COMPUTER VIRUSES - PREVENTION, DETECTION, AND TREATMENT

- C1 Technical Report 001
- Library No.: S232,522

6 June 1989

PREVENTION, DETECTION, AND TREATMENT

by

Mario Tinto

This publication contains technical observations, opinions, and evidence prepared for informal exchange among individuals involved with computer security. The information contained herein represents the views of the author and is not to be construed as representing an official position of the National Computer Security Center.

Reviewed by: _____

BLAINE W. BURNHAM

Chief, Criteria and Technical Guidelines

Division

Released by: _____

ELIOT SOHMER

Chief, Office of Computer Security

Evaluations,

Publications and Support

Executive Summary

There has been, of late, considerable interest in the topic of computer viruses. The debate has been especially brisk since the so-called "Internet Virus" of November 1988. At one extreme are those who declare that viruses are an essentially new phenomenon, against which we are powerless. At the other end

of the spectrum are those who treat viruses as more of a semantics problem than a technical one, claiming that the problems they pose have already been solved under different terminology. Where then is reality? This paper makes the case that the situation, while certainly not ideal, is not nearly as bleak as some of the alarmists would claim, and that existing technology and security-oriented procedures are extensible to the virus threat. Further, these are largely captured in the DoD Trusted Computer System Evaluation Criteria (TCSEC). However, while the available techniques are relevant, they supply only partial solutions; perfect and universal countermeasures against all possible virus scenarios do not exist. If we are to determine whether or not such are possible, much less develop them, further R&D activity is required.

I. Introduction: The Symptoms

Viruses are a form of the classical Trojan horse attacks and are characterized by their ability to reproduce and spread. However, like all Trojan horses, they first must be "imported" and executed by an authorized user; the attacker typically dupes an unsuspecting individual into accepting and executing the virus. The malicious code may be buried in what are presented to be otherwise useful utilities (e.g., spreadsheets, text editors), which then operate with the user's own authoritzations to cause harm. The offending code may be present in a code segment the user "touches," which then attaches itself to the user's program, without the user ever realizing that he is importing a virus. For instance, a virus may be implanted in system library utilities (e.g., sort/merge routines, mathematics packages) and servers.

While a virus (or Trojan Horse) is normally considered to be limited to the authorizations of the user who is executing the code, the virus can clearly exploit any flaws in the system that would allow the user to enter privileged state (although such attacks are more correctly seen as traditional penetration attacks). If the user who executes the infected code has system privileges (e.g., a system administrator), then the virus will be able to do still more severe damage, depending upon the specific privileges available to it.

The critical point is that viruses depend upon their ability to exploit the legitimate capabilities of authorized users. In order to be successful, a virus must replicate and infect other programs without detection.

II. Treatment and Prevention

As with their biological namesakes, computer viruses come in a variety of types; their missions can be modification or theft of data, or denial of service. Their methods of attack will be as numerous and varied as the weaknesses manifest in systems. Thus, perfect and universal solutions are not likely; there will be no single solution developed capable of preventing any and all virus attacks. Such a

solution is certainly not currently available. However, that is not to say that we are powerless to combat viruses, contain their effects, or limit their capability to do damage. The defenses against viruses are both technical and procedural. More specifically, the principles and mechanisms provided in the TCSEC, especially at class B2 and above, provide a variety of valid defenses against a large class of malicious code and, when applied effectively, can severely limit both the scope of the attack and the extent of the damage.

A. Technical Measures

1. Trusted Computing Base

The TCSEC has, as a central theme, the extremely strong notion of a Trusted Computing Base, or TCB (i.e., the implementation of the Reference Monitor concept). In essence, the TCB is the central policy enforcement mechanism for the computer system, mediating the actions of all system users and user processes. Among the important characteristics of the TCB is that it be always invoked (i.e., unbypassable, mediates each and every access) and self-protecting (i.e., cannot be modified by user code). The consequence of requiring architectures that provide such mechanisms is to limit the ability of hostile code to subvert the TCB. Beginning at the C1 level of trust, fundamental protection mechanisms are required that provide protection of the system programs and data from unprivileged users. Many existing systems (e.g., PCs running DOS) lack even these basic protections required at C1, thus allowing a virus executed by any user to infect any part of the system, even those most basic to system operation and integrity. Commencing with the B2 level of trust, we expect that there will be no fundamental design flaws that allow the security mechanisms to be circumvented. Thus, in the absence of penetration paths, a virus would be limited to attacking users on an individual basis. This means that the rate at which it could propagate would be reduced, as would the damage it could inflict.

