Symantec White Paper Series
Volume XXXIV

Understanding Heuristics:

Symantec's Bloodhound Technology

SYMANTEC.™

Table of Contents

*Portions of this white paper were first printed in *Virus Bulletin.*

Understanding Heuristics: Symantec’s Bloodhound Technology

Introduction

The word “heuristic” was initially coined by the Greeks; its original form was heuriskein, which meant “to discover.” Today, the term is used almost exclusively in computer science to describe algorithms that are effective at solving complex problems quickly, but yield less than optimal solutions because of the accompanying speed improvement. To illustrate this concept, the Traveling Salesman problem is a classic enigma in computer science for which computer scientists have devised many heuristic solutions.

Englebert the Twine Salesman and Heuristics

Englebert, a twine salesman, has to travel to a number of different cities, visiting each city just once. He would like to minimize his travel time so he spends as little time as possible in airplanes. If Englebert had to travel to Los Angeles, Oregon, Paris, and Rome, he would minimize his travel by visiting Los Angeles first, then Oregon, followed by Paris and then Rome. If he traveled from Los Angeles to Rome, back to Oregon and then to Paris, he would have to spend a great deal more time in airplanes. Obviously, our first solution is better; in any case, it seems like a trivial task to plan Englebert’s travel schedule.

What if Englebert had to travel to 500 cities? To find the absolute best solution to this problem would take today’s fastest computers years of computational time. Obviously, spending this much time on a computation is unfeasible, so computer scientists came up with heuristic algorithms to help solve such problems faster. A heuristic algorithm makes certain assumptions about the problem it is trying to solve. These assumptions allow it to cut down on computation time and still produce relatively good results. By its very nature, a result produced by heuristic algorithm is not guaranteed to be the best possible solution. Any time programs (or persons, for that matter) makes assumptions, they are bound to make mistakes. In many cases a heuristic algorithm yields a solution to a problem that is very close to the optimal solution, but there is no guarantee that this will be the case. Surely, Englebert will be satisfied to travel 100,055 miles if he knows that the best he could do would be 100,000 miles, especially if he doesn’t have to wait several hundred years for his travel itinerary.

Why the digression into computer science theory and twine salesmen? Well, in the anti-virus field, we have our own set of difficult problems to solve. Some of these problems are even more difficult to solve than the Traveling Salesman problem described above. In fact, some of the problems anti-virus researchers are tackling are considered unsolvable. In other words, it is impossible to come up with the exact, best solution to the problem in any finite amount of time. The only way to solve such a problem is by using heuristic algorithms.

Virus Detection: An Inexact Science

The task of determining whether a computer program is a virus is an unsolvable problem. It is impossible to write an anti-virus program that is capable of correctly telling a user, with a 100% success rate, whether a program is infected with a virus, for all possible viruses that have ever been or could be written. If it were possible to solve such a problem, all of the major anti-virus vendors and MIS personnel would be dancing through the streets with glee. It would mean the end of costly-to-develop and difficult-to-distribute monthly virus updates and the end of troublesome false positives.

Unfortunately, the general problem of discerning between viral and non-viral programs is unsolvable. Consequently anti-virus researchers have devised a number of innovative heuristic methods to help detect the tens of thousands of computer viruses. The best known technique is known as signature scanning. Most people don’t think of signature scanning as a heuristic technique, but it is.

An anti-virus program that performs signature scanning maintains a database of signatures and searches for them in every program on your computer. Each signature is a short sequence of bytes that is extracted from the body of a given virus and the anti-virus program has a different signature for each virus it is capable of detecting.

Usually this sequence of bytes is both unique to the virus and comprises only a small percentage of the total set of bytes that make up the virus logic. The former attribute reduces the likelihood of the anti-virus program falsely identifying non-infected programs as infected. The later attribute is required; otherwise anti-virus data files would be hundreds of megabytes in length (in essence, the signature data file would have to include a full copy of every detected virus in every month's definition updates).

An anti-virus program that uses a signature scanning technique can identify whether a given program contains one of its many signatures, but cannot tell you for certain that the program is actually infected with the associated virus. Usually users trust the guess of the anti-virus program when it makes such an assessment, since the odds are in its favor. However, the identified program could contain random data that coincidentally looks like the virus or it could contain actual legitimate program instructions that by chance matched those bytes in the virus signature. So, while there is an extremely high probability that an identified program is a virus, our signature scanner cannot tell us this definitively. Consequently, signature scanning is a heuristic algorithm.

The signature scanning technique is the most widely used technology in anti-virus programs today. While it does have its faults, it is very effective at identifying viruses for which the anti-virus program has a signature in its data files. Unfortunately, virus writers are ever vigilant and are constantly creating new viruses. In most cases, signatures added to the anti-virus data files to detect earlier viruses are powerless to detect new virus strains. And with the ubiquitous nature of the Internet, these new viruses are accessible to almost any user in minutes. These factors have created the need for anti-virus technology that is capable of detecting viruses without signatures and without the slow and expensive process of virus analysis.

