A Guide to the Selection of

Anti-Virus Tools and Techniques

W. T. Polk & L. E. Bassham
National Institute of Standards and Technology
Computer Security Division

December 2, 1992

Abstract

Computer viruses continue to pose a threat to the integrity and availability of computer systems. This is especially true for users of personal computers. A variety of anti-virus tools are now available to help manage this threat. These tools use a wide range of techniques to detect, identify, and remove viruses.

This guide provides criteria for judging the functionality, practicality, and convenience of anti-virus tools. It furnishes information which readers can use to determine which tools are best suited to target environments, but it does not weigh the merits of specific tools.

Contents

1 Introduction

This document provides guidance in the selection of security tools for protection against computer viruses. The strengths and limitations of various classes of anti-virus tools are discussed, as well as suggestions of appropriate applications for these tools. The technical guidance in this document is intended to supplement the guidance found in NIST Special Publication 500-166, *Computer Viruses and Related Threats: A Management Guide* [WC89].

This document concentrates on widely available tools and techniques as well as some emerging technologies. It provides general guidance for the selection of anti-virus tools, regardless of platform. However, some classes of tools, and most actual products, are only available for personal computers. Developers of anti-virus tools have focused on personal computers since these systems are currently at the greatest risk of infection.

1.1 Audience and Scope

This document is intended primarily for technical personnel selecting anti-virus tools for an organization. Additionally, this document is useful for personal computer end-users who wish to select appropriate solutions for their own system. This document begins with an overview of the types of functionality available in anti-virus products and follows with selection criteria which must be considered to ensure practicality and convenience. The body of the document describes specific classes of anti-virus tools (e.g., scanners) in terms of the selection criteria. This document closes with a summary comparing the different classes of tools and suggests possible applications.

The guidance presented in this document is general in nature. The document makes no attempt to address specific computer systems or anti-virus tools. However, at this time the computer virus problem is most pressing in the personal computer arena. Consequently, most types of anti-virus tools are available as personal computer products. As a result, some information will address that specific environment.

Certain commercial products are identified in this paper in order to adequately specify procedures being described. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the material identified is necessarily the best for the purpose.

1.2 How to Use This Document

The remainder of this section is devoted to terminology and basic concepts.

Section 2 describes the different types of functionality that are available in anti-virus tools. Several different types of detection tools are described, as well as identification and removal tools. This information should assist readers in identifying the classes of products appropriate for their environment.

Section 3 describes some critical selection factors, including accuracy, ease of use, and efficiency. The description of each of these factors is dependent on the functional class of product in question. These selection factors are used to describe product classes in the sections that follow.

Section 4 describes specific classes of tools, such as scanners or checksum programs, and the techniques they employ. This section provides the reader with detailed information regarding the functionality, accuracy, ease of use and efficiency of these classes of tools.

Section 5 presents guidelines for the selection of the most appropriate class of anti-virus tools. It begins by outlining the important environmental aspects that should be considered. Next, the information from Section 4 is summarized and a variety of tables comparing and contrasting the various classes of tools are presented. The remainder of the section provides several hypothetical user scenarios. A battery of tools is suggested for each application.

Section 6 presents guidelines for the selection of the best tool from within a particular class. Important features that may distinguish products from others within a particular class are highlighted.

This document will be most useful if read in its entirety. However, the reader may wish to skip the details on different tools found in Section 4 on an initial reading. Section 5 may help the reader narrow the focus to specific classes of tools for a specific environment. Then the reader may return to Section 4 for details on those classes of tools.

1.3 Definitions and Basic Concepts

This section presents informal definitions and basic concepts that will be used throughout the document. This is intended to clarify the meaning of certain terms which are used inconsistently in the virus field. However, this section is not intended as a primer on viruses. Additional background information and an extensive "Suggested Reading" list may be found in NIST Special Publication 500-166 [WC89].

A **virus** is a self-replicating code segment which must be attached to a host executable.[1] When the host is executed, the virus code also executes. If possible, the virus will replicate by attaching a copy of itself to another executable. The virus may include an additional "payload" that triggers when specific conditions are met. For example, some viruses display a message on a particular date.

A **Trojan horse** is a program that performs a desired task, but also includes unexpected (and undesirable) functions. In this respect, a Trojan horse is similar to a virus, except a Trojan horse does not replicate. An example of a Trojan horse would be an editing program for a multi-user system which has been modified to randomly delete one of the user's files each time that program is used. The program would perform its normal, expected function (editing), but the deletions are unexpected and undesired. A host program that has been infected by a virus is often described as a Trojan horse. However, for the purposes of this document, the term Trojan horse will exclude virus-infected programs.

A **worm** is a self-replicating program. It is self-contained and does not require a host program. The program creates the copy and causes it to execute; no user intervention is required. Worms commonly utilize network services to propagate to other computer systems.

A **variant** is a virus that is generated by modifying a known virus. Examples are modifications that add functionality or evade detection. The term variant is usually applied only when the modifications are minor in nature. An example would be changing the trigger date from Friday the 13th to Thursday the 12th.

An **overwriting** virus will destroy code or data in the host program by replacing it with the virus code. It should be noted that most viruses attempt to retain the original host program's code and functionality after infection because the virus is more likely to be detected and deleted if the program ceases to work. A **non-overwriting** virus is designed to append the virus code to the physical end of the program or to move the original code to another location.

A **self-recognition** procedure is a technique whereby a virus determines whether or not an executable is already infected. The procedure usually involves searching for a particular value at a known position in the executable. Self-recognition is required if the virus is to avoid multiple infections of a single executable. Multiple infections cause excessive growth in size of infected executables and corresponding excessive storage space, contributing to the detection of the virus.

A **resident** virus installs itself as part of the operating system upon execution of an infected host program. The virus will remain resident until the system is shut down. Once installed in memory, a resident virus is available to infect all suitable hosts that are accessed.

[1] An **executable** is an abstraction for programs, command files and other objects on a computer system that can be executed. On a DOS PC, for example, this would include batch command files, COM files, EXE-format files and boot sectors of disks.

A **stealth** virus is a resident virus that attempts to evade detection by concealing its presence in infected files. To achieve this, the virus intercepts system calls which examine the contents or attributes of infected files. The results of these calls must be altered to correspond to the file's original state. For example, a stealth virus might remove the virus code from an executable when it is read (rather than executed) so that an anti-virus software package will examine the original, uninfected host program.

An **encrypted** virus has two parts: a small decryptor and the encrypted virus body. When the virus is executed, the decryptor will execute first and decrypt the virus body. Then the virus body can execute, replicating or becoming resident. The virus body will include an encryptor to apply during replication. A **variably encrypted** virus will use different encryption keys or encryption algorithms. Encrypted viruses are more difficult to disassemble and study since the researcher must decrypt the code.

A **polymorphic** virus creates copies during replication that are functionally equivalent but have distinctly different byte streams. To achieve this, the virus may randomly insert superfluous instructions, interchange the order of independent instructions, or choose from a number of different encryption schemes. This variable quality makes the virus difficult to locate, identify, or remove.

