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 ................................. 1
   1.1 Audience and Scope .......................... 1
   1.2 How to Use This Document ..................... 2
   1.3 Definitions and Basic Concepts ............... 2

2 Functionality ............................... 5
   21 Decision Trees ................................ 5
      21.1 Decision by Saliency Analysis ............... 5
      21.2 Decision by Interaction ..................... 5
      21.3 Decision of Modification ................... 6
   22 Identification Trees ......................... 6
   23 Real Trees ...................................... 6

3 Selection Factors ........................... 7
   31 Array .......................................... 7
      31.1 Decision Trees ............................... 8
      31.2 Identification Trees ......................... 8
      31.3 Real Trees .................................. 9
   32 Ease of Use ...................................... 9
   33 Administrative Overhead ....................... 10
   34 System Overhead ................................ 10

4 Tools and Techniques ....................... 11
   41 Signature Sampling and Agglomerative Decision .... 11
      4.1.1 Functionality .............................. 12
      4.1.2 Selection Fliers ............................ 12
      4.1.3 Sinv ...................................... 14
   42 General Purpose Milters ......................... 14
      4.21 Functionality .............................. 15
      4.22 Selection Fliers ............................ 15
      4.23 Sinv ...................................... 16
   43 Access Control Sells .......................... 16
      4.31 Functionality .............................. 17
      4.32 Selection Fliers ............................ 17
      4.33 Sinv ...................................... 18
   44 Classifiers for Clique Detection ............... 19
      4.41 Functionality .............................. 19
      4.42 Selection Fliers ............................ 20
      4.43 Sinv ...................................... 21
   45 Knowledge Base Virus Real Trees ............... 21
      4.51 Functionality .............................. 21
5 Selecting Anti-Virus Techniques
   51 Selecting Anti-Virus Techniques .................................. 27
   51.1 Choosing Anti-Virus Techniques .................................. 30
   52 Identifying Tools ..................................................... 31
   53 Identifying Tools ..................................................... 31
   54 Selecting the Right Tool ............................................. 31
   54.1 Anti-Virus ......................................................... 31
   54.2 Anti-Virus ......................................................... 31
   54.3 Anti-Virus ......................................................... 31
   54.4 Anti-Virus ......................................................... 31
   54.5 Anti-Virus ......................................................... 31
   54.6 Anti-Virus ......................................................... 31

6 For Additional Information ............................................. 39

References ........................................................................ 41

Index ............................................................................... 43
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 Solicitation No. 16, Computer Viruses and Related Threats: A Management Guide [W86].

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. Delays of anti-virus tools have found 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 owners who wish to select appropriate solutions for their systems. This document begins with an overview of the types of functionality available in anti-virus products and lists selection criteria that 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 does a survey comparing the different classes of tools and suggests possible applications.

The guidance presented in this document is general in nature. This document makes no attempt to address specific computer systems' anti-virus tools. However, at this time, the computer virus problem is not pressing in the personal computer area. Consequently, most types of anti-virus tools are available as personal computer products. As a result, some information will address the specific environment.

---

Certain commercial products are identified in this paper in order to adequately describe being described. In no case does such identification imply recommendation or endorsement by the Institute of Standards and Technology, nor does it imply that the materials are necessarily the best available 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 protection tools are described, as well as identification and removal tools. This information shall 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 detscan 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 shall 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. It 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 NSF Summer Education 300-165 [WJ].
A virus is a self-replicating code agent which must be attached to a host executable. 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 unwanted (and undeletable) 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 might perform its usual, expected function (editing), but the deletions are unwanted and unrequested. Almost any 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. It propagates the copy and copies it to execute; no user interaction is required. Worms often utilize network services to propagate to other computer systems.

A variant is a virus that is created by modifying a known virus. Variants are modifications that add functionality or evade detection. The variant is usually applied to the infection in some manner and the modification is one of irritation. An example would be during the period from June 13 to Tuesday the 13th.

