Formal Model Proposal for (Malware) Program Stealth

Eric Filiol
efiliol@esat.terre.defense.gouv.fr

Army Signals Academy
Cryptology and Virology Lab
Rennes

Virus Bulletin 2007

Plan

Introduction

Definition

Stealth is the ability for a program to operate and remain undected within a system. Rootkits are conceptually just sets of stealth techniques.

- Stealth is not a new approach (*Stealth* virus - 1991).
- Two classes of techniques :
  - Classic techniques (Hoglund - 2005).
  - Virtualization-based techniques (*SubVirt*, *BluePill* - 2006).
- The critical issue is :
  - "how easy or difficult it is to detect what is supposed (or claimed) to remain undetected ?"
  - "what does detecting stealth mean ?"

Introduction (2)

- There exist only a few attempts to formalize stealth (Zuo & Zhou, 2004).
  - Use of recursive functions (Zuo & Zhou, 2004).
  - Detection of some classes of stealth techniques has a huge complexity (NP^{NP}-complete or higher ; Zuo & Zhou, 2004).
- Detection is generally (falsely) considered as a technical problem only.
  - Security policy must be prevalent over technical considerations.
  - The aim is to determine whether a system has been compromised or not.

Introduction (3)

Some other aspects must be considered :

- Computability issue : some problems have no solution at all.
  - Malware detection is undecidable (Cohen - 1986).
- Complexity issue : solving some problems is too time- or memory-consuming.
  - Detection of polymorphism is NP-complete (Spinellis - 2003).
  - Sequence-based detection of metamorphism is undecidable (Filiol ; Borello, Filiol, Mé - 2007).
- Can a (stealth) program still remain undetectable once its code/concept has been disclosed or analysed ?
  - The *BluePill* case (Rutkowska vs AV Community) !

Summary of the talk

Plan

Steganography and steganalysis

Definition

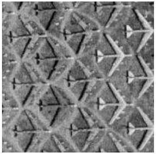
The steganography *is the set of techniques which not only enables the security of the information – COMSEC (COMmunication SECurity) aspect – but also and above all the security of the (information) tranmission channel – TRANSEC (TRANSmission SECurity) aspect. The* steganalysis *is the set of detection techniques whose purposes is to detect the use of steganography and to access the hidden information.*

- Obvious parallel between steganography and stealth :
  - COMSEC is related to the malware to hide.
  - TRANSEC is related to the malware execution and its interactions with the target system.

Example : Image Steganography

Covertext 

$\Longrightarrow$ Stegotext 

Secret message 

Malware Stealth

Statistical aspects of steganography

- We consider statistical models for both Covertext ($\mathcal{P}_\mathcal{C}$) and Stegotext ($\mathcal{P}_\mathcal{S}$) populations, with respect to some estimator E.
- Hiding a secret message into a covertext results in statistical modifications with respect to E.
- Detection is based on the behaviour of E according to either $\mathcal{P}_\mathcal{C}$ or $\mathcal{P}_\mathcal{S}$.

Application to stealth

Just find two different population distributions and one or more suitable estimators.

- *DSys* is the distribution of all possible files, structures and processes of a system that can be used as coversystem.

Important Remark

DSys *can refer to a virtual but not infected system !*

- *DStealth* is the distribution of files, structures and processes that have been effectively used with respect to a given stealth technique.
- Let us denote $\mathcal{P}_Q(x)$ the probability of x with respect to the distribution Q.

Stealth security

Definition

A stealth system is said to to ϵ-secure against a passive attack if and only if

$$D(P_{\text{DSys}}||P_{\text{DStealth}}) = \sum_{x \in Q} P_{\text{DSys}}(x) \log \left(\frac{P_{\text{DSys}}(x)}{P_{\text{DStealth}}(x)} \right) \leq \epsilon.$$

where Q denotes the space of possible measurements. If $\epsilon = 0$ then the stealth system is said to be perfectly secure.

- Consider the relative entropy $D(P_{DSys}||P_{DStealth})$ between $DSys$ and $DStealth$.
- We have $\epsilon = 0$ whenever $DSys$ and $DStealth$ are identical.

