Computer Communications 31 (2008) 4367–4375

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Virtual password using random linear functions for on-line services, ATM machines, and pervasive computing

Ming Lei [a], Yang Xiao [a,*], Susan V. Vrbsky [a], Chung-Chih Li [b]

[a] Department of Computer Science, The University of Alabama, Box 870290, Tuscaloosa, AL 35487, USA
[b] School of Information Technology, Illinois State University, Normal, IL 61790, USA

ARTICLE INFO

Article history:
Available online 16 May 2008

Keywords:
Security
Password
On-line services
User ID
ATM machines

ABSTRACT

People enjoy the convenience of on-line services, Automated Teller Machines (ATMs), and pervasive computing, but online environments, ATMs, and pervasive computing may bring many risks. In this paper, we discuss how to prevent users' passwords from being stolen by adversaries. We propose a virtual password concept involving a small amount of human computing to secure users' passwords in on-line environments, ATMs, and pervasive computing. We adopt user-determined randomized linear generation functions to secure users' passwords based on the fact that a server has more information than any adversary does. We analyze how the proposed schemes defend against phishing, key logger, and shoulder-surfing attacks. To the best of our knowledge, our virtual password mechanism is the first one which is able to defend against all three attacks together.

1. Introduction

Users with important accounts on the Internet face many kinds of attacks, e.g., a user ID and password can be stolen and misused. There are many reports on thefts on ATMs as well. The secure protocol SSL/TLS [1] for transmitting private data over the web is well known in academic research, but most current commercial websites still rely on the relatively weak protection mechanism of user authentications via a plaintext password and user ID. Meanwhile, even though a password can be transferred via a secure channel, this authentication approach is still vulnerable to attacks as follows.

- *Phishing:* Phishers attempt to fraudulently acquire sensitive information, such as passwords and credit card details, by masquerading as a trustworthy person or business in an electronic communication [2]. For example, a phisher can set up a fake website and then send some emails to potential victims to persuade them to access the fake website. This way, the phisher can easily get a clear-text of the victim's password. Phishing attacks have been proven to be very effective. Fig. 1 illustrates the total phishing reports received from September 2005 to September 2006 [2] according to the anti-phishing working group.
- *Password stealing Trojan:* This is a program that contains or installs malicious code. There are many such Trojan codes that have been found online today, so here we just briefly introduce two types of them. Key loggers capture keystrokes and store them somewhere in the machine, or send them back to the adversary. Once a key logger program is activated, it provides the adversary with any strings of texts that a person might enter online, consequently placing personal data and online account information at risk. Trojan Redirector was designed to redirect end-users network traffic to a location to where it was not intended [3]. This includes crime ware that changes host files and other Domain Name Service (DNS) specific information, crime ware browser-helper objects that redirect users to fraudulent sites, and crime ware that may install a network level driver or filter to redirect users to fraudulent locations.
- *Shoulder-surfing:* Shoulder-surfing is a well-known method of stealing other's passwords and other sensitive personal information by looking over victims' shoulders while they are sitting in front of terminals [10] or ATMs. This attack is most likely to occur in insecure and crowded public environments, such as an Internet Café, shopping mall, airport, etc. [14,18]. It is possible for an attacker to use a hidden camera to record all keyboard actions of a user for both a computer and an ATM machine. Video of the user's actions on a keyboard can be studied later to figure out a user's password and ID.

Many schemes, protocols, and software have been designed to prevent users from some specified attacks. However, to the best of our knowledge, so far, there is not a scheme which can defend against all three types of attacks listed above at the same time.

In this paper, we present a password protection scheme that involves a small amount of human computing in an Internet-based

* Corresponding author. Tel.: +1 205 348 4038; fax: +1 205 348 0219.
E-mail address: yangxiao@ieee.org (Y. Xiao).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.05.005

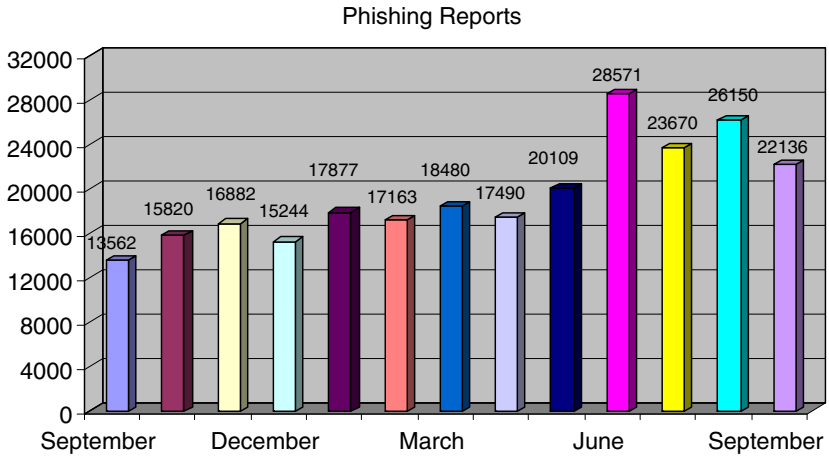

Fig. 1. Phishing report received September 2005–September 2006 [2].

