore the known techniques for exploiting online banking systems are described in the next section.

CURRENT AUTHENTICATION THREATS AND VULNERABILITIES TO ONLINE BANKING SYSTEMS

This section describes some of the known current threats and vulnerabilities to online financial systems relating to the two main authentication issues stated previously that our proposed model aims to resolve. Examples of vulnerabilities include exploiting known software bugs e.g. buffer overflows, analysing user/browser interaction and the one that is the main focus of this paper is bypassing the authentication mechanism (Renaud 2007). To do

this the attacker performs a MitM attack, to hijack another user's session and take over another user's account (Renaud 2007). The attacker can install a piece of malicious software known as a Trojan horse on the target's computer that watches activity and records user names and passwords as the user enters web sites.

These details are sent to the attacker's computer to be used to gain access to the system (Renaud 2007). These 'banking Trojans' that began to appear in mid-2003, infect the computer of an online banking customer to target the money from their online bank account. Once the customer's computer is infected, the Trojan typically harvests banking information by filtering URLs the user accesses (Ståhlberg 2007). For example, Bancos.NL includes 2,764 different bank URLs from over 100 countries (Ståhlberg 2007). Additional examples of real Trojans that have targeting financial services are presented in figure 1 to demonstrate that these attacks are not just theoretical.

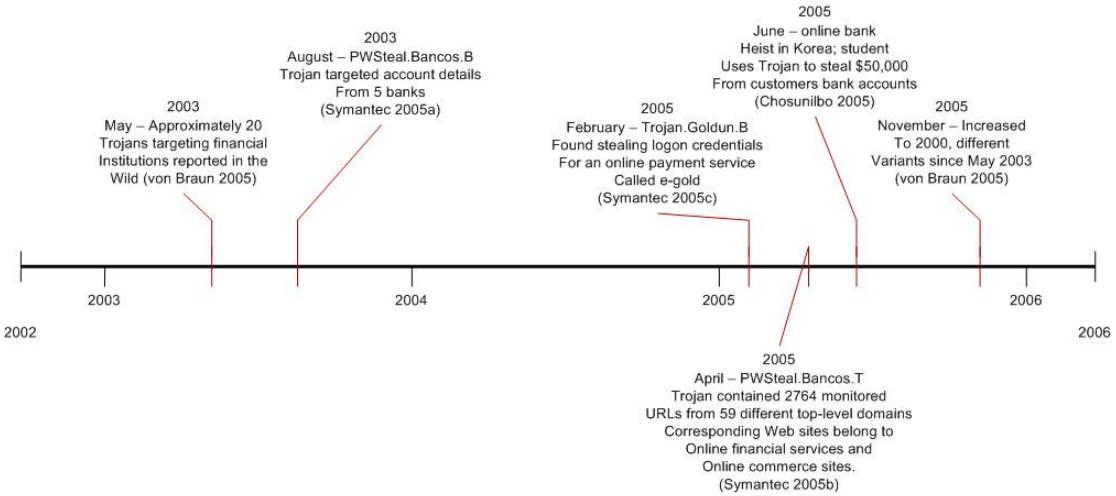

Figure 1. Timeline of real Trojan examples targeting financial services

The banking Trojan can use any one of a number of methods to determine where the user is surfing. These include (Ståhlberg 2007): Hooking e.g. inline hooks on WinInet API functions, BHO (Browser Helper Object) interface; Window title enumeration e.g. FindWindow(); DDE; Other COM (Component Object Model) /OLE (Object Linking and Embedding) interfaces; Firefox browser extensions and LSP (Layered Service Provider) interface. Once the Trojan has detected that the user is accessing a banking site, its attempts to capture the user's credentials or his authenticated banking session by spying on the data using one ore more of the following techniques: form grabbing, screenshots and video capture, keylogging, injection of fraudulent pages or form fields, pharming and Man-in-the-Middle attacks (Ståhlberg 2007). In any case, the result is that the Trojan has visibility to everything the customer does and can use his authenticated banking session to steal his money (Ståhlberg 2007). Figure 2 illustrates how the commonly perceived protection of the Secure Socket Layer (SSL) can be circumvented by a simple MitM attack.

