Computer Security by Redefining What a Computer Is

Yvo Desmedt

EE & CS Department
University of Wisconsin – Milwaukee
Milwaukee, WI 53201-0784, U.S.A.

Abstract

The security of modern networked computers is very low and must be dramatically improved. Integrity of data and programs is an essential aspect of computers. We propose approaches towards computer security in which the main trust is a cryptographically authenticated "keyboard". The achievability follows from the current trend towards personal computers, workstations and notebooks. We discuss how this could increase computer security and which problems remain to be solved in such an environment.

1 Introduction

Presently a large portion[1] of the population has used a computer directly and the number of users who have programming skills is increasing. By making our society more dependent on computers, the potential impact of computer crime, fraud and unreliability starts to become more frightening. Computers are being used in many areas where security is very critical. They control chemical plants, they are being used to design and produce electronic components and mechanical devices using CAD/CAM programs, they monitor air-traffic, and in some countries the Registry of Births, Deaths and Marriages, has been computerized.

Nevertheless much research on the topic of computer security has already been done and has been reported in international conferences on the topic, nobody unfortunately has a watertight secure solution and it seems that a lot of ongoing research is necessary before we will achieve it.

The techniques we propose here, imply that computers will have to be designed differently than now. But it seems that only radical measures will make computers more secure because the modern add-on software approach towards computer security has clearly failed.

To have decent security on a multi-user computer the existence of a secure operating system that manages the access control is crucial. (Unfortunately deciding whether a access-matrix system is safe is undecidable [1] with all the consequences!) The fact that a multiple of users introduce (new) programs to the system increases the vulnerability to computer viruses [2, 3] (and worms). So it seems that the multi-user aspect of computers makes computers more vulnerable. We propose to "abolish" multi-user computers and we propose techniques to maximize the impact of this on the security of such computers.

In Section 3 we propose the main ideas. We overview in Section 4 what the impact of our ideas is on computer security by proposing a new open problem and in Section 5 we discuss which problems still need to be addressed.

2 No need for multi-user computers

In the beginning the price of computers was above the budget of (most) single individuals. Computers were therefore shared. Out of this sharing of resources grew the multi-user operating system for mainframes.

Due to hardware development one can easily foresee that computers will mostly be used by a single user (virtually more users could be allowed: the same user being "root" and the same user being "user"). A single-user computer could allow that many pro-

[1] Computers are used in many daily life applications: *e.g.*, many libraries and overseas railway companies are abolishing their paper (and microfiche) catalogs and paper timetables replacing them by computerized ones.

grams run at the "same" time, so that some aspects of a multi-user environment would be kept. Only the fact that these processes were started up by different users would vanish. So we would have, what we call, a multi-program, single-user computer.

Many personal computers, workstations, notebooks are used by a single user. The question then is why do we need multi-user environments? An obvious answer is that many applications need multi-user environments: *e.g.*, databases, joint development of programs, etc. A database contains information which was given by people authorized to input this data and which can be consulted by some people. In fact by viewing databases as servers, there is no need that these who input or consult the data have an account on the computer as long as they are able to perform their operations.

From above it seems that there is *no* need for a user to have accounts on different computers than his own. Moreover, the rate of progress in the speed of computer networks is much higher than the rate of progress in the number of CPU instructions per second [4]. So the role of networks in the computer environment could overtake the importance of the multi-user aspect of computers.

3 The main ideas

3.1 Hardware configuration

We now describe our hardware configuration. We also discuss the main security features needed in the system.

We assume each user will in the future have a workstation at home (or at work) and will have an electronic notebook. We assume the existence of a fast reliable network which does not need to be secure. Progress in networks [4] allow us easily to assume that such fast reliable networks will be around soon.

