Quick Training Handbook for the VFind Security ToolKit(s)

The Leader In Total Security Solutions For
Linux, UNIX and Mac OS X

- Anti-Virus
- Baseline Control
- Crypto Cracking
- Pattern Analysis
- Computer Forensics
- Operational Monitoring
- Self-Healing and Repair
- Command Line or Web Based

Providing Security Tools Since 1988

Quick Training Handbook
For the VFind Security ToolKit(s)

This document can be used for an instructor-led class or as an independent study. Permission to print/copy granted for educational use, provided the document is printed/copied in whole including the copyrights and is not sold for more than the cost of reproduction.

Training Times

Abridged	4 hours
Full	6 hours

Version 3, May 3, 2005

By
Peter V. Radatti
radatti@cyber.com

Third Printing Version 3.00 May 3, 2005
Second Printing Version 1.01 April 21, 2004
First Printed April 12, 2004

This document is based upon the following copyrighted white papers located at www.cybersoft.com:

Secrets of the VFind® Security ToolKit Professional Plus, Version 1.00 Version of May 2000
Who, What, Why, Version of March 2001
The Plausibility of UNIX Virus Attacks, Version 2 of April 1996
Special Attributes of the VFind ToolKit, Version of May 1997
Non-Technical Cost Justification for CyberSoft VSTK and VSTK/P Products, Ver. of Sept. 2001
Customer Case Study Number 2, Version of January 2001
Customer Case Study Number 3, Version of January 2001
Use of the Avatar and CIT Tools for Centralized Distribution and Control, December 2003
Recommended Use of VSTK on a File System (Flash Presentation)
Anti-Virus for Multimedia Publishers version of May 1996

CyberSoft Operating Corporation
1508 Butler Pike • Conshohocken, PA 19428
+1 610·825·4748 • Fax +1 610·825·6785
www.cybersoft .com

Preface

By design, this document does not organize all of the subject matter into groups. There are several reasons for this, including the fact that a good deal of the material can belong in more than one place. In addition, I have attempted to organize the material in a way that makes for easier assimilation but, not necessarily, easier reference. You should also note that there is a self-directed laboratory with CD-ROM that is complementary to this document.

Definitions

In this paper, UNIX, Linux and Apple Macintosh OS X are used interchangeably. When using the name UNIX, we are referring to all UNIX like systems. The words Microsoft and Windows are used to mean Microsoft Windows unless referring to a non-Microsoft Windowing system such as X-Windows.

The word Virus is used to mean all forms of attack software, examples of which are viruses, worms, Trojan Horses, droppers and hacks.

VSTK refers to the VFind Security ToolKit. VSTKP refers to the VFind Security ToolKit Professional. VSTK/P is shorthand that refers to both the VSTK and VSTKP.

Legal Notices

A Short History of CyberSoft and VFind

Prior to starting CyberSoft, Peter Radatti spent over 15 years working in the aerospace and military industries for companies like Control Data Corporation (CDC) at the Naval Air Development Center, General Electric (GE) at the Valley Forge Space Center and Systems Research Applications (SRA) in Moorestown, New Jersey. He got the idea for CyberSoft's first commercial product while at General Electric. GE had no interest in the product which was an anti-virus scanner for Sun Microsystems UNIX computers. Undeterred, he purchased a Sun workstation and completed the project at home. When the program was completed, he was going to enter it into the public domain but General Electric suddenly took an interest in the program and asked him to wait.

GE's interest in the program, called "VFind," made Radatti realize that he might have something of serious commercial value. At the time, his mother Marie Radatti worked for Bell Atlantic. Bell Atlantic also showed interest in the program. With two major corporations showing an interest in the program, Mr. Radatti decided to spend his savings to demonstrate the program at the UNIX

Expo International in October 1991 at the Jacob K. Javits Convention Center in Manhattan. His trust paid off and he made enough of a profit to seriously launch his company, CyberSoft, Inc.

CyberSoft, Inc is a corporation registered in Pennsylvania. It was founded July 29, 1988 and incorporated January 1, 1993. The first product was shown in 1990 but publicly announced at the UNIX Expo International in New York City on October 30, 1991. On July 1, 2002 CyberSoft, Inc (CS) was reorganized as an intellectual property holding company, and all operations were spun off to a new corporation called CyberSoft Operating Corporation (COC).

CyberSoft Operating Corporation
1508 Butler Pike
Conshohocken, PA. 19428-1322
Telephone: +1 610/825-4748
Fax: +1 610/825-6785
Website: www.cybersoft.com

Different Versions of the VFind Security ToolKit and their Components

The VFind Security ToolKits are general-purpose computer security-oriented products. When you think of them, consider them as toolboxes filled with various tools that you could use to solve problems or build things. While these tools were all created with computer security as the primary problem set, many customers have used them to solve problems that are unrelated to computer security. As you go through this course and gain familiarity with the individual tools and the tool concept, you will begin to see ways to use these tools to solve problems that you are facing. If you are in a structured, instructor led course, you are encouraged to share problem definitions with your instructor who can help you find ways to use these tools for problem resolution.

In addition to the VSTK/P packages, there are some subset packages available. These packages are documented on our websites at www.cybersoft.com, www.cybersoftinternational.com, www.cyber.com and www.safeinternete-mail.com. Generally the subset packages like the CIT/THD Combination Package and SafeInternetEmail are goal specific, while the VSTK packages are general-purpose packages, which can be used to build solutions for a variety of problems including those problems that are not specifically computer security oriented.

There are several packaged versions of VFind ToolKits. The first package is VSTK (VFind Security ToolKit). The second package is VSTKP (VFind Security ToolKit Professional). The third package is VSTKCW (VFind Security ToolKit for Cyber War). The VSTK package is the basic product and completes the needs of 80% of all users. The VSTKP product contains everything in the VSTK product, with the addition of the Avatar tool. VSTKCW contains everything in VSTKP with the addition of tools needed for a cyber war environment. VSTKCW is intended for a very hostile or very high availability environments where downtime is unacceptable. The fourth and fifth packages are Turbo versions of both VSTK and VSTKP (VSTK-T and VSTKP-T). The sixth and seventh packages are Turbo with Proxy combinations for both VSTK and VSTKP (VSTK-TP and VSTKP-TP).

It is necessary to explain the version number system used by the VSTK product line. Version numbers have been a point of confusion for users in the past but are really simple to understand once it is explained. Each tool has its own version number. Version numbers are increased every time a change is made to a tool. The version numbers of every tool is different because the tools

have been introduced at different times and have different development schedules. Older tools will have higher version numbers because of enhancements made over the course of years. VFind started at Version 1.00 in 1990 and is currently at Version 15.4.1 in April 2005. Tools are not released individually. They are only released as a ToolKit; consequently, each ToolKit needs a version number. Version numbers of ToolKits are serial and incremental. The April 2005 version of the ToolKits will be Version 169. Every ToolKit version has specific tool version numbers associated with it. If a change is made to anything in a ToolKit then a new ToolKit is created with an incrementally increased version number. We speak of ToolKit version numbers as Package Versions.

Some Basic Theory

There are three basic ways to secure computers. Almost everything else is an extension of one of these methods. The first is compartmentalization. Compartmentalization basically means that the system is isolated and locked up. Everything else is a compromise with various strong and weak points. The second method is called fortressing. This process attempts to make a fort or castle out of the computer by hardening the security. It keeps the bad guys out. Unfortunately, modern computers are network connected and are very complex. Even with the best fortressing technology known, someone, somewhere will penetrate the fortress and gain entrance. The third method is reactive security. Reactive security assumes that unauthorized people are going to get in, or authorized people are going to make an unauthorized modification to the baseline or worse, unauthorized users will make unauthorized modification. It then deals with that issue. *The most important part of reactive security is baseline configuration control.* In other words, keep the system baseline where it belongs, keep the system under control and keep it performing its function as long as possible in a hostile environment.

Each of the three methods has drawbacks. The best overall solution is to combine all three. This is a common solution sometimes called a ring security system. The outermost ring provides compartmentalization to the extent possible. On a network this may be provided by configuration of the system, router and firewall and by physical security. The second ring is fortressing. Remove all known exploits from the system and harden the configuration to make it as hard as possible to gain entry or control while allowing enough functionality to allow the system to continue performing its functions. The innermost ring is the reactive ring. Maintain the baseline configuration at all costs. Correct unauthorized changes to the baseline. Remove all attackers that gain entry. Automate the process so that discipline is maintained.

There is also a “fourth” method, which is really a modification of the fortressing and reactive methods. I consider this a fourth method because it is very important, and yet the most complex and expensive to implement. This fourth method is Update Management. Update Management is the process in which all manufacturer and organization mandated updates are installed. On a single system with an automated or semi automated update system, with a trusted user, this is not a problem. On networks of thousands of systems with users that are both trusted, untrusted, trained, untrained, capable and not capable of using the update system this becomes one of the most expensive and difficult computer security tasks. In addition it is also an area of large risk. If an update causes a critical process to fail and an automated update system has installed that update on thousands of systems the result can be a major catastrophe. Above and beyond this, every manufacturer has their own update management system. A typical computer may have dozens of software packages installed each with a different update system.

Even with all of these problems, update management becomes a critical problem because the most common reason for an update to be released is reactive. Someone found a security hole and it must be blocked. One of the most common reasons that computer viruses and worms spread so quickly on the Internet is because there are millions of systems that have not been updated or a new hole has been discovered. The Internet is susceptible to the agricultural problem of blight. Once a new attack comes along, it can wipe out an entire crop.

Update Management will become a critical computer security tool in the future. In fact, large organizations will not be able to wait for a manufacturer update to a problem and need complete and proactive control over their update processes. Lightning fast in-house response may mean the difference between victory and defeat in an endeavor.

In addition to demonstrating ways to use the VFind Security ToolKits to solve the first three problems we will also show you methods of using these same tools to solve the Update Management problem.

CyberSoft Tools and Their Use in the Tertiary Security Ring Model

Tool	Compartment (Lock it up)	Fortressing (Harden it)	Reactive (Baseline Control)	Available For UNIX	Microsoft
VFind	yes	yes	yes	yes	yes
CIT	no	no	yes	yes	yes
THD	no	no	yes	yes	yes
UAD	yes	no	yes	yes	yes
NTI	yes	yes	no	yes	yes
NTI-Crypto	yes	yes	no	yes	yes
RMI	yes	yes	no	moot	yes
Wrappers	yes	yes	no	yes	moot
MvFilter	yes	no	yes	yes	yes
Avatar	no	yes	yes (+ update mgt.)	yes	yes
RobotMode	no	yes	yes	yes	yes
MiniWeb	yes	yes	yes	yes	yes
VFind-Daemon	yes	yes	yes	yes	no
Proxy	yes	yes	no	yes	no

Here is a table that explains what is contained in the main VFind packages as of VSTK/P Package Version Number 169.

Tool	VSTK	VSTK-T	VSTKP	VSTKP-T	VSTKCW	VSTK-OEM
VFind	Y	Y	Y	Y	Y	Y
VFind MT	Y	Y	Y	Y	Y	Y
VFind Library	Y	Y	Y	Y	Y	Y
VFind Daemon	N	Y	N	Y	N	Y
MvFilter	Y	Y	Y	Y	Y	Y
CIT	Y	Y	Y	Y	Y	Y
UAD	Y	Y	Y	Y	Y	Y
THD	Y	Y	Y	Y	Y	N
BHead	Y	Y	Y	Y	Y	N
JDIS	Y	Y	Y	Y	Y	N
VGUI (HTML)	Y	Y	Y	Y	Y	N
MiniWeb Server	Y	Y	Y	Y	Y	Y
VDL Update	Y	Y	Y	Y	Y	Y
Robot Mode	F	F	F	F	Y	N
Avatar	N	N	Y	Y	Y	N
NTI	N	N	N	N	Y	N
NTI-Crypto	N	N	N	N	Y	N
UNIX Wrappers	N	N	N	N	Y	N
RMI (Windows NT)	N	N	N	N	Y	N
Virus VDLs	Y	Y	Y	Y	Y	Y
Spam VDLs	N	N	N	N	N	Y
Custom VDL Generator	F	F	F	N	F	Y

Y - Yes Included, N - Not Included, F - Future Enhancement

Magical Immunity to Software Attacks

Lots of intelligent people seem to think that for one technical reason or another their systems are immune to software attacks such as viruses or worms. Not as often there are people who think their systems are immune to "hacker" attacks, either inside or outside. Often these people refer to firewalls, gateways and specific versions of an operating system in support of their argument.

