ELSEVIER

Computer Communications 19 (1996) 124–132

computer communications

Model for network behaviour under viral attack

Sokratis Katsikas[a], Thomas Spyrou[a], Dimitris Gritzalis[a,b], John Darzentas[a]

[a]*Department of Mathematics, 'University of the Aegean' Karlovassi 83200, Greece*
[b]*Department of Informatics, Athens University of Economics and Business, 34 Patision St., Athens 10434, Greece*

Received 3 January 1994; in revised form 19 October 1994

Abstract

Computer viruses, worms and Trojan horses pose the most severe intrusion threat against an automated environment, especially if this environment is distributed and the ability to enforce physical access control is very limited. These structures are also collectively referred to as *malicious software*. In this paper, a model for viral attacks against computer networks is being developed. The soundness of the model is being tested via simulation experiments. Interesting results that can be practically useful to network administrators are being derived.

Keywords: Security; Computer viruses; Worms; Malicious software; Intrusion; Network

1. Introduction

Computer viruses, worms and Trojan horses pose the most severe intrusion threat against an automated environment, especially if this environment is distributed and the ability to enforce physical access control is very limited. These structures are also collectively referred to as *malicious software*.

A computer virus may be informally described as a sequence of symbols in a machine's memory; what makes such a sequence of symbols an element of a 'viral set' [1], is that when the machine interprets this sequence, it causes some other element of that viral set to appear somewhere else in the system, at a possibly later time. Formal descriptions of computer viruses can be given either through a computational approach [1–3] or through an abstract formalism [4]. Recent research results [5,6] provide additional formal approaches to these structures, but have not yet been sufficiently evaluated and tested.

Even though several network-oriented intrusion detection systems have appeared in the literature [7,8], there is very little published work on the issue of how a virus spreads into a network and, what is more important, how a network is affected by this spreading. Virus spread models in non-distributed environments have been reported [1]; it has been shown [9] that a virus that has infected a single file quickly spreads and incapacitates all files within a computer system if the system has no ability to apply countermeasures against the virus. An epidemiological model of virus spreading processes has been proposed [5] that allows for antidotes to be modelled as well. In a network environment, a model for assessing the performance of a network – equipped with antidotes – under a viral attack was proposed [10]. However, the simplifying assumptions made in that work are quite restrictive and obscure several characteristics of a viral attack against a network.

The main purpose of this paper is to explore what happens when a computer virus is injected into a stable computer network, given that some nodes of the network have been equipped with antidotes against viruses. In more detail, this paper attempts to answer questions such as: what is the behaviour of the network during the viral attack, what is the network recovery time (i.e. the time required to reach stability again); what are the dependencies between the initial percentage of the nodes equipped with antidotes, those infected by the virus and the network recovery time? These questions can be answered either by analytical means or by simulation. Since the problem is too complicated to tackle analytically, we elected to take a simulation approach, thus setting the stage for further analytical work. It should be noted that the model proposed here is more realistic than that proposed earlier [10], which, to the authors' knowledge, is the only one available in the open literature to date pertaining to viral attacks against networks.

The above issues are addressed by assuming a model of the network; by assuming a model for a hypothetical

SSDI 0140-3664(95)01033-5

virus resembling those most commonly encountered in viral attacks; by establishing the rules of the virus spreading process between nodes in the network; by establishing the rules of the virus spreading process within a network node; by establishing the rules of the antidote virus removal process; and by performing several simulation experiments.

The paper is organized as follows. In the next section, the environment of the problem under study is presented and discussed. The proposed model to be used is then presented and discussed. The simulation model is given, and the simulation experiments are presented and discussed. Finally, the most important conclusions are summarized and topics for future research outlined.

2. Problem environment

Any network environment consists primarily of nodes, which are points where information flows come together or diverge, and of links, which are bi-directional paths along which information is transmitted. The properties of the links are determined by the nodes to which they are connected; if both nodes are operational, then the link is operational. If one node has been disabled, then the link is not operational, meaning that communication between the two nodes attached to the link is impossible.

In our particular case we assume a network with a bus topology. In such a network, any node can directly communicate with any other node. Therefore, the network's ability to transmit data depends primarily upon the condition of its nodes rather than upon the condition of its links. Incapacitation of a single node can never disrupt communication between other nodes, unless the incapacitated node is the node responsible for monitoring and controlling the network itself (the Server).