Suppose a user, A, wants to use his workstation (due to our assumption he will not be able to login on another one) when traveling. He takes his electronic notebook, types in his PIN (the PIN protects against fraud due to a lost or stolen notebook) and connects it then to a "networked-screen". A networked-screen is similar to an intelligent (graphical) terminal and is connected to a worldwide network, *but* it has *no* keyboard. The notebook makes, through the networked-screen, a link to the owner's workstation. The user does not login to his workstation. Instead, the notebook sends, using a *one-time-valid* authentication system [5, 6], the message: "I, A, am identifying myself to B," where B is the network address of the workstation. Hereto the notebook contains A's secret key. Because the authenticated message is only valid once, replay is excluded. Now A uses his notebook as a keyboard. The notebook authenticates, using the one-time-valid authentication system, all the commands typed by A. The output of his workstation is displayed on the networked-screen. The mouse could either be connected to the screen, or be a ball on the notebook. When the user is home, his notebook is still the keyboard (so his workstation has *no* keyboard!). Observe that screens are almost available everywhere. Any TV contains a screen. Unfortunately these in hotels are not always that easy to connect to another device.

If a lot of typing has to be done a networked-terminal can be used. To avoid lowering the security too much, most operating system commands cannot be executed when typed from the keyboard of a networked-terminal, but the keyboard of the electronic notebook is necessary to have the commands be authenticated. So to start editing a file the command "edit <filename>" has to be typed on the notebook; the user could type the editor commands from the networked-terminal. If the editor allows editing many files at once, each filename has to be entered from the notebook. It is clear that this is less secure than the networked-screen approach.

When more progress has been made on displays the notebook could have its own display. This would result in a downgraded laptop without disk and fast CPU, but resembling more a terminal. Indeed since very recently nodes on the internet network need not to be at a fixed location, but could actually move around. So, the notebook, connected through the network to ones workstation, would behave as a laptop. The output of the workstation could then be authenticated too, increasing the security dramatically. (If the user wants to have the privacy of his office (home), the communication link could be protected using encryption.) Then contemporary laptops (with disk and diskettes, etc.) will be obsolete.

To enhance the user-friendly aspect, the electronic notebook could be used to access ATMs, telephones and to replace the many plastic (chip) cards carried by just one notebook. However the risk of losing the notebook becomes enormous, even when it is protected by a PIN. Therefore *one* standard chipcard (or a more secure version) should be inserted for critical applications. This standard chipcard could be the user's identity card or his electronic passport [7]. The user should however never have to type a PIN or a pass-

word (besides the *one* PIN to start up his notebook). (The need for users to know more than one PIN has reduced the security concept of a PIN to a joke because users are writing down all their PINs.) To obtain the desired security the user should keep his standard chipcard separate from his notebook.

3.2 The role of cryptography

The security of single-user computers could be enhanced using cryptography, in particular authentication (which has often[2] been proposed to improve computer security). Observe that the role of identification in the above has merely vanished, which it should have a long time ago. Indeed a user who is logged in on a modern computer over a network is vulnerable to the most unbelievable attacks. The computers through which his communication goes (many are very insecure ones) are able to insert all possible commands without displaying them, *e.g*, deleting half of his files, etc.

The programs the user uses may contain trapdoors that allow circumventing the security measures. To prevent this the authentication should be developed in hardware so that only the legitimate user can access his workstation. However, if conventional cryptography is used, then the workstation contains the secret key and the program may succeed in having a fraudulent order being authenticated. If digital signatures [8, 9] are used the key stored in the workstation to verify the authenticity is different from the one used to generate the signature. The key used to verify the authenticity can be public. So the above problem is then eliminated (assuming that there is no conspiracy between the designer of the electronic notebook, where the secret key is stored, and the designer of the workstation/program).

A reason which has often been given against the use of cryptography is that it is too slow. Indeed the best hardware for RSA signatures (and DSS is not that much faster!) gives only a speed of 10-20 kbits/sec. [10]. However, that is fast enough to authenticate all what is being typed, because we don't type 1000 symbols per second!

3.3 Clarification

Let us first focus on what above approach exactly means. Suppose that the notebook signs, for example: "whta<del><del>at". What will be stored in the workstation's memory and eventually disk space? Will it be the result, *i.e.*, "what", or will it be what was signed. We call what was signed the *real* world and the result the virtual world. Nowadays only the virtual world is stored and the real world is lost. Storing the real world on disk has some advantages in some circumstances.