An overwriting virus will destroy code or data in the host program and replace it with the virus code. It should be noted that not all virus programs attempt to create the original host program code and functionality after infection because the virus is more likely to be affected and deleted if the programs to work. An overwriting virus is designed to spread the virus code to the physical and virtual environment and protect the original code from alteration by the virus.

A self-recognizing program is a technique whereby a virus determines whether or not an executable is already infected. The program usually inserts binary code for a particular value at a known position in the executable. Self-recognition is required if the virus is to add multiple infections of a single executable. Multiple infections can cause excessive growth in size of infected executable and cause excessive storage space, contributing to the detection of the virus.

A resident virus installs itself as part of the operating system or an executable host program. The virus will remain resident until the system shuts down. Once installed in memory, a resident virus is available to infect all executable files that are accessed.

1 An executable is an abstraction for programs, command files and other objects that can be executed. On a DOS PC, for example, this would include batch files and EXE files.
As a stealth virus is a virus that attempts to evade detection by remaining undetected in infected files. To achieve this, the virus infects system files which contain 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 execute the original, uninfected host program.

A 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 is executed, replicating or becoming resident. The virus body will include an encryptor to apply during replication. A variable encrypted virus will use different encryption keys or encryption algorithms. Encrypted viruses are easy to detect and study since the researcher can 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, interchanging the order of individual instructions, or choose from a mix 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 not been released to the public. These include the viruses 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. Anti-virus vendors and virus writers cooperate to determine the extent. 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 fabrication.
2 Functionality

Anti-virus tools perform three basic functions. These may be used to detect, identify, or remove viruses. ² Detection tools perform active detection, passive detection, or reactive detection. This is, they detect a virus before it executes, during execution, or after execution. Identification and remedial tools are more straightforward in their application rather 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, resting 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 executable files and the executable files of programs that execute them. These tools can be used in passive or reactive fashion. They can be used to detect infected code before it is introduced into a system by testing all executable files before installing software on a system. They can also be used in a 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 inhibit 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 occurs.

²A few tools are designed to prevent infection by one or more viruses. These are limited to Section 4.7.2, Inoculation, due to their limited application.
2.1.3 Detection of Modification

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

Detection of replication may also identify other security problems, such as the installation of Trojan horses. Note that this type of detection will 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 point towards other types of dangers and appropriate cleanup procedures.

2.3 Removal Tools

In many cases, one virus has infected, it is found in memory system or numerous executables on a single system. Recovery from original deletes or data loss can be a tedious process. Removal tools attempt to efficiently restore the system to its uninfected state by reversing 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 detect. Ease of use describes the typical user's ability to install, maintain 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. The factors are introduced below in depth discussions of these factors are in subsequent sections.

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 making the background abilities of the system 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 of the user. Errors will require assistance to interpret results from the tool; this can place a large burden on the organization's support staff. It is important to choose tools that your organization has the resources to support.

System overhead is important from a strict security point of view. Accurate detection, identification or removal of the virus is the important part. However, most of these tools are intended for end-users. If a tool is slower, causes other applications to stop working, end-users will dislike it. This, attention needs to be paid to the tool's ability to work quickly and to coexist 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 mistakes. It is the type of error 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 these users with the system knowledge to intelligently verify the information supplied by an anti-virus tool, array is not as great a concern. Unfortunately, many computer users are not prepared for such action. For such users, a virus infection is seen as frightening and very confusing. If the anti-virus tool is supplying false information, this will raise a false alarm warning. For these users, the overall error rate is not 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 continuous improvement of anti-infection techniques. As a result, the detection process can result in errors of two types: false positives and false negatives.

With a detection tool identifies an unaffected 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 attempts 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 result program may not work, changing the executable or will improperly change the program's reading, useful code. Either scenario results in confusion for the user and lost confidence.

With 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. Detecting 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.
3.2 Ease of Use

As an example, consider two ICOMP identification tools. The first tool consists of the set of ICOMP viruses as 30 distinct viruses. The second consists of the same set with 900 viruses. 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 virus attempts to remain with high precision. At any instance a virus as another variant of the virus for a variety of reasons. The variant may be new or analysis of samples may be 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

Remove tools attempt to restore infected executable to their uninfected state. Remove is successful if the executable, after disinfection, makes 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.