Each node in the network contains files. For the purposes of this paper, we are only interested in the number of executable files contained in each node, since viruses spread only by attaching themselves to executable files. This number may vary among nodes.

When two network nodes communicate, they exchange data or one of them calls executable files resident at the other. The precise mode of communication is governed by specific communication protocols used in the network, as well as by the operating systems employed by the nodes involved. We assume no specific protocol nor operating system herein, even though the precise behaviour of the virus injected in the network does depend upon both [1,11]. The reason for this is that we are interested mainly in assessing the overall network behaviour rather than the virus spread within each of its nodes.

Every node in the network can be assumed to resemble a user in a non-distributed computer system. How often calls are made to other nodes and how often executable files residing in the same node are called constitute, among others, the characteristics of what is termed the *node profile*; this is assumed to vary with each node. Thus, some node may make heavy use of the network by making frequent calls to other nodes, it may make heavy use of its own files, or conversely, it may make light use of the network or of its own files.

Every node in the network can be at exactly one of four allowable states:

1. *Disabled*: the node has lost its capability to communicate with other nodes due to the activation of the virus resident in some of the files residing at the node. The only way that this node may be reinstated to a normal state is by human intervention.
2. *Normal*: the node performs its designed operation.
3. *Infected*: at least one of the files residing at the node has been infected. The node still functions, but when communicating with some normal node it may infect the latter.
4. *Antidotal*: a normal node equipped with an antidote. In addition to its ordinary functions, the node can check the condition of other nodes and can also cure infected nodes by moving its correcting capability to them.

Notice that the fourth allowable state implies that nodes have the capability to initiate sequences of actions at other nodes. Even though this may seem unrealistic, it can in fact happen if one considers that instead of automatically initiating actions at some remote node, a message could be sent to the human operator of the remote node, who could then proceed with carrying out the required actions.

The network is installed in a plant where human service is available. This means that a disabled node can be serviced by humans and be reinstated to the normal state. Mean service times are allowed to vary and are assumed to follow an exponential distribution.

Finally, it is assumed that the network operates in an asynchronous parallel manner.

Among computer viruses, PC viruses are the most proliferated. However, viruses for the Macintosh and the Amiga systems also exist, as well as a few (approximately 10) Unix viruses. As stated earlier, the proposed model assumes no specific operating system. However, as we need to assume certain virus characteristics, we focus on PC viruses from now on.

The IBM High Integrity Computing Laboratory has reported [12] that just a few viruses (the Stoned virus, the Jerusalem virus, the Bouncing Ball virus, the Joshi virus, the Cascade virus, the Yankee Doodle virus and the Frodo virus) account for the majority of infection incidents detected in the real world between 1990–1991. These seven viruses account for approximately 71% of all infection incidents, whereas the next 70 more frequent viruses account for approximately 27% of reported

incidents, and the remaining known (more than 800) viruses account for a mere 2% of reported infection incidents. More recent reports [13] state that the six more popular viruses (the Stoned virus, the Mich virus, the Form virus, the 15xx virus family, the Joshi virus and the Jerusalem virus) account for approximately 75% of all the infection incidents. Therefore, the model of the hypothetical virus, used herein, should conform to the characteristics of the most commonly encountered viruses in order for it to be realistic.

The most important characteristics of a computer virus include the following [14,15]:

- *Basics*; the virus major functional specifications.
- *Infection method*; used by the virus to infect a file or a system, as well as the time required for such an infection.
- *Detection capabilities*; used to detect the presence of the virus and the time required to achieve it.
- *Activation process*; method used by the virus and the conditions that have to hold so that the virus' damage portion can be activated, as well as the time required to complete the activation process.
- *Removal method*; the method and the time required by a suitable antidote to disinfect (reinstate) an infected system.

Several types of antidotes have been proposed. These may be broadly classified into two classes: software-based and hardware-based. The former comprise, among others, vaccines, pattern matchers, simple and cryptographic checksums, software self-defence or fault tolerance, change control mechanisms and integrity shells [16]. Hardware antidotes based on a neural network approach have been proposed [8,17,18]. While the former are rather inexpensive, they impose a computational overhead on the system that uses them and they do not provide complete immunization from virus infection. On the contrary, the latter are costly, but do not severely increase the system computational burden.