Stealth classification

According to the value of ϵ, we have three possible classes of stealth security :

- *Unconditionally secure stealth* ($\epsilon = 0$)
    - Detection is not possible even with unlimited time and computing resources.
- *Statistically secure stealth* ($\epsilon = \mathcal{O}(\frac{1}{n})$ for some arbitrary n).
    - The adversary is an arbitrary unbounded algorithm (time and computing).
- *Computationally secure stealth* ($\epsilon = \mathcal{O}(\frac{1}{n})$).
    - The adversary is an arbitrarily probabilistic, polynomial-time algorithm.
- *Unsecure stealth* (ϵ is a constant).
    - The adversary is a deterministic polynomial time algorithm.
    - Consider it as a trivial subset of the previous class.

Stealth classification (2)

What about virtualisation-based techniques (aka *SubVirt* and *BluePill*) ?

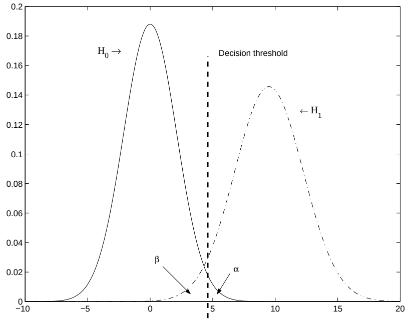
- Rootkit activity is bound to modify some estimators (to be defined).
- According to information theory, security cannot rely on the system secrecy only.
  - Security must consider some secret parameter, e.g. cryptographic key (Kerckhoff's laws).

Plan

1. Introduction
2. Formal model of Stealth
   - Steganography and steganalysis
   - Modeling Stealth
3. Formal Model for Stealth Detection
4. Practical Aspects of Stealth Detection
5. Conclusion et future work

Statistical model of detection

Any antiviral detection can be modeled as one or more statistical testings (Filiol, Josse 2007).



- The *null hypothesis* $\mathcal{H}_0$ refers to *DSys* while the *alternative hypothesis* $\mathcal{H}_1$ refers to *DStealth*.
- False positive (α) and non detection (β) probabilities are never null and are opposite.
- Whenever the code is disclosed or known, $\mathcal{H}_1$ is always known to the analyst !

Formal model of Stealth Detection

- Choose an estimator and define $(\mathcal{H}_0, \mathcal{H}_1)$.
- Compute

$$\Delta(\alpha, \beta) = \alpha \log \left(\frac{\alpha}{1 + \beta} \right) + (1 - \alpha) \log \left(\frac{1 - \alpha}{\beta} \right).$$

Theorem

In stealth system that is ϵ-secure against passive detection, the non detection probability β and the false positive probability α satisfy

$$\Delta(\alpha, \beta) \leq \epsilon.$$

If *DSys* and *DStealth* are equal then $\Delta(\alpha, \beta) = 0$ (class of unconditionally secure stealth).

A new definition of Stealth

A worse situation : the attacker (e.g. a rootkit) uses detection techniques against the defender.

- He performs *statistical testing simulability*.

Definition

Simulating a statistical testing consists for an adversary, to introduce, in a given population $\mathcal{P}$, a statistical bias that cannot be detected by an analyst by means of this testing.

- Strong simulability (just design a new, unknown technique not managed by the existing testings).
- Weak simulability (make *DStealth* looks like to *DSys*).

Consequences

Consider a known malware (code/concept has been disclosed).

- Stealth model is equivalent to the ability to remain undetected by using testing simulability (simulating *DSys*).
- This is possible with respect to known estimators E only.
- It is intuitively impossible to simulate $\mathcal{E}$ (the infinite set of all possible estimators E).
  - Mathematical proof soon published.
- A rootkit cannot simulate (e.g. defeat) some “secret” estimator (in particular $\mathcal{H}_0$ is unknown to it).

Conjecture

The class of unconditionnally secure stealth techniques can be defined with respect to known detection techniques only.

Consequences (2)

Absolute stealth (or definitively undetectable stealth) does not exist !