Hard failure occurs if the disinfected program will no longer execute the removal program terminates without removing the virus. Such a severe failure will be obvious to detect and occur for a variety of reasons. Executables infected by the existing 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 from the actual infector.

Remove 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 have a few errors, but the user can be certain of that. A 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 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.
3.3 Administrative Overhead

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

This factor considers difficulty of installation, update requirements, and support leads re-
quired by endusers. These functions are often the responsibility of technical support staff
or system administrators rather than the enduser. Note that an enduser without technical
support may 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 resources 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 that can be applied to the anti-virus effort. This section will address the following anti-virus techniques:

- Signature scanning and algorithmic detection
- General purpose scanners
- Access control tools
- Databases for change detection
- Knowledge-based remedial tools
- Research efforts
  - Intrusion binary analysis
  - Precise identification

For detection of viruses, there are five classes of techniques: signature scanning and algorithmic detection, general purpose scanners, access control tools, databases for change detection, and heuristic binary analysis. For identification of viruses, there are two techniques: scanning and algorithmic detection, and precise identification tools. Finally, remedial tools are addressed. Remedial tools can be found in three forms: general system utilities, single virus disinfectors, and general disinfecting programs.

4.1 Signature Scanning and Algorithmic Detection

Among the 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 some limited damage 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 are 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

Sannes are linked intinnally to the detennination of known viruses. Here, as a side effect of the basic technique, some known viruses may also be detected. They are also identification tools, although the methodology is precise.

Sannes examine executables (e.g., *.EXE, *.COM, * in a MS system) for indications of infection by known viruses. Detection of a virus produces a warning message. The warning message will identify the executable and the virus or viruses that it is infected. Detection is usually performed by signature matching, special cases may be detected by algorithmic methods.

In signature scanning an executable is searched for selected binary code sequences, called virus signatures, 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 variability.

Single point scanners are the easiest of relative position to the virus signature. The code sequence is expected at a particular position in the file. It my not be detected if the position is wrong. By changing the relative position with the signature string, the number of false positives is greatly reduced. As a result, these scanners can be more accurate than bind scanning at that position.

Hybrid viruses, such as those derived from the EXPC (extension engine) [Ste], do not have fixed signatures. The virus is self-replicating or partially encrypted. We see many scanners use multiple signatures to detect possible infection by these viruses. Algorithmic detection is a more powerful and comprehensive approach for these difficult viruses.

4.1.2 Selection Factors

Array

Sannes 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 subvirus, a detection failure is unlikely with a sanner. A good sanner should be able to some extent identify any virus you are likely to encounter. Sannes have other problems, though. In the detection process, both false positives and false negatives can occur.

False positives occur when an uninfected executable inducts a byte string adding a virus signature into the sanner's database. Sannes do not test their signatures against libraries of compiled and uninfected software to reduce false positives. For additional assurance,
4.1 Signature Scanning and Algorithmic Detection

The developers perform a 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 bad signature definitions. If a virus is moving around at the time the scanner executes, the virus may hide itself. False negatives can also occur when the systems becomes infected by a virus that was unknown at the time the scanner was built.

Signatures are also prone to misidentification or lack of precision in many. Misidentification will usually occur when a variant or a derivative virus is encountered. An example, a variant may simply have a different pattern in that Jerusalem city is present. This can occur because viruses are both Jerusalem variants and share much of their code. After signature input the scanner also ignore “Jerusalem” in file names. This is accurate, but rather desperate.

Like of Life

Signatures are very easy to use in general. You simply execute the scanner and it provides concise results. The scanner may have a function 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.

Administrative Outlook

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 defined for distribution of updates. As an example, a IS that is more than a few weeks old will not detect new virus variants. (It may detect, but misidentify some variants.) Today updates are crucial to the effectiveness of any signature-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: only viruses are correctly identified. Detecting false positives will usually require some assistance from technical support. This level of support may be available from the vendor.