3. Proposed model

The model consists of the characteristics of the virus, the characteristics of the antidotes, the characteristics of the network nodes, the network configuration, the intruder characteristics, the defender characteristics, and the rules according to which the viral attack is evolving. The network configuration and characteristics of the network nodes were given above. The remaining characteristics are given in detail below.

3.1. Virus

Our hypothetical virus has the following characteristics:

- *Basics*: a TSR stealth virus, infecting the boot sector and corrupting data and files.
- *Infection method*: at load time and whenever a file access is done to an executable file. Also, when booting from a floppy disk.
- *Detection capabilities*: usually with the first initiation of a virus scanning process.
- *Activation process*: based on a sophisticated randomization algorithm, incorporating machine checks, monitor types, time, etc. Activation starts after 20 executable files have been infected or when the free hard disk space is less than 10% of the total available, or after 8 hours of work (whichever comes first).
- *Removal method*: By the antidote, as soon as the virus' presence has been detected.

3.2. Antidotes

The antidote assumed here is a software-based product, capable of detecting the presence of a virus, as well as of disinfecting any files infected by the virus. Notice that even though the antidote is assumed to have detection and correction capabilities, this does not mean that it can *always* detect or remove a virus; virus detection or removal happens with a given high probability, which is called the *efficiency* of the antidote.

Moreover, the antidote itself is not immune from infection by the virus, even though the probability of the antidote being infected is far lower than the probability of an ordinary (i.e. not antidotal file) being infected. Thus, nodes equipped with antidotes are not completely immune from virus infection, even though they are better protected than normal nodes.

The antidote also has the ability to launch instructions to a node other than the one in which it normally resides, in order to disinfect files of that node, as well as the ability to copy itself to a normal node, thus transforming the latter into an antidotal one.

It is also assumed that the antidote's operation imposes some overhead on the network's operation, thus reducing its performance; thus, it is not advisable to equip all nodes with antidotes, even though this would reduce the infection probability to negligible levels.

We also assume that a hardware-based antidote is available. Note that equipping all nodes with hardware-based antidotes would completely immunize the network against viral attacks, but this is a prohibitively costly solution. Thus, it is advisable that only safety-critical nodes are equipped with this antidote – in our particular case the network server node.

3.3. The intruder and the defender

Initially, the network is stable, i.e. all nodes are operating normally. At some point in time, an *intruder* launches a viral attack against the network by infecting some files in one or more nodes, thus turning the latter into the *infected* state. However, the network's *defender*

(the security officer) can have some of the nodes in the network equipped with antidotes. Therefore, the initial network configuration is as follows: out of the N nodes in the network, N_n are normal, N_i are infected and N_a are antidotal, where of course $N = N_n + N_i + N_a$.

Notice that, even though it was assumed that an antidote may itself be infected, in this initial configuration we assume that the sets of the antidotal and the infected nodes are mutually exclusive. This clearly does not affect the generality of the approach, since *all* infected nodes (regardless of whether they are equipped with antidotes or not) are members of the set of infected nodes. The percentage of infected nodes is a parameter depending on the intruder's intentions and on the type of the attack, whereas the percentage of the antidotal nodes is a parameter depending on the wishes of the network defender.

After this initialization, the intruder does not interfere with the network any more. This assumption may not seem very realistic at first glance, in the sense that it is likely that the intruder will also launch other attacks against the network. However, once an attack has been launched, security regulations and precautions are likely to be reinforced, thereby reducing the intruder's ability to continue attacking the network. It should also be noted that there is nothing in the model used that prohibits modelling additional intruder attacks.

3.4. Transition rules

The initial situation evolves in time because network nodes will change state. Hence, the rules of the game are completely specified by specifying the rules governing the transition between states for the nodes of the network. These transitions are events taking place at random times and with certain probabilities. Therefore, what we need to specify are the transition probabilities between states, as well as the distribution of the time taken to complete these transitions.