- Just reverse the sword against shield battle.
- The rootkit writer cannot forecast all the detection estimators that an antivirus analyst may imagine !
- All the antivirus expert's work consists in finding an efficient enough estimator.
  - **Good news : from the theoretical model, such estimators** ALWAYS **exist !**

Plan

1. Introduction
2. Formal model of Stealth
   - Steganography and steganalysis
   - Modeling Stealth
3. Formal Model for Stealth Detection
4. Practical Aspects of Stealth Detection
5. Conclusion et future work

General principles

The aim is to detect :

- either the activity of some (unusual) virtual system (while none is usually used),
    - ⇒ *DSys* will model a clean, physical system.
- or detect an usual activity within a virtual system.
    - ⇒ *DSys* will model a clean, virtual system.

⇒ We have to find one or more suitable estimators.

Detecting virtualisation

A few recent work have addressed this issue :

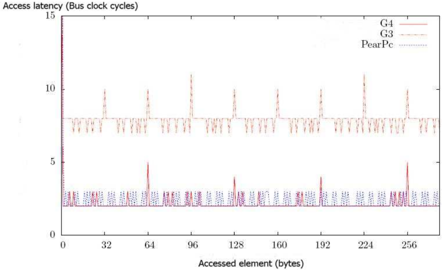
- Execution Path Analysis (Rutkowski, 2002).
- RedPill (Rutkowska, 2005).
- Transparent VMMs (Garfinkel, Adams, Warfield, Franklin, 2007).
- Samsara (Lawson - Ferie - Ptasek, 2007).

While being very interesting, no formal proof has given up to now.

Detecting virtualisation : C. Lauradoux's work (2007)

Measure the access time to array elements. Take the periodic anomalies with respect to the processor cache memory as a detection estimator.

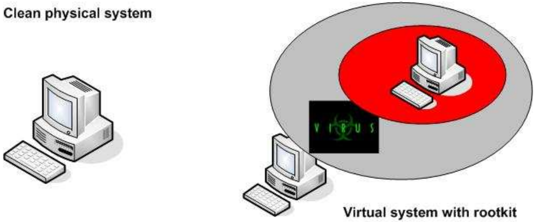
```
X = (float *)
&pageX[offsetX];
Y = (float *)
&pageY[offsetY];
time = HardClock();
memcpy(X, Y, 512);
time = HardClock() - time;
```



$$\mu(E_{Virtual}) < \mu(E_{Physical}).$$

Detecting rootkits

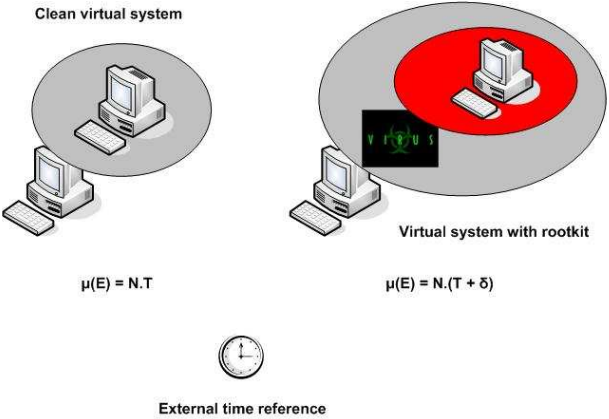
All the detection techniques proposed up to now are conceptually flawed.



You cannot compare what cannot be compared !

Detecting rootkits in a virtual system

Just model a clean virtual system. Any statistical biais must be considered as suspicious.



Plan

Conclusion

- To remain undetectable, a code (stealth or not malware) must either :
    - lie on an undecidable problem (Filiol ; Filiol - Borello - Mé, 2007), or
    - lie on a problem of untractable complexity (Spinellis, 2003 - Zuo & Zhou, 2004 - Filiol, Beaucamps, 2006).
- This is very likely to result in a far slower malware/system, in most cases.
- Another key point : for critical system, antiviral security policy must forbid virtualization. . . until an efficient detection solution has been designed.

Future work

- Define some efficient estimators and build efficient detectors.
  - Estimators based on strong cryptographic protocols are potentially excellent candidates. . . to be continued.
- Use of active detection to detect stealth.
  - Input some data and/or commands into the system.
  - This corresponds to make *DSys* vary with time.
  - The rootkit author cannot make *DSealth* vary on-the-fly (would have to forecast every possible *DSys* variation).
- Use of "polymorphic detection" techniques.

Thanks to Mary Lammer and Helen Martin for their help.

Thanks for your attention.