Efficiency

Signatures are very effective. There is a large body of knowledge about searching algorithms so the typical scanner executes very rapidly. Passive 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's release) and any such detection will result in misidentification. Scanners enjoy an especially high level of user acceptance because they meet 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 richness of the signature database. For viruses requiring algorithmic methods, the quality of the algorithm 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 effective algorithms.
- Detection of scanners usually does not require any special knowledge of the computer system.

The major limitations of scanners are:

- Scanners only look 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 creation of the payload. Trojan horses by actively intercepting malicious actions.
4.2 General Purpose Monitors

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 suspicious activity that may be viral-like or otherwise malicious. However, since a virus is a single stream, there is a very real possibility that legitimate programs will perform similar actions, causing the alarms to sound.

The design of such a system begins with a model of "malicious" behavior, then detects rules that differ from and halt attempts to perform those actions. These rules operate as part of the operating system.

4.2.2 Selection Factors

Accuracy

Monitoring programs will detect processes that are not in the set of suspicious behavior and flag unnecessary alerts. These alerts are usually valid assumptions. Many viruses may utilize normal tools that may fall outside of the set. Such a virus would not be detected by the monitoring program.

The techniques used by monitoring tools to detect viral-like behavior are also not foolproof. Standard antivirus programs detect symptoms so a program can usually circumvent any control features of the operating system. As a part of the operating system, monitoring programs are vulnerable to this as well. There are some viruses with cheat 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 if false positives, if the configuration is not optimal.

The average user may not be able to determine that programs should modify files, but programs should not. The high rate of false alarms can discourage such a user. As a result, the monitor will be turned off or ignored altogether.
4.2.3 Summary

Monitoring software may be difficult to use if it detects 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 starts performing activities the monitor does not depend on. We note, some viruses have succeeded in attacking monitored systems by gumming up 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 access control policy for the system. If a policy is disabled in some programs and the data files they may access. The access control shell will sound an alarm only if a user attempts to access or modify a file with an unauthorized software package.
4.3 Access Control Shells

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 rely on another version of the operating system to encrypt the credentials. Note that many 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 systems, then converts that description into a set of critical files and the programs which may be used to monitor them. The administrator may also select the files which require encryption.

For instance, a slipping disk might be authorized to access the inventory database with a partial program. However, that same disk may not be allowed to access the database directly with the database management software. The disk might be authorized to access the audit records generated by the trusted application, 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

Array

Access control shells, like monitoring tools, depend on 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.

Then with the access control shell, a well-behaved virus can modify any program that its user programs authorized to modify. To reduce the risk that any program 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 may 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 large, such a tool may 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 create whatever programs they require against the required data. If the access control table prevents the operation, they must go through the administrator to obtain additional privileges.

Efficiency

A access control tool modifies the operating system so that additional security procedures are performed. This implies some amount of overhead when any program is executed. The overhead may be substantial if large amounts of data must be encrypted and reencrypted upon each access.

Administrative Overhead

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

4.3.3 Summary

Access control tools 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 affect the replication of a virus or adaption of a Trojan horse.

The tool may incur high overhead processing costs or be expensive due to hardware requirements. Both false positives and false negatives may occur. False positives will occur when the access table does 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

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 since executables are self-modifying. Additionally, in a software development environment, executables may be modified by compilation. Thus, two copies may be detected as different, which is an improper 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 one on the (theoretically) permanent part of a baseline 3 for testing. During subsequent executions, the program compares the updated executable with the baseline executable. Change in the executable indicates a modification to 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 covers every executable in its entirety on the system, it is used in a proper manner, a virus cannot evade detection.

Change detection tools employ two basic mathematical techniques: Optic Identity Checks (OIC) and cryptographic checksums.

OIC Idents

OIC identifiers are used to verify integrity of packets in networks and other types of communications between computers. They are efficient and well understood. OIC-based identifiers are not extremely secure; they are based on a known set of algorithms. Therefore, they can be broken (the particular algorithms are known) by a program if it can find the identical OIC data.

OIC identifiers, like all change detection tools, can only detect that a virus has replicated. Additionally, the executable must be present in the baseline.

