Abstract
Malware or more precisely software designed to secretly access a computer system without the owner’s informed consent is one of the main sources of data loss in companies. Presently, the greatest threat is given by the incredible quick spread of infections. We aim at use concurrent calculi to model malwares and, in particular, their interactions with the network so to be able to identify immunization protocols that can be used in practice.

1 Introduction
The past thirty years have seen a continuous growth to the threats to information technology. From the first home made virus in 1981 (the Elk Cloner) that infected the boot sector of Apple operating systems to the complex mechanisms of nowadays, employed by organized crime for stealing identities and confidential data, the damage suffered by companies costs more and more.

According to [9] the 62% of large organizations in the UK were infected by malicious software in the past year (while in 2008 the percentage was around the 21% ) and the 46% of them had a leak of confidential data. Regarding costs, [9] reports that the time spent to remediate incidents has increased of 14% since 2008: a third of large organizations had at least one breach in the year that took more than ten man-days to deal with, and that on average, small organizations spent £4000–£7000 responding to their worst incident of the year, while large organizations spent £25000–£40000.

This activity has justified a trend of research that aims at classify and find sophisticated techniques for detection of the so-called malicious software or malware. According to Grimes [6]:

‘‘Malicious code is any software program designed to move from computer to computer and network to network in order to in-
This is a very general definition that aims at unifying all kinds of malicious behaviors. Roughly speaking, we can identify five main families of malware (see also [8]):

- **Viruses**: A virus is a self-propagating program that attaches itself to host programs and propagates when an infected program executes. A virus typically consists of an infection procedure, that searches for a new program to infect, and of an injure procedure, that performs the virus payload (i.e., the action that a malicious program is designed to perform on the infected machine).

- **Worms**: A malicious program that uses a network to send copies of itself to other systems is usually called a computer worm. Unlike viruses, worms do not need an host program to carry them around but rather propagate across a network. In general, worms do not contain a specific payload but they are only designed to spread. However, the growth in network traffic and other unintended effects are usually causes of major disruption.

- **Trojan horses**: Trojan horses are non-replicating programs that hide their malicious intent inside host programs that may look useful, or at least harmless, to an unsuspecting user. Trojan horses can be either corrupted legitimate programs that execute malicious code when they run, or standalone programs that masquerade as something else in order to obtain the user unaware complicity needed to accomplish their goals.

- **Spywares**: The term spyware usually refers to malicious programs designed to monitor users’ actions in order to collect private information and send them to an external entity over the Internet.

- **Botnets**: The term botnet refers to a network of infected hosts (bots) that are under the control of a human operator: the botmaster. Usually the life cycle of a botnet includes a recruiting phase where new victims are infected and the attack phase where the botmaster instructs in broadcast all infected hosts to run some form of malicious code.

The usual way of detecting an intruder consists in scanning pre defined sequences of bytes that presents a sort of fingerprint of the malicious code. Other techniques regard the monitoring of the propagation of malwares in a network both for studying properties of the considered malware and/or for finding ways to immunize the network.
We are mainly interested in this second form of detection, as propagation of malwares played a fundamental role in the overall amount of threats detected in 2009. Indeed, [10] reports that: of the top 10 new malicious code families detected in 2009, six were Trojans, two were worms with back door components, one was a worm, and one was a virus, and that in 2009, bot networks were responsible for the distribution of approximately 85 percent of all spam email. Hence, the trend in conducting attacks is to develop malware that makes use of an underlying network to spread.

The next section is devoted to the description of a possible project of research on this topic.

2 Project proposal

As underlined above, we are interested in modeling the interactions between a network of hosts and the malicious code. To the best of our knowledge there are few attempts for finding a model that is: (i) general enough to model the different families of malware; (ii) expressive enough to model the concurrent interactions that characterize the spread of those malwares. To our judgment, concurrent calculi, also as a consequence of our area of expertise, seem to be a natural choice to fulfill those aims.

The starting point of our research is obviously an analysis of the current formal models. One of the first and most used formalization dates back to 1985 and it is by Fred Cohen who in his PhD thesis [4] introduces the notion of computer virus:

“A virus is a program that can infect other programs by modifying them to include a possibly evolved copy of itself.”

More precisely a virus is defined as a sequence of symbols that when interpreted on a Turing machine writes a possibly evolved copy of itself further on the tape.

Few years later Adleman proposes a more abstract definition of computer viruses based on recursive functions [2]. According to Adleman a computer virus is a total recursive function that maps programs to infected programs, where an infected program on every input exhibits one of the following behaviors: (i) imitate, meaning that it performs the intended task of the original program; (ii) injure, meaning that it ignores the intended task and computes some other function (damage); (iii) propagate meaning that it infects another program.

These definitions, although extremely abstract and general, lacks in the fact that they do not support interactive computation and thus are not suitable to correctly characterize all forms of malware that have been developed since the ’80s.

Along the lines of these definitions but also including some concurrent features is a recent work by Jacob et al. [7]. The authors propose a model of
malware based on the Join-calculus. We think that the work is inspiring but that the use of process algebra could be more insightful. In fact we think that using process algebras for the detection of malware does not worth the effort: as first of all powerful and smart algorithms have already been developed. Secondly, deciding properties in these calculi is computationally expensive and too many implementation details should be ignored to obtain a reasonable model. Nevertheless, process algebra or more generally concurrent calculi may help in the study of protocols where we can abstract from the implementation details and focus mainly on the interactions. For this reason, we would like to use a calculus where the topology of the network can be easily represented.

Therefore, guided by the Jacob et al. work and inspired by the large number of biological proposals, we think that one possible choice is the \( \kappa \)-calculus \([5]\). The \( \kappa \)-calculus is a graph rewriting model. The rewriting part regards how the graph can evolve through time: which nodes are added or removed, how relations (edges) or internal state can change.

This calculus presents several advantages: first of all has been extensively used in the biology framework, as a consequence there are several tools implemented for the simulation of \( \kappa \)-systems \([1]\). In particular, these tools already include probabilistic models, that, therefore, can be used to represent the propagation model of worms/botnets \([3]\). Moreover, the \( \kappa \)-calculus, being a graph rewriting model, offers a natural way of representing the topology of networks. Indeed, one can profit of the visual advantages offered by graphs: objects are represented by nodes and their relations by edges.

The road-map to an interesting model for malwares would then be: first to propose a definition in \( \kappa \) of malwares, then to validate the model by instantiating it to existing examples of viruses, worms, . . . . Once that this step is complete, it would be interesting to use the existing tools to propose and validate models of propagation for worms and botnets. These models should also include the possibility of study immunization protocols: i.e. finding the weak nodes in a network so that expensive but powerful detecting tools can be used only on the crucial points of the network, hence enhancing the security of the overall system.

Concluding, the laboratory where more naturally this project fits, is UMR7126 Preuves, Programmes et Systèmes (PPS) where the \( \kappa \)-calculus has been and is developed. But also the laboratory UMR5217 Laboratoire d’Informatique de Grenoble (LIG) where I am currently collaborating is an interesting choice, as it will ensure continuity in the current research activity and also include people with great expertise in the light of the present proposal.
References


