

Anti-keylogging measures for secure Internet login: an example of the law of unintended consequences

by

Stuart P. Goring¹, Joseph R. Rabaiotti² and
Antonia J. Jones²

¹Rhose, CF62 3EY, Wales

²Cardiff School of Computer Science
Cardiff University

5 The Parade, Cardiff CF24 3AA, Wales UK

First draft: 27 June 2006

Revised: February 2007

Submitted to: Computers & Security

Anti-keylogging measures for secure Internet login: an example of the law of unintended consequences

Stuart P. Goring, Joseph R. Rabaiotti and Antonia J. Jones

17 February 2007

Abstract

Traditional authentication systems used to protect access to online services (such as passwords) are vulnerable to compromise via the introduction of a keystroke logger to the service user's computer. This has become a particular problem now that many malicious programs have keystroke logging capabilities. When banks first introduced online banking services they realised this, and added features to protect users against keystroke logging. In this paper we show, using a real online banking system as an example, that if these features are incorrectly implemented they can potentially allow an attacker to bypass them completely and gain access to a user's bank account. The vulnerability was initially noticed in a particular Online Banking Service, but any system implemented in the way we describe is equally vulnerable.

1 Disclaimer

No illegal access took place during our research. It is generally assumed that to be in a position to *prove* that a gatekeeper system has a weakness one must have broken the law¹. However, in this paper we demonstrate that this is not the case. In this case we show that by perfectly proper use of a system, which has implemented login in a particular way, and by *intelligent observation* one can logically *prove* a weakness without even passing the gatekeeper or entering the system [1]. Whilst we can do this because of a rather trivial and easily fixed design flaw, it seems to us that an interesting point of principle has been established. It should be noted that the bank in question deny that our observations constitute a security risk to their Online Banking service (see Section 3).

2 Introduction

Karl Popper once claimed that “the main task of the theoretical human sciences . . . consists in identifying the non-intentional social repercussions of intentional human actions” [13], p. 1.

Companies or organizations such as banks which offer internet-based access to services have had to develop a number of techniques to prevent compromise of user authentication codes, which are substantially more sensitive than the average website access password. One popular technique which helps prevent access codes being recorded by keystroke

¹This has the effect of discouraging people who discover such flaws from reporting them because they fear (often correctly) legal reprisals. Popper might have called the topic of this paper “an unintended consequence in a closed society”.

loggers is to ask the user for a number of randomly-chosen characters (usually 3) from the authentication code(s). Whilst someone “watching”, either physically or via a hardware or software-based keystroke logger, would obtain the account ID (and extra information such as the user’s birthday), only a few characters from the authentication code would be harvested and their positions not be known. A keystroke logger would still eventually capture all pieces of the code, but without the positional information the attacker has no data to enable these pieces to be put together within the usual three login attempts. So far so good. However, sometimes, what may seem to be a useful additional security measure can have unforeseen consequences. Here we show how an apparently minor detail of implementation can, if flawed, effectively entirely negate the anti-keylogging measures; thus providing an interesting example of the “law of unintended consequences”.

The vulnerability we describe allows an attacker to know beforehand which character positions the victim will be asked for on his next login attempt. When the victim next logs in, the attacker can view the characters which correspond to these positions with the keystroke logger, and repeat the process to get more characters from the authentication code. In general at each such move *either* the attacker gains entry *or* she learns at least one new digit (and its position). So it’s a win-win situation for the attacker in terms of information gain. In this particular case the maximum authentication code length is 10 digits. After the first move three digits (and their positions) are known, this inevitably means that in at most *eight* more moves the attacker is *guaranteed* to gain access. However, probabilistically the attacker will gain entry very much faster than this.

Keystroke loggers are not hard to obtain. Hardware loggers which are designed to look like various kinds of keyboard adapters may be purchased online relatively cheaply. These require physical access to the computer at least once, but they can be used on any computer, need no software, and are almost undetectable (other than by physical examination). Some of the more advanced types allow their data to be accessed remotely via a radio link and so require only a *single* physical penetration.

