CyberSoft White Papers

Why VFind

May 1996

Peter V. Radatti CyberSoft, Inc.

Prescript

While it is popular to not bother reading footnotes, it is worth the trouble with this paper. Some of the footnotes contain valuable information, however, they would destroy the flow and tempo of the paper and were therefore set aside as footnotes. Unless otherwise noted, all referenced papers are available in our White Papers directory.

VFind History And Initial Design Constraints

VFind is the name of the first tool to be created in what has become the VFind tool kit. The VFind program was created in 1988 by Peter V. Radatti after he encountered the Typhoid Mary Syndrome 1 on a UNIX network. At the start, the program was developed in the C language on a Sun Microsystems Sun 3/50 workstation operating SunOS. Designed for UNIX on a UNIX system, the product was unencumbered by any of the anti-virus design philosophies currently in use at that time for MS-DOS and Macintosh. Since the impetus for the design of the product was a computer virus that migrated between heterogeneous environments, it was decided that the product not only had to scan for UNIX viruses but also for MS-DOS, Apple Macintosh and Amiga viruses. Finally, it was recognized that UNIX systems, at that time, were attacked more often by forms of hostile software such as Trojan Horses, Logic Bombs and Worms than by computer viruses. UNIX computer viruses did exist and were damaging systems "in the wild" but the number of viruses were far outweighed by the number of Trojan Horses and Logic Bombs. 2 For this reason, VFind searches for all forms of hostile software

and does not limit itself to viruses. The product continued to be called a virus scanner because the general public didn't understand what the product did when we called it a "direct hostile algorithm examination and detection software package". Since the methods employed to search for the " other" forms of attack were the same as for viruses, we allowed the description of a "virus scanner" to stand.

As are most UNIX programs, VFind was designed as an object oriented tool that could be combined with other UNIX commands in order for the end user to resolve new problems that would not otherwise be resolvable without a special purpose program. The program could interact with other commands through the use of a UNIX pipe. This feature was taken advantage of in the early design of the product to replace the "target generator and the "report writer portions of a generic scanner with standard UNIX commands. The target generators job in a virus scanner is to identify files or devices that require scanning. This was easily accomplished by the UNIX "ls" and "find" commands. When generating targets for the scanner, there are significant advantages to using the "find" command. The "find" command allows the user to search selected parts of the file system tree and to restrict that search by any criteria desired, including date and time of last access. The "report writer was simplified by use of the UNIX "grep" and "awk" commands. This allows the user to reduce the verbose output from the scanner into a concise report displaying only the data the user wants, in the form desired.

In addition to files, magnetic tape was a common storage media for UNIX systems in 1988. Floppy disk drives were not common, but a few systems did have them available. It was therefore decided that VFind would have to scan all forms of storage and not just files. Since it was not possible to determine what type of storage systems the end user would have in addition to the file system, it was decided that VFind would have to include a universal interface that would allow it to read all forms of media. Fortunately, the UNIX system provided such a universal interface in the form of the "dd" command. [3] This command is able to read and convert into a byte stream any form of storage that the specific UNIX system running it is capable of accessing. For this reason, VFind is able to process everything from paper tape to the most esoteric holographic laser storage media.

The first commercial version of the product was completed and sold in 1990. At that point, CyberSoft decided to exhibit VFind at a national show. The show chosen was the UNIX Expo International 1991 in New York City. [4] This was, in fact, the first commercially viable anti-virus product for UNIX and the first virus scanner that was heterogeneous in that it was able to scan for viruses of multiple platforms other than the platform on which it was running. [5]

Storm Clouds

All of this was very heady stuff. A product that broke new ground in two important ways was developed in less than four years and appeared to be a minor financial success. As Frank Sinatra said, If you can make it in New York, you can make it anywhere appeared to be true, however, there were also storm clouds gathering. CyberSoft recognized many of the problems that would eventually become serious long before any of it's customers did.