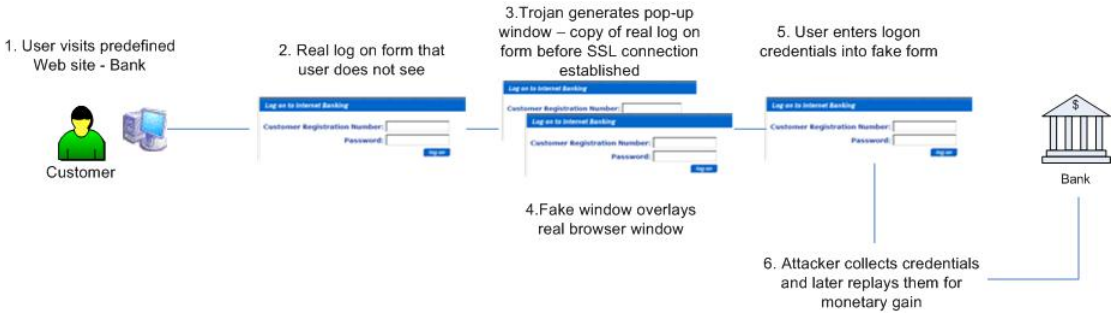

Figure 2. Circumventing SSL using a MitM attack

In nearly all security advice given by online services, traffic between the banking web site and the user are secured using SSL which is indicated by the yellow padlock symbol in the browser window (Wüest 2005). However SSL was designed to secure the channel from the user machine to the bank computer, and not the end points themselves. Whatever is done with the data before the start point and after the end point of the SSL channel is completely out of the SSL encryption context (Wüest 2005). Consequently the security issues are not related to the banks' systems per se, but the client machines that access the banks' systems via an online means. For that reason the banks have had to develop alternative strategies. These are presented in the next section.

GRAPHICAL AUTHENTICATION METHODS FOR ONLINE BANKING SYSTEMS

There is an ongoing arms race between online banks and Trojans (Ståhlberg 2007). Many banks have deployed new security mechanisms including virtual keyboards, mutual and multifactor authentication schemas, where the password page can be personalized with images or phrases (Wüest 2005) and secure one-time password hardware tokens; RSA and Vasco secure tokens are good examples of such devices (RSA Security 2008). The banks require these one time passwords not only to authenticate, but a different password for each transaction (Wüest 2005). The improved security provided is that the customer must authenticate their token details repeatedly, which makes any attack more difficult, but not impossible (Wüest 2005).

In order to highlight the need for further research in this area and to provide justification for the conceptual model presented in the next section, the following presents four examples of graphical authentication methods that are currently being used by internet banks for the protection of authentication (Wüest 2005).

Virtual keyboards: introduced to protect against key logger attacks, the customer is presented with a keyboard embedded into the internet banking user interface, and uses mouse clicks rather than keystrokes to enter their authentication credentials. The MitM attack defeats virtual keyboards. While a key logger is no longer capable of intercepting the secret information, multiple screenshots still can. Recording the position of all mouse clicks and one screenshot of the virtual keyboard contains enough information to recalculate the characters used (Wüest 2005). In addition many banking Trojans perform screen or video capture to bypass virtual keyboards (Ståhlberg 2007).

SMS challenge code: based on using the customer's registered mobile phone to receive an activation code via the Short Message Service (SMS). The customer identifies themselves to the bank with their account name and the bank responds with a random temporary password as a challenge which the customer enters into their browser to prove he or she has access to the correct mobile phone.

Image verification: based on using an image and verification phrase to form a shared secret between the bank and the customer. If the user has already been authenticated to the service, the customer identifies themselves to the bank with their username and a Device ID which is an encrypted cookie that is stored on the customer's machine. When the service determines that the username and a Device ID match the logon page is presented to the customer with the secret image and verification phrase embedded in it.

Dynamic security skins: this extends the image verification approach whereby a photographic image selected by a customer is transparently overlaid on web forms together with a visual hash that can be seen as a unique graphical pattern. Because of the link between the secure session and secure hash, which changes each session, it is infeasible for an attacker to spoof a pop-up that is a duplicate of the password prompt. The problem with this approach is that it does not provide reliable mutual authentication.

The problem still remains that a MitM attack can circumvent all of these protection approaches. Further alternatives have been suggested. Depending on the implementation a PKI based software solution may be used to protect against a MitM attack and a PKI based hardware token solution can be shown to protect against a MitM attack. However for today's Internet banking customer base there would be usability, key management and distribution issues making these solutions impractical for large scale deployment. Therefore an improved approach that builds on the protection methods discussed is required to counter the MitM attack. The approach needs to make the gained information unusable for the attacker and provide strong authentication of both parties to mutually authenticate each other (Wüest 2005).

CONCEPTUAL MODEL FOR AUTHENTICATION, DATA ENTRY AND TRANSACTION VERIFICATION

Authentication, data entry and data validation are the three primary processes involved in conducting a transaction in an online banking context. As outlined in the previous section internet banks employ a variety of methods to facilitate and secure each process. Each method provides different levels of security and each method requires a different level of complexity and infrastructure to implement and manage. Authenigraph was designed to provide a method to enable authentication, data entry and data validation that is simple to implement and manage and provides a level of security that meets the demands of industry. The following section outlines the research and development that has been undertaken, provides an overview of the how the system works and outlines how Authenigraph can be used to provide security against the variety of attacks currently known within the online transaction environment.