If one stores the keyboard's output then it is the editor's task to display the corrected text. The disk will basically store signed history, *i.e.*, a complete signed log of all modifications. So what a user sees being displayed is basically a virtual world produced by the editor, but in reality on disk all modifications, even minor ones would be stored. One of the advantages is that in the case of databases one can keep track of who modified and what. Hereto one just needs to display the real stored data which includes the signatures of all modifications. Moreover if the disk controller only allows authorized users to modify a database file, it is sufficient for the disk controller to verify the signature. So if a public key system was used the (disk controller of the) database could check whether this user has the authority to modify the data. Only if so will the updated data be stored.

A disadvantage is that editing old text can be slow. Before being able to display the data, the editor has each time to convert from real world to virtual world. This means to actually perform all the old editor's commands typed long time ago by the user. If this is too slow one could store the virtual world, but then the data is not signed anymore and the problem of protecting its integrity reappears. In the next section we discuss how this might be solvable.

3.4 Further security improvements

On a "single-user" computer the disk space could be divided securely into four types of files:

1. "root" files which are never modified. The disk drive will never allow these to be modified; read-only optical disks could even be used to achieve this, but modern implementations are often slow.

2. "root" files that are often modified.

3. manually produced user files (source files, text files, script files, etc). These files need authenticated permission from the user to be modified. The disk controller would check this authentication. A mixture of signature and authentication techniques could be used to obtain acceptable speed and security.

[2] Unfortunately the concept of authentication has often been confused with error-correcting codes, which are two completely different topics.

4. non-manually produced user files (the mail spool file, data files which are the output of a program, etc).

If an open problem we discuss in Section 4 can be solved then this induces a method to achieve security of data of Type 2 and Type 4. In the meanwhile, operating system based security measures should be used to protect these files.

The above division implies that operating system should be less configuration depended and that one can deal with this configuration dependency using the modifiable "root" files. This implies that the operating system cannot easily be updated or modified by a virus. Observe that many users do not need frequently updated operating systems.

If these updates are necessary the read-only disk should be removable. If these disks are removable the question whether the original software is used becomes a problem. So it may be advantageous to have the same (main) software being used through the lifetime of the computer. If the number of software vendors is small the authenticity could be checked when the disk is mounted using the public key of the vendors. Because the disk is read-only there is no need more to check the authenticity of the disk while running it, as long as it was checked when mounted.

4 Operator oriented signatures: an open problem

To explain what we need let us focus again on our editor example. Each editor command can be viewed as an operation. So the editor commands form an operator space. Each text which is being typed is a string from, what we call, the operand space. Now, when one corrects parts of the text one applies an operator on this text. Also the concatenation command is an operation on two texts. To explain our need let us just focus on this concatenation operation.

Due to our set-up, we have data strings: Operand_1 together with a signature and Operand_2 together with a signature. Due to our keyboard, editor commands, such as the concatenation will be signed too. In an operator oriented signature scheme now we have, very roughly speaking, that given a signature of the operator and the operands one can easily produce a signature of the result. Let us from now on denote Sign(operand) as the signature of the operand by the keyboard. Then in a operator oriented signature scheme, we would have that given: Operand_1, $\text{Sign}(\text{Operand}_1)$, Operand_2, $\text{Sign}(\text{Operand}_2)$, Operator, and Sign(Operator), one could easily produce $\text{Sign}(\text{Operator}(\text{Operand}_1,\text{Operand}_2))$, *i.e.*, even without having the notebook's secret key. However this itself would not be very useful. Indeed if a user ever signed an operator it could be used to act on whatever text one wants. So clearly, above description was indeed rough and needs to be adjusted.

To avoid above problem one should give: $\text{Sign}(\text{Sign}(\text{Operand}_1),\text{Sign}(\text{Operand}_2),\text{Operator})$. So in such a scheme one would give in total: Operand_1, $\text{Sign}(\text{Operand}_1)$, Operand_2, $\text{Sign}(\text{Operand}_2)$, Operator, $\text{Sign}(\text{Sign}(\text{Operand}_1),\text{Sign}(\text{Operand}_2),\text{Operator})$. The security requirement for operator oriented signatures is that without the above signature it should be infeasible (without knowing the user's secret key) to produce $\text{Sign}(\text{Operator}(\text{Operand}_1,\text{Operand}_2))$. This means that if we would have an *insecure* workstation, it cannot produce a fake signature of $\text{Sign}(\text{Operator}(\text{Operand}_1,\text{Operand}_2))$. Clearly the insecure workstation could produce the result $\text{Operator}(\text{Operand}_1,\text{Operand}_2)$ but if the disk controller only allows to store signed data, the result is of no help to the enemy that programmed the insecure workstation.