Since 1993, I have published papers on this topic. Here is the straight scoop. There is no safe system and you cannot make any system totally safe. Even a system that has no connections to the outside world can be attacked. What you can do is make a system as safe as possible. Generally this means you have hit the law of diminishing returns where making the system any safer also makes it non-functional.

In the past, people have asserted that UNIX and other computers that utilize hardware instructions to provide a Supervisor mode of operation were immune to attack. The use of Supervisor mode is not necessary for viral infection or even any other form of attack; therefore, the argument is moot. This argument is also supported by the fact that there are viruses in existence for UNIX and other systems that utilize Supervisor mode. A white paper written by Tom Duff and M. Douglas McIlroy of AT&T Bell Laboratories contained in the USENIX 1989 Volume 2 journal not only attests to the existence of UNIX viruses, but also provides full source code for an example.

White papers by Dr. Fred Cohen, a pioneer in the field of computer viruses, demonstrated that a computer virus could fully penetrate super user access on any computer, even properly configured security rated computers in a short period of time. Dr. Cohen performed his tests on multiple UNIX systems.

Other people state falsely that dedicated single purpose computers such as process control systems; cell phone systems and embedded computers are immune. There are already cell phone viruses (Spain was struck by the first outbreak) and single purpose embedded computers has been infected. Some people even went so far as to design, build and field "immune" computers using ROM (read only memory) based operating systems. The theory was that if there is no way to modify the operating system, and then it cannot be infected or hacked. In the paper "From Little Acorns Mighty Viruses Grow" by Alan Glover of Pineapple Software, it was disclosed that the Acorn Archimedes computer (a British educational computer system), which holds all of its operating system and windowing systems locked in hardware based Read Only Memory, has been successfully infected by computer viruses. This is an extreme case of hardware-based protection, and yet it failed. As of January 1994, there were 52 virus families totaling 84 viruses affecting the system.

If the Acorn, UNIX, Macintosh, Microsoft Windows and even cell phone computers can be infected or hacked, then any system can be. In 1993 I stated, "Those components of an operating system that are deemed necessary for practical use, such as copy, append, change permission settings and hundreds of other basic functions are the only necessary building blocks for viral code." I continued that, "Many simple and normal functions that may pass a security screen, when combined, implement a virus" or any other form of attack.

Magical Disinfection of Viruses

All software is flawed. It is the nature of software. Programs, which have run for 20 years without error suddenly, and without notice, fault. If a software executable which is flawed to begin with but at least tested (we hope) to operate properly under normal conditions is infected by a virus or modified by a Trojan or other software attack then the operability of the program is unknown and the state of the program is hostile. In addition, many viruses are poorly written and the author has no idea what effect it will have on the many different targets it may come across. Algebraically the equation would be fv(p) = p'. Function virus upon target program p yields p-prime. A disinfection program is of necessity a different program than the program that infected the file to begin with. Viruses do not run in reverse and undo the damage done. Again, the authors of disinfection program cannot test them against all combinations of viruses and target programs. The effect is a program that should never have been modified being modified twice. Algebraically the equation is fd(p') = p''. That is, function disinfection upon target program p-prime yields p-prime-prime. The result p-prime-prime is not equal to p. Even if by some rare chance the result of p'' is p, the state of the program is unknown unless a cryptographic hash of p prior to the first transformation exists and can be verified.

Condition	Operability	State
Program	Good	Good
Infected	Unknown	Hostile
Disinfected	Unknown	Unknown
Disinfected Macro	Not Trusted	Not Hostile

Programs that were infected by a virus and later disinfected by even the best disinfection program in existence are still damaged and should be at best considered untrustworthy and at worst, should be considered seriously damaged.

There is one type of disinfection, which while it will not yield the original file (p), it can yield a non-hostile result (p''). This is the case where a data file is infected. A perfect example of this is Microsoft OLE files. OLE files are used for Microsoft Word documents and other types of documents. Microsoft has announced that it will be phasing out OLE in deference to XML, but these statements will also be true for XML. Viruses, which infect OLE documents, are macros. They are interpreted by whatever is processing the file. Removal of all macros from an OLE document will yield a result that is still untrustworthy, but is at least known to be in a state that is not hostile. I say this because many macros change the contents of the file. In a word processing document this may or may not be noticed. In a spreadsheet this may not be noticed. If a macro virus changed a document directing someone to do something, it may be serious but may be detectable by common sense. In a spreadsheet, a number or formula could be modified in such a way as to not be noticeable. There are real life examples of viruses, which do this.

Disinfection of macro viruses has several drawbacks besides what I have already mentioned. First, they may leave a ghost of the virus, which will be detected by another anti-virus program. Secondly, the industry practice is to remove all macros. If a macro is essential to the proper use of the document, then the document is not recovered correctly. If a backup of the document is not available, then disinfection of these types of files are an acceptable starting place provided that the end user understands the limits of the disinfection program.

The only trustable method of disinfection of an executable program is to remove the program and replace it with a known good copy. Even if backup copies were not made, this is often an easy solution to implement by using the installation media. Disinfection should be considered an undesirable and temporary bandage until a fully trustable solution can be implemented.

Activation Keys

VSTK/P locks to a specific system using an activation key. The keys lock on nodename, not hostid or domain name. Future versions (Key Type 10) will allow locking against hostid and other identifying values. For example, a hostid for a system might be 214435 while the domain names associated with that system are www.cybersoft.com and www.cybersoftinternational.com. The nodename for that same system could be webbox. Notice that there is no relationship between these forms of identification, other than they all refer to the same physical system. The nodename is what the operating system thinks it calls itself. The domain names are how a domain name servers resolves DNS requests into IP addresses. The hostid is hardware based and is supposed to be something like a serial number but can be easily changed by someone who knows how, typically using the boot monitor software of the system.

The disadvantage of using a hostid is that it is supposed to be hard to change. Using Domain Names is not wise, since they tend to be moved from platform to platform frequently. The advantage of using the nodename is that it is easy to change for the system administrator. If a server breaks and the hard disk is transferred to another compatible hardware platform, then the nodename of the system remains the same and the activation key will operate.

In addition, if you are migrating from one type of system to another type of system, there is a good chance that the nodenames of the system will be direct replacements. Since we deliver almost all supported platforms on the delivered media, you could change between Sun Micro, IBM, HP, SGI, Linux and any other supported platform without calling CyberSoft.

Of course, this also makes it easier to steal licenses by naming two systems with the same nodename, but that is only true of thieves. A thief will always find a way to steal what they want. If we make it hard for the thief, we would make it significantly harder for the system administrator to do what they need to do. Since our goal is to provide computer security, the loss of product activation keys to thieves is an acceptable risk in order to insure that our product is actually used. Making the product easy to administer makes it significantly more likely to be used.

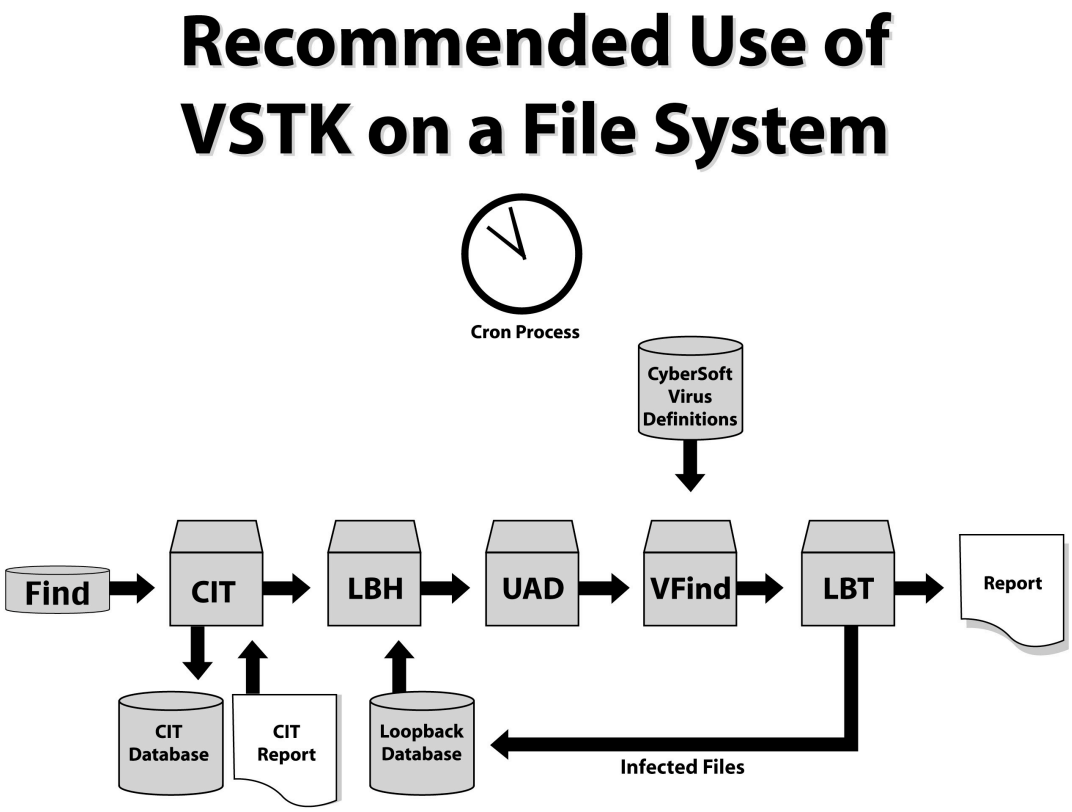


Figure 1

VFind

Many people think of VFind as an anti-virus program instead of a pattern analysis system. This is understandable since many customers need anti-virus programs and do not need a pattern analysis program. This perception is also not entirely wrong, since VFind started life as an anti-virus program and has multiple functions that are geared toward efficient processing for that purpose. In fact, all anti-virus programs are single purpose pattern analysis programs specialized to finding viruses. Since VFind is more than an anti-virus program, it has the ability to perform functions that are not normal to anti-virus programs. It has a different problem set to solve. VFind was the first anti-virus program to execute on something other than a Microsoft operating system. I believe it was also the first anti-virus program in existence to have an e-mail interface (MpScan).

VFind is a high-speed pattern analysis tool. In fact, it is the world's fastest general-purpose pattern analysis program available commercially off the shelf (COTS). When delivered as part of the VFind Security ToolKit, it is delivered with virus detection databases that make it into a virus scanner. Pattern analysis is the most difficult part of computer security. In general, if you can recognize a threat you can deal with it. Fast, accurate, flexible pattern analysis is a must. It is also a difficult problem that makes VFind significantly more valuable than a single purpose virus scanner.

VFind is unlike any other virus scanner in existence. It was the first anti-virus scanner for UNIX, the first heterogeneous virus scanner, and the first scanner to incorporate a full virus description language, CVDL. Unlike most virus scanners, it searches for attacks in a file based upon what the file contents reveal its type to be. Most virus scanners assume that the filename is a description of

the file type. VFind determines the file type by direct examination of the file's contents. This makes VFind significantly more powerful than a virus scanner that only searches in files with the ".com" and ".exe" filename extensions for Microsoft executable viruses because it is not reliant upon a filename which, in a hostile environment (such as a virus attack), could be wrong. Without this additional functionality, a mere filename change can be used as a form of stealth attack. In addition, this allows VFind to examine data in a byte stream in which filenames may not exist. This can be a significant feature if your computer is network attached.

