© British Computer Society 2001

Comment on 'A Framework for Modelling Trojans and Computer Virus Infection'

ERKKI MÄKINEN

Department of Computer and Information Sciences, P.O. Box 607, FIN-33014 University of Tampere, Finland
Email: em@cs.uta.fi

We (re-)introduce a Turing machine model for computer viruses. Despite the recent criticism of Turing machine models, they enjoy important advantages: their well-known notation and rich theory make them easy to understand and to elaborate. For many natural problems concerning computer viruses, e.g. for various decidability problems, Turing machine models provide a suitable platform of research.

Received 24 November 1999; revised 19 February 2001

1. INTRODUCTION

Thimbleby *et al.* [1] have recently introduced a framework for modelling computer viruses and other malicious programs. They also criticized the use of Turing machine (TM) models for the same purpose. This note (re-)introduces a universal Turing machine (UTM) model which originally appeared in [2] and discusses its properties in the light of Thimbleby *et al.*'s critique. We show that their points are not valid in the case of models using UTMs.

We show that (universal) TMs can serve as a basis of a model illustrating the properties of viruses and other malicious programs. An obvious benefit of our model is that TMs and their properties are so widely known. This gives a considerable 'competitive advantage' to our model and compensates for the fact that TM models are a bit clumsy.

A typical model using TMs is presented by Cohen [3]. The concept of viral sets is essential in the model. A viral set is a pair (M, W) where M is a TM and W is a set of strings over its tape alphabet. Each string w in W has the property that when M, being in its start state, starts reading w it always writes another string w' of W to somewhere else in its tape. Hence, each w in W is a virus and when M (i.e. 'a computer') reads it, another virus will appear somewhere in its tape (i.e. in its 'memory'). Cohen's model allows us to directly apply the well-known undecidability results for TMs, e.g. it follows from the halting problem of TMs that it is undecidable whether or not a given pair $(M, \{w\})$ is a viral set.

The shortcomings of Cohen's model are discussed in [2]. Most of the critiques of Thimbleby *et al.* [1] are appropriate in the case of Cohen's model.

Instead of a TM we use the UTM as a model of a computer. Viruses are then descriptions of TMs causing other descriptions to be written to the tape of the UTM. In Cohen's model the set of viruses depends on the TM on which they are interpreted. In our modification the set of viruses depends on the rules according to which the descriptions of TMs are written.

2. TURING MACHINES

We assume a familiarity with TMs, decidability and related topics as given, e.g., in [4], where unexplained concepts are to be found.

A TM is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, \beta, F)$, where

- Q is the finite set of states,
- Γ is the finite set of tape symbols,
- β is a special symbol in Γ, the blank symbol,
- Σ, a subset of Γ not including β, is the set of input symbols,
- δ is the next move partial function from $Q \times \Gamma$ to $Q \times \Gamma \times \{\textit{left}, \textit{right}\}$,
- q_0 in Q is the start state and
- F, a subset of Q, is the set of final states.

We can suppose without loss of generality that TMs have a unique final state. The structure of M can be entirely described by the set of valid moves provided that the start state and the final state can be inferred from the encoding used.

Suppose the states in Q and the alphabets in Γ are named by $\{q_1, \ldots, q_n\}$ and $\{a_1, \ldots, a_m\}$, and the directions left and right are denoted by d_1 and d_2, respectively. Then a move $\delta(q_i, a_j) = (q_k, a_l, d_m)$ can be encoded by a binary string

$$0^i10^j10^k10^l10^m.$$

A binary code for a whole TM is

$$111\langle code_1\rangle 11\langle code_2\rangle 11\ldots 11\langle code_p\rangle 111,$$

where each $\langle code_r \rangle$, $r = 1, \ldots, p$, is an encoding of a move according to δ and p is the number of such moves [4].

Given the above code for M and the initial tape contents, the UTM U is capable of simulating the computation of M. It is obvious that there are encodings whose simulation produces other encodings having this same property to the tape of U.

3. WHAT IS A VIRUS?

In Cohen's model a string w (a virus) in W has the property that when M, being in its start state, starts reading w it always writes another string w' of W to somewhere else in its tape. In our model a computer virus is a description of a TM whose simulation by the UTM causes another description of a viral TM to appear to the tape of the UTM. A virus can also modify existing TMs (i.e. programs) in the tape. As an example (sketch), consider a TM T performing the following tasks:

1. find a description of some other TM, say T', from the tape;
2. insert a special symbol into the beginning of the initial tape contents of T';
3. supplement the encoding of T' by moves having the effects described in the items (a)–(c) below,
   (a) reading the new symbol from the tape causes T' to enter into a new subsystem of states,
   (b) a copy of T is inserted into the description of T', and
   (c) the control is returned to the start state of T' and the head of T' is moved to the first cell of the original tape contents.

