INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Applied Cryptanalysis of Cryptosystems and Computer Attacks Through Hidden Ciphertexts Computer Viruses

Eric Filiol

No 4359

Janvier 2002

THÈME 2

Rapport de recherche

ISSN 0249-6399

Applied Cryptanalysis of Cryptosystems and Computer Attacks Through Hidden Ciphertexts Computer Viruses

Eric Filiol*

Thème 2 — Génie logiciel
et calcul symbolique
Projet Codes

Rapport de recherche n° 4359 — Janvier 2002 — 15 pages

Abstract: This report presents a new class of techniques which allow either the attack of a computer or to catch the keys of a cryptosystem by using a pair of (or combined) viruses, one of them being hidden by the attacker in ciphertext. These techniques are valid for any operating system and can be efficiently implemented in any programming language and for any operating system. In order to avoid detection, the viral infection is very limited and uses polymorphic techniques. Moreover the main virus erases itself after the payload action. The general structure of the two viruses is presented and the problem of protection against such attacks is finally envisaged.

Key-words: Computer security, applied cryptanalysis, symmetric cipher, ciphertext, computer viruses, steganography, PGP, AES.

(Résumé : tsvp)

also Virology and Cryptology Lab, French Army Signals School, ESAT/DEASR/CSSI, 35998 Rennes, France, *efiliol@esat.terre.defense.gouv.fr*

* Projet Codes - Eric.Filiol@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Cryptanalyse appliquée de systèmes de chiffrement par virus informatiques

Résumé : Ce rapport présente une famille de techniques permettant d'attaquer un ordinateur ou de capturer les clefs d'un système de chiffrement en utilisant une paire de virus informatiques (encore apppelés virus binaires ou combinés), l'un d'entre eux étant dissimulé par l'attaquant, dans un cryptogramme. Ces techniques sont efficaces pour tous les systèmes d'exploitation et tous les langages de programmation. Afin d'éviter la détection, l'infection est limitée et utilise des techniques polymorphes. De plus, le virus principal se désinfecte lui-même après le déclenchement de sa charge finale. La structure générale des deux virus est présentée et les possibilités de protection contre de telles attaques sont envisagées.

Mots-clé : sécurité informatique, cryptanalyse appliquée, chiffrement symétrique, cryptogramme, virus informatiques, stéganographie, PGP, AES.

1 Introduction

The recent evolution of modern symmetric cryptology has led to the widespread use of highly secure secret key cryptosystems. Among many examples (IDEA [12], GOST [8], Blowfish [18],...) the best ones are without doubt the different candidates proposed for the A.E.S [1], for the NESSIE project [16] or at CRYPTREC [5]. The same problem is encountered with highly secure steganography products (Outguess [17] and others [21]).

Despite frequent highly unrealistic claim of cryptanalysis, these cryptosystems may be considered as unbreakable for a very long time to come. Neither the brute force attack (exhaustive search on the key-space) nor the published mathematical or algorithmic analysis are likely to yield "operational, real-time" cryptanalysis for these ciphers in the next few years.

An FBI spokesman recently [4] acknowledged the fact that "*encryption can pose potentially unsurmountable challenges to law enforcement...*".

As an example, let us consider the case of the AES. Let us suppose that we use a deep-crack like computer allowing exhaustive key search of 2^{56} key in one second. Then brute force attack on the different AES version will:

- require 1.5×10^{14} centuries for 128-bit keys.
- require 2.76×10^{31} centuries for 192-bit keys.
- require 5.1×10^{50} centuries for 256 bits keys.

The situation is quite different when considering another kind of cryptanalysis that the "operational men" call "applied cryptanalysis".

Definition 1 *Applied cryptanalysis describes the set of all non mathematical means and techniques which aim at attacking a cryptosystem either on its implementation or on its everyday use in order to endanger its secret key thus allowing easy deciphering to the attacker.*

With regards to this definition, physical techniques such as timing attack [10] and power analysis [3, 11] are recent known examples. However these techniques nearly always require highly technical knowledge, costly hardware and the need to have physical access to the cryptosystem or at least to its implementation (*e.g.* smart card). Stealing cryptographic material may very likely provoke the complete change of the keys.