Overview of AuthentiGraph

Authentigraph was originally developed in 2003 to provide a simple secure authentication method for online environments (Pierce et al. 2003). Given the majority of online authentication consists of a login and a password, the primary goal was to develop new forms of authentication that overlay existing models whilst providing improved levels of security with low implementation and management costs. Since the initial prototype, the potential of the method has been identified and led to a number of studies that have examined the design and implementation options (Pierce et al. 2004b) and usability and user acceptance issues (Minne et al. 2007). Based on this research, Authentigraph has since been expanded to facilitate data entry, data validation and improved, flexible image generation options. It has been developed to provide a variety of simple, secure graphical data collection scenarios that can integrate into existing architectures.

Authentigraph is based around the concept that a user can identify characters and symbols from the screen by selecting them with a mouse. This is achieved by removing the need for keyboards to enter and collect data. ASCII codes are not used during the authentication, data entry and data verification process and therefore not transferred between the client and server (Pierce et al., 2004b). This eliminates the requirement for text based information being collected on the client and sent to the server.

Authenigraph has the potential to eliminate the possibility of spyware and Trojans relaying password strings to remote servers, reduce the capture and alteration of data and provide a secure method to present and validate data entered without the need to use external mechanisms such as SMS, that are outside the domain of the online environment.

To illustrate how Authentigraph works, the following outlines the use of Authentigraph as an authentication tool. When a client requests an authentication session with the server, an image is generated based on the associated process parameters. The image is a collection of characters randomly placed within the image. Each image for each authentication session that is generated will place the characters in different positions and if required, backgrounds, gradients, ghosting and other visual complexity is added to the image. The image presented in figure 3 consists of four character sets placed in four quadrants. Each character is positioned randomly within its designated quadrant each time an image is generated. Ghosting consists of placing characters randomly throughout the image and a gradient has been added to add complexity.

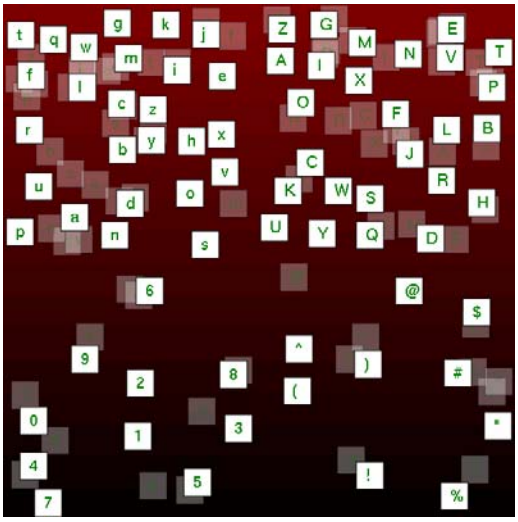

Figure 3. Four quadrant authentication image

The co-ordinates of each character rendered to the image is stored on the server in the form of a rectangle that defines the position on the image the character was drawn along with a unique session id to identify which image the client was sent.

The image is returned to the client and rendered to the browser according to the applications requirements. The user then selects the characters from the screen with the mouse. Each mouse click co-ordinates are stored on the client in the order they were collected.

When the user has completed selecting the information, the browser packages up the co-ordinates and returns them to the server. The server then processes each co-ordinate set received and attempts to map each point to a rectangle and subsequently retrieve the character that was originally drawn on the screen.

The server builds a character string in the order dictated by the co-ordinates returned and then either uses the character string to check authentication data, verify a predetermined string or uses the string as data entry for

any purpose. Regardless of the application, the information collected on the client, sent to the server and processed is complex, variable given the image is different each time and far more secure than text based collection and transmission.

The fundamental difference between this approach and many other approaches currently being used is the data transferred to the server is a collection of (x,y) co-ordinates that represent the mouse clicks collected on the client. The data has no correlation to any ASCII characters and subsequently very difficult to interpret in any meaningful way. The image delivered to the client is different for each authentication session. This means that each time the same information is selected from the image, the data co-ordinates that are collected and sent to the server will be different.

It is acknowledged that this method does not prevent the image and co-ordinates from being captured and co-ordinates mapped back to the image and the characters identified. To do this manually would be time consuming. To automate this process using a computer would require that OCR be performed on the image and rectangle co-ordinates collected. Using the ghosting would add complexity to this process as the OCR would need to identify which rectangles and associated characteristics contain the characters that the client was identifying and selecting. The co-ordinates would then need to be mapped to the rectangles to determine the character selected and a character string formed. This must be done for each image generated and each data entry session conducted as the images and associated data is different each time. Authentigraph has been developed to enable the method used to generate the image to be adjusted on the fly. This provides an added requirement for any automated process to adapt to the change making the OCR even more complex and time consuming.

