World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:9, 2008

Another Formal Proposal For Stealth

Adrien Derock, DCNS/ESIEA and Pascal Veron, IMATH

***Abstract*—** Taking into account the link between the efficiency of a detector and the complexity of a stealth mechanism, we propose in this paper a new formalism for stealth using graph theory.

***Keywords*—**Detection, eradication, graph, rootkit, stealth.

I. Introduction

AMONG the most advanced antiviral functionalities, stealth is certainly one of the most sophisticated. Up to now rootkits constitute the most accomplished programs taking advantage of stealth techniques. Modeling these techniques is a difficult business. Nevertheless we can give the following definition:

Definition 1: The main goal of stealth is to steal a program (legitimate or not) from user or any legitimate security program supervision (software utilities or appliances).

From this definition, we can extract three elementary properties :

- Stealth applies to all programs (malicious or not);
- Stealth complexity relies on the user faculty. It seems obvious that a beginner user has not the same potential as a more experienced user. In the sequel of this paper we will generalize all possible detection means by the term *detector*;
- Lastly, stealth notion for all programs involves two main abilities:
  - Resist a detection process (inactive program). We will talk about *Camouflage*;
  - Resist at a detection process during program activity. We will name that *Total stealth*.

Definition 2: A detector is a resource that enables detecting modification in a system. This can be operated with presence system tools, or simply by an investigation made by the user, or even more by IT security software or hardware.

II. Previous works

Stealth formalism outlines have been made only for theory of computer viruses. Indeed stealth has always been considered as malicious technology. Contrary to code armouring and polymorphism, it does not exist to date rigid frame for stealth concept. The first formalization was shown through Adleman [1] works dating at 1988. In 2004, a quiet more accurate definition is given by Zuo and Zhou [2]. Lastly Eric Filiol has suggested a parallel between Steganography and Stealth [3]. Moreover he has given a classification based on statistical tests analysis[4]. At this stage and in order to understand better the content of this paper, it becomes necessary to introduce the mathematical tools used in the theory of computer viruses.

Adrien DEROCK is working in the information security department at DCNS, the french naval ship builder, Toulon, France, adrien.derock@dcnsgroup.com / He is also affiliated to Operational Cryptology and Virology lab, ESIEA, Laval, France

Pascal Veron is PhD at Toulon university, and researcher at IMATH laboratory, La Garde, France

A. Recursive functions

Founders works in the theory of computer system and programs are those of Alan Turing [1] and Fred Cohen[5]. The Turing machine still remains an important model for designing computer science. The principle consists of a program loaded on the machine. A data tape with input data is processing by this program through a read/write head. The output data is then written onto the data tape. Hence a Turing Machine can be assimilated to a function f which computes $f(x)$ with the input x.

Definition 3: Among all possible functions, those who can be computed by a Turing machine are called recursive functions.

Using Gödel numbering [6], it can be proved that the set of recursive functions is enumerable. Using a proper coding each Turing machine can be represented by an integer p. Thus, one can associate to each recursive function an index p which is the coding of the associated Turing machine. Such a function will be denoted as $\phi_p(d)$.

With this classical definition, one Turing machine can only compute one particular function. That's why Turing has introduced quiet fastly the universal Turing machine : a Turing machine which can emulate any other Turing machine. Such a machine is denoted as $\phi_{P_0}(x)$ where,

- ϕ_{P_0} represents the universal recursive function with the particular program P_0 (a kind of operating system);
- $x =< p_x, d_x >$ represents the program p_x which has to be applied to the input data d_x;

The result of $\phi_{P_0}(x)$ is equivalent to the non universal Turing machine $\phi_{p_x}(d_x)$.

If we consider P, the set of all programs, a virus modifying one program into another program is actually an application v of P in itself. Then, when the program i is infected, we obtain the program $v(i)$ and the associated recursive function $\phi_{v(i)}$. Knowing the fundamental principle of recursive functions, we can now take a closer look at the proposed formalization of stealth. These important concepts are well-detailed in [7].

B. Stealth through works from Adleman

According to Adleman, the form of an infected program can response by three possibilities depending on the input provided by the user:

- infection - The program after realizing the expected features, infects other programs (it reproduces);
- functionality-added - The program, in addition to its expected functions performs other actions. These actions

may be delayed or not, whether (usually) offensive or not and their nature only depends on the initial infection;
- imitation - The program makes no infection or offensive action but just performs instructions legitimately expected.