A **research** virus is one that has been written, but has never been unleashed on the public. These include the samples that have been sent to researchers by virus writers. Viruses that have been seen outside the research community are termed **"in the wild."**

It is difficult to determine how many viruses exist. Polymorphic viruses and minor variants complicate the equation. Researchers often cannot agree whether two infected samples are infected with the same virus or different viruses. We will consider two viruses to be different if they could not have evolved from the same sample without a hardware error or human modification.

2 Functionality

Anti-virus tools perform three basic functions. Tools may be be used to *detect*, *identify*, or *remove* viruses.[2] Detection tools perform proactive detection, active detection, or reactive detection. That is, they detect a virus before it executes, during execution, or after execution. Identification and removal tools are more straightforward in their application; neither is of use until a virus has been detected.

2.1 Detection Tools

Detection tools detect the existence of a virus on a system. These tools perform detection at a variety of points in the system. The virus may be actively executing, residing in memory, or stored in executable code. The virus may be detected before execution, during execution, or after execution and replication.

2.1.1 Detection by Static Analysis

Static analysis detection tools examine executables without executing them. Such tools can be used in proactive or reactive fashion. They can be used to detect infected code before it is introduced to a system by testing all diskettes before installing software on a system. They can also be used in a more reactive fashion, testing a system on a regular basis to detect any viruses acquired between detection phases.

2.1.2 Detection by Interception

To propagate, a virus must infect other host programs. Some detection tools are intended to intercept attempts to perform such "illicit" activities. These tools halt the execution of virus-infected programs as the virus attempts to replicate or become resident. Note that the virus has been introduced to the system and attempts to replicate before detection can occur.

[2] A few tools are designed to *prevent* infection by one or more viruses. The discussion of these tools is limited to Section 4.7.2, *Inoculation*, due to their limited application.

2.1.3 Detection of Modification

All viruses cause modification of executables in their replication process. As a result, the presence of viruses can also be detected by searching for the unexpected modification of executables. This process is sometimes called *integrity checking*.

Detection of modification may also identify other security problems, such as the installation of Trojan horses. Note that this type of detection tool works only after infected executables have been introduced to the system *and the virus has replicated*.

2.2 Identification Tools

Identification tools are used to identify *which* virus has infected a particular executable. This allows the user to obtain additional information about the virus. This is a useful practice, since it may provide clues about other types of damage incurred and appropriate clean-up procedures.

2.3 Removal Tools

In many cases, once a virus has been detected it is found on numerous systems or in numerous executables on a single system. Recovery from original diskettes or clean backups can be a tedious process. Removal tools attempt to efficiently restore the system to its uninfected state by removing the virus code from the infected executable.

3 Selection Factors

Once the functional requirements have been determined, there will still be a large assortment of tools to choose from. There are several important selection factors that should be considered to ensure that the right tool is selected for a particular environment.

There are four critical selection factors: *Accuracy*, *Ease of Use*, *Administrative Overhead* and *System Overhead*. Accuracy describes the tool's relative success rate and the types of errors it can make. Ease of use describes the typical user's ability to install and execute the tool and interpret the results. Administrative overhead is the measure of technical support and distribution effort required. System overhead describes the tool's impact on system performance. These factors are introduced below. In depth discussions of these factors are in subsequent subsections.

Accuracy is the most important of the selection factors. Errors in detecting, identifying or removing viruses undermine user confidence in a tool, and often cause users to disregard virus warnings. Errors will at best result in loss of time; at worst they will result in damage to data and programs.

Ease of use is concerned with matching the background and abilities of the system's user to the appropriate software. This is also important since computer users vary greatly in technical skills and ability.

Administrative overhead can be very important as well. Distribution of updates can be a time-consuming task in a large organization. Certain tools require maintenance by the technical support staff rather than the end-user. End-users will require assistance to interpret results from some tools; this can place a large burden on an organization's support staff. It is important to choose tools that your organization has the resources to support.

System overhead is inconsequential from a strict security point of view. Accurate detection, identification or removal of the virus is the important point. However, most of these tools are intended for end-users. If a tool is slow or causes other applications to stop working, end-users will disable it. Thus, attention needs to be paid to the tool's ability to work quickly and to co-exist with other applications on the computer.

3.1 Accuracy

Accuracy is extremely important in the use of all anti-virus tools. Unfortunately, all anti-virus tools make errors. It is the type of errors and frequency with which they occur that is important. Different errors may be crucial in different user scenarios.

Computer users are distributed over a wide spectrum of system knowledge. For those users with the system knowledge to independently verify the information supplied by an anti-virus tool, accuracy is not as great a concern. Unfortunately, many computer users are not prepared for such actions. For such users, a virus infection is somewhat frightening and very confusing. If the anti-virus tool is supplying false information, this will make a bad situation worse. For these users, the overall error rate is most critical.

3.1.1 Detection Tools

Detection tools are expected to identify all executables on a system that have been infected by a virus. This task is complicated by the release of new viruses and the continuing invention of new infection techniques. As a result, the detection process can result in errors of two types: *false positives* and *false negatives*.

When a detection tool identifies an uninfected executable as host to a virus, this is known as a *false positive* (this is also known as a Type I error.) In such cases, a user will waste time and effort in unnecessary cleanup procedures. A user may replace the executable with the original only to find that the executable continues to be identified as infected. This will confuse the user and result in a loss of confidence in either the detection procedures or the tool vendor. If a user attempts to "disinfect" the executable, the removal program may abort without changing the executable or will irreparably damage the program by removing useful code. Either scenario results once more in confusion for the user and lost confidence.

When a detection tool examines an infected executable and incorrectly proclaims it to be free of viruses, this is known as a *false negative*, or Type II error. The detection tool has failed to alert the user to the problem. This kind of error leads to a false sense of security for the user and potential disaster.

3.1.2 Identification Tools

Identification tools identify *which* virus has infected a particular executable. Defining failure in this process turns out to be easier than success. The identification tool has failed if it cannot assign a name to the virus or assigns the wrong name to the virus.

Determining if a tool has correctly named a virus should be a simple task, but in fact it is not. There is disagreement even within the anti-virus research community as to what constitutes "different" viruses. As a result, the community has been unable to agree on the number of existing viruses, and the names attached to them have only vague significance. This leads to a question of *precision*.

As an example, consider two PC virus identification tools. The first tool considers the set of PC viruses as 350 distinct viruses. The second considers the same set to have 900 members. This occurs because the first tool groups a large number of variants under a single name. The second tool will name viruses with greater precision (i.e., viruses grouped together by the first tool are uniquely named by the second).

Such precision problems can occur even if the vendor attempts to name with high precision. A tool may misidentify a virus as another variant of that virus for a variety of reasons. The variant may be new, or analysis of samples may have been incomplete. The loss of precision occurs for different reasons, but the results are no different from the previous example. Any "successful" naming of a virus must be considered along with the degree of precision.

3.1.3 Removal Tools

Removal tools attempt to restore the infected executables to their uninfected state. Removal is successful if the executable, after disinfection, matches the executable before infection on a byte-for-byte basis. The removal process can also produce two types of failures: *hard failure* and *soft failure*.

A *hard failure* occurs if the disinfected program will no longer execute or the removal program terminates without removing the virus. Such a severe failure will be obvious to detect and can occur for a variety of reasons. Executables infected by overwriting viruses cannot be recovered in an automated fashion; too much information has been lost. Hard failures also occur if the removal program attempts to remove a different virus than the actual infector.