Cryptographic Checksums

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

3The original file names and their corresponding checksums.
used for efficiency. These techniques are sometimes used in conjunction with other procedures to detect system problems. These techniques are message digesting and hashing.

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

4.4.2 Selection Factors

Array

Hyphenated names, message, and digests during programs should detect every virus. This is, there are no false positives with message detection. Message detection can result in high numbers of false negatives as well. Each scan can falsely indicate a scan of every file containing executable code. If these files are deemed as they should be, a change in configuration will trigger a false alarm. Additionally, the system cannot be virus-free without the detection of all and resident viruses may fool the message detection software.

Execution

Message detection software is easy to install and it requires no setup. 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. The system administrator or high level of support can be required for the system administrator, however. A qualified staff member must be available to determine whether or not a change to a partial executable is due to a virus or simply a result of self-modification.

*Discussion of cryptographic terminology is beyond the scope of this document.*
4.5 Knowledge-Based Virus Removal Tools

Efficiency

Some detection does not impact general systems. There is, however, some storage overhead for the baseline detection. These are best stored offline with the detection program.

The additional detection is computationally intensive, the potential functions not be calculated on at least a portion of the executable. The 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-replication, or modification of the baseline. An untrusted 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 attenuated removal of virus infection is knowledge-based removal tools. These tools attempt to reverse the actions of a virus file to a file. After analyzing a particular virus to determine its effects on an infected file, a suitable algorithm is selected for disinfecting files. This is possible with at least a single virus. These single virus disinfectors are usually designed as the result of a particular virus outbreak of a virus. Once detected are general virus removal programs containing removal algorithms for several viruses.

4.5.1 Functionality

Knowledge-based 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 overwriting virus, removal is not possible. The information that was overwritten cannot be restored.
The most critical piece of information in the recall process is the identity of the virus itself. If the recall program runs under a different operating system than the one under which the virus was run, 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. True error rates are fairly high, but are not errors. This is a result of incomplete information regarding the virus and the lack of quality assurance among virus writers. Additionally, recall techniques tend to fail when a system file has been infected multiple times (i.e., by the same virus more than once, or by one than one virus).

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

4.5.3 Summary

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

Certain viruses (e.g., creeping viruses) always come impenetrable. Some viruses are initially run-time viruses, and can be disinfected by running them in a separated file. Others are not. Some viruses may not be removed from the infected system. In such a case, the file length or size of the disinfected system must be compared with the infected system. If the size of the file is different, the virus may not be removed. In such a case, it is impossible to predict the behavior of the disinfected program. This is why precise virus readers generally do little recall programs and discourage their use.

4.6 Research Efforts

The following sections describe research areas in the anti-virus field. Most of these, based on techniques developed in those and other areas, may be available in the near future.
4.6 Research Efforts

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.

Reliability

Binary analysis tools examine an executable for virus-like code. If the tool 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 Errors

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 does not use these techniques common to viruses.

Binary analysis tools are fairly easy to use. The user simply specifies a program directory to be analyzed. Analyzing the results is more difficult. Sifting 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 selectivity is appropriate for daily use on a large number of files. It is not appropriate for one-time use on a small number of files, as in acceptance testing.

Heuristic analysis programs 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 current detection software. A useful system monitor, it is difficult to detect suspicious behavior which prevents false positives and false negatives. However, these tools have been used successfully to identify executables infected by "wild" viruses in a few small outbreaks.

Heuristic binary analysis is still experimental in nature. Initial results have been sufficiently encouraging to suggest that some anti-virus packages could implement tools to augment more traditional techniques.
4.6.2 Precise Identification Tools

Precise identification tools are a must by which viruses are rid with a much higher degree of accuracy. These tools are intended to aid detection tools. One a virus has been detected, a precise identification tool would be included in order to more accurately identify the virus.