In his definition, the concept of stealth appears in the third possibility. Indeed the imitation is the main goal of stealth. Detectors must notice a normal behaviour for the system. In terms of recursive function, imitation can be considered by the following relationship:

$$\forall i \in \mathbb{N},\ \phi_{v(i)} = \phi_i$$

The infected program $v(i)$ has exactly the same behaviour compared to the original program. However, details of this imitation are not available in this relationship.

C. Stealth by Zuo and Zhou

Zuo et Zhou [2] have provided an appreciable amount of very interesting results on advanced computer viruses complexity. As regards stealth, and according to Zuo and Zhou, an execution of any program depends on two parameters : *data* and *execution context* (environment). Hence, the corresponding function ϕ depends as well on two parameters d and p:

$$d = (d_1, \cdots, d_n), \quad p = (p_1, \cdots, p_n)$$

For instance p_1 is the operating system, p_2 the text editor and so on ... Authors' aim was to present a particularly interesting granularity at resources level. A program is not self-sufficient to carry out its task. It appeals to system resources, hence the importance of execution context. This granularity enables a better modelling of stealth within a system.

1) Formal framework: Here is the proposal from Zuo and Zhou for formalizing stealth viruses. In its incubation stage, a stealth virus makes the infected program perform the same actions than the healthy one.

$$\phi_{v(i)}(d, p) = \phi_i(d, p)$$

A stealth virus when it is replicating has to select a program in order to infect that last one. He must alter the associated system call to delude detectors.

$$\phi_{v(i)}(d, p) = \phi_i(d, p[v(S(p)), h(\mathrm{sys})])$$

S means the selection function in charge of determining the target object to proceed the infection. Thus $p[v(S(p))]$ means the execution context in which an element p_s of the set p, chosen by the function S, has been infected. $h(sys)$ refers to the modification of the appropriate system call so that the element p_s will not be identified as infected. For instance, this system call can be one in charge of consulting the file system at the requested place. By replying the suitable answer, the system will appear safe. The behaviour of the system call is explained below :

$$\phi_{h(\mathrm{sys})}(x) = \phi_{\mathrm{sys}}(y), \quad \text{if } x = v(y)$$

Otherwise, the system call behaviour is normal:

$$\phi_{h(\mathrm{sys})}(x) = \phi_{\mathrm{sys}}(x)$$

Zuo and Zhou's formalization gives a good definition of the notion of stealth viruses. Here is the complete version:

$$\phi_{v(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[v(S(p)), h(\mathrm{sys})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{else.} \end{cases}$$

and

$$\phi_{h(\mathrm{sys})}(x) = \begin{cases} \phi_{\mathrm{sys}}(y) & \text{if } x = v(y) \\ \phi_{\mathrm{sys}}(x) & \text{else.} \end{cases}$$

where

- $I(d, p)$ is a boolean value which determines if the infection must be proceeded;
- $T(d, p)$ is a boolean value which determines if the code $D(d, p)$ of the virus must be launched.

D. Stealth according to Filiol

Into this general lack of modelization, Filiol has proposed to establish a parallel between steganography and stealth[3]. The two concepts have the same goal, which is to hide information. Lot of works have been done to formalize steganography. Especially, formalism of Cachin[8], [9] can be stressed on in order to formalize stealth.

Definition 4: (Steganography and steganalysis) The steganography is the set of techniques which not only enable the security of the information - COMSEC (COMmunication SECurity) aspect - but also and above all the security of the (information) transmission channel - TRANSEC (TRANSmission SECurity) aspect. The steganalysis is the set of detection techniques whose purposes is to detect the use of steganography and to access the hidden information.

This definition gives an obvious parallel between stealth and steganography:

- the COMSEC aspect is related to the malware itself (its code and its actions);
- the TRANSEC aspect relates to the malware execution and interactions with the targeted system.