It can be argued that a virus capable of infecting each and every user in the system (one that was present in the text editor, for instance) would be reasonably effective at accomplishing some missions (e.g., denial of service). Thus, the value of an intact TCB in the face of an otherwise completely infected user population is moot. However, it is still true that a strong and self-protecting TCB, at a minimum, forces a virus to infect users one at a time. It can also prevent some forms of attack (see 2.b, Mandatory Access Controls, below), and assure the existence and protection of the audit data by which viruses may be detected and traced. In fact, a strong TCB represents the central protection mechanism that a virus must overcome in order to infect the text editor in the first place.

2. Access Controls

Among the fundamental principles that provide the foundation to the TCSEC is that of policy enforcement, the need for the computer system to enforce an access control, or sharing, policy. For both technical and historical reasons, the

principle of policy enforcement translates in the TCSEC into access control mechanisms. Specifically, these are:

a. Discretionary Access Control (DAC)

Discretionary Access Control provides the mechanisms that enforce user-defined sharing, also known in some communities as "need-to-know." Beginning at C1, the TCSEC requires that it be possible for the owner or manager of each data file to specify which users may access his data, and in what modes (e.g., Read, Modify, Append). Clearly, such a mechanism provides control over both acquisition and modification of data by Trojan horses and viruses. In order for the malicious code to carry out its mission, it would have to be executed by someone who already possessed valid permissions against the data being targeted. If that user is not the owner, then the capability of the attack code to do harm would be limited by the allowed permissions (e.g., if the user who was being attacked had "READ-ONLY" access, the attack code could copy the data, but could not modify or erase it). While discretionary access control mechanisms provide relatively weak protections, they do constitute a hurdle that a virus must overcome, and can slow the rate at which the virus propagates.

b. Mandatory Access Control (MAC)

Mandatory Access Control provides those mechanisms that enforce corporate policy dealing with the sharing of data. Examples of such polices would be: "only members of the payroll staff may read or change payroll data," and "classified data may only be accessed by those having the appropriate clearances." Beginning at the B1 level, the TCSEC requires computer systems to be capable of enforcing MAC as well as DAC. That is, the system must be able to enforce those more formal rules dealing with either, or both, levels of sensitivity (e.g., DoD classification scheme) and categories of information (e.g., payroll, medical, R&D, corporate planning). Thus, the ability of a user to access and manipulate data is based upon the comparison of the attributes of users (e.g., "member of payroll department," "member of R&D staff," "management," or "clearance level") with the attributes of the data to be accessed (e.g., payroll data, R&D data, classification level). Because it is required that the TCB control and protect these attribute designators (or, "labels"), they constitute a "hard barrier" for a virus, effectively limiting the scope of what it may do; in a properly designed and implemented system a virus would be unable to effect any changes to the labels. This means, for instance, that a virus that is being executed by someone in the PAYROLL department would be limited to doing damage strictly within the set of data that is labelled accordingly. It would have the potential to modify or destroy PAYROLL data, but not access R&D or MEDICAL data. Additionally, a virus could not change any labels, which means that it is unable to prevent PAYROLL data from being passed to anyone who is not a member of the payroll staff. Likewise, a virus could not cause "SECRET" data to be downgraded. In short, MAC is an extremely strong mechanism, which prevents any process, including a virus, from making properly labeled information available to users who are not

authorized for the information. Systems that achieve TCSEC levels of B2 or greater essentially guarantee that information will not be "compromised," i.e., no malicious code can violate the restrictions implied by the labels.

