

# IMPROVING PASSORD SECURITY ON COMPUTER SYSTEMS

*Source:* <http://www.derkeiler.com/Newsgroups/comp.security.misc/2003-09/0369.html>

---

*From:* Miguel ([mivemu\\_at\\_yahoo.com.br](mailto:mivemu_at_yahoo.com.br))

*Date:* 09/16/03

Date: 16 Sep 2003 04:55:50 -0700

See original complete html text at <http://www.meucat.com/passi.html>

## IMPROVING PASSORD SECURITY ON COMPUTER SYSTEMS

With Internet worldwide use, computer systems, previously constrained to some hundred users, became almost immediately accessible to million people. Until some years ago, trying to break bank account codes required counterfeiter to be close ATM machines, at least, close to some computer terminal linked to bank systems. But now, hundred million people have the power of playing with accounts and passwords, while comfortably living in their homes all around the world.

The quantity of accounts to be checked is so high, that modern hackers use a very more profitable strategy than guessing passwords: they choose an unique and easily recorded password, to say '1234', and checkthis password against lot of accounts simultaneously, because they are sure that in few time, some of these accounts will match that password.

Even for more traditional systems, which doesn't operate plugged to Internet, traditional passwords are becoming vulnerable to modern technology. Some weeks ago, the Revenue Federal Taxes in Brazil discovered a way to cute debts for big companies, executed by dishonest Brazilian federal employees directly into computers. These employees stole passwords from authorized others, using a little plug in external ports of micro computers.

Many financial companies waste lot of resources to protect passwords on Internet, using sophisticated tools, and replacing keyboards by mice (virtual keyboard) to type passwords. Even so, they insist in the old 'fixed password' scheme to protect users, whenever they could use these resources in a more practical way. Many banks are using double checking (two different passwords), and showing random conversion tables to type characters, trying to hide real letters and numbers of passwords. These checking improve security on transactions, but also leads to more difficult performance for customers.

Even so, the current method for protecting accounts is very vulnerable. Suppose a person going to a public place, like a cyber house, to perform a bank transaction. His/her bank had lot of work to replace keyboard by mouse, to give him random tables and another tools, trying to hide private info from avid eyes. But we should recognize technology also plays on hackers flavor, and so, it is not impossible a tiny TV camera, almost imperceptible, recording all user movements and screens on his back. With these tools, bad boys can register what 'virtual keyboards' send to computer, what are returned screens, what user is pressing on keyboard (aside name, Soc. Sec. ID, addresses and so).

In this work, it is presented a more easy and secure password method than traditional one. Instead using 'fixed characters' passwords (like car licenses or girlfriend names), it is better and cheaper to use 'fixed places' passwords with variable characters, which will null all previous knowledge of password values. The present method is so secure, than user can even show his/her password to other people, being secure that other person won't get access to his account.

Imagine Alice getting money from ATM machine. After entering account number, the machine would show a little board (like this one) to enter password:

(image)

Alice knows 'her password' is formed by first upper position, followed by diagonal one, turning down left and right, as shown by letters a,b,c,d in that order.

(image)

Alice will type '1332' as password on this transaction, machine will match it against 'abcd' places on board, and then it will validate transaction.

Square board is filled randomically on every transaction, to get different characters, since the important thing here is location of characters instead characters by itself. Probably every user will have a different path to build the password on checkerboard. For instance, chess players could choose horse movements, while another ones would use clock movements. In this way, now system admin and bank controllers would ask customers to remember 'a given path into the board' instead remembering a given fixed password.

Since it is impossible for eavesdropper to know which place user is looking at when choosing digits, he will never know what the location of this number in checkerboard is. Even for sophisticated TV cameras focusing directly on user eyes, this task is almost impossible to perform. Probably computer will take care to show repeated digits on

board, to avoid someone identify what the location for given number is. In previous example, number '1' can become from 3 places, '2' from four places and '3' from two different places, totaling  $3 \times 2 \times 2 \times 4 = 48$  different combinations (or paths) able to build the '1332' password.

On next transaction, after entering account number, ATM would show this board to Alice:

(image)

so password for this transact now is '2123', which is different from previous one. Note that even if a careful eavesdropper could register that password, it wouldn't have any value for next transaction. In the Brazilian case of taxes frauds, the collecting hardware plugged on external computer ports would be completely useless.

Of course a hacker getting several samples of passwords and checkerboards from the same user, could deduce the correct path to build the passwords for all transactions. But the task of collecting several passwords is not so easy, because probably user and hacker will meet just once by mere chance at ATM machine or public terminals, and also because passwords continue to appear like jokers '\*' on display machines.

It is not difficult to prove that given a checkerboard with P places, the best quantity of characters that should be displayed on board is  $\sqrt{P}$ . If checkerboard were a square with  $P = N \times N$  ( $N = \text{size}$ ), then it is better to display N different characters on checkerboard.