One of the biggest technical problems was that UNIX attack programs were typically written in

interpretive languages such as script, and while it was possible to locate specific incidences of an attack, there was no way to generically describe the attack so that all instances of the attack could be found. Vagueness in the attack program caused by the typing and programming style of the author could take a simple "recursive bin remove attack" from a single pattern to many thousands of possible patterns. It was not practical to create and search for thousands of patterns for every attack, so a method of searching for these patterns that would be flexible, yet not create false notifications, was necessary.

It was determined that an entire pattern description language was necessary to perform this task and one was created as a high level abstract. After review of the abstract language, it was obvious that many parts of the language could be handled by the parser portion of a language compiler combined with modeling technology. 6 Now that the technology was identified to resolve the problem, we were able to revisit the problem and create the final pattern matching language.

As the final language was being defined, we decided to add additional features to it for a multitude of problems. One reason is that we continued to think in the terms of tools. We just didn't know what the end user might use this language for other than the purpose we intended it to be used for, so everywhere a new bell or whistle could be defined, it was. The second reason the language was extended was because Pete Radatti 7 was working on the technical problem of resolving complex English words and phrases within text and mixed text and binary files. This is a common problem in the Defense Department where the task is called dirty word checking. 8 Corollaries also exist in the commercial world where research reports, marketing plans and financial information is typically labeled using the words "TOP SECRET" or "COMPANY CONFIDENTIAL". Usually, this is done as a header or footer of a document but there are many variations.

The final language, CVDL, 9 was announced and released in 1992 as part of VFind Version 4.0 Release 0. By the time the language was completed, it was much more than what it started out to be. Not only was every technical problem defined resolved by CVDL, but it had become a general purpose pattern modeling system. With CVDL and the fuzzy logic afforded by proximity, Boolean and case control logic, it was now possible to define one CVDL model that would definitively locate all variations of the pattern, both known and unknown.

As a final stroke of good planning, it was decided that the CVDL language would be made fully available to VFind users through a command line interface. To further user acceptance of the language, it was published both as a technical white paper and as part of the VFind User's Guide. 10

The CVDL interface allows users to enter as many pattern models as desired into one file called a CVDL program. This feature was later utilized by VFind Version 5.0 when CyberSoft decided it was necessary to allow the end users of VFind to have control over some of the patterns searched for. CyberSoft delivered the "cyber_01.vdl" and the "winwrd.vdl" CVDL programs as part of the product. The "ìcyber_01.vdl"î program contained four models for UNIX attacks, while "winwrd.vdl" contained one model for detecting the Microsoft Word Concept Virus. 11

Monsoon

If the problem resolved by CVDL was a storm, then the second problem was a Monsoon. The problem was that end users considered VFind slow when comparing it to an MS- DOS scanner. This was not a technical problem but a more serious perceptional problem. Technical problems could be defined and resolved. We didn't know if we could resolve a perceptional problem and if we didn't, then potential customers wouldn't buy our product and we could be in serious trouble. 12

Our first problem was really caused because MS-DOS scanners did not scan every file on a system. At that time, MS-DOS virus scanners only scanned "ì*.EXEî" and ì"*.COMî" files. The MS-DOS scanners also had an advantage in that they did not scan the entire file. Most scanners only scanned parts of these files. This is a valid way to save time if all of the viruses that are being searched for only exist in the first several hundred bytes of a file.

Unfortunately, these handy conveniences did not exist in the UNIX system. UNIX does not identify its binary executables with anything as handy as a file name extension. It uses the execute permission bit setting of a file to identify a file as an executable. Not only that, but there are three different execute permission bits. These are "executable by self", "executable by others in the same group" and "executable by everyone else in the world". 13 Since these permission bits also control read ability and write ability and because the settings are traditionally expressed as three octal numbers which are a bother to correctly set, many people automatically set the value to "ì777"î which makes the file readable, writable and executable by everyone. This is still a common practice today.

Our second problem was that UNIX contained many different types of executables. For example, the SunOS system not only contained executables for Sun3 systems but also for Sun4 Sparc and Sun 386i processor types. The same source code program compiled on each of these systems, running the same version of SunOS, would produce completely different binaries. This was further complicated by the fact that different versions of the operating system could also create wildly differing binaries.