Authentigraph is not the perfect solution. It is however, simple to use, implement and manage. It does provide added security compared to text based data collection, many scamble pads and image based data collection tools currently employed within in major financial institutions within Australia (Pierce et al. 2005a, Pierce et al. 2005b).

The following section presents the conceptual model for how Authentigraph can be used for authentication, data entry and transaction verification. An example of how the system can be used is presented and an explanation of the security features outlined.

Using Authentigraph for Authentication

Authentication generally requires a user to enter a particular character combination that is validated to determine if it exists and identifies the authenticated user. The data entry generally is conducted on the client and information sent to the server for validation and authentication. The method used, strength of the passwords, condition of the clients machine and many other factors affect the level of security that exists to protect the client from having their authentication information captured and used. Securing the client is a major problem as each user is responsible for the maintenance and up keep of their computer.

Authentigraph is designed to provide a method that reduces the requirement for the client to maintain a high standard of security by removing the elements from the authentication process that are easily corrupted. The authentication process takes advantage of a humans ability to recognise characters within a random image, collects obscure data and generally provides a more complex task for those attempting to capture and determine a users' authentication details. When a client requests an authentication session, an image is generated containing all characters that may be included in a login or password as part of an authentication process. These include lower case, upper case, numeric and special characters. The user is presented the image and selects the characters from the screen using the mouse. Figure 3 represents an authentication image that accepts all lower case, upper case, numeric and some special characters. The character sets are organised into quadrants and a gradient fill and ghosting applied to add complexity to the image. The character set used to generate the image can be adjusted to suit the application.

The mouse click co-ordinates are sent to the server where they are mapped back to the corresponding characters and a string of characters created. This string is then used to validate whether the user can be authenticated. As described previously, this method requires the capture of the image and co-ordinates for each authentication session in order to determine the authentication information, but compared to text based and structured image data collection, is more complex and therefore harder to crack.

Using Authentigraph for Data Entry

Data entry that is suited to the Authentigraph system is more likely to be applied to tasks requiring short character strings such as account numbers and credit card numbers. This is primarily due to usability issues associated with identifying and selecting characters randomly positioned within an image. Images can be generated containing a specific character collection the user uses to select and build a desired character string

such as an account number. An image is generated containing all the characters available and suitable for a data entry process. For instance, the user may be required to enter their account number consisting of upper case characters A-C and numbers 0-9. In this example an image is generated containing only one instance of characters A, B and C and numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 randomly positioned within the image or separated into two quadrants. Characters A, B and C in the left quadrant and characters 0 to 9 in the right quadrant. The user selects the characters from the screen using the mouse. The co-ordinates of each mouse click are returned to the server where they are mapped back to the corresponding characters. This process creates a string of characters in the order that they were selected. This string can then be used as required.

Figure 4 shows an example of a data entry image that consists of a character set of 'a' to 'z' with ghosting using the same character set and blue gradient fill. This image allows the user to select any character in any order from the character set available to form variable data input.

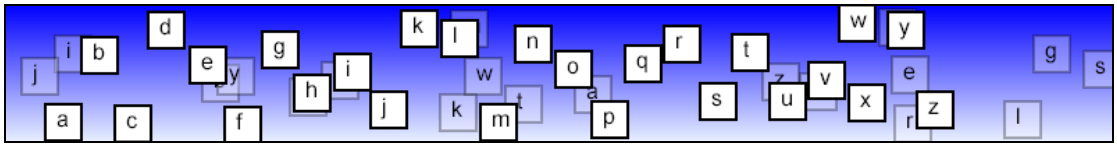

Figure 4. Data entry image.

This method prevents MitM attacks and the capture and replacement of information prior to the information being sent to the server within the period of the transaction. Any attempt to capture the image and co-ordinates would require an attacker to: map the co-ordinates to the image; determine the character string; and where replacement of a new number must occur, map the characters to the co-ordinates and replace the co-ordinates with new ones. This would be extremely difficult to execute within the timeframes that generally exist within a standard online transaction.

Using Authentigraph for Verification

Verification is the process of checking and confirming the information that is presented is the information required. This might be the verification of an account number entered to facilitate the transfer of money. Currently institutions are using SMS to send information to the client requesting they verify the information they entered is the correct information and not different information substituted.

Authentigraph enables verification of information by creating an image containing the characters ordered from left to right that is presented to the client for verification. This process requires an input string that is used to create the image. Each character of the input string is positioned starting from the left hand side of the image to the right hand side of the image. This ensures the verification string is logical and provides the user the ability to scan and identify the characters in a manner consistent with traditional keyboard data entry. The characters are organised in the y axis in random positions. The user selects the characters from the screen from left to right of the image by selecting the character on the image with the mouse. The co-ordinates of each mouse click are returned to the server where they are mapped back to the corresponding characters and a string of characters created. This string is then compared to the original string used to generate the image to verify the data.