Heuristics Logic

As we have seen, even the most ubiquitous anti-virus technologies employ some form of heuristic logic. However, in the anti-virus industry, the term "heuristics" (a noun) is invariably used to describe a specific type of virus detection technology. Specifically, heuristics is a term coined by anti-virus researchers to describe an anti-virus program that detects viruses by analyzing the program's structure, its behavior, and other attributes instead of looking for signatures. During the remainder of this paper, the terms "heuristics" or "heuristic scanner" will be used to describe this technology.

For instance, there are two ways to catch a criminal. A police officer can go to a crime scene and dust for fingerprints. If the officer finds a fingerprint at the scene, he can check his database of fingerprints back at the police station. Should he find a match, he can look up the criminal's file, then locate and arrest him.

Unfortunately, many criminals are first-time offenders and have not yet been fingerprinted. In this case, the police officer can try a different tactic. The officer can observe each person he comes into contact with and make an assessment as to the likelihood of that person being a criminal. If a pedestrian walking by the officer is wearing a bullet-proof vest and carrying a shotgun, the police officer could make a reasonable assessment that the person was up to no good and arrest him. On the other hand, the officer would probably only take a quick glance at a nanny carrying a bouncing baby boy before moving on. Of course, the police officer might inadvertently arrest an innocent person and occasionally miss some wrongdoers, but a well-trained officer will probably have a high probability of success.

Anti-virus programs that employ heuristics use an analogous approach to detect computer viruses. Each time a heuristic anti-virus program scans an executable file, it scrutinizes the program's overall structure, its programming logic or computer instructions, any data contained in the file, and a number of other attributes. It then makes an assessment of the likelihood that the program is infected by a virus. Like the police officer, sometimes the heuristics will fail to detect or recognize virus-like behavior. And, like our officer, the heuristics may inadvertently identify innocent programs as being infected with a virus.

Today's state of the art heuristic scanners achieve a 70 to 80% detection rate of new and unknown viruses, according to industry experts. These rates are commendable given the difficulty of the problem. While most heuristic scanning technologies have achieved similar virus detection rates, their propensity to falsely identify clean programs varies widely. Some popular anti-virus products regularly falsely identify clean programs with their heuristic scanner, which has given heuristics a bad reputation that is not necessarily deserved. Later, we'll see why some heuristic anti-virus programs have a penchant for crying wolf.

A big plus for heuristics is the ability to detect viruses in files and boot records before they have a chance to run and infect your computer. Just like a standard signature scanner, the user can initiate an on-demand heuristic scan of a new program or diskette before it is used. Likewise, users who run an on-access anti-virus program with heuristic scanning technology can detect a high percentage of new viruses as they are downloaded from the Internet or saved from an email attachment.

Other anti-virus technologies, such as behavior blocking or integrity checking, actually require the virus to execute on the host computer and exhibit suspicious and potentially harmful behavior before the virus can be detected and stopped. Both heuristics and signature scanning get a positive checkmark in their ability to stop a virus before it has a chance to wreak havoc on your computer.

To date, most of the research done on heuristics has concentrated on the detection of DOS executable viruses. Consequently, this paper covers the underlying concepts of heuristics by examining how anti-virus researchers have attacked the DOS virus heuristics problem. However, the techniques described below have also been used to detect boot viruses and, as we'll see, they can even be used to detect the new generation of macro viruses that are wreaking havoc in the enterprise.

How Do Heuristics Work?

Anti-virus researchers have investigated two competing heuristic scanning architectures: static heuristics and dynamic heuristics. The primary difference between these two schemes is whether the heuristic scanner employs CPU emulation to search for virus-like behavior. For now, let's ignore the differences and discuss those attributes that are common to both architectures.

The typical heuristic scanner has at least two phases of operation when scanning an executable file for viruses. In the first phase, the goal of the heuristic scanner is to catalog what behaviors the program is capable of exhibiting. The heuristic scanner starts by determining the most likely location where a virus would attach itself to the executable file, if the file were infected. This is an important step because some executable files are many hundreds of kilobytes or even megabytes in length. Performing detailed heuristic analysis on such a large program would be excruciatingly slow. Given most DOS-based computer viruses are only a few kilobytes in length, a well designed heuristic scanner can significantly limit those regions of the file to be scrutinized. Most often, this region will be the first and last few kilobytes of the file.

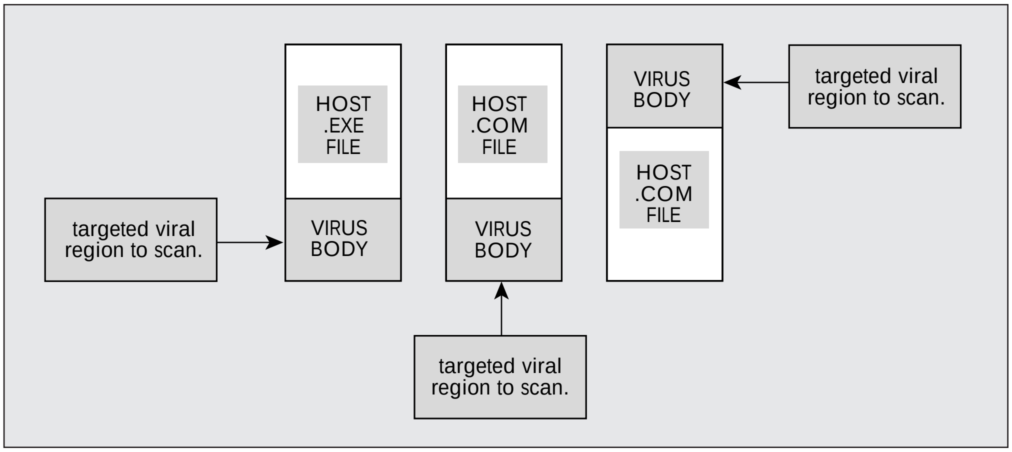
HOST .EXE FILE
VIRUS BODY
HOST .COM FILE
VIRUS BODY
VIRUS BODY
HOST .COM FILE
targeted viral region to scan.
targeted viral region to scan.
targeted viral region to scan.