Software loggers, on the other hand, are available free (or can be written by a relatively competent programmer), in principle can be remotely queried over the Internet, and in fact form a component of many already-existing malicious programs (trojans, worms, spyware, etc.).

3 The login procedure

We will use the Online Banking system as an example. This system requires a customer to select an authentication code which must consist of between six and ten numeric characters (0-9). On logging in to the service the customer must enter a banking ID number and their date of birth. These must be entered in full every time the user logs on. The customer is then prompted to enter three characters whose positions are chosen randomly from this code. It should be noted that the bank in question does not accept that our observation implies a security risk, see [3].

4 The Vulnerability

Assume the user makes a mistake in entering the three authentication code characters. When this happens, he is presented with a failure message and must re-start the login process. However, at this point the system designer has a choice: the system can either

ask for the *same* set of three digits, or ask for a *new* randomly generated set².

- The situation we analyze here is when, under these circumstances, the user is asked for *exactly the same characters* he was asked for previously - in other words, if he is asked for the first, fifth and sixth characters and gets it wrong, he will be asked for the first, fifth and sixth characters when he tries again.

For such an implementation the requested character positions only change after a successful login, and this remains the case even if the next attempt is made from a different computer at a different time.

It is thus possible for an attacker, by making a login attempt, to gain advance knowledge of which digits the user will be asked for the next time they log in. This allows the attacker to reconstruct the authentication code from the keystroke logger transcripts. The ID and date of birth will be captured by the keystroke logger the first time the victim logs in. The attacker must know these to proceed, and they also provide a convenient search string to allow her to locate the pertinent part of the logger's captured data.

Obviously, if the logger is present on the victim's machine for long enough it is certain to capture all the digits in the authentication code. However, the attacker has no access to positional data - she will have sets of 3 digits, but no idea of their sequence in the authentication code, or even of their uniqueness (the same digit may occupy more than one position in the code). Even if an attack is feasible based on this idea, it is likely to be vastly slower and less certain of success than the attack we describe.

It has been argued that this stratagem, of not requesting a different set of characters on each retry, acts as a defense against phishing sites (in this case a website that masquerades as the legitimate site and attempts to garner users login details, i.e. website forgery). It appears to us that this argument is flawed. It is only a defense if the *user* is aware of the fact that if they mistype their authentication code the genuine site *will not* request a different set of characters. We submit that many users would not draw such a distinction in practice and act on it, even if they had once been told of it.

There are other technical measures against phishing websites (see [12]). Moreover, granted sensible precautions (e.g. such as registering similar domain names so they cannot be used for phishing), in the final analysis, assuming customers have been warned, it does not seem (to us) reasonable to hold a company responsible because a customer has entered their authentication details on some other website. In any event this seems to be an increasing problem. Reports suggest that in the United Kingdom losses from web banking fraud, mostly from phishing, almost doubled to £23.2m in 2005, from £12.2m in 2004 [5], while 1 in 20 users claimed to have lost out to phishing in 2005 [6]. So phishing is undoubtedly a serious issue.

5 Attack Methodology

The attacker must be able to introduce a keystroke logging device to the victim's computer. For hardware loggers, an initial physical access is required - though software loggers could be injected using a virus or various other non-physical methods. Methods based on "social engineering" are often very successful - see [2] for a particularly novel method.

The attacker first waits until the keystroke logger has captured the ID number and date of birth (plus three authentication code digits whose position is not known which although

²Or, indeed, do something else entirely different.

they do provide useful information in this context we shall ignore). Then she makes a login attempt, noting the positions of the three requested digits. This attempt will in all probability fail (we ignore the extra ‘negative’ information gained here), but the attacker now *knows* that the victim will be entering his next three digits at those specific positions next time he logs on.