Precise Identification

Virus scanners, currently the most common virus detection method, generally employ signature scanning to detect and identify viruses. This method, however, can lead to false identification. The signature that the scanner matched could appear in more than one variant of the virus. To aid in identification, the precise virus signature is not just a subset of the virus (i.e., the signature). It is rather feasible and desirable for identification software to be distributed containing the code to all viruses it can detect. Therefore, precise identification tools utilize a "virus map" to represent the contents of the virus. The virus map contains describes values for all contact parts of the virus code. The map skips over sections of the virus that contain file information such as text or system dependent data values.

If the virus scans match the corresponding portions of the program, the program is almost certainly infected by the virus corresponding to the map. If one of the maps in the database corresponds the program is infected by a new virus (or is uninfected).

Selection Rates

The quality of the results produced by a precise identification tool is dependant 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 virus.

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

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

Summary

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

Readers 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 simulation, are included for completeness. These tools can be used to provide a measure of functionality. In general, however, these tools are worse than general anti-virus tools.

4.7.1 System Utilities

Some viruses can be detected or removed with basic system utilities.

Diagnostic tools for viruses and some Minitool viruses can be used with system utilities. System utilities can also be used to detect viruses by searching for virus signatures. These tools have a rather limited form, 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 viruses that cannot be disinfected with these tools.

Virus parasites, disinfected with system utilities will produce dependable results. A minimum amount of knowledge is required about the system and the virus itself, though. This technique can also be very tedious if a large number of systems are infected.

System utilities are an indirect 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

Automatic identification is frequently possible. Certain classes of viruses (e.g., overwriting viruses) always change the executable beyond all hope of repair. Others modify the executable in a less obvious way. Only viruses that are extremely well-behaved can be disinfected automatically. Similarly, detection with system utilities has limited application.

5 Two examples of these system utilities are Norton Utilities for the PC.
4.7.2 Immunization

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

Since viruses may have their self-recognition codes in overlapping locations, the number of viruses that can be immunized against simultaneously will be small. Thus, one way to create a new variant is to change the self-recognition code. This, then, this technique will often fail when tested by new variants of the viruses immunized against.

Immunization 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 having three or four outbreaks of a particular virus from network R5, immunization 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 where electronic information is shared. Every organization shares information in some way and is a potential victim of a virus infection. No organization should have no 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 when viruses are few or only theoretically possible.

The use of such tools is generally not required. 6 It may be desirable in situations where a single person or a small team is tasked with dealing up after an infection or where high coercivity 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 point 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. These viruses constitute this small set on a very geographically. These viruses can be distinct or different categories, due to the places in which they travel. The case, different hardware platforms will be at risk from different viruses.

Identification tools may be valuable to a larger set of viruses. This set may be obtained by mining 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. In this case, the only alternative to removal tools is to format the disk.
business. Organizations with outdated or installed versions of virus writers are partially at risk [100] and are more likely to encounter new viruses.

Risk from new viruses is an important consideration. Savers are limited to their detection when new viruses occur. Detection tools are designed to detect any virus. If your organization is at high risk from new viruses, savers should not be the sole detection technique employed.

Another important criterion is the number and type of errors one should tolerate. The tolerance for a partial type of error in an organization will vary according to the application. The table shows the types of errors which should be expected. An estimate of the frequency with which each type of error is encountered (Infrequent, Frequent, or Never) is also given. Each class of tool and error type. All anti-virus tools are subject to errors, but their relative frequencies vary widely. Savers probably have the lowest overall error rate.

Table 1: Types of errors.

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

Table 2 includes a general evaluation of 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 enduser?
- Does your organization have a support staff to assist users with technical problems?

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:
### 5.1 Selecting Detection Tools

<table>
<thead>
<tr>
<th>Detection Tool</th>
<th>Criteria</th>
<th>Scanner</th>
<th>Checksum</th>
<th>Binary Analysis</th>
<th>Generic Monitor</th>
<th>Access Control Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Easy of Use</strong></td>
<td><strong>Very Good</strong></td>
<td>Requires no special knowledge of the system</td>
<td>Average</td>
<td>Easy to use; results may be difficult to interpret</td>
<td>Poor</td>
<td>Easy to use; results must be verified</td>
</tr>
<tr>
<td><strong>Administrative Overhead</strong></td>
<td><strong>Low</strong></td>
<td>Requires frequent updates. Little add’l support req’d</td>
<td>Low</td>
<td>No updates req. Assist in interpreting results</td>
<td>High</td>
<td>Few updates. Much verification of results</td>
</tr>
</tbody>
</table>

**Table 2: Read requirements**

If several tools still appear to be candidates, consider the functionality of these tools beyond virus detection. Viruses are only one of the 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.