Figure 1. DOS file viruses typically append themselves on the end of DOS .EXE files. DOS file viruses prepend and append themselves onto DOS .COM files. Other infection techniques are also possible but less common.

Once the heuristic scanner has identified the likely area of viral infection, it analyzes the program logic contained in this region in an attempt to determine what its computer instructions are capable of doing. This is itself an extremely difficult problem since there are so many different ways to write a given computer program. For instance, consider the following two sequences of instructions:

```
Example 1
MACHINE LANGUAGE BYTES (IN HEXADECIMAL)     USER-READABLE INSTRUCTIONS
B8 00 4C                                    MOV         AX, 4C00
CD 21                                       INT         21

Example 2
MACHINE LANGUAGE BYTES (IN HEXADECIMAL)     USER-READABLE INSTRUCTIONS
B4 3C                                       MOV    AH, 3C
BB 00 00                                    MOV    BX, 0000
88 D8                                       MOV    AL, BL
80 C4 10                                    ADD    AH, 10
8E C3                                       MOV    ES, BX
9C                                          PUSHF
26                                          ES:
FF 1E 84 00                                 CALL FAR [0084]
```

Figure2.

Two examples of how to write program logic on the PC to terminate a program and return to the DOS prompt. The left side shows the hexadecimal machine language bytes that the computer's microprocessor understands. The right side shows how a programmer would type the instructions into a computer.

Both of the examples above perform the same function: They cause a program to terminate itself and return to the DOS prompt. However, if we look at the sequence of machine code bytes that make up each set of instructions, they look entirely different. The first code sequence calls upon the operating system using a simple, common technique. The second sequence uses a much more round-about approach to make the same request to the operating system.

Machine language bytes for example 1: B8 00 4C CD 21

Machine language bytes for example 2: B4 3C BB 00 00 88 D8 80 C4 10 8E
C3 9C 26 FF 1E 84 00

Figure 3.

Two examples of machine language bytes that instruct the computer to terminate a program. Even though the two sequences of bytes tell the computer to do the same task, they look entirely different.

Given there are practically an infinite number of ways that one can write code, it would seem to be almost impossible for a computer program to examine the sequence of bytes that make up a program and glean any information at all. Luckily, most DOS viruses actually use straightforward techniques like the first one shown above to accomplish most of their tasks. In any case, how is a heuristic scanner supposed to detect this type of behavior? Static and dynamic heuristic implementations accomplish this task using markedly different techniques.

Static and Dynamic Heuristic Scanners

The static heuristic scanner recognizes various program behaviors using a handful of methods. First, the static heuristic scanner can maintain a large database of byte sequences (signatures) like the ones above; it associates each sequence of bytes in this database with its functional behavior. The scanner can use simple wildcards to help match information that may change from virus to virus:

#	BYTE SEQUENCE	ASSOCIATED BEHAVIOR
1.	B8 ?? 4C CD 21	Terminate program (permutation 1)
2.	B4 4C CD 21	Terminate program (permutation 2)
3.	B4 4C B0 ?? CD 21	Terminate program (permutation 3)
4.	B0 ?? B4 4C CD 21	Terminate program (permutation 4)
	Ö	
100.	B8 02 3D BA ?? ?? CD 21	Open file (permutation 1)
101.	BA ?? ?? B8 02 3D CD 21	Open file (permutation 2)
	Ö	

Figure 4.

The heuristic scanner has a database of behavior signatures. If one of the above byte sequences is found inside of a program, it indicates the program is probably capable of exhibiting the associated behavior. The "??" shown above are called "wildcards," and will match any byte value.

Program bytes: B4 09 BA 20 01 CD 21 **B8 02 3D BA 12 34 CD 21 CC B8 FF 4C CD 21**

Signature 100 Signature 1

Matching signatures: 100. B8 02 3D BA ?? ?? CD 21 This program opens a file
1. B8 ?? 4C CD 21 This program terminates itself

Figure 5.

The program bytes shown above represent a simple program that could be a virus. If a heuristic scanner were to search for all of the signatures shown in Figure 4, it would find signatures 1 and 100 in the program. Notice signature number 100 uses wildcards to match byte values of 12 and 34 in the program.

These strings look strikingly familiar to the standard virus signatures used by anti-virus products for many years, and for good reason. They are! However, standard virus signatures such as those used by traditional anti-virus scanners are used to definitively identify a specific virus strain. The signatures shown above are used to identify whether or not a given program contains the program logic to exhibit a given behavior, which may or may not be viral in its own right. If our heuristic scanner locates such a string in the program, it does not necessarily mean that the program is viral. All it indicates is that the program may be capable of exhibiting a given behavior.