When the attacker sees from the logger transcript that the victim has logged on for the second time, she notes down the digits, pairing them with the position data she already has. She now makes a second login attempt, again noting the digit-positions requested. This cycle is repeated, either until the attacker is asked for a set of digit-positions she already has (and can thus access the account), or until the attacker has the entire authentication code. Of course, if the code is a recognizable number, such as a date, then the attacker may be able to deduce it without needing to recover all the digits.

The sequence of events is then as illustrated in the flow chart of Figure 1. Note that the process does not have to ‘END’ at the stage indicated in the flow chart (the first successful login), it could continue until (with high probability) the *complete* authentication code had been retrieved.

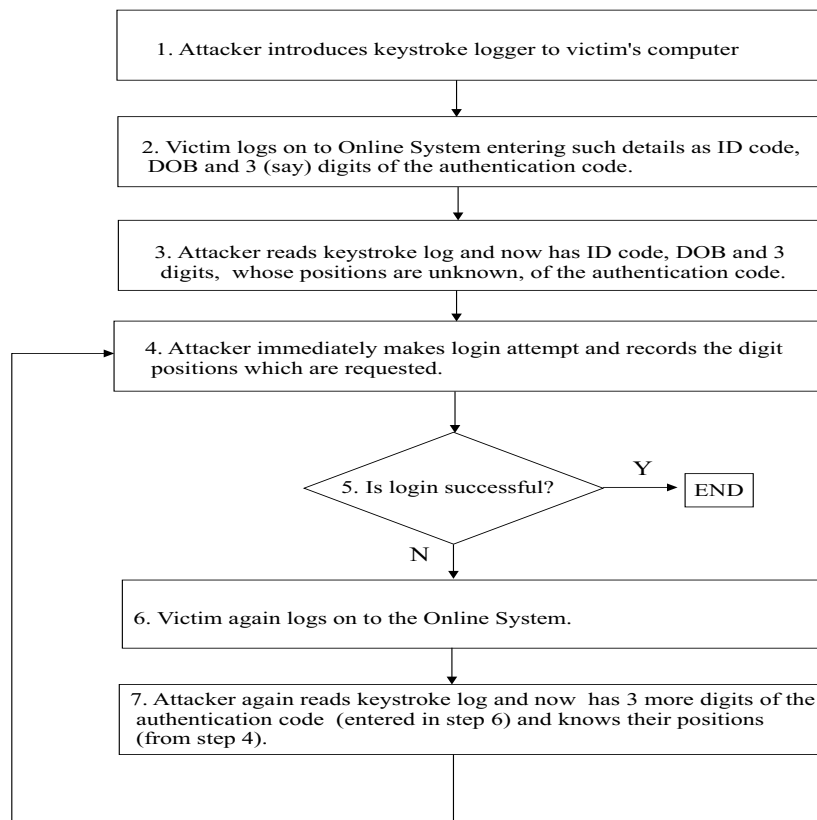


Figure 1: The sequence of attack steps described as a flow chart.

The time frame within which login can be accomplished, or the complete code recovered, depends on how frequently the victim uses the service and the dedication of the attacker.

It has been argued that this is a complex attack which would be time consuming and dependent on frequent user logins. Whilst perhaps true for a single target we feel that this objection misses the essential point. The methodology is easily automated by the dis-

semination of key-logging trojans. As these payloads are successfully inserted and report back via the Internet to their designer, a large database of victims could be automatically constructed and eventually exploited (perhaps simultaneously), all without any further work on the part of the attacker. It becomes a matter of statistics: some login credentials will be harvested and some will not, but with patience on the part of the attacker the scale of the eventual damage could easily be significant.

In case this seems far-fetched, we note that at least one exploit with approximately similar features has already taken place, see [6] and [7]. In the United States in 2005, a Miami businessman sued his bank after \$90,000 was lifted from his firm's online banking account following a computer virus attack [4] in which it appears authentication details were compromised by a trojan (CorelFlood).