An example of an attack that VFind can detect but other anti-virus programs cannot is a two-body problem. A normal virus is a single body problem. An example is that the virus infects an executable. The executable is named in such a manner as to allow it to be recognized as an executable by the system and consequently the virus. When the virus infected executable runs the payload is delivered. In a two-body problem the payload is hidden in a file that is not normally scanned. An example of that is "payload.data". A normal virus scanner will not scan this file because the operating system and the virus scanner do not recognize it as an executable. In this case the contents of the file are considered moot. The second body is an uninfected program, which reads the payload file, loads it into memory and executed it. A virus scanner may be able to clean up the results of the infection in recognized executables but the infection will reoccur every time the clean program executes since the payload file is never scanned. Since VFind evaluates every file and determines the contents by direct examination this problem cannot occur because VFind would recognize the "payload.data" file as an executable. Special Note: There are viruses that will "infect" jpeg graphic files.

VFind is also heterogeneous. This is critical in a server environment. Often a server will contain executable programs for network attached client systems of a different architecture. A very common example of this is a UNIX server providing network disk services to Microsoft Windows workstations. In this case, the UNIX system can harbor viruses for the Windows system, even though it is itself immune to that attack. This problem was documented by me in a white paper published in 1991, "Heterogeneous Computer Viruses In A Networked UNIX Environment" where the phase "Typhoid Mary Syndrome" was coined to describe the problem. VFind solves this problem by simultaneously searching for UNIX, Microsoft (MSDOS, boot sector, Win-32, etc and OLE Macro), Macintosh, Amiga, Java and any other type of attack programs we deem important. Note: In current versions of VFind the Amiga virus library is defaulted to off.

Another interesting aspect of VFind is that it simultaneously solves for all patterns. This makes it very fast when there are hundreds of thousands of patterns to search for. In addition, newer versions of VFind are multithreaded and can scan more than one file at a time. Multithreading also allows VFind to take advantage of multiprocessor systems.

VFind includes the CVDL pattern analysis language. This system allows CyberSoft or the user to define new attacks or any other type of information that can be examined by advanced pattern analysis. CVDL is implemented in text so no binary modification is required. In addition to the proprietary CVDL operators, the language includes the entire Extended Regex language as a subset (newer versions only). CVDL is very powerful, includes macro capability, meta-VDL capabilities and is constantly being expanded.

A meta-VDL is a VDL that refers to other VDLs in order to decide if a pattern is solved. This is not a macro. As far as I can tell, meta-VDL capability is unique and powerful. An example where a meta-VDL is useful is if several VDLs are all solved for a dataset. Each of those VDLs would

normally report; however, the useful information that ALL of them solved would not be revealed. The use of meta-VDLs solves this problem. Meta-VDLs allow the CVDL programmer to inform the user that a significant event has taken place. As an aid to the programmer, VDLs can be set so as to not report to the user, but will set a flag for the meta-VDL.

One use for the CVDL system is to search for words or phrases that are not allowed on a system. These phrases could be proscribed as part of a organization policy such as sexual harassment, or could be part of a compartmentalization policy for handling classified information, or for restricting what programs may reside on a computer by direct examination. It also allows for reactive processing of espionage attacks in which data is being moved within a system. Finally, CVDL allows for very fast updates of the VFind tool to search for new attacks without the need for replacement of the binary executable. One example where the CVDL language is being used for simultaneous multiple purposes are in the SafeInternetEmail program. One copy of VFind loaded with multiple CVDL programs provides the pattern analysis for virus scanner, unsolicited bulk e-mail (spam), sexual/racial harassment and organizational information blocking such as is required for HIPPA or FTC compliance.

Customer case study number 2 demonstrates a typical use of VFind after a break in. This part of the document was written by the customer, with edits for brevity and clarity.

We installed VFind on four dedicated e-mail firewalls. Two in the United Kingdom, one in the United States and one in Sweden. Another is planned for Australia. Each server is a Sun Micro 400 series with 4 processors. (We get a lot of mail). A few years ago we had external consultants draw up an anti-virus strategy document. They said, in summary: Use more than one anti-virus product, but definitely keep VFind on the firewalls. Most of the document was about selecting anti-virus for the desktops but essentially said that this decision could be made on costs grounds as long as VFind remained on the boundary.

We merged with another company that had no firewall virus scanning. (They were not VFind customers.) When the Melissa virus hit the world, both halves of the company were infected and disconnected from the Internet. The day after the outbreak we have a signature installed from CyberSoft and isolated our infected PCs so we could reconnect our half of the company to the Internet. The other half of the company remained off-line for most of the week. While managing the desktops was difficult for both halves of the company we only had to update the three VFind servers to reconnect to the world. This allows us to update the desktops away from the crisis. The other half of the company installed VFind on its servers after this incident.

VFind has several engines. Each engine solves a different type of pattern analysis problem. The scanning engines are: “Basic Grunt,” “Case Insensitive,” “Jade,” “Intel Emulator,” “Entry Point Engine”, “Cryptographic Hash” and “Bayes”. There used to be an engine called “Sieve,” but it was removed as redundant to the newer, better engines. There are also support engines: “CVDL Compiler” and “Control System” but they are moot from a user standpoint.

The “Basic Grunt” engine is neither basic nor grunt, but is a very complex system developed over more than a decade of time. It is a fast parallel search engine. The search time in this engine is independent of the number of patterns being solved.

The “Case Insensitive” engine is the same as the “Basic Grunt” engine but was specifically designed to operate on case insensitive patterns.

The “JADE Engine” (Java Advanced Disassembly Engine) does a fast disassembly of Java Byte Code then searches for patterns in the disassembly. This is also available as a stand-alone command line tool for security analysts called JDis (Java Disassembly). It is important to disassemble Java Byte Code prior to virus scanning because it is the only way to associate constant pool entries with op-codes. This association allows the virus scanner to scan for what is actually going on in the program instead of guessing. VFind is the only tool that does this.

The “Intel Emulator” can do several different things. It identifies the entry point and initialization information of an executable and it emulates the processing of instructions for Intel processors. It can identify several types of things including suspicious code; obfuscated code that appears to be hostile and code that modifies instructions in memory. This final process is important to identify polymorphic viruses.

The “Entry Point” engine uses information generated by the “Intel Emulator” to apply its VDLs for pattern analysis starting at the entry point for a specified number of bytes. This is basically what is referred to in the industry as a scalpel engine.
The “Cryptographic Hash” engine is a subset of the Cryptographic Integrity Tool, CIT. It does not perform all of the functions of CIT but does process the danger-file database. The danger-file database is a flat file that contains a list of the names of known Trojan Horses and their MD5 cryptographic signatures. Since Trojan Horses and some other types of software attacks are not infectious and are not polymorphic there is no need to search the entire file for a pattern when the entire file itself is the pattern. These attacks never change by themselves, cannot be made safe and should always be deleted upon detection. Consequently we can calculate an MD5 hash value for every file processed and then compare that hash value to the database of known hash values. If anything matches, it is an exact match. Of course, this feature can also be used for the detection of objects that are not software attacks. Anything that is static can be digested into a hash value and placed in the database.

The “Bayes” engine was actually developed for the analysis of Unsolicited Bulk E-mail, UBE (sometimes referred to as Spam). Our implementation of Bayes is different from most other implementations; in addition, we allow it to be applied to any file you desire. The applicability of the algorithm is dependent upon the Bayes database. In short, the Bayes engine allows you to statistically classify an object (file or data stream) as belonging to a class or family of objects without specifically having knowledge of the object under evaluation. The determination of belonging to a family of objects is determined by whatever is programmed into the database. While not infallible, this engine allows you to detect needles in a haystack in a statically meaningful way without knowing what kind of needle might be in the stack. Of course, this engine can be made totally worthless or valuable depending upon the quality of the database.

The “CVDL Compiler” changes text-based programs written in the CVDL language to the internal structures required for identification. It supports both the CVDL proprietary language along with both the “POSIX Basic Regular Expressions” and “POSIX Extended Regular Expressions”. It also interprets the VDL.LIST control file and manages all file type restrictions.

The “Control” engine is basically the superstructure that manages all of the engines. This engine is available to end users via command line options.

VFind Iterations

There are different iterations of VFind. The first and oldest version is the standard single threaded command line version. Later we added the other versions to solve specific customer problems.

VFind Iteration	Single Threaded	Multi Threaded
VFind Command Line	Y	Y
VFind Linkable Library	Y	Y
VFind Pipe Daemon – Script	Y	Moot
VFind Daemon Binary	N	Y

Multithreaded programs allow you to take advantage of multiprocessor systems. While multithreading causes a slow down on single processor systems we were also able to speed up VFind by almost the same increment. This is not a problem since we continue to support the single threaded version. The VFind Pipe Daemon is obsolete with the availability of the VFind Daemon Binary. It is maintained for those users who wish to "roll their own" daemon using script languages.

VFind Linkable Library

The VFind library provides a software interface to the powerful pattern matching and virus detection capabilities of VFind. The VFind library allows an application to scan single files directly or scan archived files via transparent application of UAD. On platforms supporting multi-threading, VFind library takes advantage of SMP processing power by allowing several files to be scanned in parallel.

The VFind library API supports global configuration of the VFind scanning engines, and local configuration of the scanning process for each scan request. Results of the scan request are returned via callback functions specified to the library as part of initialization of the library interface. This provides a flexible and speedy mechanism to do analysis of the scan results within a single process. One could do sophisticated statistical analysis of the results, or simply modify the strings, which VFind typically writes to standard output when a virus is detected. It should be noted that VFind itself is simply an application using the VFind library.

VFind Daemon Binary

The VFind Daemon does everything VTSK does, plus more. VFind Daemon is a heterogeneous, highly configurable, high performance multi-threaded server for the detection of virus-infected files, spyware, other forms of malware, and spam. VFind Daemon scans simultaneously for Microsoft Windows, UNIX, Linux, and Macintosh viruses, including Denial of Service attacks, Back Door attacks, Spyware, hostile Java Applications and Applets, OLE/VB5 Macro viruses, and common hacks. VFind Daemon also allows customers to add their own VDL definitions. VFind Daemon includes all the services and features provided by the VFind and UAD programs. VFind Daemon's capabilities are based on a powerful, general-purpose pattern detection engine, the CyberSoft Virus Definition Language (CVDL), and CyberSoft's commitment to providing our customer's systems with superior protection from malicious and unwanted data.

VFind Daemon Benefits

The VFind Daemon provides user applications virus scanning and detection services at a high level of performance. Running as a daemon process eliminates the need to re-initialize the scan engines on each request. Without the overhead of initialization, files are processed as they are received, improving response time and minimizing the effect of virus scanning on the main application.

VFind Daemon makes VFind's scanning and virus detection services accessible to any application running on a user's system. An application connects to VFind Daemon using sockets. The easy to use message interface makes passing a file scan request and processing the result a straightforward task. VFind Daemon's multi-threading capability enables it to scan requests from multiple applications concurrently.

Applications can access VFind Daemon services through an easy-to-use message interface. The Simple Virus Scanning Protocol (SVSP) is a text-based, request/response interface that gives applications full access to VFind Daemon's services. SVSP includes commands that enable the program to set scanning options on a per-request basis and to specify the file to be scanned. Requests can be tagged so that the subsequent responses can be matched. This allows the application to submit multiple scan requests and be able to match the asynchronous responses.