Traditional virus signatures (found in Virus Bulletin magazine):

Paulus.1804: B9 D5 00 8B DE ?? ?? 27 06 53 ?? ?? 07 86 CA ?? ?? 86 CA 2E 88 07 4A ??
V.974: 9C 80 FC AA 75 04 B4 BB 9D CF 80 FC 4B 74 0B 80 FC AB 74 06 9D 2E FF 2E

Figure 6.

The above virus signatures detect specific computer virus strains instead of an isolated behavior such as opening a file or formatting the hard drive.

In addition to this simple database of behavior signatures, the static heuristic scanner may also use more elaborate, hard-coded programs (written by an anti-virus researcher in programming languages such as C++ or assembly language) to seek out and recognize more complex virus behaviors. For instance, encrypted and polymorphic DOS viruses often have computer instructions to unscramble (decrypt) themselves when an infected program is launched. The bytes that comprise these computer instructions can vary widely in appearance; even so, it is possible for the anti-virus researcher to write fairly simple subroutines to recognize a large percentage of virus decryption routines. If the heuristic scanner's detection subroutine happens to locate what it believes is a decryption routine, it will also catalog this behavior.

While the static heuristic scanner relies on simple signatures and code analysis subroutines to catalog a program's behavior, the dynamic heuristic scanner uses CPU emulation to gather its information. After some initial sanity checks, the dynamic heuristic scanner loads the suspect executable file into a virtual computer and emulates its execution. The program being emulated has no idea it is running inside of a simulated computer; it believes it's running on a real system. As the program runs within the virtual computer, it exhibits behaviors that are cataloged by the dynamic scanner.

While the program is running within the virtual computer, the dynamic heuristic scanner can monitor all requests (interrupt calls) it makes to the operating system. Any time the virtual operating system is called on by the emulated program, the scanner records the behavior and then allows the emulated program to continue executing. Since the vast majority of DOS viruses rely heavily on the operating system to infect new programs, this monitoring is a robust way to determine virus behavior and gives the dynamic scanner an advantage over the static heuristic scanner.

The following analogy illustrates the difference between the static and dynamic heuristic scanners. Let's assume I'm the ambassador (and a crafty spy) at the U.S. embassy in Votslovia. If I want to spy on the top Votslovian nuclear scientist who works four blocks away, I can take many different paths to reach him. I could go north one block, east two blocks, and north again one block. Alternatively, I could go west one block, north two blocks, and east three blocks. Both paths would take me to the scientist's laboratory. There are many such paths I can take to reach his laboratory, just as there are many possible sequences of computer instructions that will achieve the same goal.

A Votslovian intelligence official could determine that I was spying in several different ways. First, he could choose to monitor a likely route between the embassy and the scientist's lab. If he observed me walking down this route, he could report this information to authorities. However, what if I took an alternate route? In this case, he would probably miss me. Of course, he could choose to recruit an additional agent to monitor another route. But, there are many, many possible routes that I could take.

Alternatively, the intelligence officer could stake-out the scientist's laboratory. If he does this, he doesn't need to concern himself with how I got to the nuclear lab. Instead, he just snaps a picture showing that somehow I arrived there.

The former strategy is analogous to that used by static heuristic scanners during the program analysis phase. The static heuristic scanner looks for different behavior and its success is highly dependent on how the suspected program implements its logic. If the program being analyzed uses an obfuscated method of calling the operating system, the static scanner may fail to detect this behavior, just as the intelligence officer would fail to detect my prying if he monitored only one or a small number of routes to the scientist's lab.

The later strategy is analogous to that used by the dynamic scanner. The dynamic scanner lets the program run freely within the virtual machine. The program can use any logic it likes to get its job done, but eventually it will call on the operating system and when it does so, the end result of all its computations and machinations will be made clear.

It would seem as though the dynamic heuristic scanner would be much more effective at analyzing and identifying the behaviors of a program. In many cases, it is. However, dynamic heuristics can also be much slower than their static counterpart. CPU emulation is a relatively slow process, and is usually much slower than scanning for strings in a limited region of memory. Furthermore, CPU emulation is susceptible to the logic and whims of the program being emulated. For instance, what if we wanted to detect the following virus using the dynamic technique:

Virus in pseudo-code:

```
1.      If the current hour is even, skip to instruction 3.
2.      Go to step 2.
3.      Infect a new program using simple, identifiable computer instructions.
4.      …
```

Figure 7.

A computer virus that uses a logic trick to hide its infection instructions from a dynamic heuristic scanner. The virus only infects a new program under very specific circumstances. When such a virus is emulated in a virtual computer, the logic trick may cause the dynamic heuristic scanner to miss the infection logic.

The dynamic heuristic would emulate the above program within its virtual computer. Immediately, it would stumble on the first instruction which would check the current time. If the current hour during the emulation session happened to be odd (1 p.m., 3 p.m., and so on), the emulated program would execute instruction 2 indefinitely and the dynamic heuristic scanner would fail to observe any of the virus' additional behaviors (on lines 3 and beyond). In general, if the CPU emulator fails to provide the virus with what it wants, the virus' logic may prevent it from executing its tell-tale behaviors and giving the virus away. Unfortunately, the CPU emulator is not a fortune teller.