6 Probabilistic Analysis

In the following, a "move" consists of the following sub-steps:

1. Victim logs in to the Online Banking system.
2. Attacker notes new digits.
3. Attacker makes login attempt and notes which positions are requested.

Number Known k	Overlap			
	d = 0	d = 1	d = 2	d = 3
k = 3	0.2917	0.1525	0.175	0.0083
k = 4	0.1667	0.5	0.3	0.0333
k = 5	0.0833	0.4167	0.4167	0.0833
k = 6	0.0333	0.3	0.5	0.1667
k = 7	0.0083	0.175	0.525	0.2917
k = 8	0	0.0667	0.4667	0.4667
k = 9	0	0	0.3	0.7

Table 1: Probabilities $P(10, k, d)$ for the overlap

Suppose the authentication code length is n ($6 \leq n \leq 10$) and that we already know the values (and positions) of k digits³. We write $P(n, k, d)$ for the probability that, at the next move, exactly d of the three positions requested are amongst the k we already know. We call d the *overlap*. So $P(n, k, d)$ is the probability that at the next move the overlap will be d . $P(n, k, d)$ may be calculated as follows:

$$P(n, k, d) = \frac{\binom{k}{d} \binom{n-k}{3-d}}{\binom{n}{3}} \quad (1)$$

This is because of the total $\binom{n}{3}$ ways of choosing 3 positions from n precisely $\binom{k}{d} \binom{n-k}{3-d}$ have an overlap of d with the k known positions. Calculating P for different values of d and k produces Table 1. Using this table we can construct the relevant probability tree for the worst case in which $n = 10$. The first few nodes are shown in Figure 2.

³We talk of such positions as 'known'.

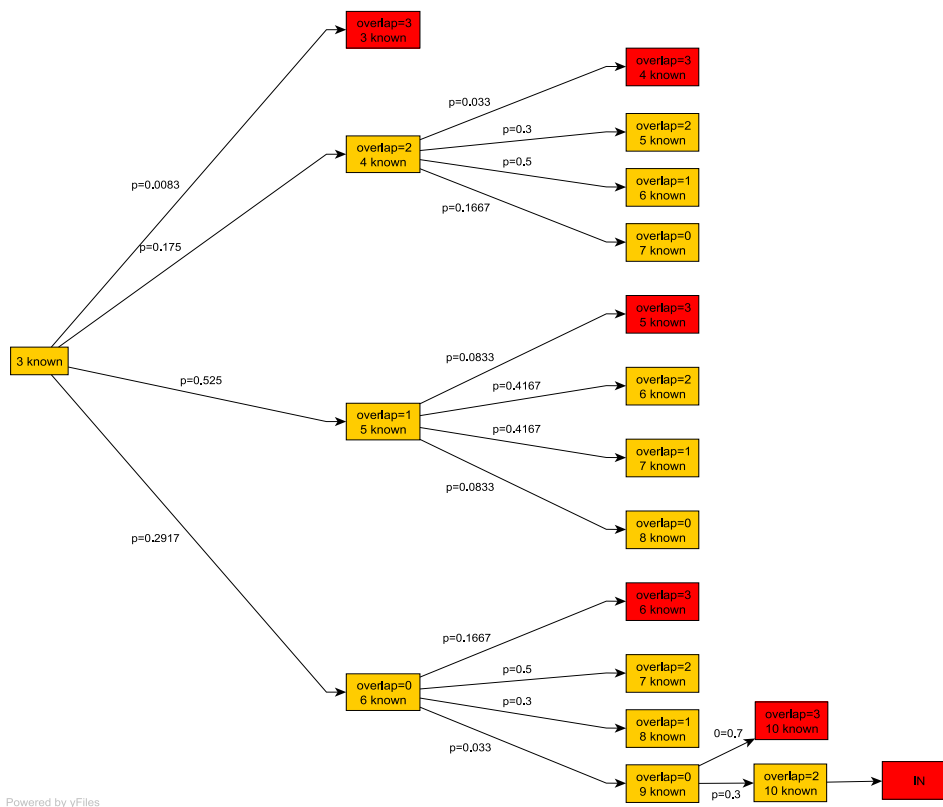


Figure 2: The first few nodes of the probability tree. Terminal nodes at which access has been gained are shaded (red). Only a few nodes beyond level three are shown.