The VFind Daemon also supports the interfaces for other available virus scanning daemons, for example: ClamAV's clamd. This makes it possible to incorporate VFind Daemon into an existing system with minimal software changes and enables applications to migrate towards utilizing VFind's additional capabilities as required.

VFind Daemon's multi-threading capability enables it to scale gracefully and take advantage of systems with multiple processors. The number of threads used by VFind Daemon is configurable and can be set to match the available computing power. A rule of thumb is to set the number of threads to the number of processors plus one.

VFind Daemon Ports

VFind Daemon has been ported to most popular UNIX/Linux platforms:

- Linux for Intel, Linux for PowerPC
- Solaris SPARC, Solaris Intel
- FreeBSD, bsdi-x86
- Mac OS X
- HPUX 11
- Tru64
- And others….

The VFind Daemon is a multi-threaded server process. It provides all the functionality of the UAD and VFind programs. VFind Daemon supports file the expansion of archive files and parses mail messages and html files. Multiple scan engines are supported: the Basic Grunt parallel scanning engine, Case Insensitive engine, Intel emulator engine, Cryptographic Hash engine, Entry Point engine, Bayes engine, and Java Disassembler. The VFind Daemon is highly configurable, enabling the user to specify whether to scan for viruses or spam or both. VFind Daemon can be configured to scan for information that is of concern to a specific business or site.

Applications connect to the VFind Daemon using sockets and using the Simple Virus Scanning Protocol (SVSP) interface. SVSP is a text-based, request/response interface. SVSP commands allow an application to set scan options and to send a request to scan a file. The scan options can be enabled/disabled on a per request basis. The results of a scan are returned in an easy to parse messages. If a virus is detected in a file, a response is returned to the application that identifies the infected file and identifies the virus. SVSP supports message tags, that is, if a request is tagged, the resultant response is returned with the same tag. This feature allows the application to send multiple scan requests and to use the tags to match the responses.

The VFind Daemon supports interfaces defined for other virus scanning daemons., for example, the ClamAV clamd interface. VFind Daemon accepts the message formats used to submit requests to clamd and returns the results in the expected format.

VFind Speed (Comparing Apples to Oranges)

While Windows based anti-virus programs only concern themselves with scanning selected parts of binary executable files for viruses, VFind does not have the same luxury. First, Windows anti-virus programs restrict what they scan by filename. Only programs ending in file extensions that are executable such as EXE, BAT or COM are scanned.

Secondly, with the exception of self-extracting archives (which are executables), most Windows based anti-viruses programs do not scan the entire file but restrict themselves to scanning selected parts of the file where viruses are known to reside. By restricting themselves to Windows they can take advantage of low level operating system directives to read only selected portions of each file, and not bother processing the bulk of each file. They don't run into a disk drive bandwidth bottleneck. This doesn't fly in a heterogeneous environment such as modern day servers and in-transit processing systems like firewalls. First, you cannot trust filenames to be accurate descriptions of the contents. Secondly, viruses that infect UNIX and heterogeneous environments are more complex than Windows viruses and may infect an executable program at any point in the file. There are also many more types of encapsulating technologies, of which the Microsoft based technologies are just a subset. Basically, when you need to protect a complex system you can't do things half way. Since VFind does not do things half way, people have the incorrect perception that VFind is slower than some Microsoft Windows based scanners; consequently, we must address the problem of this incorrect perception.

To prove 100% coverage of every file, VFind has to scan every byte of every file. It also has to do the work of identifying everything by its contents instead of blind faith. This makes VFind appear slow but it allows it to do things other scanners cannot do. CyberSoft has clocked VFind as significantly faster than Windows based anti-virus programs at processing data. This means that VFind is faster but does a lot more work so it takes longer. Now that there are some competing products on the UNIX platform, they are slower than VFind. Another way to understand this problem is that a Windows based anti-virus scanner can scan a very large disk drive in a matter of minutes. If you reviewed the total amount of data being scanned, and divided that number by the total number of minutes the scan took, you would learn that in fact the program appears to be processing data at a rate that exceeds the maximum data transfer rate of the disk drive. This is of course not possible, so logic dictates that the Windows based scanners are in fact not processing all of the data.

This is no longer true of some of my competitor's programs that run on UNIX and heterogeneous environments. It is also not true of Java based anti-virus programs, which are painfully slow, but for reasons that have to do with Java and not what was already stated. My competitors learned that doing things half way in these complex environments does not work. In fact, when we compared VFind to our competitor's UNIX offerings, we found that VFind was significantly faster. This however, is no help since people's perceptions were created on the Windows platforms that cheat on the work in order to gain speed.

You now understand that the issue of speed is a perceptional problem and not a technical problem. Our solution has been twofold. The first was educational. This has been somewhat successful, but we cannot educate the world. The more successful solution has been technical. We have been making VFind faster. It is now very fast. It took us almost 10 years of constant effort, but our algorithms are now significantly faster than any other commercial product without any reduction in the features. VFind is doing more work than anyone else, but we do that work faster. The next thing we implemented was context switching of VDLs based upon content. We now have the option of restricting looking for EXE based viruses to EXE files. In the past, we searched every file for every type of virus. There is a small risk to this. Since this feature is optional, we only use it where we deem the risk to be low or nonexistent. One example where this strategy cannot be used is that OLE documents must now be searched for some types of binary executable viruses. Due to the flexibility of the CVDL language, it is trivial to direct VFind to perform this task and to do so without changes to the VFind binary executable.

Having said all of that there are still two ways in which speed can be compromised and which you as the end user had direct control. The first is the way in which you use VFind. If you are using VFind inside of another program an example of which is a script then you should not restart VFind for every file you want to scan. VFind has a large startup overhead. Over the course of thousands of files that overhead is negligible but over the course of one file, it is large. Use the daemon, library, pipe daemon or any other method you choose but don't restart VFind for every file that you want to scan.

The second issue is that the CVDL language is full featured. Some of the operators are slower than others. In addition, you can generate very complex scanning programs. VFind will do what you tell it to do. If you give it a pattern analysis program, that means that it must do millions of operations more than usual then it will follow your directions. Take the CVDL course to learn how to optimize pattern analysis programs in the CVDL language.

CVDL Language

The CVDL language was first released to the public in VFind Version 4.0 in 1992. Using the CVDL fuzzy logic afforded by proximity, Boolean and case control logic it is possible to define one CVDL statement that can definitively locate all variations of a pattern, both known and unknown. *In the terms of a computer virus this means that one VDL statement can be created that can detect more than one type of virus*. In 2004 the CVDL language was expanded to include all of the "extended regular expressions (regex)".

One CVDL definition is called a "VDL statement" or just "VDL" for short. A collection of two or more statements is called a VDL program. Sometimes we also refer to VDL programs as VDL files.

Training on the CVDL language can take an entire day by itself and therefore cannot be included in this class. A CVDL primer has been included in Appendix A and a free video training course is provided on www.cybersoft.com. The video training course has not yet been updated to include all of the new features. You can also look on the website for the CVDL Training Handbook when it is ready.

VFind and Computer Viruses (An open eyed examination of viruses)

Before we get too far into this section of the training guide I need to dispel several incorrect urban legends about computer viruses. Here are the facts:

1. Computer viruses are not created by the Anti-virus industry (We don't have to. It would be too risky and other people are already doing it). (Special bonus: Discuss viruses as part of the former USSR national policy as an economic weapon against the West and how it backfired.)

2. One infected computer can destroy most of the bandwidth of a network if the network is not configured with security in mind.

3. There are a lot of viruses that were created over the course of many years but in reality there are significantly less than the industry tries to make people believe there are. There are several reasons for this including the fact that many of the older viruses will not run on modern computers or the viruses require a transport format that is basically obsolete such as boot sector viruses and floppy diskettes. There is also the fact that some anti-virus companies will search for viruses that are moot. This is done solely for marketing reasons. See http://www.cybersoft.com/whitepapers/archives/eimpact.shtml for more information on computer virus extinction. (Special bonus: Discuss the lack of naming standards and count inflation.)

4. There are only 100 to 150 new *effective* virus threats a year. This number has been stable for many years. There are a few thousand new viruses that will never infect anyone each year. (There is an exception to this rule called virus creation tools and virus wars but they really don't count.) You can easily tell what is important by viewing the Wild List. www.wildlist.org (Please note that the Wildlist International was purchased by the ICSA Labs www.icsalabs.com, a division of TruSecure Corporation www.trusecure.com, in 2003 and is no longer an independent organization.) This means that a virus scanner must scan for the effective viruses plus the present threats but does not have to scan for anything else. (Special Bonus: Discuss the law of diminishing returns and how it applies to anti-virus.)

5. The best way to tell if an anti-virus product really works in a way that will protect you is independent testing. Anti-virus testing is not easy and is often done wrong when performed by computer magazines. See http://www.cybersoft.com/whitepapers/archives/open_letter.shtml for more information on bad virus testing. There are only a few organizations that perform virus testing well and not all of them publish results. One that does publish results is West Coast Labs, CheckMark. Any anti-virus product that is CheckMark Level 1 certified will detect all of the viruses in the Wild List. See www.check-mark.com for details. The VFind product is Check Mark Level 1 certified.

Super Critical Systems and Anti-Virus

In May 1995 I published a paper entitled Anti-Virus for Multimedia Publishers. This paper was to the best of my knowledge the first time anyone proposed using more than one brand of anti-virus

program on the same system. In fact, I suggested using three. Today this is common on super critical systems such as central servers and medical equipment.

The paper explained that each manufacturer of anti-virus makes different trade-offs when designing their products. These trade-offs are necessary in order to make the product useable. Reference the law of diminishing returns. I suggested that super critical systems required a higher set point for diminishing returns than a normal system; consequently, I suggested using more than one manufacturers anti-virus and using multiple methods. An example is to use three virus scanners by different companies, one integrity system and one baseline control system. The more critical the system the more tools and methods should be implemented. You can implement all phases of this concept except for the multi-vendor idea using the standard VSTK/P products. Be careful that some manufacturers tools may not work well together.

In addition to the idea of multi-vendor products protecting the same system this paper also presented the CyberSoft Half Dozen Rules of Anti-Virus Common Sense. Here is an updated version:

CyberSoft Half Dozen Rules of Anti-Virus Common Sense

1. Always virus scan new software prior to installation. Especially true of Internet downloads.
2. Save a cryptographic database of your system on removable media. Do this prior to an infection so you can detect the changes.
3. Scan the system and use the integrity system often, at a minimum daily. Update the virus definitions daily or more often.
4. Use the keyboard or screen lock to keep people out of your system when you are not there.
5. Scan software copied from other people's systems prior to use.
6. Backup your system often. You can always retrieve destroyed data if you have it archived.

Lexical Analysis using VFind's CVDL

The objective of this paragraph is to portray CyberSoft's approach of meeting the challenging pattern analysis demands common in today's complex e-mail market. Our e-mail product SafeInternetE-mail (www.safeinternete-mail.com) uses VFind and UAD to do this. All lexical analysis that is done in SafeInternetE-mail is done by VFind. Accommodating for dynamic patterns such as the example below cannot be met by conventional scanning methodologies. This is an important consideration when scanning messages for the medical field, for example, where the word Viagra can be found in legitimate communication. Using CVDL matching dynamic unsolicited messages becomes an attainable reality while bypassing messages that are most likely routine correspondence.

One feature of the CVDL language that contributes to the successful creation of dynamic pattern analysis is the ability to go beyond matching individual words and phrases. An analyst is able to expand upon Boolean expression logic utilizing an array of CVDL functions. Some of these functions may be seen in the example below.