On the other hand, our static heuristic scanner would detect all the behaviors in the above program since it is not constrained by the program's logic. The static scanner looks throughout the virus' body for behaviors, regardless of whether they would be reached during a typical execution of the virus. As you can see, both approaches have their benefits and drawbacks.

After analyzing the program's logic and instructions (using either static or dynamic techniques), the heuristic scanner also searches for any strings of bytes or suspicious data stored within the likely viral region. Virus writers often include expletives or the word "virus" in their viruses. Since most programs don't normally have this type of data, the presence of a text string is often a strong indicator that the program may be infected. The heuristic scanner logs these strings and other tell-tale signs of infection for use during the second phase of the heuristics process.

Once a set of possible behaviors and attributes has been obtained, the second phase begins: analysis of the observed behaviors. In general, the second phase is the same for both dynamic and static heuristic scanners. The heuristic scanner now has a list of all behaviors, virus-like attributes, and other information that it was able to glean from the target program. At this point, it must make an assessment as to whether the set of behaviors that was detected looks virus-like or not.

The heuristic scanner must know, for instance, that appending .COM viruses may use the following behaviors when infecting a new executable file.

Ask the operating system to:

1. Locate the first COM file in the current directory.
2. Open the file.
3. Seek to the end of the file.
4. Seek to the top of the file.
5. Write out 3 or 4 bytes to the top of the file.
6. Seek to the end of the file.
7. Write out several hundred bytes at the end of the file.
8. Close the file.

Figure 8.

If a heuristic scanner observed all of the above behaviors during the first phase, it could report with high confidence in the second phase that it had detected a virus. Unfortunately, the first phase rarely obtains such a complete list of behaviors. Behavior identification and cataloging technology, be it static or dynamic, has its flaws and occasionally fails to detect certain behaviors of the target program. Therefore, the second phase must be able to make educated guesses as to the "virusness" of a program based on the (possibly incomplete) observed set of behaviors.

For instance, what if only behaviors 4 through 8 above were observed during the behavior cataloging phase? For most virus researchers, this subset of behaviors would still raise a big red flag. However, it's one thing for the heuristic scanner to report a possible infection to a researcher who can examine the executable file and verify the scanner's assessment. It's another thing entirely for the scanner to report this to an end-user who has no virus analysis skills.

Given the set of observed behaviors may be incomplete, the behavior analysis component of the heuristic scanner must be designed with extreme care. If this component is too stringent in its requirements, it will have difficulty detecting a significant number of viruses. On the other hand, if the anti-virus programmer designs the analysis component to be too lenient in its behavioral requirements, the anti-virus product may be overly susceptible to false identifications. You can now see why some anti-virus products with heuristics frequently cry wolf: Their designers opted for higher detection rates with fewer behavioral requirements at the expense of occasional false identifications.

Thus far, a number of different approaches have been used to make this behavioral assessment. IBM AntiVirus, for example, uses a neural network to analyze behavioral information in its heuristic boot virus scanner. Another example is Symantec's Bloodhound technology, which uses an expert system to analyze the cataloged behaviors and assess the likelihood of viral infection. There are probably as many different behavior analyzers as there are heuristic scanning products, and it is likely that this technology will evolve significantly over the coming years.

Bloodhound: The Next Generation in Heuristics

All future Norton AntiVirus products will be equipped with Symantec's patent-pending Bloodhound technology. Researchers at the Symantec AntiVirus Research Center (SARC) have developed two types of heuristics for the Norton AntiVirus. The first, Bloodhound, is capable of detecting upwards of 80% of new and unknown executable file viruses. The second, Bloodhound-Macro, detects and repairs over 90% of new and unknown macro viruses.

Bloodhound: Detects up to 80% of New and Unknown Executable Viruses

Symantec's Bloodhound technology represents a complete departure from traditional heuristic scanners in a number of key areas. First and foremost, the Bloodhound system does not use the classical static or dynamic behavior cataloging algorithms. Instead, it uses a hybrid technology that enjoys the benefits of both schemes.

As we have seen, traditional heuristic systems catalog behavior using one of two different techniques. Static cataloging algorithms have the advantage of being very quick; however, these algorithms are poor at identifying obfuscated program logic and often completely fail to recognize even slightly non-standard program logic. As virus writers become more savvy, static heuristic scanners will be less and less effective at detecting new computer viruses.

On the other hand, dynamic behavior cataloging algorithms excel at identifying obfuscated program logic, but are often slower than their static counterparts. Dynamic algorithms may also fail to catalog behaviors due to logic tricks implemented by the virus. Virus writers may insert logic tricks into their viruses to intentionally confound dynamic scanners (see Figure 7); however, in many cases ordinary viruses that use conventional logic will also evade the dynamic scanner.

For instance, many viruses only infect programs that are within a given size range. If this type of virus attempts to infect new programs and locates an executable file that is too big or too small, it will skip over it and continue to search for other appropriate files. Consequently, the virus fails to exhibit the virus-like behavior required for detection. Other viruses, for example, refuse to infect programs that have a seconds value of 12 (each file has a time stamp indicating when they were last modified in HR:MIN:SEC format), since the virus uses a time stamp to indicate that it has already infected the file. Some viruses won't infect any files with names ending in AV, since many virus scanners such as NAV.EXE and TBAV.EXE have filenames that end in these letters and infecting a program with AV in the filename would alert the user to the infection. Unfortunately, each virus has its own unique set of conditions, and each condition is a potential roadblock for the dynamic scanner.