Human compromise of the secret key user or secret key theft are less well known but have been shown to be very effective during previous wars as spying history teaches us [9]. The main drawback of this approach comes from the fact that it is both risky and incautious. Moreover, close contact with the user is always directly or indirectly necessary. Once again compromission of the keys may be, with a good probablility, detected thus leading to the replacement of the keys.

Another technique is to have access to the hard disk when considering encryption softwares and to make it "speak"; in most cases, cryptographic softwares are so badly implemented that swap mechanism between processeur and hard disk nearly all the time, (unprotected) secret keys may be present on the hard disk for a very long time. A simple analysis of the hard disk will find them. Of course, the major drawback is to have physical access to the target computer, and the "victim" will no longer use its keys.

A more interesting, operational approach consists of using "infecting softwares". One of the first known attempt is the Caligula virus [20] but it was totally inefficient, since this macro virus stole encrypted, thus useless, secret key of PGP. Recently [4] the FBI has acknowledged the use of *Magic Lantern* Technology in a broader project called "*Cyber Knight*". The aim is to capture the user's secret keys by installing Trojan-like powerful software over the Internet which will perform eavesdropping of the target computer keyboard buffer. Such an approach has been very recently observed with the new worm BadTrans [20] which essentially steals passwords by keyboard buffer eavesdropping too. Unfortunately for the attacker, due to their high replicating power, worms are bound to be detected very quickly. in the other hand viruses embedding many functionalities are generally of large size. It implies that their infectious capabilities are rather limited and moreover they are liable to be easily detected due to their size.

The aim of this paper is to present a new, efficient technique of applied cryptanalysis using other less-known infecting malwares: binary (combined) computer viruses. In particular, we present an operational approach which effectively allowed us to remotely get the secret key of a target user, during the various experiments we conducted, without detection by recent anti-virus softwares and of course the "target". In our experiments we considered several now very famous "unbreakable" cryptosystems.

Each time we managed to get the user's secret key very easily. The main trick was to use the ciphertext as a "vehicle" for our attack (but other variants are now being implemented: e-mail attachement, sniffer module,...).

Combined (or pairs of) viruses have been used in this attack (*i.e.* two computer viruses which combine their respective payload to obtain a given final effect). For details on these little known computer viruses see [7]). One resident, polymorphic, stealth virus detects the presence of a second virus, hidden by the attacker in the ciphertext, and launches it. This latter then conducts the attack itself by selectively infecting cryptosystems executables and allowing the final secret keys to escape. Just after its payload action the second virus will disinfect itself, leaving no trail of its presence. Moreover it attributes polymorphic and stealth properties to the first virus.

The main interest of two complementary viruses instead of one (which could itself steal the secret keys), is that infection is broken in two independent parts which never simultaneously occur. As a result, we always deal through the different steps of the attacks with small thus stealthier viruses. A virus alone could not work so efficiently and with such stealth capabilities.

Thus physical access to the host computer is no longer necessary. The attack is very easy to conduct and required very limited means. Moreover it has been made completely stealthy in order to avoid detection. Otherwise it would inevitably induce the user to change their secret keys.

The paper is organised as follows. Section 2 presents the general overview of these attacks. In particuliar we define precisely the operational context. In Section 3 the first infection level is presented while Section 4 describes the second level of infection. Section 5 details the different possible attacks themselves which essentially aims at attacking the cryptosystem and at "key release". The case of a simple computer attack with a Trojan horse is presented as well. In Section 6 we try to envisage possible protection against these attacks.

Warning and Claim : the purpose of this paper is purely research and is not in any way to promote, help or encourage such attacks or the use of viruses. Our motivation is to draw the attention of security specialists to new attacks which have been proved dramatically efficient. We emphasise the fact that such attacks, as well as use of computer viruses, are harshly punished by the law in most of countries.