Consider an e-mail message containing suggestive Viagra text such as:

“Click here to get your sample of ¥ 1ÄGR@”

The use of the CVDL language in the example below will find the word “click” regardless of case, and the word “Viagra” with any character substitution as defined in the macros for A, I, and V around “g” and “r”, within two characters of one another. This allows us to detect the word Viagra even when it only appears to be “Viagra”. False Matches are prevented by the NOT ~"viagra" (note the correct spelling) command at the outset of the VDL, and the WP1 (one or more white space or punctuation characters) surrounding the “viagra” alternative.

```
$define letterA "Aa@ÀÁÂÃÄÅàáâãäå"
$define letterI "Ii1;!¡_ÌÍÎÏìíîïl|:"
$define letterV ({"Vv¥"} | ~"\/")
:SPAMVDL, NOT ~"viagra" AND ~"click" AND WP1,$letterV,@-2,{$letterI},@-
2,{$letterA},@-2,~"g",@-2, ~"r",@-2,{$letterA},WP1 # ; Matches Viagra language with special
characters.
```

This Viagra example provides a glimpse of CVDL practicality in pattern matching. Its simple yet sophisticated approach provides a unique combination of tools to match the most dynamic of patterns while minimizing the possibility of false matches. The use of macros provides an additional edge not found in conventional scanning engines.

The CVDL language is a work in progress as it has been for the last decade. We are constantly adding new constructs to the language in order to solve complex problems. This is because the problem of lexical analysis is becoming more and more difficult. The original driving force behind the lexical analysis using CVDL was dirty word checking used by the government and industry. Think of words and phases such as “TOP SECRET”, “COMPANY PROPERTY” and “GOVERNMENT COMPLIANCE.” Today’s more complex problems are unsolicited bulk e-mail. As an example of a lexical problem in the extreme, that is in itself an explanation of the problem, read the following paragraph:

Aoccdrnig to rscheearch at an Elingsh uinervtisy, it deosn’t mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer is at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae we do not raed ervey lteter.

The prior paragraph has been circulating on the Internet as a joke. If you didn’t understand the point of the paragraph then go back and look at the spelling. A certain percentage of people will read that paragraph without noticing most of the spelling “errors”. The ability of the mind to “fill in what is not there” very well defines some of the problem of lexical analysis. The CVDL language is very well suited to these kinds of problems.

MvFilter

The MvFilter program disinfects OLE documents (Microsoft Word, Excel and PowerPoint) from macro viruses (both VBA and Word Basic). It does this in the same way that all anti-virus programs disinfect macro viruses, by removal of all of the macros. The difference is that MvFilter was designed as a tool and significantly, the way in which it removes the macro. Very often anti-virus programs can detect a virus infection in a program that was disinfected by a different manufactures program. This is because the virus is actually left intact but disabled. A few bytes of the virus code are changed so that the product will not trigger and the macro will hopefully not run. Since each manufacture determines what bytes to change these bytes may not overlap. Consequently Norton might detect a virus in a sample disinfected by McAfee. This problem is called a "ghost virus". MvFilter does not have this problem because it totally removes the virus by zeroing out the entire virus and virus macro name from the table of objects. There is nothing for another virus scanner to detect.

As such, MvFilter can be used for compartmentalization purposes in addition to its reactive disinfection role. As a compartmentalization tool, MvFilter can be used to proactively prevent all macro virus infections, including new unknown infections, by automatically stripping all macros from OLE documents as they enter a system. This is a reasonable policy since it provides 100%, infallible protection in exchange for losing the ability to use macros. Since very few people use macros, this should not be troublesome to them. Users who need macros can use authorized methods to allow the macros they need while still preventing damage from unauthorized macros.

A second use for MvFilter is as part of a document warehouse archival and management system such as Documentum®. MvFilter has been successfully interfaced and operating with these systems for a few years. Generally, these systems manage hundreds of thousands to millions of critical documents. MvFilter automates the process of document baseline control by providing a consistent format, free of all macros, in addition to preventing attacks.

CIT

Special Note for discussion: CIT and VFind are very special tools in that you can accomplish a great deal more with them than meets the eye. It is always reasonable to consider these two tools when trying to solve a computer security problem. One of the overlooked features of CIT that is often valuable is that it is an intelligence gathering and analysis system. Using CIT, you know what has happened in a system as reflected in its file system. Knowing what has happened in a system also tells you what people have been doing in the system. Remember a computer system IS its file system, not the hardware!

CIT is a fantastic tool that complements Avatar. It has multiple uses and can tighten baseline configuration control down to a single bit or be used with surgical precision on an entire system or single file. The CIT tool produces a database of cryptographic hash values for every file it is directed to manage. In normal operation, this is every file on the system with the exception of special files like /dev, FIFO and proc files. The /dev files should not be scanned except explicitly. Otherwise you will end up scanning every device on the system several times. The FIFO and proc files are never ending. If you start to scan one of these you will be stuck. Once the database is produced, it will report on every file that has been modified, added, removed, duplicated or otherwise noted as dangerous. It can also be utilized to locate any known file by the files hash signature, after the fact. Additionally, it produces a target opportunity list of all files that have been

added or modified on the system for virus scanning by VFind. When utilized with LBH (Loop Back Head) and LBT (Loop Back Tail), which provides a feedback loop for infected files, it allows rapid virus scanning of a system without the danger of missing content. Files that are known to be clean and have not changed do not need to be scanned a second time.

CIT is also the best Baseline Configuration Control (BCC) tool. It allows the security officer to rapidly detect BCC problems. Since all changes to a system are tracked, it becomes a trivial effort to detect baseline configuration problems. This is especially true if a hacker breaks into a system or an authorized user makes unauthorized changes. It can even detect changes caused by hardware degradation or outside electronic forces such as magnetic pulses or radiation. One more benefit of using CIT as a BCC tool is that it becomes possible to diagnose and correct configuration problems with computers in a substantially shorter period of time than would be possible without the tool.

Another use for CIT is personnel monitoring. A trained security officer can determine a great deal about the user activity on a system by examining CIT reports. It is usually possible to tell if a user is not doing their job or is attempting to tamper with the system baseline configuration. The following examples highlight this feature. (Note this report is not real. It was formatted for this document and is representational. In addition while there are stealth cloak programs there is no hack66 program.)

Representational Output of CIT (CIT.RPT)

Added Files:	**Modified Files:**	**Deleted Files:**
/home/george/abc.c	/home/george/main.c	/home/harry/a.c
/home/george/def.h	/home/george/a.c	/home/harry/c.c
/home/harry/xyzzy.c	/home/george/b.c	/home/harry/1.c
/home/harry/567.c	/home/george/c.c	/home/harry/2.c
/home/harry/xyz.jpg	/home/harry/123.c	/home/harry/3.c
/home/harry/report.txt	/var/adm/sulog	
/var/log/uucp/Log	/var/mail/horse	

Duplicate Files:

/home/george/abc.c	with	/home/harry/567.c
/home/george/def.h	with	/home/harry/xyz.jpg
/home/george/main.c	with	/home/harry/report.txt

Danger Files:

/tmp/123ac96fc is “stealth cloak program hack66”

When a security analyst views the example just given, the first call made should be to have Harry removed from the building. The second call should be to convene an emergency meeting. Why? A little more information needs to be gathered but you know that Harry deleted all of the work he did, copied work from George, which he should not have done, and was attempting to hide the fact that he copied critical files from George. You can also tell that George has been very busy doing his assigned task. At first glance it appears that Harry is at worse a spy or a malcontent or at best someone that is breaking the rules by nosing around the system and attempting to hide the fact. A detailed examination of the file contents along with other standard security investigation methods

will reveal if this is a false alarm. If a false alarm then Harry is still showing behavior that is undesirable.

Another piece of intelligence that was picked up on this report is the fact the /var/log/uucp/Log file was created. This means that someone on that system was running the uucp command. The uucp command was the primary method of moving e-mail and files between systems prior to the invention of the network. It also allowed people to execute commands on remote system using e-mail. Since the use of the uucp program on this system was unauthorized it appears to have been used as a back channel. You should also notice that a mail file was modified for a user called horse. Since there is no authorized user called horse this raised a serious red flag. The fact that it was modified means that it existed in the past and additional information can be gathered by viewing old CIT logs and system backups. This is also an indication of what the unauthorized uucp connection was doing and the fact that another channel besides uucp may be in use since the uucp log was new while the mail file was already in existence.

Yet another bit of information is that the /var/adm/sulog was modified. The sulog file is a record of all attempts by users on the system to execute the `su` command. Each time `su` is executed, an entry is added to the sulog file. The su command gives someone the ability to switch user identities. It is often used to become the system super user.

CIT reported changes in system log files reveal which processes have been executed and does so in a known timeframe. It is not necessary to utilize the highly malleable system timestamp on files, you know that the detected activity took place in the window between CIT executions. This is in addition to identifying saved e-mail, edited files, database files that were modified, insuring that files that must not exist (such as known Trojan horses or obsolete programs/documents) don't, and generally determining what is happening and not happening in the system.

One of the interesting things revealed by this report is timing. The system date and time logs are of little value since they can be easily modified to values in the past. On the other hand since CIT is run nightly we know that modifications to the file occurred within the 24-hour period between runs. Knowing the actual time window in which something happened is critical when doing an investigation. Since the /var/log/uucp/Log was new but the /var/mail/horse was modified this indicates that this problem may be longer standing than it would appear at first glance. Attempts to use the su command combined with our other indicators indicate that hacking attempts may have been in progress. In fact, my starting guess would be that Harry already had a back door and was using an smtp transport to send/receive e-mail but decided to add the uucp system. I would investigate the file /tmp/123ac96fc which is identified as a "stealth cloak program hack66". In addition, I would guess that Harry may not have had the root account password but was using some other method of getting control. All of this information is based on the results of the CIT run shown above. You should also note that all of the past CIT logs are available on the backup tapes! A wealth of information on Harry's past activities can now be learned with timing nailed down to 24-hour windows.

Recap of why Harry needs to be removed from the building pending investigation

1. Harry deleted all of his work. This was not authorized and destruction of company property is normally a firing offence.
2. Harry made unauthorized copies of George's work. This is suspicious.
3. Harry tried to hide the fact he made copies of George's work. This is very suspicious and dishonest.
4. Someone placed a known hacker tool on the system.
5. Someone was using an unauthorized back channel communications tool on the system. This is very suspicious.
6. Someone has fully or partially penetrated security on the system as evident since they could execute uucp and either did or was planning on hiding their path using a known stealth tool.

Other uses for CIT include ensuring that the contents of a tape, disk or e-mail attachment are what they are suppose to be. By referring to a secure database of known CIT hash values, it is possible to determine if the contents have degraded or been modified. This is extremely valuable when dealing with programs or critical documents distributed over a network or archived in off line storage. It is a 100% reliable way of knowing that what was sent, was received, and what you thought was on removable media, is in fact, what is on the media. The CIT MD5 hash values are one-way traps. It is not possible to create the contents of a file by knowing the hash value. This means that it is possible to securely distribute hash values over open channels such as fax or low speed modem communications for even the most sensitive of documents.

The output from CIT is highly formatted which allows it to be read by data reduction programs. This means that CIT reports can be collected from very large networks into pools at central locations where data reduction techniques can produce an aggregate report. Aggregate reports of this type can reveal stealth attacks, including very hard to detect slow stealth attacks over a large number of systems. This report can be used as a quick way to determine if a very large number of networks attached computers remain true to their baseline configuration. When operating in a hostile environment, this means that all of the computer equipment can be verified from one report as being correctly configured and operational. Systems that do not conform to the baseline configuration can be identified easily. Depending upon which options are selected, the central pool of CIT reports can also be utilized to locate a single file on one computer in a network of hundreds of thousands of computers.

