Abstract Detection of Computer Viruses

G. Bonfante, M. Kaczmarek, and J.-Y. Marion

Loria, Calligramme project, B.P. 239, 54506 Vandœuvre-lès-Nancy Cédex, France,
and École Nationale Supérieure des Mines de Nancy, INPL, France.

1 Introduction

Computer viruses are an omnipresent issue of information technology, a lot of books discusse thier practical issues, see [6] or [9]. But, as far as we know, there are only a few theoretical studies on this topic. This situation is amazing because the term "computer virus" comes from the seminal theoretical works of Cohen [2–4] and Adleman [1] in the mid-1980's. We do think that theoretical point of view on computer viruses may bring some new insights to the area, as it is also advocated for example by Filiol [5], an expert on computer viruses and cryptology, or Zuo and Zhou [10]. Indeed, a deep comprehension of viral mechanisms is from our point of view a promising way to suggest new directions on virus detection and defense.

This being said, the first question is what is a virus? We try to give an answer in sect. 2. Then, sect. 3 studies detection strategies widely used by antiviruses. The proofs of the theorems and some illustration are given in appendix.

2 Abstract Virology

We briefly introduce our formal framework for virus modeling. We are considering an acceptable enumeration of recursive functions φ

First, to build a computer virus, you have to find vulnerability for infection spreading, typically an exploit that allows remote execution. Then, you have to write a self-reproducing program using this vulnerability. Moreover, you could add a payload or furtiveness modules.

To follow this scheme, we describe the propagation mechanism by a computable function $\mathcal{B}$. $\mathcal{B}(\mathbf{r}, \mathbf{p})$ is the program that spreads the program $\mathbf{r}$ over the program structure $\mathbf{p}$. An example is provided in A.1.

Then, a virus w.r.t. the propagation mechanism $\mathcal{B}$ is a program $\mathbf{v}$ which propagate itself. It follows the definition.

Definition 1 (Virus). *Assume that $\mathcal{B}$ is a semi-computable function. A virus w.r.t. $\mathcal{B}$ is a program $\mathbf{v}$ s.t. for each $\mathbf{p}$ and x in $\mathcal{D}$, $\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{r},\mathbf{p})}(x)$.*

$\mathcal{B}$ is the propagation function of the virus $\mathbf{v}$, and $\mathcal{B}(\mathbf{v}, \mathbf{p})$ is called the infected form of $\mathbf{p}$ w.r.t. $\mathbf{v}$ and $\mathcal{B}$. A construction is provided in A.2.

More formally, to build a virus, we consider a propagation mechanism and through iteration theorem we specialize it in $\mathcal{B}(\mathbf{v}, \mathbf{p})$ e.g. the program that spreads $\mathbf{v}$ over $\mathbf{p}$. Then, to add self-reproduction, we use the recursion theorem to build $\mathbf{v}$, the program that spreads itself. In fact, this is a generic construction. An illustration is provided in sect. A.3.

Theorem 2 (Generic Virus). *If f is a semi-computable function, then there is a virus $\mathbf{v}$ s.t. $\varphi_{\mathbf{v}}(\mathbf{p}, x) = f(\mathbf{v}, \mathbf{p}, x)$.*

3 Detection Strategies

Virus Detection. A method widely used for virus detection is file scanning. It uses signatures that only match the considered viruses and not healthy programs.

To thwart this detection a polymorphic virus changes some parts of its code to look different on duplication. Clearly, it used the fact that for all semi-computable function f there are infinitely many programs that compute f – see padding lemma [7]. A formal approach is given in sect. B.1.

As a result, to detect all polymorphic forms, one has to decide the set of viruses $V_{\mathcal{B}} = \{\mathbf{v} \mid \forall \mathbf{p}, x \ : \ \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) = \varphi_{\mathbf{v}}(\mathbf{p}, x)\}$. Unfortunately the following holds.

Theorem 3. *There are some functions $\mathcal{B}$ for which $V_{\mathcal{B}}$ is Π_2-complete.*

But there is some hope, the following theorem is, as far as we know, one of the first positive results concerning the detection.