For more details on the link between stealth and steganography, the reader can refer to [3]. E. Filiol also used statistical tests simulability to model [4] the problem of detecting stealth mechanisms. The interest of testing simulability depend on the fact that stealth can be defined as the capacity of defeating detection and remaining undetected by using testing simulability. In other words, stealth consists in simulating the distribution D_{Sys} of a clean system. A strong consequence is that if such non detection successfully occurs (up to the statistical risks) it is only possible with respect to a given testing and a given estimator. But it is intuitively impossible to design stealth techniques in order to simulate all possible testing and taking the infinite set E of all possible estimators. Current research aims at proving this claim on a mathematical basis.

III. New formal proposal

We detail in this part a new formal definition for stealth. We want to take into account all the aspects of the problem, by considering all the works ever done and the latest techniques which have changed the deal. To this end, as compared to the previous models, the detector notion has been fully integrated in our formal proposal. Indeed, stealth is in the balance of power with the environment in which it is practicing its faculties. Moreover, we will suggest a deeper granularity of the system closer to the reality. Finally, stealth's category used, camouflage or total stealth, must appear in our formalism.

A. System's Diagram

The overall diagram is based on several components definition. Traditionally, a computer system SYS is considered as a stack of layers. Each layer is linked up to the next layer down until the last one (the hardware layer). Deeper is the layer, higher is the privilege. The privilege is a notion much spreading. That concept allows to establish a program hierarchy. The main idea is to separate applications according to their importance, their criticality.

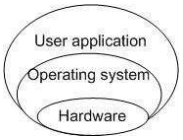

Fig. 1. General diagram of a system

This diagram is no longer enough to represent a system that has evolved considerably. Indeed, with the emergence of virtual machines, an additional layer must be added. The hypervisor[1] layer has to be considered[10]. The figure 2 show us that layer.

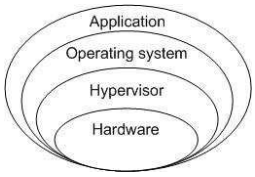

Fig. 2. Evolved diagram of a system

Thanks to the precedent diagram, we can introduce the following concepts : a system SYS is a set with layers L_i. These layers are contiguous and can communicate with one another. That relation between the layer is represented by ILC, that is to say *Inter Layer Communication*. We can classify all those layers in several categories by considering the privilege level used in the layer. This classification depends on privilege class in which we can extract four main classes:

- $Priv_{\text{User}}$: user space privilege. A user application has all its instruction executed in this class of privilege;
- $Priv_{\text{Kernel}}$: kernel space privilege. Most device drivers and operating system services need this privilege;
- $Priv_{\text{Hypervisor}}$: hypervisor level privilege. Virtual machine monitors have program executed with this privilege;
- $Priv_{\text{Hardware}}$: Hardware layer. Instruction dealing with the hardware use this privilege class.

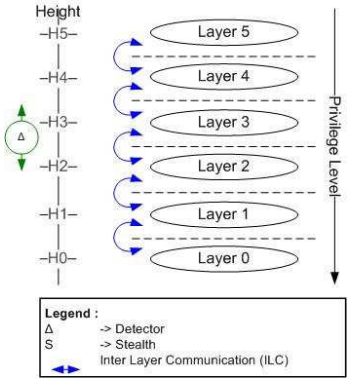

Fig. 3. Model of the system

In order to reduce the complexity of our model, we consider a low granularity of the system. More the system's granularity considered is complex, bigger is the number of layers. By considering the granularity level, a layer can be divided in several layers or many layers can be regrouped in only one. Anyway, fundamentals of our model are not impacted by this notion of granularity. The model can be illustrated with an oriented graph (see figure 4). Since there is a bidirectional link

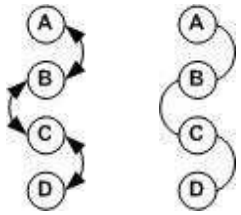

Fig. 4. An oriented graph and a non oriented graph

between each layer, we simplify the graph by considering a non-oriented graph (see figure 4). Thus, this illustration can be modelled using graph theory.

Definition 5: A non oriented graph G is a couple (P, E) in which P is the set of all vertices. E represents an element from $(P \times P)$. The number of vertices defines what is called the order of the graph (the cardinality of P).

The reader can find more details about graph theory in [11]. We defined as a simple graph, the one illustrated in figure 4. In a deeper granularity, each graph vertex can reveal a tree (see figure 5).