The added functionality of these may not be needed by your organization and should accordingly be evaluated by your organization.

Additional functionality are outlined in Table 3.

<table>
<thead>
<tr>
<th>Detection Tool</th>
<th>Scanner</th>
<th>Checksum</th>
<th>Binary Analysis</th>
<th>Generic Monitor</th>
<th>Access Control Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Additional Functionality</strong></td>
<td>Identification; May also detect known trojan horses</td>
<td>Detection of trojan horses and altered data</td>
<td>Detection of trojan horses</td>
<td>Detection of trojan horses</td>
<td>Enforcing organizations security policy</td>
</tr>
</tbody>
</table>

**Table 3: Additional functionality**

The final selection criteria to be considered involves the tool detect viruses. Passive detection allows the user to keep viruses off a system by testing incoming software.

---

7Some scanners can also detect known Trojans, horses.
tools only allow one chance of detecting a virus (upon initial introduction to the system). Active detection tools interfere during the replication phase itself. Passive detection tools can be used anytime after a virus has entered the system. Additionally, passive tools are not as rigorous in their demands on system performance. Table 4 shows which detection tools affect viruses.

<table>
<thead>
<tr>
<th>Detection Tool</th>
<th>Scanner</th>
<th>Checksum</th>
<th>Binary Analysis</th>
<th>Generic Monitor</th>
<th>Access Control Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Executable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replication</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Infection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Which tools affect?

5.1.1 Concerning Detection Tools

To not completely protect all data, it is best using tools with different methods and protect against different classes of viruses. For instance, when used together a scanner and a decompiler 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 not detected by the scanner, but it will be detected by the decompiler.

The two tools do have different functionality (see table 3) to form the not comprehensive security package. For instance, the combination of a decompiler and an access control shell will also detect Trojan viruses and enforce organizational security policy in addition to virus detection. On the other hand, doing a binary analysis to a system that already employs decompiling will not provide additional functionality.

If you use two scanners, be sure that they use different search strings. Number of tools are based on lists that are used to scan a computer. The tools currently utilize the same public domain signature databases. To 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

Cautiously scanned are the only effective means of identifying viruses. As discussed in Section 3.1.2, the array to which scans are identified can vary. In the future, precise identification tools should greatly increase accuracy.

5.3 Removal Tools

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

In large organizations with high connectivity, automatic 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. Non-knowledge-based removal tools, disinfection of executables is not always reliable (see Sec. 3.3). Yet all disinfectant 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 presents 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 the job. Such a user should not be required to obtain detailed knowledge just to use anti-virus tools. This implies that scans are probably not appropriate for the average end user. At other times, all require expert from a technical support team or computer security incident response team. One method for the remaining tools, the best option is a desktop program. By creating the desktop program regularly, for example, weekly or monthly infections will be detected within a limited timeframe.

Another possibility is to give the users of the responsibility of detecting viruses only. If a technical support team is already providing other services (e.g., backup), the support team may use any additional anti-virus tools deemed necessary.
5.4.2 Rover Users

Rover users, those with detailed knowledge of their computer systems, will be better equipped to handle a large variety of anti-virus tools. A rover user is more able to determine whether a change detected by a desk setup program is in fact legitimate. Additionally, a rover user is going to be better equipped to configure some of the other tools, such as general purpose routers and access control skills.

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 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, e.g., 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 everyday use. Activation of anti-virus tools is required to adequately perform this function, which must detect both known and future viruses. In particular, a desk setup program is not useful. Even if the trigger conditions for the payload are not set, the virus will still most likely attempt to replicate. It is the result of the replication process that a desk setup program is.