Removal results in a *soft failure* if the process produces an executable, which is slightly modified from its original form, that can still execute. This modified executable may never have any problems, but the user cannot be certain of that. The soft failure is more insidious, since it cannot be detected by the user without performing an integrity check.

3.2 Ease of Use

This factor focuses on the level of difficulty presented to the end-user in using the system with anti-virus tools installed. This is intended to gauge the difficulty for the system user to utilize and correctly interpret the feedback received from the tool. This also measures the increased difficulty (if any) in fulfilling the end-user's job requirements.

Ease of Use is the combination of utilization and interpretation of results. This is a function of tool design and quality of documentation. Some classes of tools are inherently more difficult to use. For example, installation of the hardware component of a tool requires greater knowledge of the current hardware configuration than a comparable software-only tool.

3.3 Administrative Overhead

This factor focuses on the difficulty of administration of anti-virus tools. It is intended to gauge the workload imposed upon the technical support team in an organization.

This factor considers difficulty of installation, update requirements, and support levels required by end-users. These functions are often the responsibility of technical support staff or system administrators rather than the end-user. Note that an end-user without technical support must perform all of these functions himself.

3.4 System Overhead

System overhead measures the overall impact of the tool upon system performance. The relevant factors will be the raw speed of the tool and the procedures required for effective use. That is, a program that is executed every week will have a lower overall impact than a program that runs in the background at all times.

4 Tools and Techniques

There is a wide variety of tools and techniques which can be applied to the anti-virus effort. This section will address the following anti-virus techniques:

- signature scanning and algorithmic detection
- general purpose monitors
- access control shells
- checksums for change detection
- knowledge-based removal tools
- research efforts
  - heuristic binary analysis
  - precise identification
- other tools
  - system utilities as removal tools
  - inoculation

For detection of viruses, there are five classes of techniques: signature scanning and algorithmic detection; general purpose monitors; access control shells; checksums for change detection; and heuristic binary analysis. For identification of viruses, there are two techniques: scanning and algorithmic detection; and precise identification tools. Finally, removal tools are addressed. Removal tools come in three forms: general system utilities, single-virus disinfectors, and general disinfecting programs.

4.1 Signature Scanning and Algorithmic Detection

A common class of anti-virus tools employs the complementary techniques of signature scanning and algorithmic detection. This class of tools is known as scanners, which are static analysis detection tools (i.e., they help detect the presence of a virus). Scanners also perform a more limited role as identification tools (i.e., they help determine the specific virus detected). They are primarily used to detect if an executable contains virus code, but they can also be used to detect resident viruses by scanning memory instead of executables.

They may be employed proactively or reactively. Proactive application of scanners is achieved by scanning all executables introduced to the system. Reactive application requires scanning the system at regular intervals (e.g., weekly or monthly).

4.1.1 Functionality

Scanners are limited intrinsically to the detection of known viruses. However, as a side effect of the basic technique, some new variants may also be detected. They are also identification tools, although the methodology is imprecise.

Scanners examine executables (e.g., .EXE or .COM files on a DOS system) for indications of infection by known viruses. Detection of a virus produces a warning message. The warning message will identify the executable and name the virus or virus family with which it is infected. Detection is usually performed by signature matching; special cases may be checked by algorithmic methods.

In signature scanning an executable is searched for selected binary code sequences, called a virus signature, which are unique to a particular virus, or a family of viruses. The virus signatures are generated by examining samples of the virus. Additionally, signature strings often contain wild cards to allow for maximum flexibility.

Single-point scanners add the concept of relative position to the virus signature. Here the code sequence is expected at a particular position within the file. It may not even be detected if the position is wrong. By combining relative position with the signature string, the chances of false positives is greatly reduced. As a result, these scanners can be more accurate than blind scanning without position.

Polymorphic viruses, such as those derived from the *MtE* (mutation engine) [Sku92], do not have fixed signatures. These viruses are self-modifying or variably encrypted. While some scanners use multiple signatures to describe possible infections by these viruses, algorithmic detection is a more powerful and more comprehensive approach for these difficult viruses.

4.1.2 Selection Factors

<u>Accuracy</u>

Scanners are very reliable for identifying infections of viruses that have been around for some time. The vendor has had sufficient time to select a good signature or develop a detection algorithm for these well-known viruses. For such viruses, a detection failure is unlikely with a scanner. An up-to-date scanner tool should detect and to some extent identify any virus you are likely to encounter. Scanners have other problems, though. In the detection process, both false positives and false negatives can occur.

False positives occur when an uninfected executable includes a byte string matching a virus signature in the scanner's database. Scanner developers test their signatures against libraries of commonly-used, uninfected software to reduce false positives. For additional assurance, some developers perform statistical analysis of the likelihood of code sequences appearing in legitimate programs. Still, it is impossible to rule out false positives. Signatures are simply program segments; therefore, the code could appear in an uninfected program.

False negatives occur when an infected executable is encountered but no pattern match is detected. This usually results from procedural problems; if a stealth virus is memory-resident at the time the scanner executes, the virus may hide itself. False negatives can also occur when the system has been infected by a virus that was unknown at the time the scanner was built.

Scanners are also prone to misidentification or may lack precision in naming. Misidentification will usually occur when a new variant of an older virus is encountered. As an example, a scanner may proclaim that *Jerusalem-B* has been detected, when in fact the *Jerusalem-Groen Links* virus is present. This can occur because these viruses are both *Jerusalem* variants and share much of their code. Another scanner might simply declare "Jerusalem variant found in *filename*." This is accurate, but rather imprecise.

<u>Ease of Use</u>

Scanners are very easy to use in general. You simply execute the scanner and it provides concise results. The scanner may have a few options describing which disk, files, or directories to scan, but the user does not have to be a computer expert to select the right parameters or comprehend the results.

<u>Administrative Overhead</u>

New viruses are discovered every week. As a result, virus scanners are immediately out of date. If an organization distributes scanners to its users for virus detection, procedures must be devised for distribution of updates. A scanner for a DOS PC that is more than a few months old will not detect most newly developed viruses. (It may detect, but misidentify, some new variants.) Timely updates are crucial to the effectiveness of any scanner-based anti-virus solution. This can present a distribution problem for a large organization.

Installation is generally simple enough for any user to perform. Interpreting the results is very simple when viruses are correctly identified. Handling false positives will usually require some assistance from technical support. This level of support may be available from the vendor.

<u>Efficiency</u>

Scanners are very efficient. There is a large body of knowledge about searching algorithms, so the typical scanner executes very rapidly. Proactive application will generally result in higher system overhead.

4.1.3 Summary

Scanners are extremely effective at detecting known viruses. Scanners are not intended to detect new viruses (i.e., any virus discovered after the program was released) and any such detection will result in misidentification. Scanners enjoy an especially high level of user acceptance because they name the virus or virus family. However, this can be undermined by the occurrence of false positives.

The strength of a scanner is highly dependent upon the quality and timeliness of the signature database. For viruses requiring algorithmic methods, the quality of the algorithms used will be crucial.

The major strengths of scanners are:

- Up-to-date scanners can be used to reliably detect more than 95 percent of all virus infections at any given time.
- Scanners identify both the infected executable and the virus that has infected it. This can speed the recovery process.
- Scanners are an established technology, utilizing highly efficient algorithms.
- Effective use of scanners usually does not require any special knowledge of the computer system.