Definition 6: A tree is a graph with no cycles such that any two vertices are always connected by exactly one path.

In the graph we consider, all vertices are numbered by considering the appropriate layer range. The first vertex (smallest number) is graph's root. Actually, the hardware layer is numbered with 0. An integer $n > 0$ is affected to the user

[1] virtual monitor or hardware virtualization are some technologies targeted by hypervisor concept

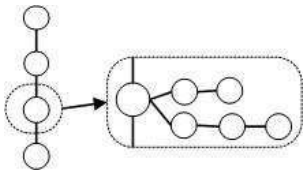

Fig. 5. Zoom on the graph in relation to system granularity

application layer and all layers between are numbered with an integer between 1 and $n-1$. Each vertex owns one and only one successor and predecessor.

Definition 7: The graph's height H is the minimum number of vertices of a path needed to reach one vertex from the root. We define also the vertex degree d by considering the number of adjacent vertices.

We have delibaretely used the label P as system layers set. That label reminds us the environment in which recursive function are executed in Adleman's and Zuo and Zhou's formalism.

B. Stealth mechanisms in our diagram

According to the diagram proposed to model the system, how is included the stealth mechanism named $Stealth$? $Stealth$ is considered like a reflector, or even more a mirror which always sends an image of a safe system despite its infection. In case of camouflage, the environment P is modified so that the detector used cannot be warned (COMsec in steganography). In case of total stealth, data D and communications E are even more modified. (TRANsec in steganography).

The mirror can be positioned between two layers, for instance L_3 and L_4 or even more replace completely one or several layers (see figure 6). All the layer placed above, (L_5) will see a normal image of the part of the system under the mirror. To achieve that, Inter layer Communication canal (ILC_{3-4}) is intercepted and modified. The modification is done in the set E of communications. Once the mirror installed, lots of customizations can be done. These modifications are done in the set of layers, P. The position of the mirror notices us about the kind of $Stealth$ used which depends on the class of the layer affected :

- $Stealth_{\text{user}}$ for user space stealth mechanisms;
- $Stealth_{\text{kernel}}$ for kernel space stealth mechanisms;
- $Stealth_{\text{hypervisor}}$ for hypervisor space stealth mechanisms;
- $Stealth_{\text{hardware}}$ for hardware space stealth mechanisms.

Remark: here is given a taxonomy of stealth mechanisms. This taxonomy was already proposed by several authors[12] (often without hypervisor aspect) from a technical point of view whereas, our classification depends on our formal model.

We can now introduce our formal proposal. A stealth mechanism belongs to the set

$$\{Stealth_{\text{user}}, Stealth_{\text{kernel}}, Stealth_{\text{hypervisor}}, Stealth_{\text{hardware}}\}$$

and will be denoted as $\phi_{\text{Stealth}(i)}$. After the stealth mechanism installation, we obtain a new environment and modified data

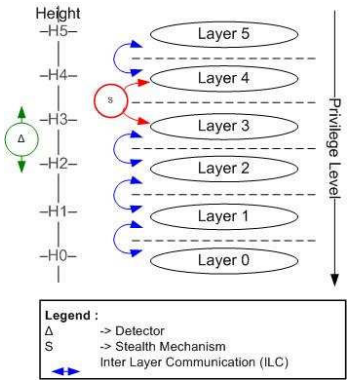

Fig. 6. Model of the system with Stealth

and we named Ω and Λ the functions associated to these changes. We obtain then the following relations:

$$\phi_{\text{Stealth}(i)}(\Omega(d), \Lambda(p)) = \phi_i(d, p) \text{ (Total stealth)} \quad (1)$$

$$\phi_{\text{Stealth}(i)}(d, \Lambda(p)) = \phi_i(d, p) \text{ (Camouflage)} \quad (2)$$

This formalism distinguishes the two aspects of stealth. However, stealth depends strongly on detector features. Let's take an interest for the problem of detector position with regard to stealth position.

C. Stealth vs detector

It has been mentioned above, that an important relation links up stealth and detector. We introduce thus, relative stealth concept. Absolute stealth exists only if the infinite set of detectors is simulated by stealth mechanism. But this is theoretically impossible. So contrary to what is said by some, it does not exist and it will never exist absolute stealth.