The third problem was that there were dozens of executable types on a UNIX system. Binaries are only about half of the executable programs contained in a typical UNIX system. Many programs are written in script languages of which there are three popular versions, Bourne, C Shell and Korn Shell. There are also dozens of other interpretive languages such as perl, awk, yacc and so on. Each of these languages could be used for attacks since most UNIX viruses were written in the shell script languages. 14

The fourth problem that had to be overcome was the fact that compound files are common in the UNIX environment. At the time, the only compound file that existed in MS-DOS was "zip" files and many virus scanners did not search them. UNIX systems typically contained "tar", "cpio", 15 "gzip" and pseudo-volume files. A pseudo-volume file appeared as a single file to the UNIX operating system but was actually an entire file system that was used by an emulation system for MS-DOS or Apple Macintosh. Many of these compound files were further encapsulated using UNIX or GNU compression. This is in addition to all the other compound files that can be typically found on MS-

DOS, Macintosh, Amiga and any other system that makes use of the UNIX system as a file server via NFS. 16

To recap, the problems encountered by VFind in attempting to increase its speed were:

1. UNIX didn't identify its executables in a useful way for scanning.

1. UNIX systems held many different types of binary executables.

1. UNIX has dozens of types of executables, not just binary programs.

1. Compound files are common in the UNIX environment.

The solution was that VFind had to scan every byte of every file in order to be able to resolve these problems. This made VFind appear slow compared to an MS-DOS scanner. The comparison was really unfair since CyberSoft clocked VFind at processing raw data at twice the rate of some of the MS-DOS scanners of that time. This was definitely not a technical problem, but the customers were not satisfied and only a technical solution would be effective.

The way we decided to attack this problem was to decide the customer was right. The product was too slow and we needed a way to make it faster. We started by optimizing the code but we also doubled the internal database so the product stayed at the same speed. Our next solution was clever and attacked the problem sideways instead of head- on. We decided that if MS-DOS scanners didn't have to scan all the files on a system, then we didn't have to either. It was easy to decide which files we didn't need to scan. The only files we didn't need to scan on a system were all those files that had already been scanned and had not changed since they were last scanned. This was a wonderful solution because we estimated that an average UNIX file system not used for database operation only modified approximately 10% of its files during a normal work week. If the 10% were evenly distributed, that meant that we only had to scan 2% of a system on a daily basis. The problem was how to determine what two percent needed to be scanned.

We first looked at the problem of determining which files needed to be scanned by looking at the date of the last modification field attached to every file in the system. There was even an easy way to utilize that field using our old friend the UNIX find command. We could just tell "find" to locate every file on the system that was created/ modified since the day before and we would be done. Our easy solution was dashed when we realized that anyone could modify a file and then reset the date of the last modification/creation using the UNIX "touch" 17 command. The next solution we looked at was to use CRC values. CRC values were easy, well understood and fast to compute, however, we didn't use CRCs when we became aware of the fact that an attack program could create a CRC value for a target file, infect it and then pad the file until the CRC value once again matched the original file. 18 The new caveat to our problem became that the method used must be impossible to crack. The only algorithms that even came near to that type of mathematically provable certainty were cryptographic. We finally decided to use the RSA MD5 hashing algorithm. RSA supplied us with a C source code function that performed the hashing and we started work on our cryptographic integrity tool, CIT. 19

The first beta version of CIT [20] was released with VFind Version 4.0 Release 4. As it was, it was difficult to interface with VFind and required that the end user periodically run a maintenance module on the CIT database. This was not the solution we wanted, but at least we had the program on the street and it worked.

Version 1.0 of CIT was delivered with VFind 5.0 in the winter of 1995 (releases 0 through 2). Version 1.0 resolved all of the problems associated with the earlier version. It interfaced with VFind via UNIX pipes and automatically performed all database maintenance on the fly. The first time we ran CIT with VFind, the combined tools took longer to run than VFind alone. The reason for that was that CIT was building its database and every file in the system was considered a new file to CIT. Therefore, not only did CIT have a lot of overhead but VFind continued to scan every file on the system.