The major limitations of scanners are:

- A scanners only looks for viruses that were known at the time its database of signatures was developed. As a result, scanners are prone to false negatives. The user interprets "No virus detected" as "No virus exists." These are *not* equivalent statements.
- Scanners *must* be updated regularly to remain effective. Distribution of updates can be a difficult and time-consuming process.
- Scanners do not perform precise identification. As a result, they are prone to false positives and misidentification.

4.2 General Purpose Monitors

General purpose monitors protect a system from the replication of viruses or execution of the payload of Trojan horses by actively intercepting malicious actions.

4.2.1 Functionality

Monitoring programs are active tools for the real-time detection of viruses and Trojan horses. These tools are intended to intervene or sound an alarm every time a software package performs some suspicious action considered to be virus-like or otherwise malicious behavior. However, since a virus is a code stream, there is a very real possibility that legitimate programs will perform the same actions, causing the alarms to sound.

The designer of such a system begins with a model of "malicious" behavior, then builds modules which intercept and halt attempts to perform those actions. Those modules operate as a part of the operating system.

4.2.2 Selection Factors

Accuracy

A monitoring program assumes that viruses perform actions that are in its model of suspicious behavior and in a way that it can detect. These are not always valid assumptions. New viruses may utilize new methods which may fall outside of the model. Such a virus would not be detected by the monitoring program.

The techniques used by monitoring tools to detect virus-like behavior are also not foolproof. Personal computers lack memory protection, so a program can usually circumvent any control feature of the operating system. As a part of the operating system, monitoring programs are vulnerable to this as well. There are some viruses which evade or turn off monitoring programs.

Finally, legitimate programs may perform actions that the monitor deems suspicious (e.g., self-modifying programs).

Ease of Use

Monitoring software is not appropriate for the average user. The monitor may be difficult to configure properly. The rate of false alarms can be high, particularly false positives, if the configuration is not optimal.

The average user may not be able to determine that program A should modify files, but program B should not. The high rate of false alarms can discourage such a user. At worst, the monitor will be turned off or ignored altogether.

Administrative Overhead

Monitoring programs can impose a fairly heavy administrative workload. They impose a moderate degree of overhead at installation time; this is especially true if several different systems are to be protected. The greatest amount of overhead will probably result from false positives, though. This will vary greatly according to the users' level of expertise.

On the other hand, the monitoring software does not have to be updated frequently. It is not virus-specific, so it will not require updating until new virus *techniques* are devised. (It is still important to remain up-to-date; each time a new class of virus technology is developed, a number of variations emerge.)

Efficiency

Monitoring packages are integrated with the operating system so that additional security procedures are performed. This implies some amount of overhead when any program is executed. The overhead is usually minimal, though.

4.2.3 Summary

Monitoring software may be difficult to use but may detect some new viruses that scanning does not detect, especially if they do not use new techniques.

These monitors produce a high rate of false positives. The users of these programs should be equipped to sort out these false positives on their own. Otherwise, the support staff will be severely taxed.

Monitors can also produce false negatives if the virus doesn't perform any activities the monitor deems suspicious. Worse yet, some viruses have succeeded in attacking monitored systems by turning off the monitors themselves.

4.3 Access Control Shells

Access control shells function as part of the operating system, much like monitoring tools. Rather than monitoring for virus-like behavior, the shell attempts to enforce an access control policy for the system. This policy is described in terms of programs and the data files they may access. The access control shell will sound an alarm every time a user attempts to access or modify a file with an unauthorized software package.

4.3.1 Functionality

To perform this process, the shell must have access to identification and authentication information. If the system does not provide that information, the access control shell may include it. The access control shell may also include encryption tools. These tools can be used to ensure that a user does not reboot from another version of the operating system to circumvent the controls. Note that may of these tools require additional hardware to accomplish these functions.

Access control shells are policy enforcement tools. As a side benefit, they can perform real-time detection of viruses and Trojan horses. The administrator of such a system begins with a description of authorized system use, then converts that description into a set of critical files and the programs which may be used to modify them. The administrator must also select the files which require encryption.

For instance, a shipping clerk might be authorized to access the inventory database with a particular program. However, that same clerk may not be allowed to access the database directly with the database management software. The clerk may not be authorized to access the audit records generated by the trusted application with any program. The administrator would supply appropriate access control statements as input to the monitor and might also encrypt the database.

4.3.2 Selection Factors

Accuracy

Access control shells, like monitoring tools, depend upon the virus or Trojan horse working in an expected manner. On personal computer systems, this is not always a valid assumption. If the virus uses methods that the access control shell does not monitor, the monitor will produce false negatives.

Even with the access control shell, a well-behaved virus can modify any program that its host program is authorized to modify. To reduce the overhead, many programs will not be specifically constrained. This will allow a virus to replicate and is another source of false negatives.

False positives can also occur with access control shells. The system administrator must have sufficient familiarity with the software to authorize access to every file the software needs. If not, legitimate accesses will cause false alarms. If the system is stable, such false positives should not occur after an initial debugging period.

Ease of Use

These tools are intended for highly constrained environments. They usually are not appropriate for the average user at home. They can also place a great deal of overhead on system administrators. The access control tables must be rebuilt each time software or hardware is added to a system, job descriptions are altered, or security policies are modified. If the organization tends to be dynamic, such a tool will be very difficult to maintain. Organizations with well-defined security policies and consistent operations may find maintenance quite tolerable.

This software is easy for users, though. They simply log in and execute whatever programs they require against the required data. If the access control shell prevents the operation, they must go through the administrator to obtain additional privileges.

Efficiency

An access control shell modifies the operating system so that additional security procedures are performed. This implies some amount of overhead when any program is executed. That overhead may be substantial if large amounts of data must be decrypted and re-encrypted upon each access.

Administrative Overhead

An access control shell should not require frequent updates. The software is not specific to any particular threat, so the system will not require updates until new techniques are devised for malicious code. On the other hand, the access control tables which drive the software may require frequent updates.

4.3.3 Summary

Access control shells may be difficult to administer, but are relatively easy for the end-user. This type of tool is primarily designed for policy enforcement, but can also detect the replication of a virus or activation of a Trojan horse.

The tool may incur high overhead processing costs or be expensive due to hardware components. Both false positives and false negatives may occur. False positives will occur when the access tables do not accurately reflect system processing requirements. False negatives will occur when virus replication does not conflict with the user's access table entries.

4.4 Checksums for Change Detection

Change detection is a powerful technique for the detection of viruses and Trojan horses. Change detection works on the theory that executables are static objects; therefore, modification of an executable implies a possible virus infection. The theory has a basic flaw: some executables are self-modifying. Additionally, in a software development environment, executables may be modified by recompilation. These are two examples where checksumming may be an inappropriate solution to the virus problem.

4.4.1 Functionality

Change detection programs generally use an executable as the input to a mathematical function, producing a *checksum*. The change detection program is executed once on the (theoretically) clean system to provide a baseline[3] for testing. During subsequent executions, the program compares the computed checksum with the baseline checksum. A change in the checksum indicates a modification of the executable.