In the sequel, the symbol a → b, where a,b ∈ {1,2,3,4} will denote the transition from state a to state b. All times are expressed in minutes. Clearly, transitions between identical states are meaningless, since there is no global clock in the network; a node remains at the same state unless something happens that transforms its state. The rules of transition between states are as follows:

1 → 2: This may only happen by means of human intervention. It is a random event happening with probability 1, taking time distributed exponentially with mean 20.
1 → 3: This transition is impossible.
1 → 4: This transition is also impossible; a disabled node may not become antidotal without becoming normal first.
2 → 1: This transition is also impossible; a normal node cannot be disabled, unless it has first been infected.
2 → 3: This is a random event, depending on whether a normal node will establish contact with an infected node. We assume that any node equiprobably communicates with any other node. Hence, the probability that a normal node will establish communication with an infected node is $K_i/(N-1)$, where K_i is the total number of infected nodes at any time and N is the total number of nodes in the network. The distribution of the time required to complete the transition depends upon the node profile (i.e. how often the node communicates with other nodes), but it is assumed to be uniform, its limits being allowed to vary from node to node.
2 → 4: This transition may occur in two distinct cases: when a disabled node is serviced; and when a normal node communicates with an antidotal node and it is decided to install the antidote to that node as well. When a disabled node is serviced, thereby becoming normal, it is decided to manually install the antidote as well. The probability of this happening is a parameter controllable by the network defender and it is allowed to vary, whereas the time distribution is exponential with mean 10. When a normal node communicates with an antidotal node, it is decided to install the antidote to that node as well. The probability of this happening is the product of two factors: first, the probability of the node to communicate with an antidotal node, which is equal to $K_a/(N-1)$, where K_a is the total number of antidotal nodes in the network at the time; and second, the percentage of normal nodes that the network defender wishes to transform into antidotal nodes, which is a parameter controllable by the network defender and which is allowed to vary. The time required to perform the transition in this case is assumed to be negligible.
3 → 1: This is a random event, depending upon the virus activation process and the process of virus spread within a single node. For our purposes, we assume the following regarding this process: Each node i stores n^i executable files, each of them equiprobable to be executed. Thus, the probability of an infected file being executed (thus making the virus memory resident) is equal to k_i^i/n^i, where k_i^i is the number of infected files in node i. The probability of an uninfected file being executed (thus turning to infected) is k_i^i/n^i, where k_n^i is the number of normal files in node i. It is assumed that once the virus is memory resident, it infects all subsequently executed files with probability one and takes negligible time. The distribution of time intervals between successive file executions is uniform and depends upon the profile of the user associated with the node, hence it varies with each node. As discussed previously, the virus is activated (i.e. disables the node), with probability one, when 20 files have been infected or after 8 hours of work, whichever comes first.
3 → 2: Even though a node may be infected, it still communicates with other nodes. If it establishes communication with an antidotal node, the latter may disinfect the former, thus reinstating it to the normal state. This is a random event happening with probability $k_a/(N-1) \times$ eff,

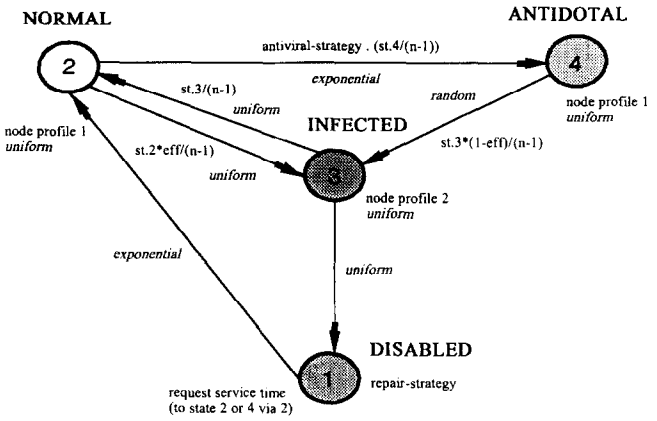


Fig. 1. State transition diagram.

where eff is the efficiency of the antidote, i.e. the percentage of cases when the antidote successfully detects the presence of a virus and removes it; clearly $0 \geq \text{eff} \geq 1$. The time distribution for the transition is uniform, its limits being determined by the node profile (i.e. how often the node communicates with other nodes), as well as by the time taken to scan and clean all files in the node.
$2 \rightarrow 4$: This transition is impossible; an infected node cannot be made antidotal without becoming normal first.
$4 \rightarrow 1$: This transition is impossible; an antidotal node cannot become disabled unless it becomes infected first.
$4 \rightarrow 2$: This transition is impossible; an antidotal node has no reason to become normal unless the network defender decides so, perhaps due to an overload in the network traffic. However, we have assumed that this may not happen, even though the model does allow for such interventions.
$4 \rightarrow 3$: Even though a node is antidotal, it may still become infected, since the antidote has not been assumed to be totally foolproof. This is a random event, happening with probability $K_i \times (1 - \text{eff})/(N - 1)$. The time required to perform the transition is again uniformly distributed with limits determined by the node profile.