Figure 5 shows an example of an image consisting of a verification string containing '01980471' positioned from left to right with ghosting using the same numbers and blue gradient fill.

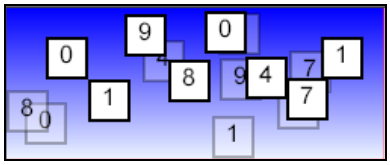

Figure 5. Verification numeric image.

Figure 6 shows an example of an image consisting of a verification string containing 'jasonwells' positioned from left to right with ghosting using the same characters and blue gradient fill.

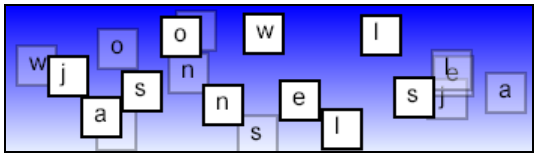

Figure 6. Verification alphabetic image.

Figure 7 shows an example of an image consisting of a verification string containing symbols derived from the Wingdings font with ghosting using the same characters and blue gradient fill. In this example the user is required to select the characters as they appear from left to right in consecutive order. There is no requirement for them to understand the content. This type of verification can be used to verify they are human as apposed to verifying a specific string.

Figure 7. Verification symbol image.

This method of verification prevents MitM attacks and the capture and replacement of information prior to the information being sent to the client for verification within the period of the transaction. Any attempt to capture the image and co-ordinates would require an attacker to: map the co-ordinates to the image; determine the character string; and where replacement of a new number must occur, map the characters to the new co-ordinates; replace the co-ordinates with new ones, and generate and replace the image. Again this would be extremely difficult to execute within the timeframes that generally exist within a standard online transaction.

Image Generator configuration

The key to Authentigraph is the generation of the image. To facilitate the creation of authentication, data entry and data verification images dynamically on the server, and to adjust the image properties on the fly, a configuration system has been developed. Each image that is generated is configured using a set of parameters stored within an XML file. An example is shown in figure 8.

```
<config>  // Encapsulates the configuration file.
  <gradient> // Encapsulates the gradient configuration options
      <enabled>true</enabled> // Indicates if the gradient has been enabled  (true) or disabled (false),
      priority over bgcolour
      <startcolor>Blue</startcolor> // Indicates the starting colour of the gradient
      <endcolor>AliceBlue</endcolor> // Indicates the ending colour of the    gradient
  </gradient>
  <screen> // Encapsulates the image size and background colour properties
      <bgcolor>Gray</bgcolor> // Sets the image background colour (ignored if         gradient is
      enabled)
      <width>400</width> // Sets the image width
      <height>300</height> // Sets the image height
  </screen>
  <box>// Encapsulates character box properties
      <bordercolor>black</bordercolor> // Sets the colour of the character box'         border
      <fillcolor>white</fillcolor> // Sets the colour of the background of the    character box.
      <bordersize>1</bordersize> // Sets the border size (width)
  </box>
  <watermark> // Encapsulates the watermark properties
      <enabled>true</enabled> // true: enables the watermark, false: disables         the watermark.
      <usecharset>false</usecharset> // true: uses provided character set as the background for the
                                          watermark, if no character set is provided then default
                                          character set used, false: uses provided input string as
                                          watermark background.
    <opacity>0.3</opacity> // Sets the level of transparency of the watermark
  </watermark>
  <font> // Encapsulates the font properties
      <type>Times</type> // Sets character font
      <size>20</size> // Sets font size
      <color>Red</color> // Sets font colour
    </font>
    <randomorder>false</randomorder> // false: output string in sequential    order, true: output string in
    random order
    <charset>abcd|ABCD|123456|$%&</charset> // User configurable character set to be displayed in the
          image. Quadrants separated by '|'
    <directory>data/</directory> // Stores the path to the folder that the images are stored in.
</config>
```

Figure 8. Example XML configuration file.

A configuration utility is provided to enable administrators to select and store the image configuration data. Adjusting the configuration manually provides the opportunity to adapt and shape the images generated according to their application. The ability to configure gradients and ghosting characters increases the complexity of the image reducing the ability of OCR but there are usability issues that must be considered. The configuration that is used will depend on the level of threat that is perceived and the level of user satisfaction that must be maintained as a consequence of using the system.

In some cases the components of the image are predetermined by the input and consequently the image output. Long verification strings may not fit into specific images sizes. Short strings with large fonts may not fit into specific images sizes.

Colours selected may clash reducing usability. It each case it is important to experiment and test the input and output to ensure it meets the systems needs.

The following describes the various configuration options available to generate images.

Font configuration

Authentigraph is designed to allow any font set stored and available on the server to be used to generate the individual character within the image. This enables unique, unseen and even symbol character sets to be created and used.