5.4.5 Multi-User System

Although viruses for the desktop 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 are as well.

Currently administrators of multi-user systems have a limited number of options for virus protection. Administrators of these systems cannot monitor or scan. Since there are no known viruses, there are no signatures to search for or expected virus behavior to detect. Therefore, the ability to detect multi-user systems is largely dependent on.

My
5.4 Example Applications of Anti-Virus Tools

of these systems are already equipped with a disk program. Access control shells are another possibility for any system. Like access control, though, they are not usually designed for virus detection.

5.4.6 Network Server

Network servers present an interesting problem. They support a variety of natures, but run an entirely different operating system. For instance, a Unix server may support a network of P-Card Macintosh workstations.

The Unix system cannot be infected by the Maryland Unix viruses, but infected files may be stored on its disk. If the network server has infected files on it, the workstations it supports will rapidly be 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 cards, scan all bus transfers. Software implementations include both resident and resident software for the automatic scanning of diskettes.

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 capture. Resident scanners may be more efficient to stealth viruses.

Although not scanners use similar detection techniques, notable differences among products exist. Options 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. The frequency should be proportional to the rate at which new viruses are discovered on that platform.
- Can the user add signatures? It is very important if a particularly harmful virus emerge 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.
- Is the tool effective? Users are less likely to use a slower tool. 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, fat-filled manuals, other packages are a single floppy disk with a few ADF files describing installation and products.
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 detect, but they are not foolproof. The following questions should be considered when selecting a general purpose monitor:

- How visible are the configuration files? Can different parts of the monitor be disabled?
  The monitor be configured so that certain executables 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. An ideal configuration to select from the set of features is desirable.
- Can the monitor be reconfigured to scan for additional virus techniques? As updates provide 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 elements. If encryption capabilities 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 mechanisms does the shell provide and does it fit your security policy?
- If encryption is employed, what is the strength of the algorithm used? In general, publicly available algorithms are 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? [HRB] 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 digests.
The following questions should be considered when selecting a change detector:
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 viruses are assigned to those 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.
- Will names be used for the identification tool? Many viruses have various "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 method as an available catalog.

Besides 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, "80% of detections will result in a working executable.") Ask for a list of viruses that cannot be removed. The ratio for the basis of a rough comparison.
- Get a scanner and removal tool that work from the same mirror. The removal tool works on the basis of the virus you receive. You need to supply it with the names by which it knows the virus. Mike did; 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 and Micros);
- technical papers on viruses, worms, and other threats;
- anti-virus freeware and shareware; and
- archives of the VRS List.

Basically, the alerts contain signature strings to update scanners. The anti-virus product review examine and evaluate specific tools. The papers provide an extensive body of basic knowledge regarding these threats. The VRS List has served as a worldwide discussion forum for the exchange of information regarding viruses since April 1993. The past issues are available for download.

Access Information

The NIST Center for Computer Security Research (NIST-CS) can be accessed via dial-up or through the Internet via telnet:

Dial-up access: (31) 985-5417 (200 baud or less)

(31) 985-5420 (9600 baud)

Internet: telnet cs-bbs.ncsl.nist.gov (129.65.3.1)
References


Index

access control skills, 11, 16-18 35
arsenal 79
administrative control 7, 10
digital detection see samus
baseline, 19
BI 3
bulletin board systems see BI
change detection 11, 19-22, 35
desar 19
desar see change detection
OC 19
cryptographic desar 19
Globally Refined Clock see OC
detection 5
case of use, 7, 10
centable 3
guard process monitors, 11, 14-16, 35
human binary analysis, 11, 23
identification 6
invocation 11, 26
integrity dating 6
keystore digesting 20
NSTSPID 16, 1, 2
precise identification 11, 24 37
read 6
samus, 11-14, 35
signatures, 12
self-recognition 3
signature sensing see samus
system overload 7, 10
Talant 3

viat, 3
vino 3
crypted 4
in the wild 4
mooring 3
orating 3
polyphonic 4, 12
research 4
residt 3
stealth 4
verbatim-crypted 4
VRSI 39
ver 3