A $100,000 keying error

Kai A. Olsen, Molde University College and University of Bergen

$100,000 is nothing. We have all heard of stockbrokers that lost millions by hitting the wrong key. This case is different, however. An ordinary bank customer, Grete Fossbakk, used Internet banking to transfer a large amount to her daughter. She keyed a digit too many in the account number, and the money went to an unknown person. This person managed to gamble away a large part of this sum before the police managed to confiscate the remainder. The case has received a lot of media coverage in Norway. The Minister of Finance has criticized the user interface, and has requested new and improved regulations for Internet banking. All of a sudden, the risk of Internet banking has become apparent to both government and ordinary citizens.

Who is to blame? Clearly, the user has made a slip. She had also the chance to correct the typo before she hit the "confirm" button. However, as we shall see, the system had also every opportunity to catch her mistake. This did not happen. A simple check to ensure correct input was missing.

The case raises important questions. Should we not expect a minimum of validation procedures from a banking system developed for ordinary users? As system designers, is it not our responsibility to aid users in avoiding errors? Today the user operates alone in front of the computer, after intermediates and colleagues have been replaced by computer systems. Then it is important to have interfaces that can offer as good as, or even better error detection than in the previous manual systems.

THE FOSSBAKK CASE

The Fossbakk case provides an interesting example. The Internet system she employed when making her fatal mistake was common to a large group of Norwegian banks. An inspection of this case will give us insight into the types of typos made by users, the psychology behind "confirmation", and the pitfalls of many Web systems.

Her daughter's account number was 71581555022, but she keyed 715815555022, i.e. inserting an extra 5. The user interface accepted only eleven digits in this field (the standard length of a Norwegian account number). Thus, the number became 71581555502. The last digit is a checksum based on a modulo-11 formula. This will detect all single keying errors and errors where two consecutive digits are interchanged. By inserting an extra 5 both the ninth and tenth digit were changed. On the average the checksum control will only catch 93% in the cases when more errors are made. For Fossbakk, the final eleven-digit number was a legal account number. However, only a small fraction of all legal account numbers is in use. Further, the chance of hitting the account number for a dishonest person without income or assets is also very low in a homogeneous country as Norway. Our user was thus extremely unlucky. The person that received the $100,000 has received a prison sentence, but this does not help Fossbakk getting the money back.

To court

Fossbakk took the case to the Norwegian Complaints Board for Consumers in Banking. This board deals with disputes between consumers and banks. The board has two representatives for the consumers and two from the banks, with a law professor as chair. In a three-to-two vote, she lost. The chair voted for the bank. His argument was that “she made an error and has to take responsibility”. He also regretted that Norwegian regulations set no limit for consumer’s loss in these cases, as there would have been if Fossbakk had lost her debit card.

Fossbakk is now taking the case to court, backed by the Norwegian Consumer Council. Her argument is that she typed twelve digits, and that the bank system should have given an error message in this case, instead of ignoring all typed digits after the first eleven. She has acknowledged that she would have no case if only eleven digits had been typed. The bank argues that she cannot prove by any measure of probability that she keyed twelve digits. They further state that there cannot be different rules of responsibility depending on the number of digits given. Finally, they stress the fact that she confirmed the $100,000 transaction.

At this point, I was called in as an expert witness for Fossbakk. In my opinion, and I should expect that of most other computing professionals, a system should give an error message when the customer types a too long number. Clearly, such a test can be inserted with a minimum of effort. In fact, The Financial Supervisory Authority of Norway has required all banks to implement this functionality based on the Fossbakk case. It is likely that we can argue that the bank showed negligence when developing the user interface in question. However, can we prove, beyond doubt, that Fossbakk keyed twelve digits? Since any digits beyond eleven were stripped off in the HTML form, there exists no log information that can tell us what happened.

BANK SIMULATOR

The answer is not to be found in the literature. It seems that researchers lost interest in keying errors when the keypunches disappeared. There is some usability data on cell phone keyboards but that is hardly relevant here. We therefore decided to get our own. This was done by implementing an “Internet bank simulator”, a simple interface that worked similarly to the system that Fossbakk used. It consisted of two forms. In the first form all data, date, customer identification number or a message, amount and account number were entered. After hitting the “pay” button on this form the data were presented in a new form for confirmation, allowing the user to “confirm” or “edit”. Students from college and high schools, 69 test persons altogether, were engaged to enter thirty transactions each from a predetermined test set. This gave data on 1778 transactions after some outliers had been removed.