Theorem 4. *There are some functions $\mathcal{B}$ for which $V_{\mathcal{B}}$ is decidable.*

The propagation function is strongly linked to iteration function. Thus, it could possible to investigate partial evaluation based on evaluators associated with decidable sets of viruses. This could lead to an execution environment where all virus will be detectable.

Infected Programs Detection. When a new computer virus is identified, antivirus editors broadcast updates in order to detect all new infected programs. This strategy correspond to decide, the infection set $I_{\mathbf{v}} = \{\mathcal{B}(\mathbf{v}, \mathbf{p}) \mid \mathbf{p} \in \mathcal{D}\}$. But this is not effective. For some viruses, you cannot decide infected programs.

Theorem 5. *There is a virus $\mathbf{v}$ s.t. $I_{\mathbf{v}}$ is Σ_1-complete.*

Behavior Detection. Some antiviruses use viral behavior detection, trying to isolate a set which includes all infected programs and no “sane” program.

The set of all programs that behave as an infected program is called the germ $G_{\mathbf{v}} = \{\mathbf{q} \mid \exists \mathbf{p} \ : \ \varphi_{\mathbf{q}} \approx \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}\}$. As $G_{\mathbf{v}}$ is an extensional set, Rice theorem reminds us that it can’t be decided. However, there is a clever detection strategy associated with this set.

A virus $\mathbf{v}$ is said isolable within its germ if there is a decidable set R s.t. $I_{\mathbf{v}} \subset R \subset G_{\mathbf{v}}$. Programs of the germ behave as infected programs, thus they produce infected forms, so they are detected by deciding R. Moreover, R contains only programs behaving as infected forms, e.g. no sane program is detected. Unfortunately, the following theorem, conjectured by Adleman in [1], holds.

Theorem 6. *There is a virus $\mathbf{v}$ which is not isolable within its germ.*

References

1. L.M. Adleman. An abstract theory of computer viruses. In *Advances in Cryptology — CRYPTO'88*, volume 403. Lecture Notes in Computer Science, 1988.
2. F.B. Cohen. *Computer Viruses.* PhD thesis, University of Southern California, January 1986.
3. F.B. Cohen. Computer viruses: theory and experiments. *Comput. Secur.*, 6(1):22–35, 1987.
4. F.B. Cohen. Models of practical defences against computer viruses: theory and experiments. *Comput. Secur.*, 6(1), 1987.
5. E. Filiol. *Les virus informatiques: thorie, pratique et applications.* Springer-Verlag France, 2004.
6. M.A. Ludwig. *The Giant Black Book of Computer Viruses.* American Eagle Publications, 1998.
7. P. Odiffredi. *Classical recursion theory.* North-Holland, 1989.
8. H.Jr. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw Hill, New York, 1967.
9. P. Szor. *The Art of Computer Virus Research and Defense.* Addison-Wesley Professional, 2005.
10. Z. Zuo and M. Zhou. Some further theorical results about computer viruses. In *The Computer Journal*, 2004.

A Abstract Virology

A.1 Propagation Function

With $\mathbf{p} = (\mathbf{p}_1, ..., \mathbf{p}_n)$ a sequence of programs and $x \cdot y$ the concatenation of x and y, the following function Δ, spread $\mathbf{r}$ over $\mathbf{p}$.

$$\Delta(\mathbf{r}, \mathbf{p}) = (\mathbf{p}_1 \cdot \mathbf{r}, ..., \mathbf{p}_n \cdot \mathbf{r}) \ .$$

As Δ is computable, there is a program $\mathbf{e}$ s.t. $\varphi_{\mathbf{e}}(\mathbf{r}, \mathbf{p}, x) = \Delta(\mathbf{r}, \mathbf{p})$.