Let us take as an example the delete character operator. Viewing this operator as delete this character in a string is ambiguous, indeed, where is "this" character? Let us discuss how we solve this problem. To delete "this" character the notebook signs each cursor movement operation, to be more precise produces Sign(Sign(String),cursor operator). In a operator oriented signature scheme this would allow the insecure workstation (given above and the string: String) to reply with: Position and Sign(Position). The notebook verifies the signature and if satisfied sends Sign(Sign(string),delete character at Position).

To be very precise we have to observe that in above example an insecure computer could have stored an old position and an old signature (from a previous cursor operation) and return these. To avoid this problem it is clear that one-time-valid authentication is more useful than signatures.

4.1 The Impact

It seems that the impact of our notebook based proposal on enhancing computer security is rather small. Indeed it is obvious that it protects the data users typed themselves. However, most programs that are being executed have not been typed by the user.

However, the idea of operator oriented signatures and operator oriented one-time-valid authentication

does not seem to be limited to just editor operators. Any program could be viewed as an operator and when a user runs a program he would sign the fact that the operator can be applied on certain data. If operator oriented signature and authentication schemes would exist, then whatever the insecure program (that claims to fulfill this operator) does, it will fail to produce the right signature.

4.2 A dream?

It is not clear whether such operator oriented signature and authentication schemes can be constructed. Even if they can, it is not clear for which set of operators and operands they can. However, we hope that above makes it clear that if they could be developed, these would have a major impact on computer security.

5 Remaining problems

5.1 What has been solved

By using authentication the problem of break-in by users who guess the password of a user has been dramatically reduced. It has not been completely eliminated. Indeed the person stealing the notebook and its PIN can still break in. However this requires physical access, while modern computer break-ins are much easier and do not require such a physical break-in.

The problem of having an insecure network when using remote login has been solved with our approach as long as the network is reliable (*i.e.*, the communication is not jammed).

The problem of illegitimate access has been dramatically reduced because the system is no longer multi-user. If data should be accessible by others, as in databases, the authority is being checked using cryptographic techniques. Although the use of cryptography to verify access has been proposed before, the multi-user aspect of the computer made this a complicated task. The *physical separation* between the signing operation of read commands (performed in the notebook) and the verification of the authority to read (a part of) a specific file (performed by the remote computer) is an essential feature.

The problem of *new* viruses has been reduced due to the ideas of putting the operating system on a read-only disk and the use of authentication.

5.2 Remaining problems

Our approach clearly does not eliminate all computer security problems. There are many remaining problems. The user has to trust the hardware of his electronic notebook and hardware of the disk drive. Making the notebook a suitable tamperproof device increases the security, but the user still need trust in the manufacturer.

We have not dealt with the issue of privacy. If we would, the problem that any electronic equipment radiates [11, 12] needs to be solved using proper shielding techniques (which are not obvious). Validation by a trusted authority of hardware is required, but if a lot of freedom is given to the designer, it is not an easy task [13]. The concept of operator oriented signatures can very easily be generalized towards operator oriented encryption. Roughly speaking we would have that, given an encrypted text corresponding with a plaintext and a signed operator it would be easy to compute Operator(Plaintext) without revealing anything additionally about the rest of the plaintext. Such tools would allow one to "search" in encrypted data without decrypting it.

In the early stages of unclassified research on cryptography privacy homomorphisms [14] were introduced to solve the problem of computation with encrypted text. Moreover, a "proof" was given that if some operations were possible the resulting scheme would leak. The "proof" relies on cryptosystems which have the property that two encryptions of the same plaintext produce the same ciphertext. The work of Goldwasser and Micali [15], which argues that randomness should be used in the encryption process to avoid leakage, implies that the "proof" of non-existence of such privacy homomorphisms is folklore. Moreover, operator oriented encryption is also different from privacy homomorphism. Indeed operator oriented encryption requests a signature to enable the execution of the operator.