It needs to be noted that the way in which mandatory controls are typically used is to prevent compromise, which is to say that the emphasis is on preventing "high" data from being written into a "low" file. This does not, in itself, prohibit viruses from propagating, either via a "low" user writing into a "high" file, or a "high" user importing software from a "low" file. However, it should be noted further that the mandatory controls provide the opportunity for implementing similar controls for writing (or importation) as for reading. Such controls are usually seen as implementing mandatory integrity policies, such that the ability to modify files is based upon a set of integrity labels, analogous to the classification labels used to regulate the reading of data. Some systems exist (e.g., Honeywell SCOMP) that have implemented such mechanisms.

3. Audit Trails

The collection of audit data is a traditional security mechanism that provides a trace of user actions such that security events can be traced to the actions of a specific individual. The TCSEC requires, commencing at class C2, that the TCB "...be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects." Because an effective virus depends upon its ability to infect other programs and carry out its mission without detection, audit data provides the basis not only for detecting viral activity, but also for determining which users have been infected (i.e., by identifying which user is responsible for the events in question). Clearly, the collection of data is merely the foundation for detection. To fully implement a sound program, audit reduction and analysis tools are also required. These are also provided for in the TCSEC. Considerable advancement in this arena is reflected by the recently developed intrusion detection systems; sophisticated real-time audit analysis and event-reporting systems, some based on artificial intelligence (or, "AI") techniques. These typically provide extensive capability for detecting a variety of anomalous behavior, and thus can be "tuned" for known or suspected viral patterns. While the available systems are still largely developmental, the early results are quite promising.

4. Architecture

While it is certainly important to identify the correct set of security features that are needed in a system, it is equally important to provide the assurances that the features work as intended, are continually present, and are uncircumventable. Such assurances are provided by the underlying architecture, namely, the hardware support for the features, and the hardware and software design. The TCSEC stresses the importance of architecture and adequate hardware support for the security mechanisms. Even at the lowest level of trust defined by the

TCSEC (i.e., C1) fundamental protection mechanisms are required that provide protection of the systems programs and data from unprivileged users. Such protection is usually implemented by multistate hardware. Starting with B2, the TCSEC places strong emphasis upon design, design analysis, and architectural features that provide for isolation of user programs from each other as well as isolation of system programs from user programs. Such mechanisms not only prevent viruses from casually infecting system programs (e.g., the TCB), but also make it more difficult for the virus to spread from user to user.

As an example of the gain to be realized by the right choice of system architecture, type-enforcement architectures are worthy of special note. These systems provide the potential for extremely fine-grained control of executing code, such that a virus would be incapable of performing any action that is not explicitly allowed by the type-enforcement mechanism. And, because all access to data and resources is via a common, central mechanism (i.e., the type manager), protection need only be focused on the code authorized to manipulate the data and resources, rather than attempting to protect all user programs. By way of illustration, such systems could quite easily enforce the following policy, or set of access rules, which a bank might wish to enforce:

- Tellers may make changes only to those accounts for which they are authorized.
- They may only make changes to specific fields (e.g., may not change the account number, depositor name).
- They may only make the changes authorized between the hours of 9:00 a.m. and 5:00 p.m., Monday through Friday.
- Transactions that exceed $1,000 require the authorization of a supervisor, while transactions that exceed $5,000 require the authorization of the bank manager.

The capabilities of a virus that attached itself to a teller's process in such a system would be, mildly speaking, somewhat circumscribed.

5. Least Privilege/Role Enforcement

A virus that is executed by a user with privilege (i.e., a user that is permitted by the system to circumvent some part or all of the system's security policy) provides an enormous threat to the entire system, because, in assuming the legitimate user's identity, it would be able to circumvent the normal controls that protect other users' programs and data. In many systems, the virus would also be able to circumvent the controls that protect the system itself from modification.