The second time we ran CIT with VFind, the entire process was over so fast that we thought something was wrong. Nothing was wrong! Everything had gone wonderfully right. Our file system had less than 1% of its files modified and the number of files that VFind had to scan were small. The overhead created by running CIT was insignificant compared to the execution of VFind. We were looking at something like a 998 times decrease in turnaround time in scanning our system using CIT and VFind for execution "1 + X". We decided that this was just not a normal system since we had lots of files that were static and never changed. Our guess is that a real world user would encounter a smaller decrease in turnaround. We guessed that if their file system was modified 10% per week and the scan number is "1 + X" then given 100% of the file system minus 10% that was modified equals 90% saved. That is 90% that does not have to be scanned and a 90% decrease in turnaround for CIT and VFind. Lets assume an additional 10% overhead [21] for CIT, that yields an 80% savings in turnaround or a yield of 800 times faster. [22] If we do this on a daily basis instead of a weekly basis, the entire thing is divided by 5 workdays per week while keeping overhead for CIT at 10%. The savings in turnaround is 88% or 880 times. Put another way, if VFind took 10 hours to fully scan a system, CIT with VFind might take only 2 hours to work or 1.2 hours per day.

It appears that the combination of CIT and VFind solved our speed problems and other technical problems that were worrying us. VFind, like all virus scanners, can only locate viruses that it knows about. CVDL allows it to find viruses that it doesn't know about but it at least knows the model for the family of attack. What was worrying us was the fact that since VFind could only locate viruses that it knew, or could model, then how could we protect our customers from new attacks that we didn't yet know about? The answer was again CIT. CIT was able to detect any change in any file or byte stream, even single bit flips. It was able to detect any file that was added to the system because the file would not have appeared in its database and it could detect all files that were deleted from the file system because files would appear in its database and were no longer in the file system. Anything that was added or modified was a potential threat. A System Administrator could review a report created by CIT and very quickly determine if system components or critical files were modified that should not have been. As an example, if the "csh" program was modified and the System Administrator didn't modify it, then one could safely bet that it was attacked by a hacker [23] or virus and probably now contained a back door. If the System Administrator didn't want to review the CIT report on a daily

basis, he/she could have "cron" automatically run a report generator 24 that would search the report for critical changes and urgently notify the administrator.

Trojan Horses

This, in effect, closed some of the loops for the VFind tool kit. VFind could now locate all known attacks, in addition to all attacks that fit a CVDL model while CIT could determine unknown attacks. This was good in that we could now find known and unknown attacks. There was still a limitation in that VFind knew what was attacking a file while CIT only knew that a file was attacked, not what was attacking it. Nothing could locate attacks that were unknown, did not fit an attack model, or did not modify a file. CIT could detect that a file was added but very few people would review all of the files added to a system to see if it was something that VFind couldn't identify. This could happen with certain types of Trojan Horse attacks. We felt that we could close this gap even more by the addition of a third tool to the tool kit. That tool was our Trojan Horse Detector, THD. 25 We recognized that many Trojan Horses and even some UNIX viruses were launched using an old trick. That trick was simply to write a program or create an empty file and then name it the same as a system command. It would then be put somewhere in the file system where it might be executed prior to the real command. When the user executed the attacked command, they would execute the Trojan. This type of Trojan Horse is called a chameleon. 26 Chameleons are hard to spot because there are hundreds of commands on a system and each user's path statement may be different. We decided that we wanted a universal way of detecting chameleons.

When designing THD, our first decision was that like VFind and CIT it had to run as a background task initiated by the "cron" 27 system. All of our tools were designed to have low impact on the System Administrator. We decided that if the administrator had to individually run each program, then he/she would always have something more important to do and would never run it. In this way, our packages can run automatically and the administrator only becomes involved on an as-needed basis. In addition, we wanted THD to maintain the tool concept that is the hallmark of UNIX programs. After reviewing the design requirements of THD, we determined that the same mechanism that was used to generate target file names for VFind and CIT would also work for THD. In this case, THD accepts file names from standard input and writes its results to standard output. This allows THD to utilize almost the same scripts as VFind and CIT. THD could even be integrated with CIT and VFind through the use of the UNIX "tee" 28 command.