6 Conclusion

Modern computers have many security problems. The approach of trying to fix those has been analyzed for — in light of the results achieved — too long a time.

So, it has been clearly demonstrated that security is not an added feature, but must be taken into account before designing the hardware and software of the computer. Instead of trying to fix the problem, it

may be better to throw (almost) everything away and start from scratch. This would include:

- to design the hardware of the computer to support security as much as possible using cryptographic techniques,
- to avoid "Open Sesame" security (as identification is) but to authenticate and verify each command (and sub-commands),
- and to avoid modern based security.

In other words, it may be better to redefine what a computer is such that it can be sufficiently protected. Indeed, it may be better to have in fifteen years time some secure hardware-software around (which one might still call a computer) than having updated versions of todays computers that became useless due to the increase in computer fraud (caused directly or indirectly by internetworking, viruses, and the fact that there will be more trained users).

In this paper we have explained the need for authentication at the source, which means the moment the text is typed. To be as secure as possible, we propose that each user only uses his own input device. It is clear that this idea on itself has a potential impact on computer security, but that impact is limited. Tricks could increase the impact. We have opted not to focus on these tricks, but to put forward an open problem which, if solvable, could have a very large impact.

Clearly, one could question whether the open problem (whether one can design operator oriented authentication schemes) is nothing more than a dream. In this context the author reports that major, but theoretical, progress has been made recently towards operator oriented encryption [16]. Reporting about its details is beyond the scope of this paper.

Acknowledgement

The author would like to thank Catherine Meadows for her encouragements.

References

[1] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM*, vol. 19, pp. 461–471, August 1976.

[2] F. Cohen, "Computer viruses: Theory and experiments," in *Seventh National Computer Security Conference*, pp. 240–263, September 1984. Also appears in Second IFIP International Conference on Computer Security.

[3] L. M. Adleman, "An abstract theory of computer viruses," in *Advances in Cryptology — Crypto '88, Proceedings (Lecture Notes in Computer Science 403)* (S. Goldwasser, ed.), pp. 354–374, Springer-Verlag, 1990. Santa Barbara, California, U.S.A., August 21–25.

[4] K. Kümmerle, "High bandwidth communication systems: Where do we go?," May 1992. Guest Speaker at IEEE Infocom '92, Florence, Italy.

[5] J.-J. Quisquater, "Signatures, identifications et controles d'accès." Lecture given at INRIA (France), December 16, 1986.

[6] Y. Desmedt, "Major security problems with the "unforgeable" (Feige-)Fiat-Shamir proofs of identity and how to overcome them," in *Securicom 88, 6th worldwide congress on computer and communications security and protection*, pp. 147–159, SEDEP Paris France, March 15–17, 1988.

[7] G. I. Davida and Y. G. Desmedt, "Passports and visas versus IDs," *Computers & Security*, vol. 11, pp. 253–258, May 1992.

[8] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT–22, pp. 644–654, November 1976.

[9] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 294–299, April 1978.

[10] E. F. Brickell, "A survey of hardware implementations of RSA," in *Advances in Cryptology — Crypto '89, Proceedings (Lecture Notes in Computer Science 435)* (G. Brassard, ed.), pp. 368–370, Springer-Verlag, 1990. Santa Barbara, California, U.S.A., August 20–24.

[11] W. van Eyck, "Electromagnetic radiation from video display units: An eavesdropping risk?," tech. rep., P.T.T., Leidschendam, The Netherlands, April 16, 1985.

[12] W. van Eyck, J. Neesen, and P. Rijsdijk, "On the electromagnetic fields generated by video display," in *Proc. Symp. EMC*, (Zürich), March 1985.

[13] Y. Desmedt, "Is there an ultimate use of cryptography?," in *Advances in Cryptology, Proc. of Crypto '86 (Lecture Notes in Computer Science 263)* (A. Odlyzko, ed.), pp. 459–463, Springer-Verlag, 1987. Santa Barbara, California, U.S.A., August 11–15.

[14] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation* (R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, eds.), (New York), pp. 169–179, Academic Press, 1978.

[15] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, pp. 270–299, April 1984.

[16] M. Burmester and Y. Desmedt, "Private computation with shares." In preparation.