Least Privilege is a familiar concept in the computer security community, and deals with limiting damage through the enforcement of separation of duties. It refers to the principle that users and processes should operate with no more privileges than those needed to perform the duties of the role they are currently

assuming. That is, a user who may take on more than one role or identity (e.g., administrator and unprivileged user, Project A and Project B), should only be given the authorizations needed at the moment, rather than all the privileges he can assume for any and all roles that may be assumed. In contrast, many current systems support only a single, all-powerful system administrator (note especially, the UNIX role of "superuser"). Beginning at the B2 level, trusted systems limit the capabilities of privileged users to those capabilities necessary to accomplish the prescribed task. Beginning at the B3 level, privileged users cannot, in their privileged roles, execute any non-TCB code. The consequence is that, in such a system, a virus could not infect a privileged user's programs, and thus could not exercise his privileges. In addition, at B3 and higher, privileged functions that may modify any security-critical system data or programs require the use of "trusted path" (i.e., require an explicit, unforgeable, action from the privileged user) in order to prevent these actions from being performed without the explicit knowledge and cooperation of the privileged user. This means that no virus could affect security critical data or programs surreptitiously, since it could not cause any modifications without the privileged user becoming aware of the requested actions, thus making the virus visible.

6. Identification and Authentication

Identification and authentication ensures that only authorized users have any access to the system or information contained on the system. It also forms the basis for all other access control mechanisms, providing the necessary user identification data needed to make decisions on requested user actions. While passwords are the oldest and perhaps the most familiar form of personal identifiers used to authenticate users to computer systems, also available today are biometric techniques and "smart card" devices.

B. Procedural and Administrative Measures

While technical measures are necessary for controlling what code segments a process may access, what actions it may take, and the conditions under which it can operate (i.e., what goes on inside the computer), total system security also involves effective site security procedures and system management. This is particularly true because poor procedures can negate the positive effects of some of the technical controls. As an example, audit data collected by the system, and the availability of even the most sophisticated audit analysis tools are of little value if theaudit logs are never reviewed, nor action ever taken as a result of questionable activity.

The following should in no way be seen as an exhaustive list of procedures and management practices effective in addressing the virus threat. Rather, it is intended to be merely illustrative of the manner in which procedural controls are complementary of technical capability.

1. Passwords and Password Management

Historically, passwords have been among the first targets on which an attacker would focus attention. They have traditionally been an easy target with high payoff potential. Because a person's password is often the key to all his data and authorizations, they are analogous to a safe combination. By extension, attacking the password file is akin to targeting the safe that holds the combinations to all the other safes in the building. Thus, good password management and practices can go a long way toward limiting virus attacks. A virus counts on its ability to infect other programs. Thus, either the target must import the virus and execute it as his own (i.e., with his own privileges and authorizations), or the virus must be able to "become" the user to be infected by invoking his password. (It might be noted, in passing, that the November 1988 Internet virus contained extensive password attacks). If the virus cannot successfully log in as an arbitrary user (e.g., by stealing or guessing valid passwords), then it is limited to attempting to fool users into executing the virus code. The trivial ease with which user passwords can be guessed and entire password files can often be attacked is usually nothing short of shocking. Truly effective countermeasures to such attacks are easy to implement and relatively inexpensive. They often amount to not much more than sensible management.

2. Configuration Control

A virus represents code that was not intended to be part of a program or the system. Thus, procedures for maintaining valid and known system configurations, for validating and approving shared code (e.g., software library routines), and for distributing approved programs and media (e.g., diskettes) can provide further obstacles to viral infestation.

3. Operational Procedures

While there may be some commonality across computer sites, it is also true that each site will offer its own unique set of problems. Thus, operational procedures typically need to be tailored to fit the needs of the particular environment, and defenses against viruses will need to be designed into the procedures that govern the day-to-day operation of the site. As an example, recovery from a known or suspected virus attack might require a clean copy of the system. This, in turn, implies procedures for verifying the source and correctness of the backup copy, protecting it from modification until it is to be installed, and for installing it safely. Likewise, management policies and procedures dealing with the importation of code can also provide a measure of resistance to viruses. The establishment of the policy will tend to heighten awareness of the danger of bringing unknown software into the work environment, while effective procedures for controlling the importation of software will make it more difficult for a virus to be introduced.