It should be noted that all the above assumptions simply reflect real-life situations. Probabilities and timings involved have been experimentally obtained by monitoring a university network. Fig. 1 depicts the sequential nature of state transitions.

4. Simulation model

The simulation tool used is SIMSCRIPT. Several simulation experiments have to be conducted, each aiming at exploring the significance of network parameters, node profile, defender's options, intruder's options, virus characteristics, antidote characteristics, etc. Clearly, the number of parameters involved is too large to establish relations of the network behaviour with all of them, so some have to be fixed. Their choice is based upon intuition and experience with real-life situations.

We assume the following fixed parameters:

1. *Network topology*: as already stated, a network with bus topology has been assumed.
2. *Number of executable files in each node*: affects the behaviour of the network, since it directly affects transitions from state 3 to state 1 and from state 3 to state 2. This is why a fixed *configuration* is assumed, whereby each node contains a fixed number of executable files, but this number is different for each node. It is determined at random, by sampling a uniform distribution in [10,80], for each node.
3. *Node profile*: determines how often a specific node accesses other nodes in the network. Again, this parameter does infect the network behaviour, since it directly influences all state transitions. This is why we have assumed a fixed *configuration*, whereby each node has a fixed profile, but nodes are classified in three classes: heavy, medium and light, corresponding to their access of the network. Heavy nodes access the network at times distributed uniformly in [20,30), medium nodes access the network at times distributed uniformly in [30,80) and light nodes access the network at times distributed uniformly in [80,200]. The percentages of heavy, medium and light nodes are allowed to vary. The type of each node is determined by tossing a biased three sided coin.
4. *User profile*: determines how often a user (corresponding to a node) accesses his own files. This parameter affects the behaviour of the network, since it directly affects transitions from state 3 to state 1 and from state 3 to state 2. This is why a fixed *configuration* is assumed, whereby each user has a fixed profile, but users are classified in three classes: heavy, medium and light, corresponding to their access of their own files. Heavy users access their files at times uniformly distributed in [5,10], medium users access their files at times distributed uniformly in [10,30], and light users access their files at times distributed uniformly in [30,50]. The type of each user is determined, again, by tossing a biased three sided coin.
5. *Number of hardware-based antidotes*: only one node (the server) is equipped with an antidote, immune from viral attacks.
6. *Human intervention service time*: this affects the network behaviour since it directly influences transitions from state 1 to state 2 and onwards to state 4. This is assumed to be exponentially distributed with mean 20 for making the transition from state 1 to state 4 (it involves reformatting hard disks and reinstalling network software) and exponentially distributed with mean 10 if the decision to move onwards to state 4 is taken (it involves manual installation of the antidote at the node).

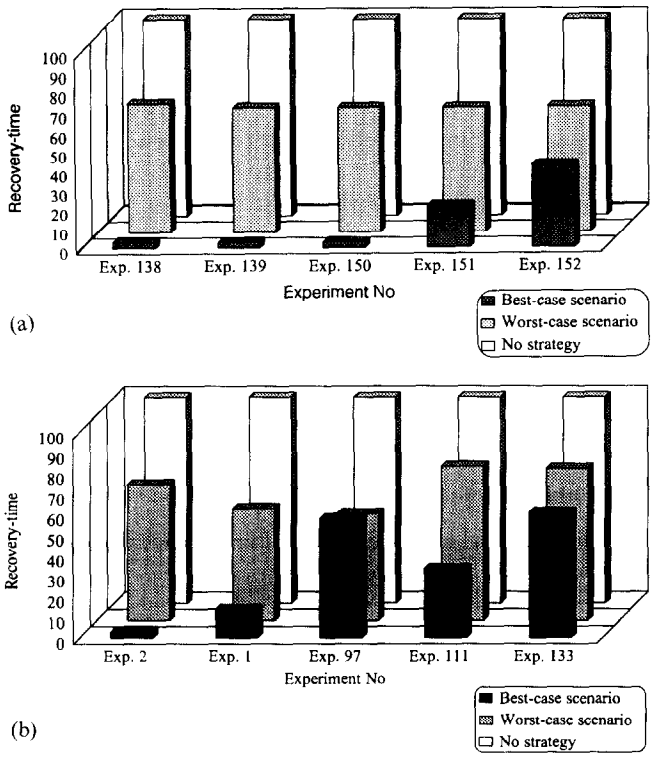


Fig. 2. (a) Network recovery-time improvement as a function of protection strategy; (b) network recovery-time improvement as a function of protection strategy.