Change detection tools are reactive virus detection tools. They can be used to detect any virus, since they look for modifications in executables. This is a requirement for any virus to replicate. As long as the change detector reviews every executable in its entirety on the system and is used in a proper manner, a virus cannot escape detection.

Change detection tools employ two basic mathematical techniques: Cyclic Redundancy Checks (CRC) and cryptographic checksums.

<u>CRC-Codings</u>

CRC checksums are commonly used to verify integrity of packets in networks and other types of communications between computers. They are fairly efficient and well understood. CRC-based checksums are not extremely secure; they are based on a known set of algorithms. Therefore they can be broken (the particular algorithm can be guessed) by a program if it can find the checksum for a file.

CRC checksum tools, like all change detection tools, can only detect that a virus has replicated. Additionally, the executable must be appear in the baseline.

<u>Cryptographic Checksums</u>

Cryptographic checksums are obtained by applying cryptographic algorithms to the data. Both public and private key algorithms can be used. In general, private key algorithms are

[3]The original file names and their corresponding checksums.

used for efficiency. These techniques are sometimes used in conjunction with two other procedures to decrease system overhead. These techniques are message digesting and hashing.[4]

In *Message Digesting*, hashing is used in conjunction with cryptographic checksums. The hash function, which is very fast, is applied directly to the executable. The result is much smaller than the original data. The checksum is computed by applying the cryptographic function to the hash result. The final result approaches the cryptographic checksum for security, but is much more efficient.

4.4.2 Selection Factors

Accuracy

Properly implemented and used, change detection programs should detect every virus. That is, there are no false negatives with change detection. Change detection can result in high numbers of false positives, however. Programs tend to store configuration information in files containing executable code. If these files are checksummed, as they should be, a change in configuration will trigger the change detector. Additionally, the system must be virus-free when the checksums are calculated; resident viruses may fool the change detection software.

Ease of Use

Change detection software is more challenging to use than some other anti-virus tools. It requires good security procedures and substantial knowledge of the computer system. Procedurally, it is important to protect the baseline. The checksums should be stored off-line or encrypted. Manipulation of the baseline will make the system appear to have been attacked.

Analysis of the results of a checksumming procedure is also more difficult. The average user may not be able to determine that one executable is self-modifying but another is not. False positives due to self-modifying code can discourage such a user, until the output of the change detector is ignored altogether.

Administrative Overhead

Change detection software is easy to install and it requires no updates. The baseline must be established by a qualified staff member. This includes the initial baseline, as well as changes to the baseline as programs are added to the system. Once in operation, a high degree of support can be required for the average end-user, however. A qualified staff member must be available to determine whether or not a change to a particular executable is due to a virus or simply a result of self-modification.

[4] Discussion of cryptographic terminology is beyond the scope of this document. Please see [Sim92]

Efficiency

Change detectors do not impose any overhead on general system use. There is, however, some storage overhead for the baseline checksums. These are best stored off-line with the checksum program.

The calculation of checksums is computationally intensive; the mathematical functions must be calculated on at least a portion of the executable. To be exhaustive, the function should be calculated on the entire executable.

4.4.3 Summary

If change is detected, there are several possibilities: a virus infection, self-modification, recompilation, or modification of the baseline. A knowledgeable user is required to determine the specific reason for change.

The primary strength of change detection techniques is the ability to detect new viruses and Trojan horses. The limitation of change detection is the need for a knowledgeable user to interpret the output.

4.5 Knowledge-Based Virus Removal Tools

The primary means of automated removal of virus infection is knowledge-based removal tools. These removal tools attempt to reverse the modifications a virus makes to a file. After analyzing a particular virus to determine its effects on an infected file, a suitable algorithm is developed for disinfecting files. Tools are available which address only a single virus. These single virus disinfectors are usually developed as the result of a particularly virulent outbreak of a virus. Others detectors are general virus removal programs, containing removal algorithms for several viruses.

4.5.1 Functionality

Knowledge-based removal tools restore an executable to its pre-infection state. All modifications to the original executable must be known in order to accomplish this task. For example, if a file is infected with an overwritting virus, removal is not possible. The information that was overwritten cannot be restored.

The most critical piece of information in the removal process is the identity of the virus itself. If the removal program is removing *Jerusalem-DC*, but the host is infected with *Jerusalem-E2*, the process could fail. Unfortunately, this information is often unavailable or imprecise. This is why precise identification tools are needed.

4.5.2 Selection Factors

Disinfecting software is not very accurate, for a variety of reasons. The error rates are fairly high; however, most are soft errors. This is a result of incomplete information regarding the virus and the lack of quality assurance among virus writers. Additionally, removal techniques tend to fail when a system or file has been infected multiple times (i.e., by the same virus more than once, or by more than one virus).

These programs are relatively easy to use and can disinfect large numbers of programs in a very short time. Any system overhead is inconsequential since the system should not be used until the virus is removed.

4.5.3 Summary

Accurate removal may not be possible. Even if it is theoretically possible, precise identification of the virus is necessary to ensure that the correct removal algorithm is used.

Certain viruses (e.g., overwriting viruses) always cause irreparable damage to an executable. Some extraordinarily well-behaved viruses can be disinfected every time. Most viruses fall somewhere in between. Disinfection will often work, but the results are unpredictable.

Some executables cannot be recovered to the exact pre-infection state. In such a case, the file length or checksum of the disinfected executable may differ from the pre-infection state. In such a case, it is impossible to predict the behavior of the disinfected program. This is the reason virus researchers generally dislike removal programs and discourage their use.

4.6 Research Efforts

The following subsections describe research areas in the anti-virus field. New tools, based on techniques developed in these and other areas, may be available in the near future.

4.6.1 Heuristic Binary Analysis

Static analysis detection tools, based upon heuristic binary analysis, are a focus of research at this time. Heuristic binary analysis is a method whereby the analyzer traces through an executable looking for suspicious, virus-like behavior. If the program appears to perform virus-like actions, a warning is displayed.

Functionality

Binary analysis tools examine an executable for virus-like code. If the code utilizes techniques which are common to viruses, but odd for legitimate programs, the executable is flagged as "possibly infected." Examples include self-encrypted code or code that appears to have been appended to an existing program.

Selection Factors

Both false positives and negatives are sure to result with use of this type of software. False positives occur when an uninfected program uses techniques common to viruses but uncommon in legitimate programs. False negatives will occur when virus code avoids use of those techniques common to viruses.

Binary analysis tools are fairly easy to use. The user simply specifies a program or directory to be analyzed. Analyzing the results is more difficult. Sorting out the false positives from real infections may require more knowledge and experience than the average user possesses.

Heuristic analysis is more computationally intensive than other static analysis methods. This method would be inappropriate for daily use on a large number of files. It is more appropriate for one-time use on a small number of files, as in acceptance testing.

A heuristic analysis program will require updates as new techniques are implemented by virus writers.

Summary

Early examples of this class of tool appear to have fairly high error rates as compared with commercial detection software. As with system monitors, it is difficult to define suspicious in a way that prevents false positives and false negatives. However, these types of tools have been used successfully to identify executables infected by "new" viruses in a few actual outbreaks.

Heuristic binary analysis is still experimental in nature. Initial results have been sufficiently encouraging to suggest that software acceptance procedures could include these tools to augment more traditional technology.