- **Font Type:** Font name.
- **Font Size:** Integer ranging from 8 to 50 for font size.
- **Font Colour:** Valid HTML colour.

Gradient configuration

A gradient can be drawn as a background on the image generated to add additional complexity to the image. More complex images make OCR more difficult.

- **Gradient Enabled:** When true is selected, the gradient background is enabled, conversely, false will disable the gradient background.
- **Gradient Start Colour:** Valid HTML colour.
- **Gradient End Colour:** Valid HTML colour.

Character Box configuration

Each character is drawn within a specified rectangle. Each rectangle may have border size and colour surrounding the rectangle and a specified rectangle background colour. Previous usability research found that the rectangle helps define the character and improves the ability of the user to identify and select characters (Minne et al. 2007).

- **Border Size: I**nteger ranging from 0 to 10 for font size.
- **Border Colour:** Valid HTML colour.
- **Box Background Colour:** Valid HTML colours.

Image configuration

An image can be any size and drawn with a specific background colour. It should be noted that the image size will affect the time required to create the image. Each rectangle and associated character must be placed randomly within the image. Reducing the image size reduces the available search and image space resulting in increased image generation times depending on the number of rectangles that have to be positioned. Usability research also determined that small complex images increases the time the user requires to identify and select a given character set (Minne et al. 2007).

- **Background Colour:** Valid HTML colour.
- **Image Width:** Integer ranging from 100 to 3000 pixels.
- **Image Height:** Integer ranging from 100 to 3000 pixels.

Water mark configuration

A watermark or ghosting image may be included in any image. The ghosting images are characters and associated rectangles positioned randomly on the image and may overlap other ghosting images. A character set can be defined to provide the ghosting data or the original character set of verification characters used. The

ghosting effect is determined by the opacity level and gradient fill used and is designed to provide more complexity to the image making it more difficult for OCR to applied to the image.

- **Watermark Enabled:** When true is selected, the watermark background is enabled, conversely, false will disable the watermark background.
- **Use Character Set:** When true is selected, the watermark will use the given character set. If no character set is provided a default character set will be used. When false is selected, the string input by the user will be used to produce the watermark.
- **Opacity Level:** Possible range of opacity levels range from 0.0 (total transparency) to 1.0 (total opacity).

Character layout configuration

Authentigraph provides the facilities to create images using 4 character sets. These include character a to z, A to Z, 0 to 9 and certain special characters such as !@#$%^*(). Character set may be organised and grouped into a maximum of 4 quadrants. A four quadrant character set of all available characters is defined as follows :

abcdefghijklmnopqrstuvwxwz|ABCDEFGHIJKLMNOPQRSTUVWXYZ|0123456789|!@#$%^*()

Each character set is separated by the '|' character. The order in which the character sets are defined will determine the quadrant they are assigned to. Where two characters are used only two quadrants are used and so on. Previous usability research found that dividing character sets into distinct quadrants improves the users' ability to indentify and select required character strings (Minne et al. 2007).

The image represented in figure 9 is used for authentication or data entry and is divided into 4 quadrants. Lower case characters are randomly placed in the upper left, upper case characters in the upper right, numbers in the lower left and special characters in the lower right.

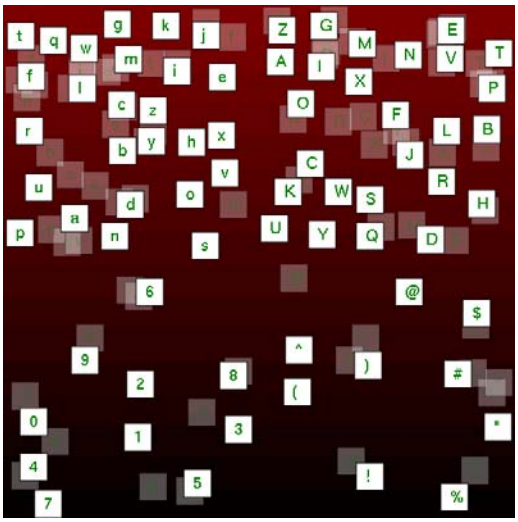

Figure 9. Four quadrant authentication image

- **Character Set 1:** Characters entered into the first quadrant. If only this character set is entered then there will be only 1 quadrant.
- **Character Set 2:** Characters entered into the second quadrant. If only this and character set 1 is entered then there will only be two quadrants.
- **Character Set 3:** Characters entered into the third quadrant.
- **Character Set 4:** Characters entered into the fourth quadrant.
- **Random:** When true is selected, the output string is presented in sequential order relating to the order the string is entered by the user, conversely, false will output the string entered by the user in a random order.