2 General Overview of the attack

We now present the general description of this family of attacks before entering into details in Sections 3, 4 and 5 and then the operational condition of our experiments.

We will not recall the definition of a virus and basic concepts on viruses that are used in this paper. Readers keen to know more will easily find out more in all common handbooks dealing with computer viruses and particularly in [7, 13].

2.1 Description of the attack

Let us consider the following setting for our attack. Two users, Alice and Bob, communicate via encrypted files. To do so they both use a strong symmetric cryptosystem denoted **S**. Alice first encrypts her plaintext **P** with secret key **K**, producing ciphertext **C** which is finally sent to Bob.

The attacker Charlie intercepts **C** and just appends a virus executable file V_2 which will perform the attack on Bob's computer. Finally $(\mathbf{C}||\sigma||V_2)$[1] is sent to Bob after insertion of a signature σ. Here the word "signature" must be taken in its viral meaning that is to say a caracteristic string allowing viral detection (viral signature). When dealing with the concept of signature as used in cryptography, we will use the term "cryptographic signature" in this paper. Thus possible confusion between these two concepts will be avoided.

Thus at a preliminary stage, Bob's computer has been infected by another virus V_1 which is of a resident, polymorphic, stealth kind[2]. It is designed in such a way that each time Bob switches his computer on, virus V_1 is launched and stays resident. Its role is to permanently look for a received file with V_2 appended to it. When found and identified through signature

[1] Symbol $||$ denotes text or string concatenation

[2] One may find that this constitutes a very unlikely hypothesis, since one can estimate that generally users are very careful of their computer security. In fact, this belief is false. Real-life experiments we conducted actually showed that it is always easy to infect someone's computer, whatever may be the user. Social engineering provides a lot of tools and tricks that make infection really easy, maybe too easy.

σ, V_2 is launched by V_1, and ciphertext integrity is restored, that is to say V_2 and signature σ are removed from the ciphertext.

When launched V_2 looks for cryptosystem executables **S** to infect and then do so. As a consequence V_2 is a very low-infecting, victim-specific virus (generally one executable depending on to the victim's computer). When deciphering the ciphertext Bob in fact first launches virus V_2 which performs the following tasks:

1. Turn V_1 into a dormant state and later on management of V_1 for polymorphic purposes.
2. Capture of Bob's secret key **K** used for deciphering the received ciphertext. V_2 stores it.
3. Finally returns control to the cryptosystem executable **S** which lastly deciphers the received ciphertext.

If Bob performs other decipherings before enciphering, V_2 in the same way will capture the key and store it with the previous ones if different from them. Let us suppose that secret keys $\mathbf{K}, \mathbf{K}'$ have been previously stored by V_2. When Bob performs a first enciphering (that is to say first enciphering after infection by V_2) with secret key $\mathbf{K}''$, V_2 once again is first launched and performs the following tasks:

1. Loading of the previously stored secret keys.
2. Comparison of $\mathbf{K}''$ with previously stored secret keys. If different $\mathbf{K}''$ is kept otherwise discarded. Let us suppose that V_2 has captured three different keys $\mathbf{K}, \mathbf{K}', \mathbf{K}''$.
3. Gives temporary control to **S** to produce ciphertext **C'**.
4. Takes again control and conceals secret keys $\mathbf{K}, \mathbf{K}', \mathbf{K}''$ in ciphertext **C'**[3].
5. Disinfects **S** from V_2 (on the hard disk).
6. Finally returns control to **S**.

The file $(\mathbf{C}', \mathbf{K}, \mathbf{K}', \mathbf{K}'')$ is sent by Bob. Charlie again intercepts this file and get the concealed keys $\mathbf{K}, \mathbf{K}', \mathbf{K}''$. The attack is completed.

Remark.- Since V_1 is still present and remains undetected the attack can be conducted as often as required by the attacker.