4. Facility Management

While a computer system may provide a variety of security-related mechanisms, they must be used and, more importantly, used correctly, if any measure of protection is to be achieved. Large, complex systems offer a special challenge, in that there are typically a variety of configuration options, and can support a large number or users, which may be grouped into different "communities" and classes, each with unique attributes, security restrictions and privileges, and with a different view of the system. This translates into a particularly difficult job for the system security administrator; it is imperative that he get everything right simultaneously. There will be many opportunities to configure the system such that needed security features are not active, or that the choice of options invalidates the action of a security feature that was activated. The second case is probably worse, because the security administrator believes that he has activated a security feature when, in fact, he has inadvertently caused the desired protection mechanism to be rendered ineffective. In short, the desired security characteristics of the system, while achievable, can easily be lost in the complex detail of configuring and maintaining an operational environment. Thus, it is critical that there be support for the system administrators such that they can make effective use of the available security features of, and configure and provide life-cycle support for, the level of policy enforcement needed. Toward this end, the TCSEC, at all levels, demands that the vendor provide the purchaser of the product a "Trusted Facility Manual," a document that describes, in a single volume or chapter, all the security mechanisms supported by the system, and provides guidance on how to use them. It is a document aimed explicitly at the system security administrator, and as such, it provides the information necessary to fully understand system security mechanisms, how to use them properly, and the potential harm of poor implementation and configuration choices (e.g., insufficient auditing).

5. User Awareness

Virtually every shared-resource system available today provides facilities for users to specify some level of protection for their data. These may be in the form of User/Group/World mechanisms, Access Control Lists (ACLs), or other features that allow users to specify how, and with whom, information is to be shared. However, in order to be effective, the features must first be used, and they must also be used properly. This clearly means that the users need to be cognizant of the protection features that are provided to them, and understand how they operate. Here also, the TCSEC provides support for this level of user awareness in that it requires that the vendor provide a separate document (i.e., the Security Features User's Guide), explicitly aimed at system users, which apprises them of the security mechanisms that are available to them. While, as noted earlier, most user-specifiable protection mechanisms are not proof against determined hostile attack (at least, not in most current implementations), such protection features do provide a barrier that a virus must overcome; it is clearly easier to steal or

damage files that are not protected than those that are. It is certainly easier for a virus to escape detection if there exist no system-enforced prohibitions against the actions it is attempting to carry out.

6. System Evaluations

It is standard practice, at least within the DoD and Intelligence communities, to have systems undergo an accreditation process, a formal and reasonably well-defined process for determining the acceptability of systems. The critical facet of the process is centered about the "certification," which involves the assessment of the system capabilities as measured against the original requirements definition (e.g., the RFP, system specifications), and typically also takes into account any system vulnerabilities that have been discovered. The certification process is a technical assessment of the system, and thus subjects the system to some level of technical scrutiny. Thus, any flaws, either in system design or in implementation detail, are more likely to be discovered. This is a direct benefit of the current evaluation process directed toward the evaluation of products against the TCSEC. The evaluation process will, in addition to assuring that the TCSEC requirements are satisfied, tend to discover and correct poor design, poor implementation choices and, in some cases, will discover and correct penetration paths. Clearly, processes that find and correct errors and eliminate penetration paths will tend to raise the cost to the attacker.

C. Synopsis of Countermeasures

As discussed earlier, the mission of a virus can be classified as one or more of the standard threats to information security, namely, unauthorized modification, unauthorized disclosure, and denial of service. Technical as well as procedural and administrative countermeasures exist that address these threats, and thus will, in general, limit the success of malicious code attempting to carry out such attacks.