CONCLUSION AND FUTURE WORK

Currently a robust version of Authentigraph has been developed and implemented in a number of web based application used within Deakin University. Usability studies have enabled the refinement of the image generation format, size and layout and facilities have been developed to monitor the system. Data is being

collected for each authentication session to determine the user accuracy, authentication duration, image generation time, effect on the server during peak load and usability of the system when gradients and ghosting is added to the image. Future research will examine this data to determine where and how the system can be improved. A study is currently underway to position the Authentigraph method within the array of other security methods and an examination of the vulnerabilities that exist with the system undertaken to uncover any potential enhancement to further improve the system.

REFERENCES

Basu, A., and Muylle, S. (2003). Authentication in E-Commerce, *Communications of the ACM*, Vol. 46, No. 1, pg 159 – 166.

Blonder, G. (1996). Graphical Passwords, United States Patent 5559961.

Birget, J. C., Hong, D., and Memon, N. (2003). Robust Discretization: with an Application to Graphical Passwords, Cryptology ePrint Archive, Report 2003/168.

Chosunilbo. (2005). Breaching Online Banking Security Proves Easy as Pie, URL

http://english.chosun.com/w21data/html/news/200506/200506050007.html, Accessed 23 November 2007.

Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D. and Mitchell, J.C. (2004). Client-side defense against web-based identity theft, 11th Annual Network and Distributed System Security Symposium, URL http://crypto.stanford.edu/SpoofGuard/webspoof.pdf, Accessed 15 March 2006, Accessed 2 July 2006.

Dhamija, R. (2000). Hash Visualisation in User Authentication, Proceedings of the Computer Human Interaction 2000 Conference, The Hague, Netherlands.

Dhamija, R. and Perrig, A. (2000). Déjà vu: A User Study Using Images for Authentication, 9th USENIX Security Symposium.

Dhamija, R. and Tygar, J.D. (2005). Phish and HIPs: Human Interactive Proofs to Detect Phishing Attacks. In H. Baird and D. Lopresti (eds.), Human Interactive Proofs: Second InternationalWorkshop, pp. 127–141, Springer. URL http://www.cs.berkeley.edu/˜tygar/papers/Phishing/Phish_and_HIPs.pdf, Accessed 12 October 2006.

Furnell, S. M., Papadopoulos, I., and Dowland, P. (2004). A Long Term Trial of Alternative User Authentication Technologies, *Information Management and Computer Security*, 12, 2, 178-190.

Furnell, S.M., Charrau, D., and Dowland, P.S. (2005). PassImages: An Alternative Method of User Authentication, Proceedings of the ISOneWorld 2005, Las Vegas, USA.

Jakobsson, M. (2005). Modeling and Preventing Phishing Attacks. In Financial Cryptography and Data Security, vol. LNCS of 3570/2005, p. 89. Springer, 2005. URL http://www.informatics.indiana.edu/markus/papers/phishing_jakobsson.pdf, Accessed 12 August 2007.

Jackobsson, M. and Young, A. (2005). Distributed Phishing Attacks, Cryptology ePrint Archive, Report 2005/091, URL http://eprint.iacr.org/2005/091.pdf, Accessed 12 August 2007.

Jain, A., Hong, L., and Pankanti, S. (2000). Biometric Identification, *Communications of the ACM*, 43, 2, 90-98.

Jermyn, I., Mayer, A., Monrose, F., Reiter, M., and Rubin, A. (1999). The Design and Analysis of Graphical Passwords, the 8th USENIX Security Symposium.

Johnson, M. and Moore, S. A New Approach to E-Banking (2007). University of Cambridge.

Kambil, A., and van Heck, E. (1998). Reengineering the Dutch Flower Auctions: A Framework for Analysing Exchange Organisations, *Information Systems Research*, 9,1, 1 – 19.

Leyden, J. (2005). Trusted search software labels fraud site as ‘safe’. The Register, September 2005, URL http://www.theregister.co.uk/2005/09/27/untrusted_search/ Accessed 21 June 2007.

Minne, P., Wells, J.G., Hutchinson, D. and Pierce, J.D. (2007). An investigation into the usability of graphical authentication using AuthentiGraph, Proceedings of the 5th Australian Information Security Management Conference, Perth, Australia.

Parno, B., Kuo, C. and Perrig, A. (2006). Phoolproof Phishing Prevention. In G. Di Crescenzo and A. Rubin (eds.), Financial Cryptography and Data Security, vol. LNCS of 4107, pp. 1–19. Springer.

URL http://www.springerlink.com/content/2jj472626750x844/, Accessed 17 May 2007.

Perrig, A., and Song, D. (1999). Hash Visualisation: A New Technique to Improve Real World Security, Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce, City University of Hong Kong.

Pierce, J., Wells, J., Warren, M. and Mackay, D., (2003). A Conceptual Model for Graphical Authentication, 1st Australian Information Security Management Conference, Perth, Australia.