Now we define $\mathcal{B}(\mathbf{r}, \mathbf{p}) = S(\mathbf{e}, \mathbf{r}, \mathbf{p})$, it follows

$$\begin{aligned} \varphi_{\mathcal{B}(\mathbf{r},\mathbf{p})}(x) &= \varphi_{S(\mathbf{e},\mathbf{r},\mathbf{p})}(x) && \text{by definition of } \mathcal{B} \\ &= \varphi_{\mathbf{e}}(\mathbf{r}, \mathbf{p}, x) && \text{by iteration theorem} \\ \varphi_{\mathcal{B}(\mathbf{r},\mathbf{p})}(x) &= (\mathbf{p}_1 \cdot \mathbf{r}, ..., \mathbf{p}_n \cdot \mathbf{r}) && \text{by definition of } \mathbf{e} \ . \end{aligned}$$

The program $\mathcal{B}(\mathbf{r}, \mathbf{p})$ spreads $\mathbf{r}$ over $\mathbf{p}$ using the mechanism described by Δ. Throughout, $\mathcal{B}(\mathbf{r}, \mathbf{p}) = S(\mathbf{e}, \mathbf{r}, \mathbf{p})$ is the specialization of propagation program $\mathbf{e}$ for the propagated code $\mathbf{r}$ and the program structure $\mathbf{p}$.

In other words, one can see $\mathcal{B}$ as a vulnerability of the programming environment. Indeed, $\mathcal{B}$ is a functional property of the programming language φ which is used by a virus to propagate itself into the system. From biological view, $\mathcal{B}$ is the vector which carries and transmits the virus.

The Δ propagation could be illustrated with the following bash code.

```
for FName in $(ls $2);do
  if [ ./$FName != $1 ]; then
   cat $1 >> ./$FName
  fi
done
```

This program concatenates its first argument at the end of every file in the path given as the second argument.

A.2 Virus

Continuing example provided in A.1 and considering $\varphi_{\mathcal{B}(\mathbf{r},\mathbf{p})}(x)$ as a function f of $\mathbf{r}$, $\mathbf{p}$ and x, the recursion theorem provide a program $\mathbf{v}$ s.t.

$$\varphi_{\mathbf{v}}(\mathbf{p},x) = f(\mathbf{v},\mathbf{p},x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \ .$$

By definition, $\mathbf{v}$ is a virus w.r.t. to $\mathcal{B}$. Effectively, $\mathbf{v}$ is a program which propagate its own code.

$$\varphi_{\mathbf{v}}(\mathbf{p},x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) = (\mathbf{p}_1 \cdot \mathbf{v}, ..., \mathbf{p}_n \cdot \mathbf{v}) \ .$$

The execution of $\mathbf{v}$ adds $\mathbf{v}$ at the end of every program of the structure given as its first entry. This could be illustrated programmatically with the following bash code.

```
for FName in $(ls $1);do
  if [ ./$FName != $0 ]; then
   cat $0 >> ./$FName
  fi
done
```

Notice this bash code is obtained by digitalization of sect. A.1's example.

Many real computer viruses use this kind of propagation. `Jerusalem`, the plaid of 1988, behaves the same way, it adds itself to executable file – that is, `.COM` or `.EXE` files.

A.3 Generic Virus

Theorem 7 (Generic Virus). *If f is a semi-computable function, then there is a virus $\mathbf{v}$ s.t. $\varphi_{\mathbf{v}}(\mathbf{p},x) = f(\mathbf{v},\mathbf{p},x)$.*

Proof. Recursion theorem implies that the semi-computable function f has a fixed point that we call $\mathbf{v}$. We have $\varphi_{\mathbf{v}}(\mathbf{p},x) = f(\mathbf{v},\mathbf{p},x)$.

Next, let $\mathbf{e}$ be a program computing f, that is $f \approx \varphi_{\mathbf{e}}$. The propagation function $\mathcal{B}$ induced by $\mathbf{v}$ is defined by $\mathcal{B}(\mathbf{v},\mathbf{p}) = S(\mathbf{e},\mathbf{v},\mathbf{p})$, since

$$\begin{aligned}
\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) &= \varphi_{S(\mathbf{e},\mathbf{v},\mathbf{p})}(x) && \text{by definition of } \mathcal{B} \\
&= \varphi_{\mathbf{e}}(\mathbf{v},\mathbf{p},x) && \text{by iteration theorem} \\
&= f(\mathbf{v},\mathbf{p},x) && \text{by definition of } \mathbf{e} \\
\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) &= \varphi_{\mathbf{v}}(\mathbf{p},x) && \text{by definition of } \mathbf{v} \ ,
\end{aligned}$$

$\mathbf{v}$ fulfills the theorem.