To get this result, we assume there is a trade off between quantity of characters displayed on the board, and quantity of possible path needed to form a given password. If password have size T (it is formed by T characters string), and S different characters can be used to buildit, then the total arrangement is  $S^T$ . For instance, with 26 letters ( $S=26$ ) and four letters passwords ( $T=4$ ), it can be formed  $26^4$  different combinations, so chance to guess this password is  $1/456.976 = 0.0000021$ .

On the other hand, if there are too many characters displayed on the board (to try maximize combinations), it will be easy to rebuild the path for that password. Again suppose in board there is just a different character on every place, then there is an unique path to build password, and while guessing password will be a difficult task, once known a password, it is easy to discover the generator path, risking all security building.

Given S different symbols, each one with quantity  $Q_1, Q_2, \dots, Q_s$  on the board, the total of different paths to build a password is given by  $Q(s_1) \times Q(s_2) \dots \times Q(s_t)$ . If board has 2 characters '1' and 3 characters '2', and if the hacker discovers password is '1122' ( $T=4$ ),

then possible paths to build this password is  $2 \times 2 \times 3 \times 3 = 36$ .

On opposite side, if the board has just a single character filling all places (to say '3'), the quantity of possible paths to build password is the maximum, but then to guess the password is very easy, since there is only one possible password '3333'. This can be expressed as 'when more paths to build passwords, less passwords can be built', and this is our trade off.

We want to make the  $S^T$  value (quantity of password) very high, and also want to make quantity of paths very high (highest possible). A way to solve this, is by placing the same quantity of characters on the board, so  $Q_1 = Q_2 = \dots = Q_s$ , so quantity of possible paths become  $Q_1 \times \dots \times Q_t = Q^T$ .

If there are  $S$  different symbols in checkerboard with  $P$  places, then each symbol will appear  $P/S$  times, so  $Q = P/S$  and equation leads to get  $(P/S)^T = S^T = \text{maximum}$ . That means  $P/S = S$  or  $S = \text{sqr}(P) = N$  (for square boards).

For the  $3 \times 3$  square board displayed above, the more sure arrangement are passwords formed with 3 different characters, and boards displaying 3 different characters, every one 3 times. These values will difficult the hardest way for any spy trying guess passwords or paths.

The  $3 \times 3$  board displayed here can be modified in many ways, and systems can evolve from cool boards to more friendly ones with pictures of faces, landscapes and other squared images, to help users choose characters into. A  $20 \times 20$  board can be hard to manage, unless you paint some beautiful face on background, letting user remember that his characters are 'around right eye, left ear and nose'. This way, the chances for eavesdropper to guess correct passwords and paths would evaporate. On  $20 \times 20$  board, the quantity of passwords and paths should be astronomic.

(image)

Using image and color resources from modern operational systems, it is possible to extend passwords security to near perfection. One weak point in the system we are explaining, lies on fact that the same information typed on keyboard, is also present on the checkerboard. This way, if a hacker get at least two passwords and two checkerboards, will have some chance of break the system. Although this is a great improvement over traditional passwords, where just one discovered code will crash security, we can try correct these fails by erasing the info on checkerboards.

Without this information, a hacker won't have any chance of entering the system, even after stolen hundred of passwords and related chessboards. The tiny TV camera recording at back all that users type

and watch on screens, will be useless.

To get such a security level, the user will need recall two things: a) the path to build password on the board and b) a little associative table between images and characters (or numbers). On our first example, suppose Alice, our bank customer, is asked to remember the path on board, and also the following relation: white = zero, black = one, red = two and blue = three.

When Alice makes a transaction, ATM machine would display the following board to her:

(image)

Then, recalling previous relation, she would type '3321' for password. Note that even some hacker getting both the password and the board, he won't know where these numbers came from, because the relation 1 = black, 2 = red and 3 = blue is just into Alice brain and computer system.

There are many such relations that could be used, for instance Alice could remember associative letters from animal names. Bank could ask Alice to record that 'zebra' = zero, 'orc' = one, 'tweety' = two and 'tiger' = three. In this case, Alice would see this board:

(image)

and would type '1122' to validate transaction. Computer could display noise pictures on some places, like birds or else, fooling any possible hacker trying to deduce what the relation between animals and numbers. It is possible the user choose the images he want to put on every place of checkerboard (his car, girlfriend, mother) and make mental associations between these images and other characters. Since these images has no meaning for other person, nobody can be able to guess the way this user chose his passwords.

Routines for 'typing from mouse instead keyboard' are already developed from almost all big companies involved with money, and so typing 'by place' instead typing 'by character' is just a matter of habit for most users. A single change of habits can save lot of money, time and work for all of us, but it appears old companies love old usage, and they pay a high price for this.

Curitiba – Brazil – 14 Sep 2003

You can see original complete html text at <http://www.meucat.com/passi.html>