Thimbleby *et al.* [1] have a finer classification of malicious programs in their model. We could also increase the concreteness of our TM model by defining masquerades, Trojans etc. However, the appropriate level of the model depends on its actual use. If we are interested in the basic undecidability results concerning viruses and their detection, the present level is sufficient.

In what follows, we show that contrary to Thimbleby *et al.*'s critique [1, Section 2], TM models are useful in modelling computer viruses.

The first argument of Thimbleby *et al.* concerns the level of abstraction in TM models, i.e. concepts like 'masquerading' and 'infection' are not present in TM models. This is true for existing models, but there is no reason preventing us sharpening the above model based on UTMs to any level of fine granularity.

Steps 1–3 above describe one possible way to perform infection. Masquerading, in turn, involves a naming convention of programs. In our case, a name of a program could be a bit string in its encoding. Masquerading means that the encoding of a malicious program contains the same bit string falsely naming the program. This is not difficult to implement in the tape of a UTM irrespective of the function of the TM containing the masquerading bit string.

Their second argument concerns the 'other' programs. In conventional TM models there are no 'other' programs to be infected. This, of course, is not a problem in UTM based models where the tape may contain any number of programs, 'normal' programs as well various malicious ones.

The third argument of Thimbleby *et al.* deals with self-awareness of infection: it should be possible to infect a program such that it cannot tell that the infection has happened, i.e. the reliable judgement whether or not a program is infected depends on an internal mechanism that is not affected by the infection. More specifically, Thimbleby *et al.* find it problematic that in most TM models of virus infection the effect of the virus is not visible; the running of the program is either unchanged or it becomes incorrect.

Although this critique holds for the TM models cited in [1], there should be no problems in sharpening the TM models in this respect. Consider again steps 1–3 above. In step 3(a) the infected TM jumps to perform some undesirable activities. This jump can be easily made dependable on a condition outside the infected TM, e.g. on the contents on a certain tape position. Now the effect of the infection can vary in different executions of the infected TM.

Fourth, Thimbleby *et al.* suggest that the model should allow self-replication. Lee [5] has described a TM capable of outputting its own description (see also [6, Problem 7.4-3.]). Such a TM is quite sufficient for our purposes. Thimbleby *et al.* find it restricting that this kind of self-replication property is up to representation. We do not find it as a restriction. It fact, the whole model is up to representation. Namely, we fix the representation for TMs appearing in the tape of the UTM. Another coding would end up with different kinds of representation for all parts of the model.

The fifth and last argument by Thimbleby *et al.* concerns time and space requirements. They point out that some viruses do their damage by consuming time and space, and that this has no consequences in TM models where speed and space are immaterial. Again, this is a matter of the level of abstraction. Time and space requirements for TMs can be defined, but it is questionable whether the effects of such viruses are meaningful to measure in an abstract model.

4. CONCLUDING REMARKS

We have (re-)introduced a universal TM model for computer viruses, and have evaluated its merits against the critique by Thimbleby *et al.* The key concept is the level of abstraction. What are you doing with your model? For many purposes, especially including those related to the basic undecidability questions concerning computer viruses, the UTM model discussed seems to be quite appropriate.

ACKNOWLEDGEMENT

This work was supported by the Academy of Finland (Project 35025).

REFERENCES

[1] Thimbleby, H. W., Anderson, S. O. and Cairns, P. (1998) A framework for modelling Trojans and computer virus infection. *Comp. J.*, **41**, 444–458.

[2] Kauranen, K. and Mäkinen, E. (1990) A note on Cohen's formal model for computer viruses. *ACM SIGSAC Rev.*, **8**, 40–43.

[3] Cohen, F. (1989) Computational aspects of computer viruses. *Comp. Security*, **8**, 325–344.

[4] Hopcroft, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

[5] Lee, C. (1963) The construction of a self-describing Turing machine. In Fox, J. (ed.), *Mathematical Theory of Automata*, pp. 155–164. Polytechnic, Brooklyn, NY.

[6] Minsky, M. (1972) *Computation: Finite and Infinite Machines*. Prentice-Hall, London.