1) Detector position: A basic condition on the detector Δ is necessary to complete the formalism. That condition depends on the position of the detector. If the detector is underneath stealth mechanism, modifications provided to the system are visible.

$$\phi_{\text{Stealth}(i)}(\Omega(d), \Lambda(p)) = \phi_i(\Omega(d), \Lambda(p)), \text{ if } H(\text{Stealth}) > H(\Delta) \quad (3)$$

Otherwise, the detector sees nothing abnormal.

$$\phi_{\text{Stealth}(i)}(\Omega(d), \Lambda(p)) = \phi_i(d, p), \text{ if } H(\text{Stealth}) < H(\Delta) \quad (4)$$

Equality between Stealth level and detector level will remain unclear.

$$H(\text{Stealth}) = H(\Delta) \quad (5)$$

A naive solution can be given to avoid this case. The solution would consist to put the detector even lower down to allow modifications' detection. Nevertheless we have to take the following relation into account:

$$H(\text{Stealth}) = H(\Delta) = 0 \quad (6)$$

This case shows us how blurred is the equality relation above. How will we do then ? We will deal about this case later.

2) Detector's sensitivity and efficiency: Detector's position is necessary but not sufficient. If the detector is underneath $Stealth$, and if this last one is designed to deceive anyway the detector, then our proposition above has to be completed. So, we have to take into account the ability of $Stealth$ to deceive the detector [4]. Detector can response by different ways, however we can distinguish two functional means :

- *On access detection* : entity listening activity,
- *On demand detection* : entity which will scan the whole system.

In order to model the ability of the $Stealth$ mechanism to deceive the detector, we will introduce and re-use notions detailed by Filiol on [4] and [3]. These notions call up concept like indistinguishability of two population distributions. The distributions which are built thanks to detector are submitted to algorithms which decide if they are identical or not. More accurate is the detector's distribution extraction process, more the decision will be sure. For instance, let consider two almost identical photos:

- One taken with 2 Megapixels camera;
- The other one taken with 10 Megapixels camera;

If we decide to compare the two photos with a screen or an impression with a resolution below 2 megapixels, we will rule on that they are the same photos. At the contrary, with an upper resolution, the two photos will be revealed completely different. All depend so on sensitivity of the detector. We then distinguish two cases of simulability :

- Strong simulability: $Stealth$ is fixed at a lower down level face to detector. This case has already been treated.
- Weak simulability: $Stealth$ is positioned above the detector. Despite, it simulates all answers in order to produce a statistical distribution $D^{\Delta}_{\text{Stealth}}$ indistinguishable with regard to system distribution D^{Δ}_{SYS}. This case is formalized below.

Remark : Notice that the system distribution depends on the detector which generates it. That's why it is denoted as D^{Δ}_{SYS}. The following relation must be so considered (the relation is true for any integer n big enough and for any polynomial P):

$$\sum_{\text{x}\in\text{answers}} \left| Prob_{D^{\Delta}_{\text{Stealth}}}[x] - Prob_{D^{\Delta}_{\text{SYS}}}[x] \right| \leq \frac{1}{P(n)} \quad (7)$$

- $Prob_Q[x]$ is the probability to obtain the event x according to a given distribution law Q.
- n represents the security coefficient[4].

We can now introduce the computational indistinguishability. That is to say, it exists a polynomial algorithm that can distinguish two given distributions. Therefore we have, for any probabilistic polynomial algorithm *A*,

$$\left| Prob_{x\leftarrow D^{\Delta}_{\text{Stealth}}}(A(x)=1) - Prob_{x\leftarrow D^{\Delta}_{\text{SYS}}}(A(x)=1) \right| \leq \epsilon \quad (8)$$

Where $x \leftarrow D$ means that x is chosen with distribution D. The relation gives us the notion of computational indistinguishability for two distributions and with regard to a detector. We remark that the two aspects of indistinguishability are relative to the detector . We name by:

- $IND_S(D^{\Delta}_{\text{stealth}}, D^{\Delta}_{\text{SYS}})$ the predicate which is true if both distributions satisfy the relation 7. Then we will say that they are statistically indistinguishable.
- $IND_C(D^{\Delta}_{\text{stealth}}, D^{\Delta}_{\text{SYS}})$ the predicate which is true if both distributions satisfy the relation 8. Then we will say that they are computationally indistinguishable.