4.6.2 Precise Identification Tools

Precise identification tools are a means by which viruses are named with a much higher degree of assurance. These tools are intended to augment detection tools. Once a virus has been detected, a precise identification tool would be invoked in order to more accurately identify the virus.

Functionality

Virus scanners, currently the most common virus detection method, generally employ signature scanning to detect and identify viruses. This method, however, can lead to misidentifications. The signature that the scanner matched could appear in more than one variant of the virus. To avoid mis-identification the whole virus must match, not just a subset of the virus (i.e., the signature). It is neither feasible nor desirable for identification software to be distributed containing the code to all viruses it can detect. Therefore, prototype precise identification tools utilize a "virus map" to represent the contents of the virus. The virus map contains checksum values for all constant parts of the virus code. The map skips over sections of the virus that contain variable information such as text or system dependent data values.

If the checksums generated by the corresponding portions of the program match, the program is almost certainly infected by the virus corresponding to the map. If none of the maps in the database correspond, the program is infected by a new virus (or is uninfected.)

Selection Factors

The quality of the results produced by a precise identification tool is dependent upon the quality of the virus map database. If that has been done well and kept current, these tools are extremely accurate and precise when identifying known viruses. Conversely, if the virus is new or has no corresponding entry in the database, the precise identification tool should always "fail" to identify the viruses.

This type of tool is easy to use. The user simply specifies an executable, and the tool returns a name, if known. The results are straightforward; it is virus "X," or unknown.

Precise identification tools are slow due to the intensive nature of the computations. These tools may be used to perform an identification pass after the use of a more efficient detection tool. Such a plan would provide the user with the benefits of precise identification without great overhead. Once a virus has been detected, the user wants to know exactly what virus he has and time is not a significant factor.

Summary

Users want to know more about the virus infecting their systems. Precise identification will help them obtain more complete information and can also facilitate automated removal.

Researchers will also wish to use this type of tool. It will allow them to separate samples of known viruses from new ones without performing analysis.

4.7 Other Tools

The remaining tools, system utilities and inoculation, are included for completeness. These tools can be used to provide some measure of functionality. In general, however, these tools are weaker than general anti-virus tools.

4.7.1 System Utilities

Some viruses can be detected or removed with basic system utilities.[5] For example, most DOS boot sector infectors and some Macintosh viruses can be removed with system utilities. System utilities can also be used to detect viruses by searching for virus signatures. These tools have a rather limited focus, though.

Viruses that can be disinfected "by hand" are generally the extremely well-behaved, highly predictable viruses that are well understood. Such viruses are the exception, not the rule. There are many more viruses that cannot be disinfected with these tools.

Where possible, disinfection with system utilities will produce dependable results. A reasonable amount of knowledge is required about the computer system and the virus itself, though. This technique can also be very laborious if a large number of systems are infected.

System utilities are an inefficient means of detection. Generally, only one signature can be handled at a time. This might be a useful technique if a specific virus is to be detected.

Summary

Accurate removal by system utilities is frequently impossible. Certain classes of viruses (e.g., overwriting viruses) always damage the executable beyond all hope of repair. Others modify the executable in rather complicated ways. Only viruses that are extremely well-behaved can be disinfected every time. Similarly, detection with system utilities has limited application.

[5] Two examples of these system utilities are Norton Utilities for the PC and ResEdit for the Macintosh.

4.7.2 Inoculation

In some cases, an executable can be protected against a small number of viruses by "inoculation." This technique involves attaching the self-recognition code for the virus to the executable at the appropriate location.

Since viruses may place their self-recognition codes in overlapping locations, the number of viruses that can be inoculated against simultaneously will be small. To make matters worse, a common way to create a new variant is to change the self-recognition code. Thus, this technique will often fail when tested by minor variants of the viruses inoculated against.

Inoculation is no substitute for more robust anti-virus tools and procedures. It *might* be useful, though, if an organization has had recurring infections from a single virus. For example, after cleaning three or four outbreaks of a particular virus from a network of PCs, inoculation might be considered as a desperation measure.

5 Selecting Anti-Virus Techniques

The selection of the appropriate class of anti-virus tools requires answers to the following set of questions:

- What is the probability of a virus infection?
- What are the consequences of a virus infection?
- What is the skill level of the users in your organization?
- What level of support is available to the end-user?

The first two questions address risk; security should always be commensurate with need. The third and fourth questions address the limitations of the tools and personnel. The answers will be different for each person or organization.

Every organization is at some risk of virus infection. Virus infections can occur whenever electronic information is shared. Every organization shares information in some way and is a potential victim of a virus infection. Most organizations should have some tools available to detect such an infection.

Personal computer users may benefit from tools to identify viruses, since so many viruses exist. Identification tools are not necessary where viruses are few or only theoretically possible.

The use of removal tools is generally not required.[6] It may be desirable in situations where a single person or a small team is tasked with cleaning up after an infection or where high connectivity can result in rapid spread of the virus (such as networks).

5.1 Selecting Detection Tools

The first point to consider when selecting a detection product is the type of viruses likely to be encountered. Approximately 95 percent of all virus infections are accounted for by a small number of viruses. The viruses that constitute this small set can vary geographically. The common viruses can be distinct on different continents, due to the paths in which they travel. Of course, different hardware platforms will be at risk from different viruses.

International organizations may be vulnerable to a larger set of viruses. This set may be obtained by merging the sets of viruses from different geographical regions where they do

[6] Exceptions, such as the DIR-2 PC virus, may be extremely difficult to remove without appropriate tools. In this case, the only alternative to removal tools is to format the disk.

business. Organizations with contacts or installations in locations where virus writers are particularly active [Bon91] are also more likely to encounter new viruses.

Risk from new viruses is an important consideration. Scanners are limited by their design to known viruses; other detection tools are designed to detect any virus. If your organization is at high risk from new viruses, scanners should not be the sole detection technique employed.

Another important criteria to consider is the number and type of errors considered tolerable. The tolerance for a particular type of error in an organization will vary according to the application. Table 1 shows the types of errors which should be expected. An estimate of the frequency that this class of error is encountered (*Infrequent*, *Frequent*, or *Never*) is also given for each class of tools and error type. All anti-virus tools are subject to errors, but their relative frequencies vary widely. Scanners probably have the lowest overall error rate. Checksummers do not produce false negatives.

Detection Tool / Error Types	Scanner	Checksum	Binary Analysis	Generic Monitor	Access Control Shell
False Positives	**Infrequent** Signatures can occur in valid files	**Frequent** Every time a program is modified	**Frequent** In our test, 15% errors	**Frequent** Whenever a legitimate program performs virus-like actions	**Frequent** Whenever a legitimate program performs virus-like actions
False Negatives	**Infrequent** May not detect variants; won't detect new viruses	**Never** Viruses always change executables	**Frequent** In our test, 8% errors	**Frequent** Viruses that circumvent OS can be missed	**Frequent** Viruses that circumvent OS can be missed

Table 1: Types of errors.

The third and fourth items to consider when selecting anti-virus tools are the ease of use and administrative overhead required for each tool. Questions to consider are:

- What is the average skill level of your organization's end-user?
- Does your organization have a support staff to assist user with more technical problems?

Table 2 includes a general evaluation of the ease of use and administrative overhead imposed by each class of tools.