One of the biggest problems with customer support is overhead cost associated with a help desk. These costs generally have nowhere to go but up. Our studies have indicated that the bulk of a customer service call is spent doing diagnosis. Generally the end user does not know why their system is not working but they are sure they were not the cause (grin). For a complex operating system such as UNIX or Windows, this can take up to 2 hours to resolve. Using CIT, a system can be baselined when it is installed and updated. At anytime in the future if the system fails, the help desk can run CIT and it will reveal every file which has been added, deleted, modified, marked as a having dangerous content or is a duplicate of other files already on the system. This can reduce the time for diagnoses to about 15 minutes while the duplicate files feature can save significant amounts of disk space.

Using CIT can provide significant amounts of forensic evidence of the activities on a system. Every activity on a system leaves fingerprints. CIT can be used as a data reduction method to find these fingerprints quickly. This is especially useful when dealing with insider attacks that may be performed by disgruntled employees.

If systems are under tight baseline configuration and control, then models of baselines can be created using CIT. Any deviation from the baseline will be reported. This is especially useful for auditing departments and if a system has never run CIT but has been in production for a long time.

In a network CIT can be used to ensure the contents of a file transmitted to another system is what was received. There are many reasons why the file may be modified, but a quick check of the hash code on both ends ensures the accuracy of the data. If the hash code is transmitted via a secondary channel then it can also be used to reveal third party tampering by a "man in the middle" attack.

A related problem is the identification and verification of removable media, such as backup tapes. Using CIT, create a hash of the entire contents of the tape. That hash value can be used as the serial number of the tape and will also verify that the contents have not degraded.

By using a little imagination this can be extended to all types of operations and can provide a new level of security for equipment in areas that are subject to content degradation such as hard radiation areas used in power plants, food processing and high orbit.

The following customer case study demonstrates a typical use of CIT after a break in. This part of the document was written by the customer with light edits for brevity and clarity.

As an Internet Administrator whose main servers are Sun UNIX systems, I always seem to be fighting that never ending battle of keeping crackers off my computers. Even with the security patches, router access control and just simply phoning providers after finding someone suspiciously banging on our door, someone got in.

One morning I tried to log into our main server after a long weekend and found that my terminal was booby-trapped. No matter what key I hit garbage was displayed. I did a telnet into the server from another computer and looked around. Using CIT I found that the /etc/rc boot and other files were modified. If I had rebooted the rc files would have taken effect giving the intruder more control. I found all of the modified files using CIT and using a backup tape replaced them. Using CIT it took 10 minutes to figure out where the intruder came from. I must say that CIT has really made my life administering my systems easier. Besides saving my skin that day and using it other times to check for suspicious changes, I look at the output every day to give me an indication of what is going on in the whole system. Great tool!*

LBT & LBH

The Loop Back Head (LBH) and Loop Back Tail (LBT) programs provide a method of performing locked loop back logic for the purpose of ensuring that no target files are missed when virus scanning utilizing CIT for target reduction. These target files are identified as files that were known to be infected at the last run.

Explanation: The CIT tool produces a target list of files that have been modified or added to the system. If the administrator chose to ignore the warnings of infected files from a system scan, then there is a chance that those files will not change between the prior system scan and a current system scan. If there was no change, then these filenames will not be promoted as target files for scanning in the current scan because the CIT database is normally updated on every run. It is not acceptable to allow virus-infected files to exist on a system due to human error. The LBT program

extracts the filenames of all infected files from the output of VFind. It stores those filenames in the loop back database (LB.DB), which is a simple text file. When the Loop Back Head program first runs it reads the LB.DB and sends the contents to standard output before accepting target files promoted by CIT. This insures that files that were previously known to be infected are scanned again and are in fact scanned first before new files.

In addition to these features, the LBT program provides the “restrict” feature that allows the users to eliminate the use of the “grep” command when parsing output from VFind.

UAD

The UAD tool solves two difficult problems, identification and decomposition. Decomposition of a file to its smallest indivisible parts (universal atomic disintegration using classical language meanings) is a difficult problem. First, the program must have infallible identification of the file in order to decompose it. *This is not an issue for UAD, which identifies the file by direct examination of its contents.* Most decomposition tools assume the contents of a file by its filename. If the file is named "xyz.zip" most decomposition tools will assume that the file is a "zip" compressed composite file. UAD does not make any assumptions. This also allows UAD to identify data in a byte stream where filename information may not exist. This is important in a network environment.

Secondly, decomposition is critical to proper pattern analysis. There is no value in virus scanning a compressed, composite or encoded file, since the encapsulating technology will hide the contents from examination. This is why UAD is able to decompose e-mail, including attachments in uuencode and MIME formats. It is also able to decompose tar, gnu gzip, pkzip, zip2exe, UNIX compress and other formats. UAD will continue decomposition recursively until every part of the file has been decomposed into a state that is known to be a terminus (atomic state) or has been decomposed into an unknown format. Most unknown formats are already in the atomic state or are moot.

A benefit of the UAD system besides its uses for virus scanning is its ability to decompose many formats of encapsulated files. This can save a lot of time when the file format is not directly compatible with the system on which it resides. The user just executes UAD with the file he wishes to decompose, and UAD performs the rest. Unfortunately, many times when a user uses a tool other than UAD to decompose a file into its parts, the tool will place the decomposed files in multiple places on the system. UAD solves this problem by forcing the current working directory to be the top-level directory for the purposes of decomposition. This allows a user or system administrator to have full control over the installation of a new program without "splating" programs and data all over the system in an uncontrolled way.

As an intelligence gathering program, UAD can detect the actual identity of a file even though its filename may be a lie. This can be valuable information when doing an investigation. Example, xyzzy.txt is identified as a tar file with a dozen graphic files. This should be a red flag requiring additional investigation.

Newer versions of UAD also include automated decryption, which is important for the latest class of encrypted viruses that carry the encryption key as part of a text message. While you would never think someone would decrypt a file, then execute the contents, these attacks have been very successful.

To view a partial list of expanders that UAD can decompose use the command “uad –elist”. As of April 2004 this includes: uuencode, MIME, tar, zip, self extracting zip archive format (.EXE), gzip, compress (.Z), binhex, TNEF, Hqx8, RAR, CAB, EXT, encapsulating text and html formats. You can also add external expanders to extend the coverage. To see the list of file types that can be identified use the command “uad –tlist”.

Avatar (Self Healing)

As computers become indispensable to our efforts, they become more valuable targets. For example, if a command and control system can be knocked out by a computer virus, it can affect the efforts of an entire theater battle group. A virus that may have cost only a few hundred dollars to develop then controls hundreds of millions of dollars worth of men and machines. The same analogy can be made for civilian systems. Avatar answers this problem by providing rapid return of control of a system through reactive baseline management. Depending upon how it is configured, it can detect and correct prior to major system fault.

Avatar maintains the system Baseline Configuration. It does so by executing system security policies that act as an intrusion detection and response system. The most important function of Avatar is response. If the system Baseline Configuration is modified, for any reason, it will be detected by Avatar and returned to the correct Baseline Configuration. The value of Avatar’s response system is that it enforces discipline by a non-subjective automated process that can execute many times per day.

Intrusion is defined as any unauthorized modification to the system Baseline Configuration. The reason for this broader than normal definition is that it can detect unauthorized modifications by authorized and unauthorized personnel. When an unauthorized person breaks into a computer, their actions will always be dictated by their goals. If they are a passive reader, then their activity will be captured in the system logs. If they are using the system as a platform for further attacks, or they want to ensure future access then they will download attack programs for execution. To ensure future access, they will have to change the Baseline Configuration. Modification of the system logs such as changing permissions, insertion of Trojan back doors into critical system applications, modifications of the Baseline in any form, or just plain destruction of critical system files, can all be detected and corrected by Avatar. The addition of new inappropriate files to a system can be detected by CIT.

The ability to maintain the Baseline Configuration also provides extensive immunity to new unknown software attacks within the Baseline. If a binary or script virus infects a file, then the file will be overwritten by the Baseline version of the file. This effectively destroys the virus, and is far superior to any form of virus disinfection used by any other company. When a virus infects a file, it modifies it. In the process of infecting the file, it is common for the file to be damaged. [$f_v(a) = a'$] The disinfection process used by most anti-virus companies may or may not remove the actual virus. It is most common not to remove the virus, but merely change program pointers so that the program executes around the virus without executing the virus. If necessary, the virus is then modified so that it is no longer detected by the same anti-virus program as a live virus. This preserves the damage created by the virus and potentially adds new damage if the pointers are modified incorrectly. [$f_d(a') = a''$, $a \neq a''$] In addition, not all viruses, especially new unknown viruses, can be disinfected. None of these problems exist with Avatar since a captured copy of the

original file is used to overwrite the infected file. [$f_{avatar}(a') \equiv a$] This also works for all forms of software attacks in Baseline configured programs, not just viruses or hacker attacks.

Avatar security policies can be maintained on a file-by-file basis or on an entire system. Security policies that can be maintained are:

(e - EXISTENCE) The existence rule states that the file(s) must exist. It does not infer any other rules. This is extremely valuable for files that must exist but whose contents change constantly. Two examples of files of this type are log files and password shadow files.

(p - PERMISSIONS) The permissions rule states that the permissions of a file must not deviate from the baseline configured permissions. Generally all system command, log and configuration files have critical permission settings that must not change. This can be combined with the "e" and "o" rules to provide maximum protection for files whose contents need to change over time.

(o - OWNERSHIP) The ownership rule states that the owner of the file must not deviate from the baseline configured ownership. For example, the ownership of the password shadow file can be used as a back door into a system, while ownership of a log file can be used to stealthily erase evidence of activity on a system.

(s- CRYPTOGRAPHIC SIGNATURE) The cryptographic signature also known as a hash value states that an alarm should be activated if the contents of the file change but no further action should take place. This is extremely useful when used within other programs and when attempting to catch hackers. An additional use is to allow a script or program to verify that a critical file conforms to the baseline configured contents without overwriting any deviations from the baseline.

(c - FILE CONTENTS) The contents rule states that the contents of a file must not change. If the contents change for any reason, the file is overwritten with the correct baseline configured contents.

A rule exists (!) to force a pathname to not be baselined. This is most valuable for scratch file areas such as the UNIX "/tmp" and "/var/spool" areas. The opposite also exists (R - RECURSIVELY) which forces Avatar to baseline the entire path recursively. Finally a shorthand rule (E - EVERYTHING) exists which has the same operation as selecting the "eposcR" rules.

The Avatar Database was designed so that it can be read only. This means that the Avatar Check, Avatar Correct programs along with an Avatar Database can all exist on a CD-ROM or other read only format (NFS, DVD, Zip, etc). A read only version of Avatar cannot be hacked. Once invoked by a background process such as UNIX Cron or remotely invoked from a Security Server, the Avatar Correct system will automatically detect and correct any problems.

A significant feature is that an alternative Avatar Database can be defined for automatic fail-over if the primary Avatar Database has a failure. For example, if the primary Avatar Database is located on a network disk and the network failed, Avatar can continue processing with a local database. In fact, multiple Avatar Databases can exist and be invoked in any order necessary. This allows Avatar to be used for multiple purposes, including rapid system software update distribution and configuration for thousands of workstations without the need for personnel to visit them. System customization can also be accomplished by a smaller Avatar database, which can be stored locally or at a central location.

One of the advantages of the Avatar system for Internet based systems such as web servers is that shortly after a hacker modifies a web page, the Avatar system can be automatically invoked and restore the damaged page. If Avatar were run dozens of times per day then the hacker would literally have to break in and modify the system dozens of times per day. This is a huge incentive to leave the system alone.