3) General Formalism of stealth: The previous formalism is so enriched and leads to this new definitions :

. Total Stealth

$$\phi_{\text{Stealth}(i)}(\Omega(d), \Lambda(p)) =$$

$$\begin{cases} \phi_i(d,p), & \text{if } H(\text{Stealth}) < H(\Delta) \\ \phi_i(d,p), & \text{if } H(\text{Stealth}) > H(\Delta) \\ & \text{and } IND_{C_{or}S}(D^{\Delta}_{\text{Stealth}}, D^{\Delta}_{\text{SYS}}) \\ \phi_i(\Omega(d), \Lambda(p)), & \text{if } H(\text{Stealth}) > H(\Delta) \\ & \text{and } \neg IND_{C_{or}S}(D^{\Delta}_{\text{Stealth}}, D^{\Delta}_{\text{SYS}}) \end{cases} \quad (9)$$

. Camouflage

$$\phi_{\text{Stealth}(i)}(d, \Lambda(p)) =$$

$$\begin{cases} \phi_i(d,p), & \text{if } H(\text{Stealth}) < H(\bar{\Delta}) \\ \phi_i(d,p), & \text{if } H(\text{Stealth}) > H(\bar{\Delta}) \\ & \text{and } IND_{C_{or}S}(D^{\Delta}_{\text{Stealth}}, D^{\Delta}_{\text{SYS}}) \\ \phi_i(d, \Lambda(p)), & \text{if } H(\text{Stealth}) > H(\bar{\Delta}) \\ & \text{and } \neg IND_{C_{or}S}(D^{\Delta}_{\text{Stealth}}, D^{\Delta}_{\text{SYS}}) \end{cases} \quad (10)$$

$\bar{\Delta}$ represents an on demand detector. Indeed, this kind of detection is essential for this category of Stealth [3].

D. Resolution of Particular cases

In section III-C, the model is undefined for the following particular case $H(\text{Stealth}) = H(\Delta)$. Worse, it becomes unclear when $H(\text{Stealth}) = H(\Delta) = 0$. We can give a solution to this problem for particular detectors. Let us denote by Δ_{Stealth} a stealth detector, that is to say a detector which hides its distribution $D^{\Delta_{\text{Stealth}}}_{\text{SYS}}$ of the system. Then it is theoritically impossible for an aggressor to build a stealth code generating a distribution indistinguishable from the true distribution hidden by Δ_{Stealth}. That is to say that $IND_{C \text{ or } S}(D^{\Delta_{\text{Stealth}}}_{\text{Stealth}}, D^{\Delta_{\text{Stealth}}}_{\text{SYS}})$ is always false. As a result, when $H(\text{Stealth}) = H(\Delta_{\text{Stealth}})$, then from equation 6, we have

$$\phi_{\text{Stealth}(i)}(\Omega(d), \Lambda(p)) = \phi_i(\Omega(d), \Lambda(p)).$$

E. Detection and eradication of stealth mechanisms

Thanks to previous formalism and the model proposed with graph theory, the problem of detecting stealth mechanisms is more affordable[2]. The detection process has been discussed many times [12], [13], [4]. Our approach is based on the formalism proposed above. So Total stealth modifies both edges (the nature of communications) and vertices of the graph while the camouflage does not alter those last ones. The presence of a stealth mechanism refers to the presence

[2]we consider from here the detection of total stealth.

of an additional vertex or modified one and/or the presence of additional edges or modifications upon ones which are adjacent. The degree of granularity of the graph considered will affect the nature of the change. The various cases to be taken into account are summarized below.

Case 1: Vertex modified (the stealth mechanism has been integrated on the system).

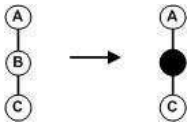

Fig. 7. Modification of vertices

Case 2: Vertex added (the stealth mechanism has been inserted on the system).

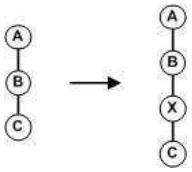

Fig. 8. Add of vertices

Case 3: modification of paths .

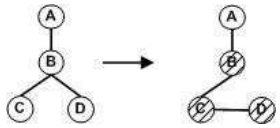

Fig. 9. Modification of paths

Case 4: modification of edges.