Criteria \ Detection Tool	Scanner	Checksum	Binary Analysis	Generic Monitor	Access Control Shell
Easy of Use	**Very Good** Requires no special knowledge of the system	**Average** Easy to use; results may be difficult to interpret	**Poor** Easy to use; results must be verified	**Poor** Results are difficult to interpret	**Average** Easy to use; Configuration is an impediment
Administrative Overhead	**Low** Requires frequent updates. Little add'l support req'd	**Low** No updates req. Assist in interpreting results	**High** Few updates. Much verification of results	**High** Few updates. Much verification of results	**High** Few updates. Much verification of results

Table 2: Personnel requirements.

If several tools still appear to be candidates, consider the functionality of these tools *beyond* virus detection. Viruses are only one of the many threats to computer security. All detection tools except scanners have general security applications beyond viruses. Scanners are limited in application to viruses, but have the added functionality of virus identification.[7] Consider the added functionality which is most needed by your organization and choose accordingly. The alternatives are outlined in table 3.

	Detection Tool				
	Scanner	Checksum	Binary Analysis	Generic Monitor	Access Control Shell
Additional Functionality	Identification; May also detect known trojan horses	Detection of trojan horses and altered data	Detection of trojan horses	Detection of trojan horses	Enforcing organizations security policy

Table 3: Additional functionality.

The final selection criteria to be considered is when does the tool detect viruses. Proactive detection tools allow the user to keep viruses off a system by testing incoming software. These

[7]Some scanners can also detect known Trojan horses.

tools only allow one chance of detecting a virus (upon initial introduction to the system). Active detection tools intervene during the replication phase itself. Reactive detection tools can be used any time after a virus has entered the system. Additionally, reactive tools are not as rigorous in their demands on system performance. Table 4 shows when these different tools detect viruses.

Detection Tool / Point of Detection	Scanner	Checksum	Binary Analysis	Generic Monitor	Access Control Shell
Static Executable	Yes	No	Yes	No	No
Replication Phase	No	No	No	Yes	Yes
After Infection	Yes	Yes	Yes	No	Yes

Table 4: When tools detect?

5.1.1 Combining Detection Tools

The most complete protection will be obtained by combining tools which perform in radically different fashion and protect against different classes of viruses. For instance, when used together a scanner and a checksum program will protect against both known and unknown viruses. The scanner can detect known viruses before software is installed on the system. A virus can be modified to elude the scanner, but it will be detected by the checksum program.

The two tools should have different "additional functionality" (see table 3) to form the most comprehensive security package. For instance, the combination of a checksum program and an access control shell would also detect Trojan horses and enforce organizational security policy in addition to virus detection. On the other hand, adding a binary analyzer to a system that already employs checksumming would not provide additional functionality.

If you must use two scanners, be sure that they use different search strings. A number of tools are based on published search strings; shareware tools commonly utilize the same public domain signature databases. Two different scanner engines looking for the same strings do not provide any additional protection of information.[8]

[8] Algorithms for detection tend to be independently developed.

5.2 Identification Tools

Currently, scanners are the only effective means of identifying viruses. As discussed in Section 3.1.2, the accuracy to which scanners identify viruses can vary. In the future, precise identification tools should offer greatly increased accuracy.

5.3 Removal Tools

The most dependable technique for virus removal continues to be deletion of the infected executable and restoration from a clean backup. If backups are performed regularly and in a proper manner, virus removal tools may be neglected.

In large organizations with high connectivity, automated removal tools should be obtained. Virus eradication through the removal of infected executables may require too much time and effort. Knowledge based tools will disinfect the largest number of different viruses, but proper identification of the virus prior to disinfection is critical. Even with knowledge based removal tools, disinfection of executables is not always reliable (see Sec. 3.1.3). Test all disinfected executables to be sure they appear to execute properly. There is still a chance, however, that soft errors will occur.

5.4 Example Applications of Anti-Virus Tools

This section provides hypothetical scenarios for the use of anti-virus tools. For each application, a battery of tools is suggested. There are several ways these tools can be applied to the same scenario; this text represents just one set of rational solutions.

5.4.1 Average End-User

Detailed knowledge of the computer system is not required for the average end-user to perform one's job. Such a user should not be required to obtain detailed knowledge just to use anti-virus tools. This implies that scanners are probably most appropriate for the average end-users. Any other choice will require support from a technical support team or computer security incident response team. Of the remaining tools, the best option is a checksum program. By executing the checksum program regularly, for example weekly or monthly, infections will be detected within a limited timeframe.

Another possibility is to relieve these users of the responsibility of detecting viruses entirely. If a technical support team is already providing other regular services (e.g., backup), the support team can use any combination of anti-virus tools deemed necessary.

5.4.2 Power Users

Power users, those with detailed knowledge of their computer systems, will be better equipped to handle a larger variety of anti-virus tools. A power user is more able to determine whether a change detected by a checksum program is in fact legitimate. Additionally, a power user is going to be better equipped to configure some of the other tools, such as general purpose monitors and access control shells.

5.4.3 Constrained User

If the user is constrained by policy to run a small set of programs against a known set of data files, an access control shell may be the appropriate choice. As an example, consider a data entry clerk who is permitted to run one particular database application and a basic set of utilities: mail, word processing, and a calendar program. An access control shell can be configured so that any changes to executable files by that user are deemed illegal operations. Additionally, if the set of executable files is restricted for the user, it is difficult to introduce a virus into the system. The virus is unable to spread if it can never be executed.

5.4.4 Acceptance Testing

Acceptance testing is a means by which software is verified to be "virus-free" before it is put into daily use. This is usually accomplished by placing the software on an isolated system and performing tests that are intended to mimic every day use. A combination of anti-virus tools is required to adequately perform this function, which must detect both known and future viruses. In particular, a checksum program is most useful. Even if the trigger conditions for the payload are not met, the virus will still most likely attempt to replicate. It is the result of the replication process that a checksum program detects.

5.4.5 Multi-User Systems

Although viruses found in the wild have been limited to personal computer systems, viruses for multi-user systems have been demonstrated in a number of laboratory experiments. Therefore, the potential exists for viruses on multi-user systems. As a result, it is prudent to ensure that the security measures taken on a multi-user system address viruses as well.

Currently, administrators of multi-user systems have a limited number of options for virus protection. Administrators of these systems cannot use monitors or scanners. Since there are no known viruses, there are no signatures to search for or expected virus behavior to detect. An option that is available to administrators of multi-user systems is change detection. Many of these systems are already equipped with a checksum program. Access control shells are another possibility for many systems. Like access control, though, they are not usually designed for virus detection.

5.4.6 Network Server

Network servers present an interesting problem. They can support a wide variety of machines, but may run an entirely different operating system. For instance, a UNIX server may support a network of PC and Macintosh workstations.

The UNIX system cannot be infected by the Jerusalem-B or WDEF viruses, but infected files may be stored on its disk. Once the network server has infected files on it, the workstations it supports will rapidly become infected as well.

Since the viruses never execute on the server, the administrator is limited to static detection techniques such as scanners or change detectors. The nature of network servers allows these tools to be run automatically during off-peak periods.

6 Selecting the Right Tool

Once an anti-virus technique has been selected, an appropriate tool from that class must be selected. This section presents several features to be considered when selecting a specific product from a class of tools.