Here's a slightly unusual but accurate analogy. You might compare such a picky virus to a laboratory frog. A scientist might want to see how far the frog could jump or measure the strength of its leg muscles. The scientist could measure these attributes by placing the frog in an artificial habitat (an aquarium) and watching to see how far it jumped. However, the frog might never jump or only take small leaps. The problems experienced by the researcher are the same encountered by the dynamic heuristic anti-virus program. It's difficult to tell exactly what the virus is capable of doing, given that the virus has control over its actions and may suppress some behaviors while running within its virtual habitat (a simulated PC).

A neurologist studying a frog would not bother to place him in a tank and wait. Instead, he would attach electrodes to each of the different regions of the frog's brain and stimulate each region to force the frog to exhibit each of its behaviors whether it wanted to or not. This is like getting your reflexes tested during a physical exam; when the doctor hits your knee with her hammer, you have no control over the reflex action of your leg. While a brain stimulation technique is arguably cruel and unusual when applied to frogs or other living creatures, its perfectly legal and ethical when used with computer viruses. And its extremely effective!

Bloodhound employs artificial intelligence technology to isolate and locate the various logical regions of each program it is told to scan. It then analyzes the program logic in each of these components for virus-like behavior, stimulating them just as the neurologist might stimulate the regions of the frog's brain. It uses both static and dynamic techniques to do this analysis and stimulation, and is subsequently capable of detecting a wider variety of behaviors than either of the traditional algorithms. Because Bloodhound identifies and examines every logical component of the virus, it is impervious to most logic trick attacks and general virus pickiness. And because it uses dynamic analysis techniques, it can identify even the most convoluted and obfuscated program logic.

Once Bloodhound has stimulated the program in question and caused it to exhibit its reflex behaviors, it uses an expert system to analyze the observed behaviors and determine whether the program is a virus. An expert system is a program which makes intelligent inferences about a given problem when complete information may not be available. It is called an expert system because its rules and decision making logic is typically designed and programmed by experts in the field. Expert systems have been successfully employed in the medical field to help doctors diagnose medical conditions. They have also been used by credit card companies to detect suspicious buying patterns and credit card fraud.

Symantec researchers have spent many years of effort to produce an exceptionally robust virus detection expert system for Bloodhound. In addition, Symantec engineers have designed Bloodhound's expert system so it can be updated on a monthly basis through standard virus definition updates. SARC engineers can continue to hone and improve the Bloodhound system, delivering these improvements to customers automatically via LiveUpdate without costly inlines or product reinstallation.

The Bloodhound system also draws extensively on the technology used by Striker, Symantec's next-generation polymorphic virus detection engine. Components of the Striker engine are used to de-scramble both encrypted and polymorphic computer viruses. These viruses encrypt their computer logic in order to hide themselves from other heuristic scanners. However, Bloodhound uses the patent pending Striker technology to crack the viral encryption and thwart the virus' efforts to hide itself.

While Symantec development teams have worked extensively to insure that Bloodhound will achieve stellar detection of unknown viruses, they have also spent countless hours of research and development effort to virtually eliminate false positives. Team members of the Symantec AntiVirus Research Center have scoured the Internet, commercial software libraries, and software CDs to find viruses and test Bloodhound under every possible circumstance. The team has collected gigabytes of executable files from the United States, Europe, and as far away as Japan for testing, making Bloodhound the most robust heuristic technology on the planet.

Finally, while Bloodhound is capable of detecting upwards of 80% of new and unknown computer viruses, it does so with only minimal overhead. Bloodhound was designed to perform an in-depth analysis of programs only if they meet stringent prerequisites. In most cases, Bloodhound can determine in microseconds that it is looking at a file that could not be infected by a virus. When it makes such a determination, it immediately ceases analysis of the file and goes on, making it one of the most efficient heuristic systems available.

Bloodhound-Macro: Detects and Repairs over 90% of New and Unknown Macro Viruses

In under two years, macro viruses have become the most infectious and widespread computer viruses in history. These electronic villains are incredibly infectious, spreading through email, the Internet, on floppy diskettes, data compact disks, and over electronic bulletin board systems (BBSs). Unlike previous computer viruses, macro viruses do not infect application files or floppy disks; instead, they infect the documents and spreadsheet files that businesses and home users rely upon on a daily basis.

What is a macro virus, anyway? It is a virus that infects word processing documents or spreadsheets. Modern word processing and spreadsheet programs such as Word for Windows or Excel allow you to write simple programs, called macros, and then attach these macros to your document or spreadsheet. These macros can then be used to automate repetitive tasks, make calculations in the spreadsheet, and so on. A macro virus is merely a malicious macro program that is designed to copy itself from document to document or spreadsheet to spreadsheet rather than serving a useful purpose. Word for Windows documents are the most common carriers for macro viruses, with well over 1,500 distinct macro virus strains.

Macro viruses are especially problematic for several key reasons. First, with increased use of the Internet, email and other workgroup software, users are exchanging more information than ever before. In the past, information exchange was not dangerous because documents and spreadsheets could not contain macros, and therefore could not contain macro viruses. Today, however, the documents most commonly exchanged through email can and do contain macro viruses.