7. *Number of human servicemen*: this is assumed to be 1 for small networks and 2 for large ones. For all experiments, two network sizes are assumed; a small network with 20 nodes and a large one with 100 nodes.
8. *Number of different viruses*: it is assumed that only one type of virus is injected in the network by the intruder. This virus has the characteristics described above.
9. *Type of antidote*: it is assumed that there is only one, with characteristics as described above. It requires an average of two minutes to scan a node and one minute to clean an infected file.

5. Simulation experiments and results

5.1. Measure of interest

The measure of interest in our experiments was the network recovery time, which is defined as the time elapsed between the occurrence of the attack and the time when the network returns completely to its normal state, i.e. to a state where all of its nodes are either normal or antidotal.

5.2. Need for an antiviral protection strategy

One of the major duties of a network administration is to establish protection policies. One of these policies should aim to safeguard the network against a malicious software attack. Such a policy could be more or less effective, depending on the type of the viral attack, of the network initial configuration, of the antidote efficiency, etc.

The results of the relevant simulated experiments are described in Fig. 2. Fig. 2a depicts the results pertaining to a small network and Fig. 2b depicts the results pertaining to a large network. In both figures the network recovery time, lacking a protection strategy, is normalized to be equal to 100 time units.

Case 1: Small network

In the case of a small network, if the protection strategy followed is not effective (worst-case scenario), the network recovery time decreases to 70–75 time units, in comparison to the 100 time units required in the case of a non-protected network. This decrease is independent of the network configuration.

On the other hand, if the protection strategy followed is effective (best-case scenario), the network recovery time decreases to 5–40 time units. The largest decrease occurs in the case of a network with limited number of infected nodes ($< 20\%$).

Case 2: Large network

In the case of a large network, if the protection strategy followed is not effective, the network recovery time decreases to 60–80 time units. The largest decrease occurs in the case of a network with limited number of infected nodes ($< 20\%$).

On the other hand, if the protection strategy followed is effective, the network recovery time decreases to 5–15 time units. The decrease is significant in the case of a network with limited number of infected nodes ($< 20\%$).

Based on the above results, the following conclusions can be made:

- Network administration policies aiming at reducing the network recovery time in the event of a malicious software (malware) attack should include an antiviral protection strategy.
- The network recovery time, after a malware attack, depends upon the time of the detection of the attack. The sooner the attack is detected, the better are the chances to decrease the network recovery time.
- The network recovery time could be significantly reduced, provided that the antiviral strategy selected is effective.

5.3. Relationship between network configuration and network recovery time

The initial network configuration is assumed to include normal, infected and antiviral nodes. An important issue is whether it is possible to estimate the minimum number of nodes, that should be protected by

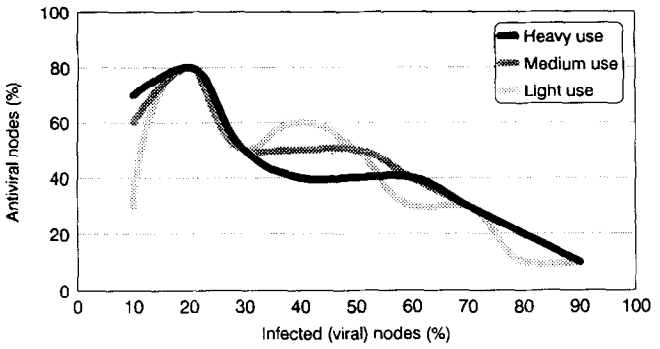


Fig. 3. Minimum time required for network recovery as a function of network configuration and mode of use.

antidotes, in order to keep the network recovery time to a minimum (provided that the number of infected nodes and the mode of use of the network are both given, or that their values can be estimated).