2.2 Experimental Settings

Different variants of this attack scenario have been conducted with several, different contexts, the context being defined as a set of the following elements:

[3]In case of embedded function in **S** aiming at digitally cryptographically signing **C'**, this concealment step must act before any cryptographic signature. An intelligence stage will provide necessary information on the software that Bob uses.

- Different platforms. We worked essentially on Intel-based or Intel compatible platforms. Current implementations now focuses on Alpha-based platforms and on Sun platforms.

- Different operating systems (OS). Both Unix (including Linux) and Windows environments have been considered. Assembly language has been used in preference for the experiments but attacks with source code viruses have been also implemented with success (for various programming languages).

- Different, recent antivirus softwares (AV) in general use. We will not disclose the product's name for the following reasons: by nature AV products are unlikeky to be able to detect new viruses, especially those with new polymorphic capabilities and techniques[4].

  Our aim is not to tarnish commercial products which are nonetheless rather good and useful products against known viruses. The interest and the force of our attack is to use very specific, low-infecting viruses which use completely new and unpublished polymorphic techniques. None of the AV products we tested has been efficient at detecting viruses V_1 and V_2. This unfortunately comes from a "natural" limitation in current state-of-the-art AV technology.

- Different cryptosystems have been attacked either as plain executable files (working in command line mode) or in embedded products: PGP, AES and Triple DES. Once again these attacks do not challenge intrinsic mathematical security. We considered applied cryptanalysis, not cryptanalysis. Experiments have also been conducted with success with steganography products where ciphertexts are replaced by images or MP3 files.

Finally, no specific assumptions have been made on Bob's behaviour concerning his computer security. We considered for our experiments that Bob was an average user with general knowledge in computer security. Current research are dealing with "paranoid" users.

3 The first infection level

Virus V_1 is designed to be a very small virus (a few hundred bits to less that two kilobytes according to the settings) whose polymorphic and stealth capabilities are managed by virus V_2 as described in Section 4. Here precisely lies the interest to use couple of viruses. First infection step is bound to remain undetected if V_1 is small enough. Thus most of its functionalities (polymorphism in particular) are insured by V_2 as described in section 4. Its characteristics, structure and functions here follow:

[4]Contrary to recent AV claims, detection of unknown viruses using unknown polymorphic and stealth techniques can be compared to the philosopher's stone. The best argument, among others, is that all these products are maintaining a viral signatures update file. Moreover, analysis of possibly infected files in order to forecast "viral nature" is by essence condemned either to false alarm detection or to limit to already known techniques. The heuristics approach, whatever may be its true nature, can only react against unknown

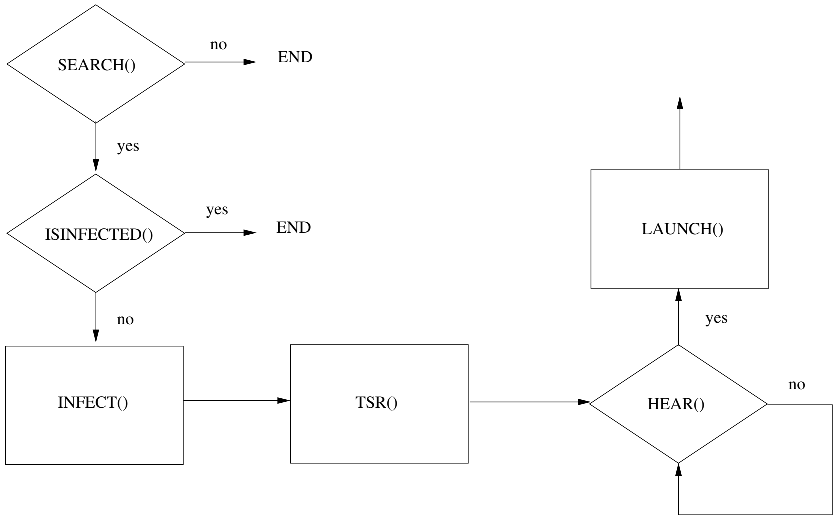

Figure 1: Virus V_1 organization chart

1. Virus V_1 is low-infecting. It only infects computers where given target cryptosystem (or steganographic) softwares are present. A **search()** routine performs this task. According to the target computer operating system, optimal search ensures quick localisation of target executable file (scanning of the registry base, of the filesystem description table, ...).