Avatar is an automated tool that runs with or without end user operation. The primary function is to automatically "repair" a damaged system. Damage is defined by the System Administrator or System Architect when they create the Avatar database. Avatar was designed for battlefield operation; therefore, it assumes a hostile environment that may in fact be unmanned. The CIT tool compliments the Avatar tool. CIT is generally used as a "big brother" type of program to allow the System Administrator or Security Officer to understand what is happening in a system. It does this by cryptographically digesting the entire contents of a system then providing several reports, one of which highlights what files were added, deleted, modified or duplicated within the system. It will also inform you of any dangerous or prohibited files in the system. This in turn can be used to understand what an end user is doing, intrusion detection, detection of stealth trojanization of a system and provides critical information to the System Administrator which can cut down on system diagnostic time by a significant factor. In one case, a System Administrator in a government organization was able to determine that their server was hacked, what the hacker modified and added to the system, and restore full operation within a couple of hours. Normally, an operation of this type would require a full system rebuild which could take three to six days. For more information on the standard operation of these tools, review the technical white paper, "Secrets of the VFind Security ToolKit Professional Plus" located on the web site www.cybersoft.com.

Using Avatar for centralized distribution and control of a heterogeneous distributed global network is within the design parameters and functionality of the program. In other words, this is a fully supported function.

The Avatar system is rule based. The rules are defined and then compiled into an Avatar database along with any files or information required to fulfill the rules. Once a database is created it can be used read only. It can be distributed on CD-ROM, DVD or on a read only exported file system. The ability to provide an Avatar database on a gold foil DVD which is impervious to radiation damage means that a lights out operations could potentially repair any damage to critical components of the file system in the event of random "bit flipping". In addition, Avatar is able to utilize multiple databases, including rollover databases. This feature allows centralized distribution of updates, upgrades, security patches, changes to the standardized baseline configuration and forced adherence to the baseline. The database(s) that Avatar uses does not have to reside on the system it is operating upon. A database can be located in a centralized network or command operations center and provided over a read only virtual private network using standard approved

encryption. Avatar is insensitive to the manner in which the database is provided as long as it can open and read the file. It is also insensitive to the location of the file system it is operating upon as long as it has full read/write permission.

One example of a potential centralized distribution method, which includes built in safe guards is to configure Avatar for automated background operation in which both a local Avatar database and a network based Avatar database are specified for processing. The local database can contain critical file system rules, which must be maintained in the event of a network failure or breach. The network-based database can be centralized, or even localized, from a central distribution point. This network database can then be utilized for rapid deployment of system updates, etc. For example, if during operations a new attack is attempted and a security patch is required to fend off the attack and Avatar systems in the field are configured to check the network database every 12 hours, then the security patch can be distributed and installed globally within that 12 hour period. Assuming that the bandwidth is available, that time period can be shortened to any practical time period. This would allow for rapid baseline management and distribution for an evolving semi-real time need. In this specific example there are several advantages:

1. Training for Avatar's security and distribution needs are the same. Anyone trained in Avatar can fulfill both functions.
2. Fully trained senior people are a valuable scarcity. The ability to bring the full power of senior people to bear globally, in a timely manner from a centralized and safe location, expands the ability to react correctly in near real time.
3. Centralized distribution using Avatar is an example of working smarter, not harder. An army of field technicians is not required to install distributions, since they can be installed automatically. This is a significant cost savings in manpower and training.
4. Avatar's database system is flexible and multiple databases can be defined for execution. This means that database creation can be broken up and distributed within bodies of authority. In a complex multi-vendor environment you could potentially breakup these databases by vendor or any other functionality with either centralized approval and distribution or multi-centralized distribution.
5. Roll over functionality means that if a specific network operations center is not functional a secondary distribution point can be specified. This secondary point can also define roll over distribution points.

The second tool under discussion with Avatar is the Cryptographic Integrity Tool, CIT. In addition to the human readable report, CIT produces a machine-readable report that can be utilized by a threat assessment program like the VFind pattern analysis system to look for attack code (a.k.a. computer viruses). This specific machine-readable report is a list of files on the protected file system, which have been added or modified since the last time CIT was run. These are prime candidates for analysis. In addition, since CIT does not rely upon the system date and time function and it uses cryptographic digests, you will know within a specified time window when a file system event took place. That time window is the period of time between any CIT database creation and the following CIT execution. This information is very valuable for a Security Analyst.

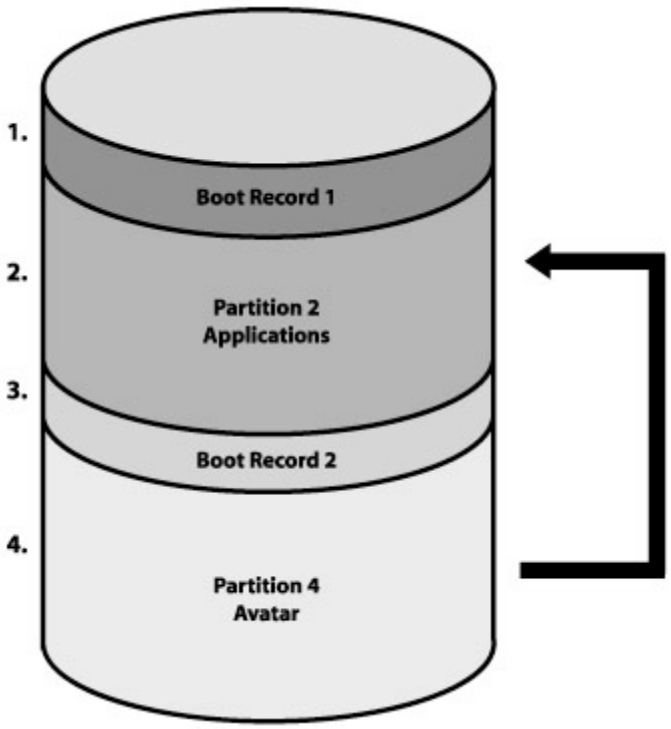
It should be noted that while Avatar only pays attention to the files and directory trees it is directed to protect, the CIT tool could evaluate the entire system. This makes the CIT tool complementary and valuable when ascertaining the condition of a system.

The CIT tool database format is open, and can be viewed by a standard text editor. This means that a collection of databases can be created and used as a diagnostic tool against any system. Such databases can reveal information, such as what revision a system conforms to. It can reveal which operational files were modified, and what the system has been doing. If these databases are downloaded to a central point, it can be used for almost instant detection of a previously unknown attack against the systems in a network. In addition, a centralized collection of operational CIT databases would allow for easy determination of baseline compliance, thereby closing a command and control loop. Digestion of these databases at the central location can produce almost any kind of report necessary for the maintenance of the systems.

The following three diagrams demonstrate some non-intuitive ways of configuring Avatar besides the standard of Avatar running on the system it is protecting.

Non-Intuitive Avatar Example Diagram 1

This diagram demonstrates how a multi-boot workstation can use "hidden partitions" to operate a backup operating system with Avatar. The Avatar system can then be used to maintain the baseline configuration of the primary partition. The advantage of this configuration is that even if the primary operating system is completely non-functional it can be returned to baseline configuration.



As a side note, this configuration can also be used with all of the VSTK/P tools and the operating systems used do not have to be the same. For example, partition 2 could be Microsoft based while Partition 4 could be UNIX/Linux based or any combination thereof. This configuration is not tied to any size system. It can be implemented on a large server or on a small laptop.

If your workstation hardware permits booting from the network then this entire configuration can be configured where the secondary operating system is network based and not system resident.

You should also note that this configuration and the configurations presented on the next page could also be used with all of the other tools in the toolkit. For example, using this configuration with CIT would allow you to diagnose a failed operating system that did not boot. Using this configuration with UAD and VFind would allow you to scan for viruses using a known clean operating system. The possibilities are endless.

Non-Intuitive Avatar Example Diagram 2

In this example, a security server has been setup to run Avatar. The application server disk drives are shared using a secure protocol to the Avatar server. Avatar controls the baseline configuration of the application server remotely over the network. If the Application server disk share "disappears" then standard off the shelf software will sound an alarm.

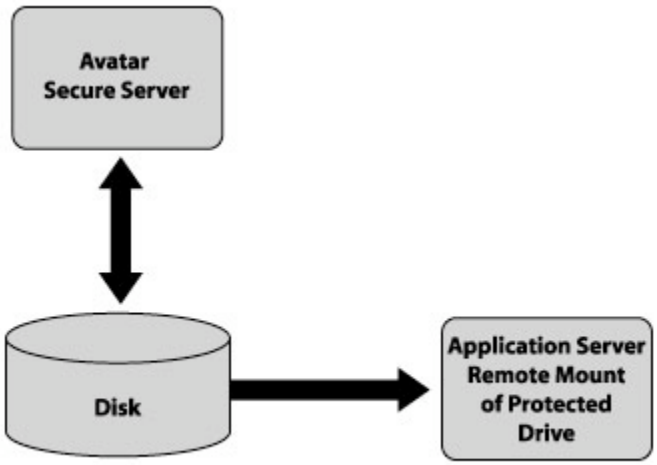


Figure 2

Non-Intuitive Avatar Example Diagram 3

This example is the inverse of example 2. In this example the application disk is attached to the Avatar server and shared out to the Application Server.

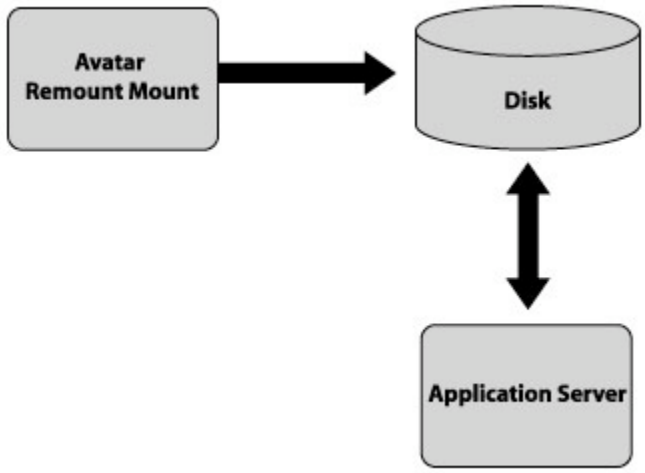


Figure 3

THD

THD answers the question, “How do you find a chameleon Trojan horse attack when there are no contents to scan?” The chameleon Trojan horse attack works because a user is able to redirect a system command to a program of the same name in a different location. The chameleon may or may not have contents. If the chameleon has contents then they can be located using VFind. If it does not have contents then THD can locate it because by definition, the filename must have the same name as another program on the system or the attack will not work.

An example of a common chameleon attack is to use the “touch” command to create an empty file with execute permission in the user path prior to the actual commands. Since many users put their home directory first in their path prior system directories these attacks are commonly performed on a per user basis. Commands on UNIX/Linux systems which are commonly attacked are: ls, cd, echo, pwd, rm, mv, mail, vi, ed, grep, login, logout, sh, csh, ksh, lp, ps and kill.

Bhead

Bhead is a simple tool. It is a program that reads a specified number of bytes from a file or byte stream and writes them to standard output. While this is not a complex task, the problems the tool can solve can be complex. One such problem is that UNIX systems running on Intel based PC architecture hardware can be infected with boot sector viruses. UNIX systems do not have a convenient way to read just the boot sector for scanning. Using the UNIX "dd" command the entire raw disk drive can be read and output as a byte stream. Scanning the entire drive would be a redundant waste of time, however, using Bhead the byte stream can be cut to just the portion of the drive that contains the boot sector. Bhead can also operate on files, floppies and tapes.

JDIS

JDIS is a tool that performs fast disassembly of Java bytecode. There is a version of JDIS called JADE (Java Advanced Disassembly Analysis Engine) built into VFind. It is important to disassemble Java byte code prior to virus scanning because it is the only way to associate constant pool entries with opcodes. This association allows the virus scanner to scan for what is actually going on in the program instead of guessing. VFind is the only anti-virus tool to provide scanning of Java disassembly.

JDIS is also provided as a stand-alone tool so that a security officer can disassemble a Java bytecode program for manual analysis. This can be valuable when confronted with a potential new, unknown Trojan Horse written in Java.