Complete analysis shows that the full tree has maximum depth eight steps from the root node. Hence an attacker is *guaranteed* access within nine moves. However, the probability of tracing this path through the tree is 0.0000879, and so needing eight steps is exceedingly unlikely. The *most probable* exact number of moves is five, with a probability of just over 0.5. This is the worst case, since we assume that the user has chosen an $n = 10$ digit random authentication code. If the user has chosen a shorter authentication code, or a code whose structure may be predicted (such as an 8 or 6-digit date), then access could be gained in even fewer moves.

7 Conclusions

The vulnerability we have described may easily be removed by ensuring that the system always asks for a new set of characters whether or not login is successful (many online banking systems already do this). Because the analysis depends on character *positions* and not on the specific types of character that are allowed in the authentication code, allowing codes to consist of a wider variety of characters (not just numbers) would not remove the vulnerability, although it might improve security in other respects. Increasing the permissible lengths of authentication codes would slow down the attack we have described, but would not alter the basic situation. It would also be possible to add further stages of authentication - for instance, asking the user for extra authentication

digits before operations such as transferring money can occur.

In summary, the key point is that anti-keylogging systems implemented in this particular way effectively negate their entire intent.

It should be possible to detect this attack using existing heuristic analysis systems that are trained to pick up patterns of suspicious login attempts. For example, a reasonably robust profile for this particular exploit might be to look for two or more single failed logins that are not followed within a short period of time by either another failed login or a successful attempt, although other possibilities spring to mind.

Assuming heuristics detect such an attack the interesting question is: how should a bank then react? This represents a commercial conundrum. If they temporarily lock the account, whilst contacting the customer, they are going to annoy customers, and so perhaps lose money. If they don't react, but could have done, then they'll take the liability.

Banks are not in the business of producing perfect security, what they want is security that is as low cost as possible yet still adequate. Like webhosts in regard to child pornography, the current state of the law might thus tempt banks, who seek to do the right thing by their shareholders, to adopt a '*don't look - so we don't know*' approach.

More generally, this is one example of the flaws emerging in single-factor authentication using end-points of unknown security. For others we might look at the E-gold trojan, see [9] and [10]. This trojan automates the burden of siphoning money from the accounts and does it from the victim's own computer. Because it uses the victim's established SSL session and does not connect out on its own, it can bypass personal and corporate firewalls and evade IDS/IPS devices. To quote Counterpane [14]

In other words, the new Trojan programs do not have to trick victims out of revealing their password. Instead, they wait for the victim to perform his normal banking business. While the victim checks his bank balance, the Trojan silently siphons money out of the account.

Other, even more creative, approaches are the Blue Pill [11], which ironically puts the user's machine into its own 'Matrix', or Metasploit [8] an Open Source project inspired by H. D. Moore of *BreakingPoint Systems*. To quote from the Meterpreter manual:

Meterpreter, short for The Meta-Interpreter is an advanced payload that is included in the Metasploit Framework. Its purpose is to provide complex and advanced features that would otherwise be tedious to implement purely in assembly. The way that it accomplishes this is by allowing developers to write their own extensions in the form of shared object (DLL) files that can be uploaded and injected into a running process on a target computer after exploitation has occurred. Meterpreter and all of the extensions that it loads are executed entirely from memory and never touch the disk, thus allowing them to execute under the radar of standard Anti-Virus detection.