Results

The students got 124 account numbers wrong, 7% of the transactions. This error rate is higher than what we would expect in a real system. Firstly, since our task is to analyze faults the simulator does not offer any error messages. However, the user will have to confirm the transaction, just as in the real system, and any errors corrected before

confirmation are not included in the count. Secondly, the test persons enter a large set of transactions. In some cases, the account number for the preceding or following transaction has been used instead. This could happen in real life, but will be much more frequent here since the transactions are entered from a list. Thirdly, the test situation does not involve any real money. We should expect that users would verify transactions more carefully when using a real system.

While the overall error rate may be higher there does not seem to be any reasons why the distribution of different types of errors should be any different from what one would find in a real system (an exception being the case when an account number is replaced by another from the data set).

In 29% of the cases with a wrong account number, the number was too long. In half of the cases where this happened, the students made the same error as Fossbakk, inserting an extra digit in a sequence of two or more identical digits. The strategy of the bank interface, of skipping digits beyond eleven, would have given the correct number in 64% of the cases (note that the user will not see digits beyond eleven, and errors here will not be caught by the users own validation). Of the remaining "abbreviated" numbers the modulo 11 test was able to capture all but three. That is, of the nearly 1800 transactions, three transactions (0.2%) would have passed the error detection routines of the banking interface. Multiply this with the near 200 million Internet transactions that are performed each year in Norway, and we see that this small percentage hides a massive problem.

In an improved interface, with a "too long" check along with the modulo 11, all errors made in our test would have been captured (except in cases where an account number from another transaction in the set were entered). Analysis of customer identification numbers, also a part of the transaction, showed the same result. It is a normal mistake to add an extra digit in a sequence. Similarly, it is easy to miss a digit in a sequence. In fact, these errors were the most common found in this test. If we ignore the error of typing another account number from the set, the "too long" occur in 41% of the error cases, "too short" in 35% and wrong eleven-digit number in 24%. Also in this respect, it seems nonchalant not having code to detect a too long number. Since none of the persons that entered an extra digit or missed a digit managed to end up with an eleven-digit number, i.e., by making yet another error, we can therefore state, with a high probability, that Fossbakk has entered a twelve-digit account number.

Confirmation

Then we are left with the argument that she confirmed a $100,000 transfer to the wrong account. So did also the students in 124 cases. In addition, for every tenth transaction the simulator replaced the typed number by a similar looking number before confirmation. For example, the number 70581555022 was replaced by 70581555502. This was done in 178 cases. In only five cases, 2.7%, did the users recognize the error and correct the number.

It seems that most people perform the inspection while keying, not when the whole number is displayed on the screen. In many ways, this is efficient. While keying we can concentrate on one digit at a time, after keying we have a large number. If this *seems* to be correct, we hit the "confirm" button. The psychologist Donald A. Norman explains

this behavior in his book, *Psychology of Everyday Things*. Here a user has confirmed deletion of his “most important work”. According to Norman, the user confirms the action, not the file name. Thus, the “confirm” part of the transaction, while having some legal implications, has minimal effect towards detecting errors.

ACCOUNTABILITY

Like many other new IT applications, Internet banking is effective. As users, we enjoy reduced costs and 24*7 availability. However, real money is transferred based on instructions from humans that may be inexperienced or perhaps just making a slip. Then it is up to the system. It must be a requirement that this intercept as many errors as possible. If Fossbakk had used the manual system instead, e.g., by writing a letter to her bank requesting the transaction, no sane bank employee would have removed the twelfth digit of the account number, hoping that this would correct the error.

We should expect more. The banking system could offer the name of the account owner as confirmation when an account number is entered. In cases where this comes in conflict with privacy issues, forename or an alias could be used. The system could give a warning whenever a previous pattern is violated. For example, if we pay a utility bill of $100 to $300 every month, we could get a warning if the amount was way off. Further, e-invoices and other automatic procedures can limit the number of transactions that have to be keyed in, thus reducing the overall error rate.

In this case, it was a banking system. In the next case, it may be a weapon system or a medical information system. We have already examples from these areas where misinterpretations between systems and users have had serious consequences. For all systems, we have a responsibility as computer professionals to protect the users from their own errors, to intercept all detectable errors, and to give informative warnings when we have reason to believe that the user may have made an error. The “she made an error and has to take responsibility” is too simple. What we need are systems that work in collaboration with the user in such a way that the overall error rate is reduced to a minimum. Yes, we need responsible users but a good system can handle most of the slips and typos they make, as illustrated in this case.