- a. Identification and authentication, discretionary access controls, process isolation, and auditing are relevant countermeasures for the virus whose mission is to destroy or modify user data. Likewise, TCB protection, least privilege, trusted path, and auditing will also serve as valuable countermeasures against the virus whose mission is to destroy or modify system programs and data structures.
- b. Identification and authentication, mandatory access controls, and discretionary access controls provide effective countermeasures against viruses whose mission is to cause unauthorized disclosure of information.
- c. Because infection requires that the virus be able to modify or replace some existing program, all of the technology and procedural countermeasures that are designed to prevent unauthorized modification of programs will make it harder for a virus to attach itself to legal user processes.
- d. The current state of the art in computer security provides only very limited

countermeasures against denial of service. Identification and authentication mechanisms ensure that only authorized users have access to system resources, while auditing allows the system administrator to determine to what extent particular users use or abuse system resources. These controls thus ensure that a virus can attack only those system resources that the infected user is allowed to use, as well as keeping a record of utilization that may make virus detection easier.

III. Summary

Clearly, as stated above, there are no universal cures; no single set of procedures and technical measures guaranteed to stop any and all possible virus attacks. However, this is not different from any other everyday security situation. Specific mechanisms tend to be designed to combat specific dangers, in the same way that vaccines are developed to combat specific diseases. Thus, preventive measures are intended to raise the cost of attacks, or to make it less likely that a specific class of attack will be successful. Similarly for viruses. While viruses can exploit any and all flaws in our computer systems and networks, they also tend to be classes of attacks with which we are already familiar. Thus, while there is valid concern for our vulnerability to virus attacks, a dispassionate analysis shows that our previous experience in computer security is relevant - the protective measures and technology we have developed are directly applicable, and provide a good baseline for making headway against these attacks. In addition, good environmental controls are critical; while technical measures are necessary for controlling what data and resources a user process may access, what actions it may take, and the conditions under which it can operate (i.e., what goes on inside the computer), total system security also involves effective procedures and system management.

On the one hand, it may be argued that viruses present no new technical challenges. The attacks they carry out are the attacks that have been postulated virtually since the advent of time-sharing. However, the intellectual process is such that one determines a threat, or attack scenario, and then develops specific countermeasures. Thus, the classical approach has led us to consider attacks and develop responses on an individual basis. A virus not only propagates, but may also carry out any or all known attacks, thus potentially presenting us with a universal set of attacks in one set of hostile code. However, what is truly revolutionary about viruses is that they change the way in which we will have to view the processing and communications support available to us, in the same way that "letter bombs" would cause us to radically change the way we viewed the postal system, i.e., from beneficial and useful to hostile and potentially dangerous. Where we have previously put great confidence in our computing resources ("If the computer said it, it must be correct"), we will now have to consider those resources as potentially hostile.

Viruses also will cause us to change our view of the very intellectual environment - the sharing of software can no longer be as casual as it was once was. Perhaps this should not be surprising. The attacks that were originally postulated and designed against (e.g., penetrations, Trojan horses, trapdoors) were predicated on a relatively uncomplicated computing environment. The communications explosion now confronts us with a considerably more complex, richly interconnected computing and communications environment. In this environment, viruses are the concern. This means that, while our previous experience is extensible to the new threats, R&D is still needed. While there is considerable debate over whether or not viruses present a completely new set of problems, there is certainly no disagreement concerning our abilities to combat them; most will concede that, at best, today we have only partial solutions. Perfect solutions may be possible, but a better understanding of the root technical issues, development of theory, and testing of countermeasures is required before we can know for certain.

In short, viruses and other forms of malicious code are seen as an extension of classical computer security threats into the current computing and communications environment. The capabilities we have already developed to combat the threats of yesterday apply perfectly well against viruses, but are not perfect solutions. If we are to develop still better solutions, R&D in this area is critical.

APPENDIX - Analysis of Internet Virus and the Evaluation Process

I. The Issue

Among the first questions asked within the NCSC immediately following the November Internet Virus attack was, "Could the attack have been prevented, or at least ameliorated, by the product evaluation process?" It is instructive to determine the impact the TCSEC requirements and the current evaluation process would have had on the virus and the flaws it was able to exploit.

The following assumes that the reader is familiar with the details of the specific attacks, and no effort is made to describe or otherwise elaborate on the technical details of the virus.