The way the THD program works is that it takes a list of file names, extracts the actual file name from the extended file name while maintaining an association between the two, and then sorts everything by the actual file name. All files ,whose actual file names are the same, appear next to each other in the work file and THD then reports them as duplicates with their absolute file name. This made the job of searching for chameleon programs possible. At the same time, we realized that THD could also perform other types of processing on file names. Some attacks always utilize the same file names. For example, the AT&T Attack Virus always creates the file "/tmp/gift". If "/tmp/gift" was located on a UNIX system, then the AT&T attack virus, some unknown descendent, or a hacker who utilized the same basic algorithm as the virus had run on the system. This was another problem to solve. We

created a text based database that contains the names of the files that THD needs to flag as "dangerous" for the user. Since the database is text based, the end user can modify it using a simple text editor.

Smarter Scanner

Now that the VFind tool kit was really cooking, it was time to add even more features. CIT was the result of an ongoing research project called CBEL. 29 A new product called SmartScan came on line out of this project. The purpose of SmartScan is to isolate and identify files at the atomic level. An atomic file is a file that is not part of a compound file and is not encapsulated in any way. Compound files have become a significant problem because of the explosive growth of the Internet. Almost everything attached to email messages is encapsulated in MIME or UUENCODED format. It is also possible to archive several files into one compound file which can then be compressed and encoded. This is very common for files that are downloaded using the web or ftp. This makes life very hard for a virus scanner.

A second problem that we wanted to solve was the fact that VFind scans every file for every type of attack. Why should we scan a Macintosh file for a MSDOS attack? This works for every combination of attack that we search for. If we were able to limit searches for attacks to only those files which can contain them, then we would be able to improve the scanner while decreasing resource utilization. Of course, we would have to process the file into its atomic level if it was not already there.

SmartScan was charged with solving these problems and did so. It resolves a file from its raw format into as many atomic files as are contained within the raw file. To do this, SmartScan recursively examines a file and attempts to unarchive, decompress or otherwise decompose the file into as many files at the atomic level, as possible. Once this is accomplished, each atomic file is examined and an identification is attempted. If an atomic file cannot be identified, then it is labeled as type "unknown". The atomic files and their associated types can then be passed on to VFind via its SmartScan interface. 30

Extra Features

One of the things that VFind was designed to do, but many people never think of, is the ability of using the CVDL language built into VFind to search massive amounts of data for specific information. This makes a lot of sense because VFind and CVDL are general purpose search tools. One example of using VFind to search for data is to define a CVDL model that fits the data of interest and then allow VFind to search the entire system for the data. If it finds data that fits the model, it will report back the file name containing the data. This is especially useful if you are searching through an entire corporate network and you are not entirely sure of how the data may be represented. This feature can also be used to perform security audits of restricted or compartmentalized data. How many people in the building have information dealing with the new drug or precursors that are under development? The answer may surprise you and only a pattern modeling language like CVDL can provide the answer with any measure of assurance.

An extra feature of CIT is its ability to act upon raw data to provide integrity for that data. If you are sending massive amounts of data across the Internet, Federal Express or the Postal Service and you want to make sure that the data is not corrupted or modified in transit, then CIT will help. Use CIT to generate a hash of the data at the transmitting end. Send the hash to the receiver via a secondary method such as fax, voice phone, direct dial modem or just print it on paper and include it in the box with the media. Once the receiver has the data, they can run CIT against it and compare the answer to the answer generated at the transmitting end. If the answers are not the same, the data was modified. This is a useful feature considering the increasing industrial espionage that everyone is burdened with today.