2. Virus V_1 is a resident virus. In other words, immediately after first infection and each time the computer is switched on, V_1 payload is active in memory. Thus it primarily infects (OS dependant) executable files which are naturally resident, or called up during the boot sequence or are very likely to be launched by the user. Another technical possibility under Windows, is to configure the registry base during the first infection in order to launch it automatically during all next OS sessions. But many other possibilities exist which have been successfully tested.

   From a technical point of view, V_1 has an **isinfected()** routine to check whether the host computer is already infected or not (it is worth noticing that this routine is able to detect not only the current version of V_1 but all the polymorphic version of this virus, since the attacker knows how to produce polymorphic versions from a previous one) and an **infect()** routine looks for specific files and infects them. Finally a **tsr()** routine makes it resident and introduces (limited) stealth capabilities (in particuliar CTRL-ALT-DEL or PS -AUX commands must not show V_1 process).

viruses using known techniques (stealthiness, polymorphism,....) that malware programmers borrow without improving them. See also Section 6.

3. The last module of V_1 to be considered is its payload. While resident, it looks for received files in order to detect signature σ (routine **hear()**. When found, **launch()** routine launches virus V_2 and restores ciphertext integrity (erasure of σ and V_2 executable code). For this last part, monitoring system calls (for example by "hooking" Windows API) will be used to efficiently intercept mail and network activity before any cryptographic signature checking. Moreover, Bob is not able to detect the temporary growth of the ciphertext due to V_1 action.

Figure 1 summarises V_1 structure.

4 The second infection level

Virus V_2 benefits from V_1 action. Thus whilst this latter is a very small in size, V_2 is larger and exhibits more complex functionalities and structure. Here precisely lies the interest and power of combined viruses. In turn, V_2 complements V_1 action by ensuring the complex stealth and polymorphic nature to V_1. This is done essentially by turning V_1 into a dormant state during V_2's life in Bob's computer and modifying its code before waking it up. Another possibility is to really disinfect V_1 form Bob's computer and to reinfect it by a modified form of this first virus. We developed two main variants of virus V_2. The first one aims to attack

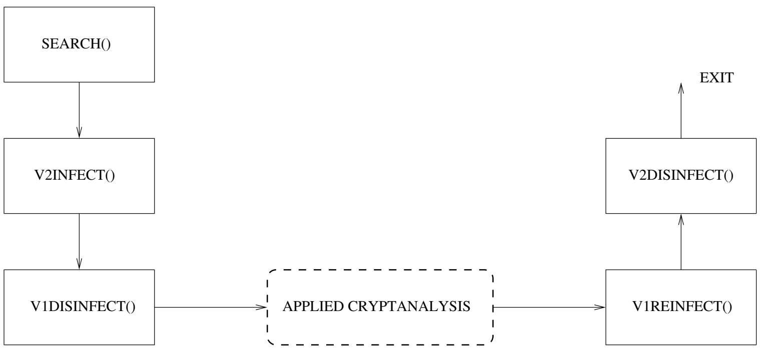

Figure 2: Virus V_2 organization chart (infection part)

only one target cryptosystem. The second is able to treat several target cipher systems in a row. Without loss of generality here is a presentation of the first variant.

1. Virus V_2 first looks for specific cryptosystem binaries that we wish to attack. A **search()** routine performs this task. It has been optimised vis-à-vis the target computer OS, in order to optimally limit the search (reading of the FAT or equivalent filesystem structure). Another possibility is to pinpoint the exact location of target cryptosystem executable file by consulting values stored by V_1 in a previous step. This

suppresses the risk of being detected by the AV computer activity monitoring. Thereafter infection is performed by a **v2infect()** routine. At this step V_1 is first put into a dormant, inactive state. One important task of **v2infect()** routine is to ensure a very high degree of stealthiness to V_2.