VGUI

All graphical user interfaces for VSTK/P have been called VGUI. It stands for VSTK/P Graphical User Interface. There are actually several versions of a graphical user interface for the VSTK/P tool kits. The first version was X-Window based using xview. It was discontinued when xview became obosolete. The second version was TCL/TK based. The source code for this version is available free on www.cybersoft.com but is unsupported. The third version was again X-Window based using Motif and is still supported for some customers. The fourth version is HTML/CGI based and is fully supported. All versions except for the HTML based version are obsolete. The HTML based GUI is supplied with its own web server called MiniWeb.

Since VGUI is delivered and executed in text source code it is easy for an authorized user to modify. This means that implementations in the field can be modified to meet the customers specific needs by the authorized end user. This would never be possible with compiled binary programs. For example, if the user needs to combine VSTK/P functionality with another program it is trivial and inexpensive to do so by modifying the html and cgi code.

The current web based VGUI allows access to VFind, CIT, Avatar, and other functions via virtually any web browser homed to the MiniWeb Server running on the target machine. The user can scan the system for viruses via VFind, baseline the system via AVATAR, and integrity check the system via CIT. The VGUI allows even a non-UNIX user access to the great majority of the VSTK/P tools. Most functions can be executed by one or two simple button clicks.

All of the VGUI underlying functions are supported by CGI scripts that are written in portable shell. Perl has not been used, as it may not be available on all platforms.

Web based VGUI is in its infancy. Future enhancements are planned

VSTK/P 166 and later has a full graphical user interface with portable Bourne shell scripts for the CGI applications. No version of the VGUI uses Perl.

MiniWeb Server

The MiniWeb Server is a compact web server based on the HTTP 1.1 standard. It supports the HEAD, GET, POST, and PUT access methods, the .htaccess file for access security, and Secure Socket Layer.

The MiniWeb Server is configured via one ASCII text file. The file permits customization of about 15 parameters, most of whose default values are adequate for almost all users. The two parameters that need to be customized by everyone are the IP address of the machine and the port number.

Concurrency is achieved in the server using POSIX threads. Each connection is handled by a separate thread. The number of threads is customizable if required. The default is 20.

The Server is implemented on all platforms supported by VSTK/P, except those on which POSIX threads are not available. Those platforms are HPUX-10, IRIX-6.2-MIPS, and OSF-ALPHA.

A debug mode is supported to a file or to the console for easy problem resolution.

While the Avatar and CIT systems can use remotely mounted file systems for their databases and execution, this only allows for near real time operation in that execution is scheduled. While this is a good thing because it allows for an orderly progression and an event window can be kept secret, or even random, there is still a need for real time deployment in the event of a serious attack. Avatar, CIT and all of the other tools in the VSTK/P can be deployed in real time for centralized distribution using the MiniWeb tool. In fact, MiniWeb can be used to control products developed by companies other than CyberSoft. MiniWeb is a miniature, high security, stripped down web server written from scratch.

MiniWeb has a very small memory foot print, requires minimal configuration and can coexist with all other web servers. MiniWeb can be bound to the system it is operating upon so that only a local operator can access it, or it can be bound to a specific list of systems or even opened. This list can include a centralized operations office. MiniWeb is intended as the server for the VSTK/P graphical user interface. It uses standard HTML 1.1 code and has the ability to execute both script and Perl CGI applications. Perl installation is not required; however, if Perl is not installed then Perl based CGI programs will not function. Using MiniWeb a centralized distribution point can force the execution of Avatar and/or CIT using a simple script thereby providing real time functionality.

Using Avatar, CIT and MiniWeb together provides most of the functionality needed for centralized distribution in addition to their security role. In recap:

1. Centralized distribution, baseline management and automated repair via Avatar.
2. Centralized control and intelligence via CIT. (Closing the loop.)
3. Real time control using MiniWeb.

As a final note to the MiniWeb program, I will once again point out that it can be used with any html or cgi scripts. Developing or changing graphic user interfaces in html is very cost effective and significantly reduces the risk of code bugs.

The MiniWeb Server manual is located in Appendix C.

RobotMode

This is the default mode for the Motif version of the VGUI. It is also a stand-alone program that can be executed by the operating system's clock function, by hand, or remotely by an authorized process.

RobotMode examines the entire VSTK/P security suite of tools and determines which tools need configuration. If the user has already configured a tool then no further configuration will be made by RobotMode. The program then configures any tools that still require configuration-using factory defaults, creates any databases that have not already been created and then executes all of the security tools in the most logical sequence possible.

If a CIT database has not been created, then RobotMode will create a CIT database for the entire system while simultaneously virus scanning the system using UAD and VFind with Loop Back. If an Avatar configuration file has not been created then it will create one followed by executing Avatar Create to create the Avatar database. It will also run THD and all of the other security tools as predetermined to be appropriate by CyberSoft.

The second time RobotMode runs, all of the tools will have already been configured and the databases built, either by RobotMode or by the user. At this point, RobotMode will run CIT, VFind, UAD, LBH with LBT, THD and Avatar Check/Correct. If necessary, it will also run MvFilter to remove any macro viruses.

RobotMode will continue to expand over its lifecycle. It is the first attempt to force discipline into the process. RobotMode encapsulates best practices for the use of the VSTK/P tools.

NTI

The Network Traffic Interceptor is a new concept in virus scanning network traffic. Instead of using proxies like other products, the NTI program actually implements an anti-virus firewall in the computer itself. This is a new standard in virus scanning because it intercepts any potential threat prior to its access to user application memory.

This is very important for the following reasons.

1. If a large number of people all attempt to use a centralized firewall then it will suffer bandwidth throughput problems. By locating the most process and bandwidth intensive part of the firewall on the local system there is no aggregate problem.

2. [*Most Important*] By intercepting viruses at the IP packet layer, they can be analyzed prior to reaching an application where they can potentially execute. This is especially true of e-mail and web browsers with extensions such as Java.

3. In addition, NTI with VFind allows end users to permit Java and JavaScript by default in their Web Browsers while insuring that known Java attacks are blocked, no matter what the source. This can be important when using smart web sites that utilize Java and JavaScript. Without this ability, the user has to choose between not accessing those sites, turning Java on and off on a per site basis or accepting the risk of leaving it on all of the time.

NTI is capable of scanning any traffic going into or out of a system. In NTI version 2.3, the first commercial version, CyberSoft has preconfigured NTI to virus scan all ftp file transfers, HTTP downloads, SSL secure sockets layer, HTTPS downloads, pop3 e-mail, SMTP e-mail and SMIME e-mail attachments (See NTI- Crypto below for SSL, https and SMIME information.)

NTI can be executed on both the local workstation and on servers. As of March 2000, NTI is available for Solaris 2.5.1 and above, HPUX 11.0 and above and Microsoft Windows NT 4.0 with Service Pack 5. It can be ported to any version of UNIX that supports the STREAMS module.

The NTI and NTI- Crypto systems are too big to put in this manual. They can take up a 2 or 3-day training session by themselves and therefore will not fit in a one-day course. For more information on NTI view the white paper
http://www.cybersoft.com/whitepapers/product_exp/nti_handbook.shtml .

NTI-Crypto

The NTI-Crypto option of the NTI program allows for effective virus scanning of encrypted data. Unless data is decrypted prior to scanning, any viruses that may exist will be hidden by the encryption algorithm along with the intentional hiding of data. NTI- Crypto deals with SSL encryption and SMIME encryption.

SSL encryption is used for all encrypted web page transactions (HTTPS) and is the backbone of e-commerce. NTI- Crypto uses the NTI system to intercept all encrypted SSL traffic prior to user access, then performs real time cracking and scanning of the SSL encrypted connection. If the data contains viruses, then the connection is broken and the user is sent a message as to the nature of the problem. The virus never enters the system application layer.

NTI- Crypto also intercepts and virus scans all SMIME encrypted e-mail messages and attachments. It does this by reading the SMIME decryption keys from the Netscape Browser SMIME key database. It then uses these keys to decrypt the message for scanning. If the message or its attachment is infected with a virus, then the message is blocked prior to entering the system application layer.

The NTI- Crypto technology is one of a kind. CyberSoft invented (patent pending) this sole source technology. No other computer security company has a product that can scan encrypted data until after it has entered the system and has been decrypted at the application layer. Once a virus is at the application layer the system is threatened.

UNIX Wrappers

The UNIX Wrappers tools are programs that take the place of and envelope the standard UNIX tar, cpio and mount commands. The tar and cpio programs are common ways of reading data into a UNIX system from removable media, such as tape or floppy diskette. The mount command is how file systems are mounted onto the UNIX file system for access. This includes network drives supplied by NFS (Network File System) and CD-ROM drives. When any of the wrapped tools are used to import files into the protected UNIX system the files are automatically virus scanned by VFind. If the files are uninfected they are allowed to enter. If they are infected or contain proscribed data they are blocked. This is a proactive tool that prevents infection.

Windows Removable Media Interceptor (Windows NT)

The Windows Removable Media Interceptor (RMI) intercepts all removable media mounts such as floppy diskette, CD-ROM and Zip disks prior to user access for virus scanning. If the media is infected then the user is denied access. RMI prevents infected files from entering a system.

Supported Platforms and License Information

The VSTK/P ToolKits are supported on almost every version of UNIX and Linux in current use and many that are no longer in common use. It is also supported on Apple Macintosh OS X, which is a UNIX like system. Finally, it is supported on Microsoft Windows NT and later (2000, XP). Generally, CyberSoft will port to and support any platform for the cost of the system. Many customers provide specialized systems to CyberSoft for this purpose. As long as we have a working system we will support it.

VSTK/P licenses are per system. We don’t care how many users or processors are on the system. The cost of the product is flat and the optional yearly maintenance and upgrade is also a flat cost per system. The first year of maintenance and upgrades via the Internet is free.

The deliverable VSTK media (CD-ROM) contains almost all supported platforms. The customer can install on any system for which he has a license from a single CD-ROM. Activation keys are text based and are obtained from CyberSoft. Due to the fact that activation keys lock to nodename (the name the computer thinks of itself as), the license key can be moved from any single system to any other single system with the same nodename, even binary incompatible computers and by reinstalling from the CD-ROM transfer the license without help from CyberSoft. This makes hardware upgrades trivial. Of course to do this legally, you need to remove the product from the first system. If you do not wish to keep the same nodename, you can contact CyberSoft for instructions on obtaining a new activation key.

VDL Update

VDL Update is a UNIX shell script supplied with all VSTK toolkits. Its purpose is to give the user an extremely easy manual or automatic means to update their virus definitions. When used from the UNIX command line, the user can manually update their definitions. When used from the UNIX cron system, the script runs automatically at the frequency and time set by the user. A file locking mechanism is employed to prevent updating of the VDL database while it is currently being used by VFind.

VDL Update only downloads VDL's if there has been a VDL change since the users last run. This is done by supplying md5 hash codes for the VDL database. If your local hash code does not match the hash code on the CyberSoft's server, an update is required.

After the VDLs are downloaded they are run against VFind as a final verification check. If all goes well, the new VDLs are accepted. As part of the installation of the VSTK ToolKit, the user is given an option of configuring VDL Update for automatic execution. By choosing this option, the user is assured that his VDL database is up-to-date as of the time of execution. Users are urged to automate the execution of VDL Update and to run it every hour.

OEM Information

Other Equipment Manufacture (OEM) sales of CyberSoft product, especially VSTK/P, are a significant source of income for the company. OEM companies incorporate our technology into their products. In general they do this by keeping our product as separate binaries and communicating with them either using our SmartScan technology, named pipe daemon or in the latest version of VSTK/P a standard socket daemon. The advantage of incorporating our technology in this way is that when we release a binary upgrade the OEM customer does not have to recompile their program.