Another factor that accounts for macro virus success is the availability of the popular office applications. Applications such as Word for Windows or Excel were designed to share the same documents and spreadsheet files regardless of where the documents originated. Therefore, a user could send a macro virus-infected document created on a PC to a coworker that has a Macintosh. If the Macintosh user edits or views the infected document, the virus takes up residence on that system. On the other hand, traditional PC viruses will only work on the type of machine for which they were originally designed.

The macro virus' success has also been fueled by their simplicity: Macro viruses are extremely easy to construct. In the past, only programmers with low-level assembly language programming skills could create computer viruses. With the introduction of user-friendly application macros, almost any savvy user can spend an afternoon writing a macro virus with only a copy of Microsoft Office and a few computer manuals. Furthermore, an unscrupulous user can easily study the programming logic of an existing macro virus and then modify the macro virus' logic to work in a different fashion. And it isn't very difficult to add a little something extra and construct macro viruses that encrypt data, format hard drives, or even silently change the numbers in spreadsheets.

Each of the above factors has contributed, in part, to the tremendous prevalence of macro viruses in the workplace. Yet one factor has arguably caused more problems than all the rest combined. In the Word for Windows word processor, when a macro virus spreads from document to document, the virus runs a small risk of having its macros (that is, its programming logic) corrupted. This corruption problem is most likely due to a software bug in Word for Windows.

Consequently, each time a macro virus tries to spread itself, Word for Windows may inadvertently corrupt or mutate it. In some cases, this corruption is lethal for the new macro virus because it prevents the macro virus from spreading to additional documents. Unfortunately, the corruption can also create a completely new macro virus that is perfectly capable of spreading. In many ways, this corruption is analogous to the random mutation that occurs in nature. Many mutations cause the offspring to die immediately; however, other mutations allow some offspring to survive and flourish in the environment.

As we have seen, most anti-virus vendors maintain a database of known viral macro signatures and regularly ship this information to customers in the form of regular virus updates. When scanning a document, if the anti-virus program locates macros that match signatures in its database, it reports that the document or spreadsheet is infected with a macro virus. This process is similar to that used by an FBI agent who identifies criminals based on their fingerprints. Once a fingerprint lifted from the scene of a crime is located in the FBI database, the perpetrator can be identified and arrested.

Unfortunately, these tactics don't work nearly as well for macro viruses. If a macro virus' programming logic changes (for instance, because it has been corrupted), the new, mutated virus effectively has a different fingerprint than that stored in the anti-virus database, and will evade detection by the anti-virus program. Traditional computer viruses are not affected by this sort of corruption, which means that users are likely to encounter the same traditional viruses over and over again. It also means that anti-virus software should be fairly effective at detecting and removing them.

Inadvertent macro corruption has created hundreds of new macro viruses without the aid of a single virus writer In the past, only virus writers created new viruses. Now, new viruses are being created every day on countless machines around the world. These new viruses aren't being created in virus research labs or in back alleys; they're being created in the workplace and on end-user machines where they cause the most damage.

As if this were not enough, macro viruses even mate with each other! There are many documented cases of two or more macro viruses combining in the same document to form wholly new macro virus strains. The new macro virus strains do not have the same fingerprint and often cannot be detected by traditional anti-virus software. These issues have fundamentally changed the nature of the virus problem, and made macro heuristics a necessity.

Given the large numbers of new macro viruses encountered in the workplace, traditional anti-virus software is unable to detect and remove these new macro virus strains. Consequently, users are encountering more viruses than ever before that can't be detected and repaired, and they're spending more time waiting for fixes from their anti-virus vendor. To compound the problem, there is an unacceptable lag between the emergence of new viruses and traditional anti-virus software's ability to detect and remove them.

Symantec's Bloodhound-Macro technology addresses these fundamental detection problems. It detects and repairs more than 90% of all new and unknown macro viruses automatically, reducing the need for costly and repetitive interaction with the anti-virus vendor. Let's examine how Bloodhound-Macro is able to provide this functionality.

Bloodhound-Macro employs a patent-pending, hybrid heuristic scheme to detect and repair macro viruses. Every time the Norton AntiVirus scans a document file, Bloodhound-Macro sets up a complete virtual Word for Windows environment and loads the document into this simulated environment. The macro (programs) contained in the document are then coaxed into running as if they were running in the actual word processor.

Bloodhound-Macro monitors the macros as they run in the virtual Word for Windows environment and watches for the macros to copy themselves from the host document to another virtual document. This behavior alone is suspicious, since most macros have no need to copy themselves from one document to another. However, this behavior by itself is not sufficient to report a virus infection.

Why? Well a number of companies have developed macro packages that install themselves into your Word for Windows environment by having the macros copy themselves into your documents. However, once these macros have installed themselves in the Word for Windows environment, they do not copy themselves to other documents. However, when a macro virus copies itself into a document, it makes sure to copy the programming logic and data required to further spread itself to other documents. This critical detail distinguishes these macro packages from macro viruses.