CIT has a secondary function that is most useful to help desks. Using CIT to control baseline configuration, help desks can reduce turnaround time on complex problems from hours to minutes. The only stable "fact" in help desk lore is that the system is down and the end user never touched anything. Of course, the end user modified the system but would never admit it. It could take a help desk hours to discover what was modified so they could fix it. If CIT was used to document the system when it was first installed, it could then be executed by the help desk to create a report of everything that was added, deleted or modified in the system. This is normally all a help desk needs to know in order to be able to solve a software problem. It can even help with virus and hard disk failures. If all of the executables on a system are modified, then a virus is in the system. If large amounts of data in the system is modified, both executables and data files, then the system may be corrupting data and will fail soon. No matter what the problem is, CIT can reduce the amount of time a help desk needs to solve it.

Serendipity

Not everything that the VFind tool kit does was designed into it as a feature. Some of the things that it does are side effects, serendipity or happy accidents. VFind was never designed to predict disk drive failure but it seems to work. 31 Disk drives are electromechanical and electronic devices. There are lots of effects that act on them, including the stretching and tearing of the magnetic substrate that is bonded to the physical platter of the disk drive. Remember that the substrate is spinning at high speed and the normally simple physical interaction of centrifugal force with the media is complex. I don't even begin to know why, but UNIX systems tend to start producing lots of soft read errors when running VFind about 3 weeks prior to total failure. The best that I have been able to detect a disk drive failure on a system that did not run VFind was about 2 days, usually the answer is about 20 minutes. I don't know why. I can't prove it, it has never been scientifically tested, but it works. We even count on this effect here. Our main server is called "rinc1". It started life as a Sun 3/50, moved to a Sun Sparc station 1 and is now a Sun 4/110 server. It has always used the same disk drive. About two weeks ago, we noticed lots of soft read errors being reported when VFind was run. We did a complete system backup onto tape. Last week, the hard drive came to a hard stop. We replaced the drive, restored the system from the backup and were fully operational within several hours. It would have been a mess without the recent backup. 32

One of the features we never thought of but were advised of by a third party testing the VFind tool kit is the ability to use THD to detect illegal copies of commercial software on a system. If the THD database is programmed with the names of files corresponding to programs that you don't want installed on a system, then THD will report the fact of their existence every time it is run. It can also be used in license management. If you program THD with the file names of all the packages you have licensed and then run THD, it will generate a report of what is installed on the system. This is especially nice if you own floating licenses with a cap. You can easily prove that you have not exceeded your license cap and if you did, correct it.

The Final Frontier

This brings us to the subject of what is CyberSoft going to do for an encore? Has VFind hit the pinnacle apex? Is there anything left undone? The answer is really simple. CyberSoft has pushed the edge of technology with VFind, often breaking its own records. The reason we did that instead of sitting back and producing a "me-too" type of product is because we like the edge. The challenge of going just a little bit further than anyone else is an addiction. At the same time, we are fiercely loyal to our customers. Pushing the edge feels good and is good for our customers. They end up with new technology that solves problems better and we get to "feel the burn". The best part of being at the technological top is that you can always see a little further than anyone else. There is always someplace higher to go. Remember when Mount Everest was the top? Then we landed a man on the moon and now one of our man- made satellites is preparing to leave the solar system. It is always possible to go higher.

While we don't want to give away too much of our future plans, we can tell you that multiple long term research projects are coming close to developing new technology. Within a couple of years, you can expect to see more answers from CBEL, Big Crunch and our Hot Glue projects. What are these project and what do their names indicate? That would be telling! Sit back and enjoy the ride because we are about to redefine the top..again.

Footnotes

1. Typhoid Mary Syndrome was described in Heterogeneous Computer Viruses In A Networked UNIX Environment.

2. Reference "computer Viruses In UNIX Networks" for some insight on how Trojan Horses and Logic Bombs operate under UNIX. See Section 2.

3. For more information about "dd", enter the command "man dd" at any UNIX prompt.

4. UNIX Expo 1991 at the Jacob Javits Center. CyberSoft had a medical theme where the booth staff wore lab coats and pill bottles full of candy decorated the exhibit. Toy hypodermic needles from a joke shop (blood from a stone type) were also displayed and stolen!