We have come full circle. In contrast to a 'closed worlds' philosophy, that instinctively hides weakness, Moore exemplifies the 'open worlds' approach (which arguably is now inevitable) of the Internet driven information space we now inhabit: vulnerabilities published are vulnerabilities more likely to be addressed.

Having no real assurances regarding the security of the user-point machine poses real problems for secure transactions which rely on purely software solutions. It is also the case that the time interval between when a vulnerability is published and when exploit code appears has shrunk, as criminals try to best take advantage of the 'window of exposure' before systems are patched or signature based Virus-checkers are updated, so that even knowledgeable and conscientious users cannot be confident regarding their system integrity.

These examples, together with the particular login issue we have discussed, raise doubts about the wisdom of having *any* sensitive system online with purely software access, and perhaps about online banking in general. One cannot always foresee the consequences of new technical developments. As bandwidth increases, and hardware memory chips becomes smaller and incredibly capacious, today's keystroke loggers might easily, for example, become tomorrow's 'keystroke plus screen' loggers.

8 Acknowledgements

We particularly thank the anonymous referees for their excellent suggestions which we (at least) feel have significantly improved the paper. AJJ would also like to acknowledge the courteous and patient treatment she received from NG (you know who you are), under what must have been somewhat exasperating circumstances, when our observations were first made public.

References

- [1] "Security flaw leaves 3m HSBC online accounts open to fraud (10 August 2006)", Bobbie Johnson and Ian Cobain, The Guardian, <http://business.guardian.co.uk/story/0,,1841853,00.html>
Accessed 16/02/2007
- [2] "Social Engineering the USB Way", Steve Stasiukonis, Secure Network Technologies Inc., http://www.darkreading.com/document.asp?doc_id=95556&WT.svl=column1.1
Accessed 27/06/2006.
- [3] "Corrections and Clarifications (21st September 2006)", The Guardian, <http://www.guardian.co.uk/corrections/story/0,,1877068,00.html>,
Accessed 14/10/2006.
- [4] "Florida man sues bank over \$90K wire fraud (8th February 2005)", The Register, http://www.theregister.co.uk/2005/02/08/e-banking_trojan_lawsuit/,
Accessed 08/02/2007.
- [5] "UK phishing fraud losses double (7th March 2006)", Finextra, <http://www.finextra.com/fullstory.asp?id=15013>,
Accessed 08/02/2007.
- [6] "Brits fall prey to phishing (3 May 2005)", The Register, http://www.theregister.co.uk/2005/05/03/aol_phishing/
Accessed 08/02/2007.
- [7] "Trojan phishing suspect hauled in (4 April 2005)", http://www.theregister.co.uk/2005/04/04/estonian_trojan_suspect_cuffed/
Accessed 11/02/2007.
- [8] "Metasploits Meterpreter (26 December 2004)", www.metasploit.com/projects/Framework/docs/meterpreter.pdf
Accessed 11/02/2007.

-
- [9] “Win32.Grams E-Gold Account Siphoner Analysis (4 November 2004)”, LURHQ Threat Intelligence Group,
<http://www.lurhq.com/grams.html>
Accessed 10/02/2007.
- [10] “Bank of America Seeks Anti-Fraud Anodyne (15 May 2006)”, Baseline,
<http://www.baselinemag.com/article2/0,1540,1962470,00.asp>
Accessed 10/02/2007.
- [11] Joanna Rutkowska, “Introducing Blue Pill (22 June 2006)”, The official blog of the invisiblethings.org,
<http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html>
Accessed 11/02/2007.
- [12] “Phishing”, Wikipedia, <http://en.wikipedia.org/wiki/Phishing>
Accessed 08/02/2007.
- [13] Boudon, R. (1977). “The unintended consequences of social action”, New York, St. Martins Press.
- [14] “2005 Attack Trends & Analysis”, Counterpane & MessageLabs,
<http://www.counterpane.com:80/cgi-bin/attack-trends4.cgi>
Accessed 10/02/2007.