The results of the relevant experimentation are described in Fig. 3. The conclusions drawn from this figure are as follows:

- When the percentage of infected nodes is high (> 60%) then the minimum network recovery time is attained when all other nodes of the network become antidotal. This is independent of the mode of use of the network nodes.
- When the percentage of infected nodes is low (< 20%) then the minimum network recovery time depends upon the mode of use of the network nodes. Faster recovery is usually achieved by protecting many nodes.
- When the percentage of infected nodes is medium (between 30% and 60%) then the minimum network recovery time is attained when the percentage of antidotal nodes is between 40% and 50%.

As a result of the above conclusions, a network administration may devise a protection policy, which will minimally affect the performance of the network, cost

Table 1
Effect of the antiviral strategy on network recovery time

Network configuration	*Antiviral strategy effect on network recovery time*	
Exp 1:		
10% viral	eff ∈ [0, 40) no effects	
10% antiviral nodes	eff ∈ [40, 90) best results with middle values	
	eff ∈ [90, 100] best results with high values	
Exp 2:		
10% viral	eff ∈ [0, 30) no effects	
20% antiviral nodes	eff ∈ [30, 100] best results with middle values	
Exp. 97:		
10% viral	eff ∈ [0, 20) no effects	*Small network*
70% antiviral nodes	eff ∈ [20, 100] best results with high values	
Exp. 111:		
30% viral	No effects	
40% antiviral nodes		
Exp. 133:		
80% viral	No effects	
10% antiviral nodes		
Exp. 138:		
10% viral	eff ∈ [0, 10) no effects	
20% antiviral nodes	eff ∈ [10, 60) best results with middle values	
	eff ∈ [60, 100] best results with high values	
Exp 139:		
10% viral	eff ∈ [0, 10) no effects	
30% antiviral nodes	eff ∈ [10, 60) best results with middle values	
	eff ∈ [60, 100] best results with high values	
Exp. 150:		
10% viral	eff ∈ [0, 10) no effects	
70% antiviral nodes	eff ∈ [10, 60) best results with middle values	*Large network*
	eff ∈ [60, 100] best results with high values	
Exp. 151:		
50% viral	No effects	
40% antiviral nodes		
Exp. 152:		
70% viral	No effects	
10% antiviral		

little and probably defend effectively against malware attacks.

5.4. Effects of the antiviral strategy on the network recovery time

A transition from state 2 to state 4 can occur in two distinct ways: directly or indirectly. The indirect transition occurs when the network administration decides not only to restore an infected node, but also to make it antidotal. This may be the case as such decisions can be taken when the malware attack is still underway. The percentage of nodes that are not only restored to state 2, but are further made into state 4, is the administration's antiviral strategy.

The results of the effects of the antiviral strategy on the network recovery time are shown in Table 1. The conclusions drawn by studying these results are as follows:

- If a malware attack is detected early enough (i.e. the percentage of viral nodes is low), then the antiviral strategy should increase with increasing antidotal efficiency.
- If a malware attack is not detected early (i.e. the percentage of viral nodes is high), then the antiviral strategy has no effect on the network recovery time.
- If a malware attack is detected early enough, minimum recovery time can be achieved by:
  (a) increasing the antiviral strategy with increasing percentage of antidotal nodes, in small networks, or
  (b) increasing the antiviral strategy and the efficiency of the antidote with increasing the percentage of antidotal nodes, in large networks.

6. Conclusions

Malware attacks against computer networks constitute an area of research of growing interest, since their results can be quite disastrous. In this paper, a model describing the spread of a viral attack against a computer network, complete with possible counter-measures that the network administration may have taken, has been developed and subjected to preliminary tests. These tests were performed by using simulation rather than analytical techniques.

The results of these tests indicate the soundness of the model, in the sense that they confirm real-life situations and further produce several interesting conclusions that can effectively be used by network administrators as guidelines towards network protection policy making.

Acknowledgements

This work has been supported in part by the Commission of the European Communities RACE Programme, SECURENET I (R2057) Project. The authors wish to thank their students at the University of the Aegean for carrying out most of the simulation experiments.