2. Once infection is completed, V_2 kills V_1 and disinfects the hard disk with a **v1disinfect()** routine.

3. V_2 then waits for the applied cryptanalysis itself. We describe this in Section 5.2.

4. Once V_2 has collected all the secret keys and has released them, routine **v1infect()** reinfects the host computer with a modified (polymorphic) variant of V_1 (in the same way as if it were V_1 itself[5]. In particular the signature σ is changed to σ' for new potential attack.

5. Finally, last, routine **v2disinfect()** clean the target computer and remove virus V_2. The attack is completed and new conditions are set up in order to carry out this attack again.

The action of virus V_2 allows powerful polymorphism of V_1 and V_2 itself. Once again only binary viruses are able to perform this to such a high level. Figure 2 summarises V_2 structure.

5 The Payload Action

5.1 Computer Attack

We will not concentrates too much on computer attack since we primarily are interested in applied cryptanalysis. However, attacks with binary viruses in a purely computer security problem are fully and easily adaptable. We performed experiments where virus V_2 was a home-made Trojan horse (for LAN or WAN attack context). Ultimately in each experiment, we took complete control of the target computer, surrendering it up to the attacker without being detected by current antivirus softwares.

Note that in this case, it may be not possible to attach virus V_2 to a ciphertext (since the target computer owner may not use encryption at all). Thus it may be necessary to attach the virus to another kind of file. Fortunately this does not change the efficiency of the attack (see Section 6 for the other solutions).

5.2 Applied Cryptanalysis of a Cryptosystem

Whilst remaining general overall, we shall now focus on the version we developed for when only one cryptosystem is present in the target computer. In the case of several cryptosystems

[5]Thus V_1 is again a resident and stealth virus but has a very different code. We mainly used dedicated, optimal encryption combined with rewriting techniques for the decryption and launch subroutines in **v1infect()**

attacked in a row, action and disinfection of V_2 occur according to the attacker's aim. Thus, the structure is just more complex but essentially the same.

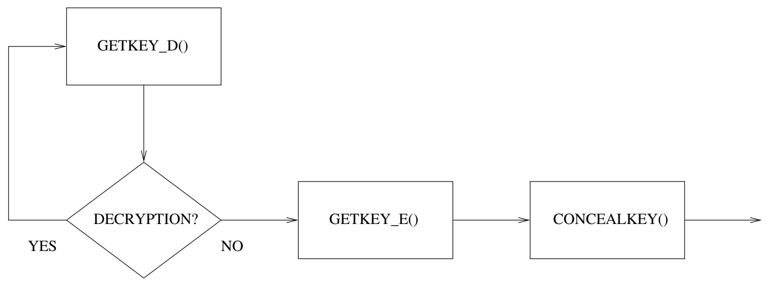

Figure 3: Virus V_2 organization chart (applied cryptanalysis part)

1. On the first round of deciphering (after V_2 infection process), routine **getkey_d()** catches the user's secret key and stores it somewhere on the hard disk (location is different from attack to attack; **getkey_d()** moreover includes a ciphering subroutine which encrypts the key before storage).

   For this part, structure of V_2 is very specific to the nature of cryptosystem binaries **S**. Indeed it must infect it in such a way that V_2 is able to temporarily pass control to **S** during its own action and before definitively abondoning control to it. Another very powerful way is to hook Windows API. Thus parameters (notably the secret keys themselves) of the different routines can be efficiently eavesdropped.

   During each next decryption process the caught key is compared to previously stored keys. When different, a counter is incremented and new key is stored in the same way. V_2 then returns control to cryptosystem executable **S**. Note that infection occurs in such a way that key catching takes place at a very low level in the cryptosystem binary code in order to always have access to the plain secret key, not an encrypted version of it. Macro viruses like Caligula [20] made the mistake of stealing the key in an encrypted form.