Pierce, J.D., Warren, M.J., Mackay, D.R., and Wells, J.G., (2004a). Graphical Authentication: Justifications and Objectives, Proceedings of the 2nd Information Security Management Conference, Fremantle, Australia.

Pierce, J.D., Warren, M.J., Mackay, D.R., and Wells, J.G., (2004b). Graphical Authentication: an Architectural Design Specification, Proceedings of the 2nd Australian Computer Network and Information Forensics Conference,, Fremantle, Australia.

Pierce, J., Mackay, D., Warren, M. and Wells, J., (2005a). A Review of Australian Internet Banking Client-Side Authentication Models, in Graeme Pye and Matthew Warren (eds), Proceedings of 6th Australian Information Warfare and Security Conference, School of Information Systems, Deakin University, Geelong, Australia

Pierce, J., Mackay, D., Warren, M. and Wells, J., (2005b). An Observational Survey of Australian Internet Banking Authentication and Client-Side Security: Does it Merit the Fees? Proceedings of 6th Australian Information Warfare and Security Conference, School of Information Systems, Deakin University, Geelong, Australia

Rash, W. (2002). Password chaos threatens e-commerce; ZDNet, URL http://articles.techrepublic.com.com/5100-10878_11-1039349.html, Accessed 5 April 2003.

Renaud, K., (2007). A process for supporting risk-aware web authentication mechanism choice, *Reliability Engineering and System Safety*, 92, 9, 1204–1217.

Renaud, K. and Smith, E. (2001). Jiminy: Helping Users to Remember Their Passwords. Annual Conference of the South African Institute of Computer Scientists and Information Technologists. SAICSIT'2001. Pretoria, South Africa.

Ross, B., Jackson, C., Miyake, N., Boneh, D. and Mitchell, J.C. (2005). Stronger Password Authentication Using Browser Extensions, Usenix Security Symposium.

RSA Security. (2008). RSA SecurID Authentication, http://www.rsa.com/node.aspx?id=1156, Accessed 19 September 2008.

Sandhu, R., and Samarati, P. (1996). Authentication, Access Control, and Audit, *ACM Computing Surveys*, 28, 1, 241 – 243.

Ståhlberg, M. (2007). The Trojan Money Spinner, Virus Bulletin Conference, URL, http://www.virusbtn.com/conference/vb2007/abstracts/Stahlberg.xml , Accessed 15 September 2008.

Stubblefield, A. (2004). Inkblot Authentication, Microsoft Research Technical Report MSR-TR-2004-85, URL http://www.research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=790, Accessed 18 September 2005.

Suo, X., Zhu, Y., and Owen, G.S. (2006). Graphical Passwords: A Survey, URL URL http://www.acsac.org/2005/papers/89.pdf , Accessed 8 January 2008.

Symantec. (2005a). PWSteal.Bancos.B,

URL http://www.symantec.com/security_response/writeup.jsp?docid=2003-073117-3108-99, Accessed 10 September 2008.

Symantec. (2005b). PWSteal.Bancos.T,

URL http://www.symantec.com/security_response/writeup.jsp?docid=2005-042522-3953-99, Accessed 10 September 2008.

Symantec. (2005c). Trojan.Goldun.B,

URL http://www.symantec.com/security_response/writeup.jsp?docid=2005-021612-4308-99, Accessed 10 September 2008.

Thorpe, J., and Nali, D. (2004). Analysing User Choice in Graphical Passwords, USENIX Security Symposium, URL http://www.scs.carleton.ca/research/tech_reports/2004/TR-04-01.pdf , Accessed 15 January 2005.

Varenhorst, C. (2004). Passdoodles: A Lightweight Authentication Method, Research Science Institute, URL http://people.csail.mit.edu/emax/papers/varenhorst.pdf, Accessed 15 January 2005.

von Braun, J. (2005). Internal study Symantec Sweden.

Warren, M.J., and Hutchinson, W. (2003). A Security Risk Management Approach for e-Commerce, *Information Management and Computer Security*, 11, 5, 238-242.

Waterken Inc. (2004). Waterken YURL Trust Management for Humans, URL

http://www.waterken.com/dev/YURL/Name/, Accessed 18 February 2005.

Wüest, C. (2005), Phishing In The Middle Of The Stream - Today's Threats To Online Banking, White Paper: Symantec Security Response, proceedings of the AVAR 2005 conference.

URL http://www.symantec.com/avcenter/reference/phishing.in.the.middle.of.the.stream.pdf, Accessed 29 March 2008.

Zishuang, Y. and Smith, S (2002). Trusted Paths for Browsers. In Proceedings of the 11th USENIX Security Symposium, IEEE Computer Society, URL http://www.usenix.org/events/sec02/full_papers/ye/ye.pdf, Accessed January 20 2003.

COPYRIGHT