5. The first anti-virus product for UNIX was not a scanner but was an integrity system called "Integrity Shell". Integrity Shell was created in 1985 by Doctor Fred Cohen (fc@all.net). Doctor Cohen stated during a telephone conversion on May 1, 1996, that the Integrity Shell was not commercial and called VFind the world's first commercially viable anti-virus product for UNIX. VFind was the second anti-virus program ever created for UNIX, but the first scanner created for UNIX.

6. A language compiler is a program that converts source code such as FORTRAN, Basic or C into binary executables.

7. The designers of the CVDL language were, Steve Jones, George Keen and Pete Radatti of CyberSoft. The final language definition and programming was performed by Steve Jones.

8. In this case, dirty word checking referred to searching for classified words or phrases in a document or binary file. This is a much harder problem than it appears to be on the surface.

9. Reference "The CyberSoft Virus Description Language", October 1992.

10. It first appeared in the October 1992 "Virus News and Reviews", page 495, published by the Virus Research Center of the International Computer Security Association, Washington, D.C.

11. Refer to "Heterogeneous Computer Viruses In A Networked UNIX Environment", page 4 and 5.

12. As someone wiser than me once said, "the best way to become a millionaire, is to start with two million dollars and start a company. The two million will soon be reduced to one million and you will be a millionaire".

13. For more information on permission bit settings in UNIX, enter the command "man chmod" and "man chgrp" at the UNIX prompt.

14. Reference " A Short Discussion On The Plausibility of UNIX Virus Attacks".

15. For additional information on these commands, enter the command "man tar" and "man cpio" at the UNIX prompt.

16. NFS - Network File System. Part or all of the UNIX file system appears as part or all of the file system to a networked computer. A typical arrangement is a UNIX server with UNIX and PC systems attached via PC-NFS.

17. For more information, enter the command "man touch" at a UNIX prompt. On some systems there may be multiple versions of "touch" and you may have to tell the "man"

command you want the "5bin" version.

18. This effect was first noted by a virus researcher other than one of ours. Unfortunately, we have been unable to identify who first noted this effect. Sorry!

19. We don't try being clever about naming products here. Our virus finding program is called VFind. Our Cryptographic Integrity Tool is called CIT or Cryptographic Integrity Tool. Just be glad we are not a restaurant. (pronounce CIT as "sit").

20. CIT was designed and written by Pete Radatti, Mike Ventura, Bill Barnes and George Keen of CyberSoft.

21. A 10% overhead for CIT is a very large number assuming a large to medium server, while it may be too small for a small workstation or PC. The volume of data determines the overhead, not factors that we can control in or program.

22. This was not scientifically tested, but we used wristwatches and the systems that were in our laboratory. We will perform more detailed and controlled experiments to quantify. Results are not guaranteed and are subject to change, depending upon many real world variables outside of our control.

23. The correct term is "cracker" as in safe cracker, but the popular press has redefined the word hacker to be the same as cracker. We now use the words interchangeably.

24. An example CIT report generator program written in script can be located at our web site. The name of the file is "ccheck.sh" (critical check).

25. We pronounce THD as "thud".

26. A chameleon is a lizard that changes color to fit into its background. This form of computer attack changes names in order to be something it is not, just like the lizard.

27. "cron" is a UNIX command that allows the system to start programs at any time or day. In our case, cron is used to kick off our security tools during off hours when the overhead would not be noticeable.

28. The UNIX "tee" command splits the standard input into two outputs. Execute the "man tee" command at the UNIX command prompt for more information.

29. CBEL - CyberSoft Binary Editing Language is pronounced "Sea Bell"

30. For unusually paranoid customers, they can either ignore SmartScan entirely, or tell VFind to ignore the identifications provided by SmartScan. This will allow SmartScan to decompose the file into its atomic components and VFind will then scan each atomic file for every form of attack.

31. There seems to be no end of what creative people can do with these tools. I expect we may even hear about stock marked predictions using VFind!

32. We normally do regular backups, but this backup was made out of sequence because of the soft read errors we were getting. It turned out that not having to restore incremental backups saved us a lot of time.