Again, it is worth to say that the propagation function relies on the iteration function S. In some sense, S should be considered as a vulnerability, which is inherent to each acceptable programming language.

This theorem allows adding functionalities freely to a computer virus. To illustrate this, we construct a virus which performs several actions depending on some conditions. This construction of triggers is very general and embeds a lot of practical cases. Typically, some conditions could trigger a payload, like a logic bomb.

Corollary 8. *Let $C_1, \ldots, C_k$ be k semi-computable disjoint subsets of $\mathcal{D}$ and $V_1, \ldots, V_k$ be k semi-computable functions. There is a virus $\mathbf{v}$ which satisfies for all $\mathbf{p}$ and x, the equation*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \begin{cases} V_1(\mathbf{v}, \mathbf{p}, x) & (\mathbf{p}, x) \in C_1 \\ \vdots & \\ V_k(\mathbf{v}, \mathbf{p}, x) & (\mathbf{p}, x) \in C_k \end{cases} .$$

Proof. Define

$$f(y, \mathbf{p}, x) = \begin{cases} V_1(y, \mathbf{p}, x) & (\mathbf{p}, x) \in C_1 \\ \vdots & \\ V_k(y, \mathbf{p}, x) & (\mathbf{p}, x) \in C_k \end{cases} .$$

As f is semi-computable the theorem 2 provide the desired virus $\mathbf{v}$.

B Detection Strategies

B.1 Virus Detection

Theorem 9 (Polymorphism Generator). *If f is a semi-computable function, then there is a computable function $\mathcal{G}$ s.t.*

$$\forall i \in \mathcal{D} \ : \ \mathcal{G}(i) \text{ is a virus} \tag{1}$$

$$\forall i, j \in \mathcal{D} \ : \ i \neq j \Rightarrow \mathcal{G}(i) \neq \mathcal{G}(j) \tag{2}$$

$$\forall i, x, \mathbf{p} \in \mathcal{D} \ : \ \varphi_{\mathcal{G}(i)}(\mathbf{p}, x) = f(\mathcal{G}(i), \mathbf{p}, x) \ . \tag{3}$$

Proof. In fact, $\mathcal{G}$ is fixed point generator. Let f a semi-computable function, $\mathbf{e}$ a program computing f, we define the computable function f_1 by $f_1(y) = S(\mathbf{e}, y)$.

The parameterized recursion theorem [8] provides a computable function $\mathcal{G}$ s.t.

$$\forall i \in \mathcal{D} \ : \ \varphi_{\mathcal{G}(i)} \approx \varphi_{f_1(\mathcal{G}(i))} \tag{\checkmark}$$

$$\forall i, j \in \mathcal{D} \ : \ i \neq j \Rightarrow \mathcal{G}(i) \neq \mathcal{G}(j) \ . \tag{3}$$

It follows, for all i,

$$\begin{aligned} \varphi_{\mathcal{G}(i)}(\mathbf{p}, x) &= \varphi_{f_1(\mathcal{G}(i))}(\mathbf{p}, x) && \text{by } (\checkmark) \\ \varphi_{\mathcal{G}(i)}(\mathbf{p}, x) &= f(\mathcal{G}(i), \mathbf{p}, x) && \text{by definition of } f_1 \ . \end{aligned}$$

This provides the property (2). Moreover, for all i, $\mathcal{G}(i)$ is a virus w.r.t. $\mathcal{B}(\mathbf{v}, \mathbf{p}) = S(\mathbf{e}, \mathbf{v}, \mathbf{p})$, we have the property (1). We conclude that $\mathcal{G}$ fulfills the theorem.

Theorem 10. *There are some functions $\mathcal{B}$ for which $V_\mathcal{B}$ is Π_2-complete.*

Proof. Suppose now given a computable function t computed by $\mathbf{q}$. It is well known that the set $T = \{i \mid \varphi_i = t\}$ is Π_2-complete. Define now $\mathcal{B}(y, p) = S(\mathbf{q}, p)$ and observe that a virus $\mathbf{v}$ verify $\forall \mathbf{p}, x : \varphi_\mathbf{v}(\mathbf{p}, x) = t(\mathbf{p}, x)$. The pairing procedure being surjective, $\mathbf{v}$ is an index of t.