Fig. 10. Modification of edges

1) Stealth mechanism detection: Let $G = (P, E)$ be a graph, we denote by:

- Σ_P an application from P to $\{0,1\}^n$ (vertices valuation).
- Π_E an application from $P \times P$ to $\{0,1\}^n$ (edges valuation).

Whenever we're talking about modifications, we're always talking about a comparison between a current state and a reference state. The comparison requires to save the reference state or at least its main features. Making a backup of this reference state of the system SYS consist of making a backup of the associated graph with vertices valuation and edges valuation.

We have then to take into account the two following graphs:

- The reference graph G_R with the valuation functions associated to vertices and edges, Σ_{P_R} and Π_{E_R};
- The current graph G_C with its valuation functions associated to vertices and edges, Σ_{P_C} and Π_{E_C}.

Remark: Considering that all graphs involved by our model are almost trees, they have at most $Card(P) - 1$ edges. Thus the number of valuation to be saved is linear in relation to vertices number.

Before any detection process, the current graph is created, all degree of each vertices are extracted and all valuation functions are calculated in advance. This stage corresponds in our $Stealth$ formalism to the extraction of the current system's distribution and is done using the adjacency matrix of the graph. This calculation needs, considering this $n \times n$ matrix :

- n calls to valuation functions (edges and vertices);
- n^2 additions for the calculation of vertices' degree;
- n additions for the calculation of the graph's cardinal.

The complexity of the extraction stage relies on graph building's cost and each valuation function's cost. Next to the extraction stage, there is the detection stage and the eradication stage.

a) Detection in the case of the first figure: In this case, $P_R = P_C = P$. The figure 7 represents a vertex modification in the graph. Due to this modification, the valuation of this vertex is changed. We obtain then the following relation :

$$\exists p \in P \mid \Sigma_{P_R}(p) \neq \Sigma_{P_C}(p)$$

For instance, this valuation can be calculated by a hash function applied to the data contained in a vertex.
Complexity: this detection needs at most $Card(P_R)$ comparisons.

b) Detection in the case of the second figure: The figure 8 represents the case where a vertex is added in the graph. The cardinality of the graph is so mandatory altered. The following relation is then true: $Card(G_R) \neq Card(G_C)$. Complexity : this detection needs only one comparison.

c) Detection in the third figure: The figure 9 show a modification of a path toward a vertex. That is possible especially by an edge removed and another one added. That is to say that some vertex will have their degree changed. The detection comes down to compare degrees of the vertices of G_C with ones of the graph G_R.
Remark : this detection can be made also by comparing edges and vertices valuations, what is more costly.
Complexity: n comparisons.

d) Detection in the fourth figure: The figure 10 underlined a non legitimate use of a communication channel in the system. A vertex functionality must have been hijacked to enable another kind of communication than ones allowed. We are in the case in which $G_R = G_C$ and

$$\exists e \in E | \Pi_{G_R}(e) \neq \Pi_{G_C}(e)$$

Remark: This modification should modify all adjacents vertices. Complexity: this detection needs at most $n - 1$ calls to the vertex valuation function. That is, so a linear detection.

2) Stealth mechanism eradication: The detection has been demonstrated as an affordable problem with our model. At the contrary eradication is said to be more complicated [14]. However the difficulty of the eradication problem remains also

on the extraction of a relevance reference graph.
So, we have to keep in mind that stealth is not commonly content with adding data on a system. Indeed the system is affected and by deleting the data corresponding to the stealth mechanism, the system will be damaged. That involves a strong constraint, which is the obligation to rebuild the system. A reconstruction of the system implies a full knowledge of this one. In our model this knowledge is acquired in the reference graph. The extraction of this graph is the main point of the eradication process. Once this reference graph well-extracted the eradication process turn into an elementary principle.
The graph of the current system have to be extracted in accordance with the reference graph extraction process. Then all common parts of both graphs have to be identified.