environment, ATM, or pervasive computing, which will be resistant to a phishing scam, a Trojan horse, and shoulder-surfing attacks. We propose a virtual password concept that requires a small amount of human computing to secure users' passwords in on-line environments, ATM, and pervasive computing. We adopt user-determined randomized linear generation functions to secure users' passwords based on the fact that a server has more information than any adversary does. We analyze how the proposed scheme defends against phishing, Trojan horses, such as key loggers, and shoulder-surfing attacks. To the best of our knowledge, our virtual password mechanism is the first one which is able to defend against all three attacks together.

The idea of this paper is to add some complexity, through user computations performed by heart/hand or by computation devices, to prevent the three kinds of attacks. There is a tradeoff of how complex the computation by the users can be. One goal is to find an easy to compute but secure scheme for computing.

We believe that for some sensitive accounts such as on-line bank accounts, on-line credit card accounts, and ATMs, users are likely to choose a little additional complexity requiring some degree of human computing in order to make the account more secure.

The rest of the paper is organized as follows. In Section 2, we propose the virtual password scheme. We propose randomized linear generation functions in Section 3. In Section 4, we describe implementation issues of our scheme. We describe related work about password protection in Section 5. Finally, we conclude our paper and describe our future work in Section 6.

2. Virtual password

2.1. Virtual password concept

To authenticate a user, a system (S) needs to verify a user (U) via the user's password (P) which the user provides. In this procedure, S authenticates U by using U and P, which is denoted as: $S \rightarrow U$: U, P. All of S, U, and P are fixed. It is very reasonable that a password should be constant for the purpose of easily remembering it. However, the price of easy to remember is that the password can be stolen by others and then used to access the victim's account. At the same time, we cannot put P in a randomly variant form, which will make it impossible for a user to remember the password. To confront such a challenge, we propose a scheme using a new concept of virtual password.

A *virtual password* is a password which cannot be applied directly but instead generates a *dynamic password* which is submitted to the server for authentication. A virtual password P is composed of two parts, a fixed alphanumeric F and a function B from the domain ψ to ψ, where the ψ is the letter space which can be used as passwords. We have $P = (F,B)$ and $B(F,R) = P_d$, where R is a random number provided by the server (called the random salt and prompted in the login screen by the server) and P_d is a dynamic password used for authentication. Since we call $P = (F,B)$ a virtual password, we call B a *virtual function*. The user input includes (ID, P_d), where ID is user ID. On the server side, the server can also calculate P_d in the same way to compare it with the submitted password.

It is easy for the server to verify the user, if B is a bijective function. If B is not a bijective function, it is also possible to allow the server to verify the user if it is at least injective as follows. The server can first find the user's record from the database based on the user' ID, and compute P_d, and compare it with the one provided by the user. A bijective function makes it easier for the system to use the reverse function to deduce F's virtual password. The user should be free to pick the fixed part of the virtual password.

2.2. Virtual password usage

We have introduced the concept of the virtual password, and next, we detail how to apply it in an internet-based environment.

To use a virtual password, human computing is involved or a handhold device which can be programmed to compute the dynamic password is needed. We could develop a smart application to make the complex calculation for the user, which can run at the mobile device, such as a cellular phone, PDA, personal computer, or programmable calculator, to relieve the user from the complicated calculations and to overcome any short-term memory problem. If such a helper-application is involved, we should make sure that the helper-application itself should be unique to each user account and only work for the corresponding user account.

Regardless of the approach chosen, a user's registration in the system is similar, i.e., the user submits a user ID and a fixed password. The one difference from a traditional approach is that in the virtual password scheme, there is a virtual function, which is a must, to be set during the registration phase.

The server then delivers this function information to the user via some channels, such as, displaying it on the screen or email. The user needs to remember this function together with the password they have chosen or save them in disks or emails.

We also note that a small amount of human-computing is involved in the authentication process. We have to choose B to make the calculation as simple as possible if the helper-application is not