Bloodhound-Macro recognizes this detail. In addition to verifying that the macros in the host document copy themselves to other virtual documents, Bloodhound-Macro also simulates the copied macros in the virtual environment and verifies that these second-generation macros contain the programming logic to further propagate themselves. Once Bloodhound-Macro has seen this capacity for repeated viral behavior, it can definitively tell users that they have a virus.

The Norton AntiVirus Bloodhound-Macro technology is the only product on the market to use hybrid heuristics to detect macro viruses. Other current anti-virus products that claim to use heuristics actually use simple static string scanning. String scanning can easily detect that a macro has the ability to copy itself from document to document, but it has great difficulty distinguishing between single-installation macros that copy themselves once and actual macro viruses. This type of heuristic technology inevitably falsely identifies uninfected macros as viral and can cause more problems than it solves.

As Bloodhound-Macro emulates a document’s macros within its virtual environment, it remembers which macros are involved in the replication process. This information is then used to repair the infected document. Norton AntiVirus removes only those macros that are absolutely copied during the repeated replication of the virus; the user’s macros are left unharmed in the document.

Once again, static heuristic scanners are at a disadvantage when attempting repair. They may be able to identify which macros have programming logic to copy macros from document to document, but have no way of definitively identifying other viral macros that lack this explicit copying logic. Most macro viruses are composed of two or more macros; in many cases, only a subset of these macros contain the programming logic to copy the macros from document to document. The other viral macros may contain malicious logic, but are not necessarily responsible for the replication process. Consequently, if a static heuristic scanner attempts to repair a macro virus infection, it runs the risk of removing only the identifiable subset of macros that contain virus-like program logic (macro copying logic), leaving potentially malicious macros within the document after it has been cleaned by the anti-virus program. In addition, a static heuristic scanner may inadvertently identify third-party macros as viral based on their macro copying logic, and mistakenly delete these macros as well.

In general, static heuristic implementations are more prone to false identifications. During the repair process, these heuristic systems may incorrectly delete non-viral macros or fail to delete all viral macros, leaving the integrity of the document and your network at risk. Symantec’s patent-pending Bloodhound-Macro system is, therefore, the only solution when it comes to detecting new and unknown macro viruses.

Conclusions

The Norton AntiVirus product line now incorporates Bloodhound and Bloodhound-Macro, the most advanced and effective heuristic tools in the anti-virus industry. Symantec researchers have spent many years of effort to insure that these technologies are unobtrusive, have low overhead, and are extremely effective at detecting and repairing new and unknown computer viruses.

Bloodhound is capable of detecting upwards of 80% of new and unknown file viruses, and Bloodhound-Macro can detect and repair more than 90% of new and unknown macro viruses automatically. This patent-pending Bloodhound technology helps to make the Norton AntiVirus the most effective corporate anti-virus solution on the market.

The Bloodhound system: 21st century technology brought to you by Symantec, today.

Further Reading

This document is one of a series of papers on Symantec's software strategy and its product offerings. Additional papers include:

- Understanding and Controlling Viruses in 32-Bit Operating Environments
- Why Norton Utilities is a Natural Complement to the Windows 95 Environment
- Managing Desktop Interfaces Across the Enterprise
- A Strategy for the Migration to Windows 95
- File Management and Windows 95
- Understanding Virus Behavior in the Windows NT Environment
- Intregrating Remote Communications into Enterprise Computing
- Using Outsourcing to Reduce IT Labor Costs
- Understanding and Managing Polymorphic Virsuses
- Using Outsourcing to Reduce IT Labor Costs
- Understanding Symantec's Anti-virus Stratedy for Internet Gateways

For copies of these papers or information about Symantec enterprise network products, call 1-800-453-1135 and ask for C342. Outside the United States contact the sales office nearest you (listed on the back cover).

About Symantec

Symantec Corporation is a leading software company with award-winning application and system software for Windows, DOS, Macintosh, and OS/2 computer systems. Founded in 1982, Symantec has grown rapidly through the success of its products and a series of 16 acquisitions resulting in a broad line of business and productivity solutions.

Symantec's acquisitions have strongly influenced the company's innovative organization. The company is organized into several product groups that are devoted to product marketing, engineering, technical support, quality assurance, and documentation. Finance, sales, and marketing are centralized at corporate headquarters in Cupertino, California.

SYMANTEC.™

WORLD HEADQUARTERS
10201 Torre Avenue
Cupertino, CA 95014 USA
1 (800) 441-7234
1 (541) 334-6054
World Wide Web site:
http://www.symantec.com

Australia (Sydney): +61 2 9850 1000
Australia (Melbourne): +61 3 9823 6204
Brazil: +55 11 530 8869
Canada: 1(416) 441-3676
France: +33 1 41 38 5700
Germany (Munich): +49 89 641 922 12
Germany (Ratingen): +49 2102 7453 0

Hong Kong: +852 2528 6206
Italy: +39 2 69 5521
Ireland: +353 1 820 5060
Japan: +81 3 3476 1156
Korea: +82 2 3452 1600
Mexico: +52 5 661 7978
New Zealand: +64 9 309 5620

Netherlands: +31 71 535 3111
Russia: +7095 238 3822
Singapore: +65 324 7990
Sweden: +46 8 457 3400
Switzerland: +41 71 626 20 40
Taiwan: +886 2 729 9506
UK: +44 1628 592 222