Conversely, suppose that $\mathbf{e}$ is not a virus. In that case, there is some $\mathbf{p}, x$ for which $\varphi_\mathbf{e}(\mathbf{p}, x) \neq \varphi_{\mathcal{B}(\mathbf{e},\mathbf{p})}(x) = t(\mathbf{p}, x)$. As a consequence, it is not an index of t. So, $V_\mathcal{B} = T$.

Theorem 11. *There are some functions $\mathcal{B}$ for which $V_\mathcal{B}$ is decidable.*

Proof. Let us define $f(y, \mathbf{p}, x) = \varphi_y(\mathbf{p}, x)$ computed by $\mathbf{e}$. Iteration theorem provides $S(\mathbf{e}, y, \mathbf{p})$ s.t. for all $y, \mathbf{p}, x$, $\varphi_{S(\mathbf{e},y,\mathbf{p})}(x) = f(y, \mathbf{p}, x)$. Let us define $\mathcal{B}(y, \mathbf{p}) = S(\mathbf{e}, y, \mathbf{p})$. In that case, any program is a virus.

B.2 Infected Programs Detection

Theorem 12. *There is a virus $\mathbf{v}$ s.t. $I_\mathbf{v}$ is Σ_1-complete.*

Proof. Let K a Σ_1-complete set, there is a computable function s.t. $f(\mathcal{D}) = K$.

Let $\mathcal{B}(\mathbf{v}, \mathbf{p}) = f(\mathbf{p})$ and consider $\mathbf{v}$ s.t. $\varphi_\mathbf{v}(\mathbf{p}, x) = \varphi_{f(\mathbf{p})}(x)$. $\mathbf{v}$ is a virus w.r.t. $\mathcal{B}$ and the associated infection set $I_\mathbf{v} = f(\mathcal{D})$ is Σ_1-complete.

B.3 Behavior Detection

Theorem 13. *There is a virus $\mathbf{v}$ which is not isolable within its germ.*

Proof. Consider the virus of theorem 5's proof with $K = \{\mathbf{p} \mid \varphi_\mathbf{p}(\mathbf{p}) \downarrow\}$.

Suppose that $\mathbf{v}$ is isolable within its germ, there exists R decidable s.t. $I_v \subset R \subset G_v$. Let R^c the complementary of R and $\mathbf{r}$ the program s.t. $\varphi_\mathbf{r}$ is the characteristic function of R^c, thus $\varphi_\mathbf{r}(x) \downarrow \Leftrightarrow x \in R^c$ $(\checkmark)$.

Notice that $K \subset R$, thus K and R^c are disjoint.

– Suppose that $\varphi_\mathbf{r}(\mathbf{r}) \downarrow$, so $\mathbf{r} \in K$. By definition of $\mathbf{r}$, $\varphi_\mathbf{r}(\mathbf{r}) \downarrow \Leftrightarrow \mathbf{r} \in R^c$. As a result $\mathbf{r} \in K$ and $\mathbf{r} \in R^c$, which is absurd.
– Suppose that $\varphi_\mathbf{r}(\mathbf{r}) \uparrow$, so $\mathbf{r} \notin R^c$ and $\mathbf{r} \in R$. By hypothesis $R \subset G_\mathbf{v}$, thus $\mathbf{r} \in G_\mathbf{v}$. By definition of $G_\mathbf{v}$ there exists $\mathbf{p}$ s.t. $\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})} \approx \varphi_\mathbf{r}$ $(\maltese)$ and by definition of $\mathcal{B}$, $\mathcal{B}(\mathbf{v}, \mathbf{p}) \in K$, as a result $\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(\mathcal{B}(\mathbf{v}, \mathbf{p})) \downarrow$. $(\maltese)$ provides $\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(\mathcal{B}(\mathbf{v}, \mathbf{p})) = \varphi_\mathbf{r}(\mathcal{B}(\mathbf{v}, \mathbf{p})) \downarrow$ and $(\checkmark)$ gives $\mathcal{B}(\mathbf{v}, \mathbf{p}) \in R^c$. Moreover $\mathcal{B}(\mathbf{v}, \mathbf{p}) \in K$, which is absurd because K and R^c are disjoint.

$\mathbf{v}$ is not isolable within its germ.