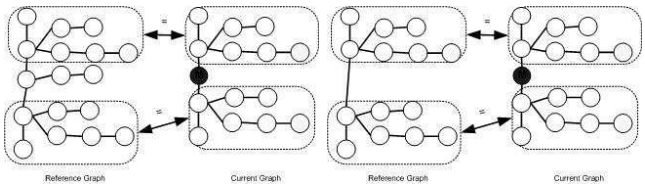

Fig. 11. Eradication of stealth mechanism

Once equal subgraphs discovered, that is to say the good parts of the system identified, the rest of the graph has to be replaced by the good one form of the reference graph. In the figure 11 we can see two examples of eradication process. The first one show us the detection of vertices modifications and the second one, a detection of vertex added. Once the equal parts detected, elements needed to be replaced are in evidence. In [12], this concept is well explicited. The antirootkit tool, pathfinder is designed like this: the vertex of the reference graph are valuated with instructions execution number (see figure 12).

```
linux:/home/tools/rktest # patchfinder -c
referenz_2.4.16
* FIFO scheduling policy has been set.
* each test will take 1000 iteration
* testing... done.
* dropping realtime schedulng policy.
test name | current | clear | diff | status
-----------------------------------
open_file | 7110| 1442| 5668| ALERT!
stat_file | 7050| 1255| 5795| ALERT!
read_file | 608| 608| 0| ok
open_kmem | 7124| 1510| 5614| ALERT!
readdir_root | 6497| 2750| 3747| ALERT!
readdir_proc | 14422| 2401| 12021| ALERT!
read_proc_net_tcp | 11750| 11750| 0| ok
lseek_kmem | 220| 220| 0| ok
read_kmem | 327| 327| 0| ok
```

Fig. 12. Example of information given by the tool Patchfinder

The same approach can be made by considering the size of functions in bytes.

IV. Conclusion

A work of general formalism was proposed in this paper. This formalism allows to frame all the aspects of stealth in the programs and evenmore in systems. The definitions which were able to be raised give us a better vision of the advantage and the inconveniences of these mechanisms which can be both used in a friendly and hostile way. The consideration of the works already made, the relativity of stealth mechanisms with the detector and the system, the use of graph theory, allow to propose an innovative formal frame. It leads quite naturally to rethink the way of protecting the system, notably by using the mechanisms of stealth in order to fight the hostile codes. We shall conclude the debate on stealth by supporting ardently that any concept of protection using it will show itself more effective than any classical mechanism of protection.

References

[1] L. Adleman, "An abstract theory of computer viruses," in *Proceedings on advances in cryptology, Crypto'88*. Springer-Verlag, 1990, pp. 354–374.
[2] Z. Zuo and M.-t. Zhou, "Some further theoretical results about computer viruses," *The Computer Journal*, vol. 47, no. 6, pp. 627–633, 2004.
[3] E. Filiol, *Techniques virales avancées*. Springer Verlag France, the english version is pending (due January 2009) under the reference Advanced Computer Viruses techniques, IRIS International Series, Springer Verlag France, 2007.
[4] ——, "Formal model proposal for (malware) program stealth," in *Proceedings of Virus Bulletin Conference, VB2007*, 2007.
[5] F. Cohen, "Computer viruses: Theory and experiments," *Computers & Security*, vol. 6, no. 1, pp. 22–35, 1987.
[6] K. Godel, "Über formal unenscheidbare sätze der principia mathematica und verwandter systeme," *Monatshefte fr Math. Phys*, vol. 37, 1931.
[7] E. Filiol, *Computer Viruses : from theory to applications*. IRIS International Series, 2nd, Springer Verlag France, 2003.
[8] C. Cachin, "An information-theoretic model for steganography," *Inf. Comput.*, vol. 192, no. 1, pp. 41–56, Mar. 2004.
[9] ——, "Digital steganography," *Encyclopedia of Cryptography and Security*, Feb. 2005.
[10] M. Myers and S. Youndt, "An introduction to hardware-assisted virtual machine (hvm) rootkits," Aug. 2007. [Online]. Available: http://crucialsecurity.com/
[11] C. Berge, *Théorie des graphes et ses applications*. Dunod, 1958.
[12] A. Bunten, "Unix and linux based rootkits techniques and countermeasures," Apr. 2004.
[13] K. kasslin, M. Stlahberg, S. Larvala, and A. Tikkanen, "Hide'n seek revisited - full stealth is back," 2005.
[14] E. Filiol, "Les virus du futur(s)," Laboratoire de virologie et de cryptologie, ESAT, Rennes, France, Oct. 2007.