   Suitable executable infection and low level action is a far more efficient approach than simple keyboard buffer eavesdropping such as in (in so far information is available) Magic Lantern technology. The latter can easily be bypassed by not inserting secret keys via the keyboard (other possible solutions being the use of removable disk, "key gun" connected to a physical external port such as e-keys, smart cards,...).

2. At the first encryption process (after V_2 infection process; that is to say a ciphertext is about to be sent) a **getkey_e()** routine catches the key (by system call or API interceptions), compares it to the previously stored ones and if different, keeps it.

3. V_2 passes temporarily control to **S** for ciphertext generation.

4. Finally V_2 takes control again and a **concealkey()** routine enters into action. All the other stored keys are loaded and all the caught keys are encrypted with a different algorithm from that used by **getkey_d()**. They are finally hidden in the resulting ciphertext (either by insertion or by replacing ciphertext blocks). The position of the caught keys in ciphertext is computed from their own value in order to ensure random position from attack to attack. Once again it is important to note that action of V_2 for this concealment part takes place before any digital cryptographic signature of ciphertext.

5. V_2 returns definitively control to **S** after completing the action of **v1infect()** and **v2disinfect()** routines.

The attack is completed. Figure 3 summarizes this scheme.

Remark.- Note that this attack is easily adaptable to public key cryptosystems. The crucial point is to infect the executable in such a way that V_2 can access the unencrypted secret key during the computing. Only the key escape itself can be slightly different in some way.

Charlie then intercepts the ciphertext sent to ALice by Bob and extract all the secret keys that are hidden in it.

6 Protection Issues

When considering protection against these attacks, the game seems to be rather in the attacker's favour. As a general rule in computer security, computer protection is nearly almost condemned to reaction, at least for the very great majority of systems. The human ability to adapt and the unbounded imagination of the attackers make such attacks generally unlikely to be forecastable and thus prevented in advance.

From the general intrusion detection (ID) point of view, it is clear that things are very difficult. Securing computers when dealing with the huge "number of rampant vulnerabilities in commonly used software" [14] and with the persistent lack of user's public awareness of basic computer security (even in sensitive areas like Defence) it is like trying to square the circle.

To protect against the family of attacks we presented in this paper, what are the possible different levels of action ?

- The combined use of several, antivirus softwares, preferably based more on heuristics than on scanning (in so far as heuristics may be more efficient in revealing new viruses), seems to have very weak probability for detecting new, carefully written, victim specific, **low infecting** viruses implementing new polymorphic techniques. These latter will always have great chances to remain undetected at least a long enough time to ensure several key escapings before detection. M. Ludwig [13, pp 394–396] proved that there will always be viral programs that remain undetected (for techniques limiting antivirus efficiency see also [13, Part IV] and [7, Part III]).

Our different experiments have been proven effective at fooling and bypassing all recent antiviral protection[6] we used both for V_1 and V_2. We consider that state-of-the-art antivirus programming cannot afford more than hindsight, especially against unknown polymorphic capabilities. Moreover the reverse-engineering of AVs will always yield enough useful information on how they work and how to fool them.

- The use of digital cryptographic signature (for definition and details see [15, 19]) for the cryptogram, could be a solution but in fact in our case it was not. The use of combined viruses was precisely intended to bypass cryptogram cryptographic signature. V_2 executable is appended to the (cryptogram || signature σ)-file but before cryptographic signature checking, in Bob's computer virus V_1 launches V_2 and restores cryptogram integrity by deleting the element appended by the attacker. All these actions are implemented by common techniques in virus programming (interrupts rerouting, hooking API,...). We only optimized them to make them more powerful and undetectable. On the other hand, concealment of the captured key in the cryptogram always occurs before cryptographic signature. The intended recipient will detect nothing but a few lost plaintext blocks. Such a loss may naturally occur through error transmission and may not necessarily draw the attention of the user.