6.1 Selecting a Scanner

Scanners are implemented in several forms. Hardware implementations, available as add-on boards, scan all bus transfers. Software implementations include both non-resident and resident software for the automatic scanning of diskettes.

Non-resident software is sufficiently flexible to meet most needs; however, to be effective the user must execute the software regularly. Hardware or resident software are better choices for enforcing security policy compliance. Resident scanners may be susceptible to stealth viruses.

Although most scanners use similar detection techniques, notable differences among products exist. Questions that potential users should consider when selecting a scanner include:

- How frequently is the tool updated? A scanner must be updated regularly to remain effective. How frequently updates are needed depends on which platform the scanner is used. Update frequency should be proportional to the rate at which new viruses are discovered on that platform.
- Can the user add new signatures? This can be very important if a particularly harmful virus emerges between updates.
- Does the tool employ algorithmic detection? For which viruses does the tool use algorithmic detection? Algorithmic detection is preferable to the use of multiple signatures to detect polymorphic viruses.
- How efficient is the tool? Users are less likely to use a slow scanner. There can be a significant difference in performance between different search algorithms.
- Does the vendor develop their own virus signatures, or are the signatures based on published search strings? There is nothing particularly wrong with published search strings, but it indicates the level of resources the vendor has committed to the product.
- What is the level of documentation? Some packages arrive with large fact-filled binders; other packages are a single floppy disk with a few ASCII files describing installation and parameters.

6.2 Selecting a General Purpose Monitor

General purpose monitors are usually implemented in software; however, hardware implementations do exist. Hardware versions may be more difficult to circumvent, but they are not foolproof. The following questions should be considered when selecting a general purpose monitor:

- How flexible are the configuration files? Can different parts of the monitor be disabled? Can the monitor be configured so that certain executables can perform suspect actions? For example, a self-modifying executable will still need to be able to modify itself.
- What types of suspect behavior are monitored? The more types of behavior monitored, the better. A flexible configuration to select from the set of features is desirable.
- Can the monitor be reconfigured to scan for additional virus techniques? Are updates provided as new virus techniques are discovered?

6.3 Selecting an Access Control Shell

Access control shells may be implemented in software or as hybrid packages with both hardware and software components. If encryption modules are required, they can be designed as software or hardware. The following questions should be considered when selecting an access control shell:

- What type of access control mechanism does the shell provide and does it fit your security policy?
- If encryption is employed, what is the strength of the algorithms used? In general, publicly scrutinized algorithms are to be preferable to secret, proprietary algorithms where you are depending on the secrecy of the algorithm, rather than secrecy of the key.
- How strong are the identification and authentication mechanisms? [FIP85] provides basic criteria for analyzing the strength of these mechanisms.
- Are the passwords themselves adequately protected? Passwords should never be stored in cleartext.

6.4 Selecting a Change Detector

Due to cost considerations, change detection tools are usually implemented in software. However, hardware implementations do speed the calculation of cryptographic checksums. The following questions should be considered when selecting a change detector:

- What kind of checksum algorithm does the tool use - CRC or cryptographic? CRC algorithms are faster. Cryptographic checksums are more secure.
- Can the tool be configured to skip executables that are known to be self-modifying? Consistent false positives will eventually cause the end-user to ignore the reports.
- How are the checksums stored? Some tools create a checksum file for every executable, which tends to clutter the file system and wastes disk space. Other tools store all checksums in a single file. Not only is this technique a more efficient use of disk space, but it also allows the user to store the checksum file off-line (e.g., on a floppy).

6.5 Selecting an Identification Tool

The following questions should be considered when selecting a scanner for identification:

- How many viruses does it detect? How many different viruses are identified? The former asks how many different viruses are detected, whereas the latter asks how many different names are assigned to these different viruses. If a scanner is using signature strings, signatures can appear in variants. These questions will give some understanding regarding the level of precision provided by a particular tool.
- What names are used by the identification tool? Many viruses have numerous "aliases," so different scanners will produce different names for the same infection. This is especially true with IBM PC viruses. The identification feature of the scanner is only useful if the scanner comes with a virus catalog or uses the same nameset as an available catalog.

Precise identification tools will be more useful when they become available, although the same limitations regarding a virus information catalog will still apply.

6.6 Selecting a Removal Tool

Removal tools are more difficult to evaluate, but the following items may be of assistance:

- Ask for a list of viruses that can be removed, and the general level of accuracy. (For example, "75% of disinfections will result in a working executable.") Ask for a list of viruses that cannot be removed. Use the ratio for the basis of a rough comparison.
- Get a scanner and removal tool that work from the same naming space. The removal tool works on the basis of the virus you name. You need to supply it with the name by which it knows the virus. Matched identification and removal tools are required to make it work.

7 For Additional Information

The National Institute of Standards and Technology's Computer Security Division maintains an electronic bulletin board system (BBS) focusing on information systems security issues. It is intended to encourage sharing of information that will help users and managers better protect their data and systems. The BBS contains the following types of information specific to the virus field:

- alerts regarding new viruses, Trojan horses, and other threats;
- anti-virus product reviews (IBM PC and Macintosh);
- technical papers on viruses, worms, and other threats;
- anti-virus freeware and shareware; and
- archives of the VIRUS-L forum.

Occasionally, the alerts contain signature strings to update scanners. The anti-virus product reviews examine and evaluate specific tools. The papers provide an extensive body of basic knowledge regarding these threats. The VIRUS-L forum has served as a world-wide discussion forum for the exchange of information regarding viruses since April 1988. The past issues are available for download.

<u>Access Information</u>

The NIST Computer Security Resource Center BBS can be access via dial-up or through the Internet via telnet:

Dial-up access:	(301) 948-5717 (2400 baud or less)
	(301) 948-5140 (9600 baud)
Internet:	telnet *cs-bbs.ncsl.nist.gov* (129.6.54.30)

References

[Bon91] Vesselin Bontchev. The bulgarian and soviet virus factories. In *Proceedings of the First International Virus Bulletin Conference*, 1991.

[BP92] Lawrence E. Bassham III and W. Timothy Polk. Precise identification of computer viruses. In *Proceedings of the 15th National Computer Security Conference*, 1992.

[Coh92] Dr. Frederick Cohen. Current best practices against computer viruses with examples from the DOS operating system. In *Proceedings of the Fifth International Computer Virus & Security Conference*, 1992.

[FIP85] Password Usage. Federal Information Processing Standard (FIPS PUB) 112, National Institute of Standards and Technology, May 1985.

[Rad91] Yisrael Radai. Checksumming techniques for anti-viral purposes. In *Proceedings of the First International Virus Bulletin Conference*, 1991.

[Sim92] Gustavus J. Simmons, editor. *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992.

[Sku92] Fridrik Skulason. The mutation engine - the final nail? *Virus Bulletin*, pages 11–12, April 1992.

[Sol92] Dr. Alan Solomon. Mechanisms of stealth. In *Proceedings of the Fifth International Computer Virus & Security Conference*, 1992.

[WC89] John Wack and Lisa Carnahan. Computer Viruses and Related Threats: A Management Guide. Special Publication 500-166, National Institute of Standards and Technology, August 1989.

Index