II. The Analysis

The question to be answered is, "What effect would trust technology and/or product evaluation have had on the effectiveness of the virus?" The responses fall into two main areas: methods of attack (i.e., which flaws or features were exploited), and the effects of the attack.

a. Infection

The virus used three methods to infect other systems: 1) a subtle bug in the "finger" daemon software, 2) the "debug" feature of the "sendmail" program, and 3) the ability of a user to determine other users' passwords.

The bug in the "finger" daemon (or, fingerd) software would likely not be caught in a C1-B1 level evaluation. There is a moderate chance that it would have been found in a B2-A1 evaluation. If discovered in any evaluation, a fix would not have been required by the NCSC as the problem would not affect the system's ability to enforce the security policy; it does not appear in the TCB, but rather in user space. Most vendors would, however, fix the bug simply to make the system more robust.

At the same time, it is important to note that this attack was successful largely because other routines, which made use of fingerd, did not perform the bounds checks required to catch the error being exploited. A system being designed against the TCSEC would be sensitive to the need for complete parameter checking, at least for security-critical or otherwise privileged codes. Additionally, the evaluation process, at any level, would likely identify fingerd as code being used by privileged programs, thus raising the probability that the flaw would either be found or obviated. This is clearly the case for systems at B2 and beyond; while the flaw that allowed the virus to attack users might still appear (depending on the implementation choices made), the B2 requirements are such that privileged processes would not be dependent on any unevaluated code.

The debug feature of sendmail had a moderate chance of being discovered in a C1-B1 evaluation. The feature would almost certainly have been discovered in a B2-A1 evaluation. When discovered, the team would only have been able to force the vendor to document the feature, as its presence would not affect the system's ability to enforce the security policy.

Here also, it is fair to point out that in a product that was designed to conform with TCSEC requirements, sendmail might well have been seen as integral to the TCB (i.e., a security-critical process). As such, it would have been more closely scrutinized and, beginning at B2, been subjected to penetration testing. To the extent, however, that sendmail is strictly within user space (i.e., not within the TCB boundary), the evaluation process is not likely to turn up flaws such as was exploited by the virus.

The ability of a user to generate other users' passwords as a result of being able to read the password file (albeit encrypted) would have been detected in any evaluation, and the vendor would have been forced to correct the problem. It is important to note that the virus contained an extensive capability to guess user passwords. While it is not clear to what extent the virus actually resorted to this attack, inexpensive and well-known password management procedures would

have a major impact on password attacks, and thus would considerably impair the propagation rate of any virus that depended on them.

b. Effects of the Virus

The primary effect of the virus was the consumption of processor time and memory to the point that nonvirus processes were unable to do any useful work. For the systems in question any valid user could have produced the same effect because the system enforces few useful limits on resource utilization. The current state of the art in trust technology provides no better than partial solutions for dealing with the issues of inequitable use of system resources. B2-A1 evaluations will address so-called "denial of service" problems, but the presence of problems of this type will not adversely affect the rating. That is, evaluators will look for, and report on, attacks that can monopolize system resources or "crash" the system. However, since no objective way yet exists to measure these effects, they do not influence the rating. Instead, it is left to the accreditation authority to determine the impact in his environment, and to implement any necessary countermeasures (e.g., quota management routines, additional auditing).

References

- 1. Continuing Education Institute, "Software-Oriented Computer Architecture," Course notes, 1984.
- 2. Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria (DoD 5200.28-STD), December 1985.
- 3. Gasser, M., Building a Secure Computer System, Van Nostrand Reinhold, 1988.
- 4. Gligor, V. D., "Architectural Implications of Abstract Data Type Implementations," Proceedings of the International Symposium on Computer Architecture, Philadelphia, PA, May 1977.
- 5. Lunt T., "Automated Audit Trail Analysis and Intrusion Detection: A Survey," Proceedings of the 11th National Computer Security Conference, October 1988.
- 6. Spafford, E. H., "The Internet Worm Program: An Analysis," Purdue Technical Report, CSD-TR-823, November 28, 1988.