- However, on the other hand, our attack, as described in this paper may be very difficult to conduct in an IPSec [6] context. However IPSec security may possibly be defeated in the future [2], at least enough to allow such viral attack. Current developments of variants of our attack seem very promising at fooling IPSec limitations. Instead of trying to concatenate virus V_2 in ciphertext, other solutions are under current testing: use of e-mail attachment, infection of specific document format (Word documents, Postscript documents, PDF documents,...), ... The variant using e-mail attachment is very interesting, since the attack will be effective even before the file has been retrieved by the mail software in the victim computer. The fact that the mail is read or not, does not make any difference.

- This attack can be done even more passive by developing the V_1 structure and functionalities in order to render it able to "sniff" network traffic. Thus V_1 will "hear" and capture V_2 without sending it specifically to the target computer.

Currently we still have not found any satisfactory, generic solution to this problem but research is under way to build one.

7 Conclusion

This kind of attack have been proven dramatically efficient. Beyond the intrincic power of computer viruses (and especially the combined viruses), the clever use of cryptologic techniques considerably helped us to make our viruses very stealthy, polymorphic and compact.

[6]Even when integrity checking or code emulation are used as part of the AV techniques.

A large and interesting field of research is to be considered in the future when considering the links between viruses (and more generally malwares) and cryptology.

Acknowledgment

This paper is dedicated to Kim Clancy and Mark Ludwig.

References

[1] **http://www.nist.gov/aes**.

[2] S.M. Bellovin, Probable Plaintext Cryptanalysis of the IP Security Protocols. In: *Proceedings of the IEEE Symposium on Networks and Distributed Systems Security*, 1997.

[3] E. Biham, A. Shamir, Power Analysis of the Key-Scheduling of the AES Candidates. *Second AES Candidates Conference*, 1999. Available online at **http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm**

[4] T. Bridis, FBI Develops Eavesdropping Tools. *Washington Post*, November 22nd, 2001.

[5] **http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html**.

[6] N. Doraswamy, D. Harkins, *IPSec*. Prentice Hall, 1997.

[7] E. Filiol, Les virus informatiques : théorie, pratique et applications. To appear, Springer, Paris, 2002.

[8] GOST 28147-89, Cryptographic Protection for Data Processing Systems. Government Committee of the USSR for Standards, 1989.

[9] D.Kahn, *The Codebreakers: the Story of Secret Writings*. Macmillan, New York, 1967.

[10] P. Kocher, Timing Attack on Implementation of Diffie-Hellman, RSA, DSS and other Systems. In: *N. Koblitz (ed) Advances in Cryptology - Crypto'96*, Lecture Notes in Computer Science 1109, Springer, Berlin Heidelberg New York, pp 104-113, 1996.

[11] P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis. In: *M. Wiener (ed) Advances in Cryptology - Crypto'99*, Lecture Notes in Computer Science 1666, Springer, Berlin Heidelberg New York, pp 388-397, 1999.

[12] X. Lai, J.L. Massey, A Proposal for a New Block Encryption Standard. In: *Damgard IB (ed) Advances in Cryptology - Eurocrypt'90*, Lecture Note in Computer Science 473, Springer, Berlin Heidelberg New York, pp 389-404, 1991.

[13] M. Ludwig, *The Giant Black Book of Computer Viruses*, 2nd Edition. American Eagle Publication, Show Low, 1998.

[14] J. McHugh, Intrusion and Intrusion Detection. *Int. Journ. Inform. Sec.* 1:14-35, 2001.

[15] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, Boca Raton, New York, London, Tokyo, 1997.

[16] **http://www.cosic.esat.kuleuven.ac.be/nessie**

[17] **http://www.outguess.org/**

[18] B. Schneier, Description of New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In: *R. Anderson (ed) Fast Software Encryption Cambridge Security Workshop Proceedings*, Lecture Notes in Computer Science 809, Springer, Berlin Heidelberg New York, pp 191-204, 1994.

[19] B. Schneier, *Applied Cryptography* 2nd edition. Wiley, New York, Chischester, Brisbane, Toronto, Singapore, 1996.

[20] **http://www.sophos.com.**

[21] **http://www.cl.cam.ac.uk/ fapp2/steganography/index.html**

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399