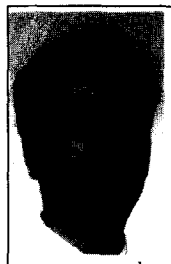
References

[1] F. Cohen, 'Computational aspects of computer viruses', *Computers & Security*, Vol. 8 No 4 (1989) pp 325–344.
[2] F. Cohen, *Computer viruses*, PhD Dissertation University of Southern California (1985).
[3] F. Cohen, 'Computer viruses: Theory and experiments', *Computers & Security*, Vol. 6 No 1 (1987) pp 22–35.
[4] L. Adleman, 'An abstract theory of computer viruses' in L. Hoffman, (ed.), *Rogue Programmes: Viruses, Worms and Trojan Horses*, Van Nostrand-Rhineholt, New York (1990) pp 307–323.
[5] J. Kephart and S. White, 'Directed graph epidemiological models of computer viruses', *Proc. IEEE Symposium Res. in Security and Privacy* (1991) pp 343–359.
[6] R. Ostrowski and M. Yung, 'How to withstand mobile virus attacks', *Proc. 10th ACM Symp. Principles of Distributed Computing* (1991) pp 51–59.
[7] T. Lunt, 'A survey of intrusion detection techniques', *Computers & Security*, Vol. 12 No 4 (1993) pp 405–418.
[8] P. Spirakis, D. Karagiannis, D. Gritzalis and M. Denault, *SECURENET: System development plan*, RACE/SECURENET Project Deliverable 5 (September 1992).
[9] D. Guinier, 'Prophylaxis for virus propagation and general computer security policy', *ACM SIGSAG Rev*, Vol 9 No 2 (1991) pp 1–10.
[10] S. Giess, *Network stability under viral attack*, Royal Signals and Radar Establishment NTIS AD-A229 274 report, RSAR, UK (July 1990).
[11] D. Gritzalis, and S. Gritzalis, *Operating Systems Security*, MIT Publications, Athens, Greece (1991) (in Greek).
[12] S. White, 'IBM's High Integrity Computing Laboratory', *Proc. 1st Int. Virus Conf.*, UK (1991) pp 75–82.
[13] J. Walker, 'The problem with scanners and compressed files', *Proc. 3rd Int. Virus Conf.*, The Netherlands (1993) pp 45–50.
[14] D. Gritzalis, *Information Systems Security*, Greek Computer Society Monograph Series, Athens, Greece (1989) (in Greek).
[15] J. Highland, *Computer Virus Handbook*, Elsevier, Amsterdam (1990).
[16] F. Cohen, *A Short Course on Computer Viruses*, ASP Press, USA (1990).
[17] H. Debar, M. Becker and D. Siboni, 'A neural network component for an intrusion detection system', *Proc. IEEE Symp. Res. in Computer Security and Privacy* (1992) pp 240–250.
[18] D. Guinier, 'Computer virus identification with neural networks', *ACM SIGSAC Rev.*, Vol 9 No 4 (1991) pp 49–59.

Sokratis K Katsikas received his diploma in electrical engineering from the University of Patras, Greece in 1982, his MS in electrical and computer engineering from the University of Massachusetts at Amherst, USA in 1984, and his PhD in computer engineering from the University of Patras, Greece in 1987. He has held teaching and research positions with the University of Massachusetts at Amherst, the University of Patras, the Computer Technology Institute, Patras, Greece, the University of La Verne Athens Campus, the Office of Naval Research, Hellenic Navy and the Technological Education Institute of Athens. In 1990 he joined the Department of Mathematics of the University of the Aegean, Greece, where he now is an Associate Professor of Informatics.

Thomas Spyrou is a PhD student at the University of the Aegean, Greece, and is working as a research assistant in the Research Laboratory of Samos, attached to the university. He has participated to many research projects funded nationally and by the European Union. His current interests include systems thinking, simulation modelling, information systems security, knowledge management in decision support systems and in user intention modelling.

Dimitris Gritzalis holds a BSc (Mathematics) from the University of Patras, Greece, an MSc (Computer Science) from the City University of New York, USA, and a PhD (Informatics) from the University of the Aegean, Greece. Currently, he is an Associate Professor at the Athens Technological Educational Institute; he also serves as Lecturer at the University of the Aegean. His current research interests include telecommunication and information systems security, privacy and information technology assessment.



John Darzentas is Professor in the Department of Mathematics, University of the Aegean, Greece and Director of the Research Laboratory of Samos, attached to the university. Prior to that, he held academic posts in the Universities of London, Reading (UK) and at the Polytechnic of the South Bank. He has an MSc from the University of Sussex, UK, and a PhD from the University of London, both in operational research. He has collaborated and led many research projects, both in the UK and Greece, as well as projects funded by the European Union. His current interests include systems thinking, decision aiding systems, simulation, aspects of knowledge management in AI, and information systems security.