VESSELIN V. BONTCHEV
UNIVERSITY OF HAMBURG
FACHBEREICH INFORMATIK
GERMANY

**ANATOMY OF THE LATEST GENERATION COMPUTER VIRUSES** *16*

# Future Trends in Virus Writing

Vesselin Bontchev, research associate
Virus Test Center, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
bontchev@fbihh.informatik.uni-hamburg.de

*The paper tries to summarize the possible prospective ideas that are likely to be used by the virus writers in the future and to suggest what kind of measures could be taken against them.*

## 1 Introduction.

The last few years we have witnessed a lot of activities in the virus writing field. Many new ideas have been invented and experimented with. Some of them have proven to be very successful, others not so much, others have as of yet uncertain effect. Some of them will be undoubtedly developed further and widely exploited by the virus writers in the future.

In this paper we try to outline the ideas that seem to be prospective from the virus writing point of view and also those ideas which have not been tried yet but seem tempting. We have no way to determine which ones will be actually used in the near future, therefore we must assume a worst-case scenario and get prepared against all of them—thus we shall try to list all of them here. In the same time, we must be as terse as possible about the particular technical details that will permit implementing these ideas in order to limit the damage this material may have if it is used by the virus writers.

## 2 Technical Dangers.

### 2.1 Glut.

The main trend in virus writing that we have observed so far is the increasing production of new viruses. There are more and more new viruses. A glut of new viruses. Statistically speaking, the curve of the virus creation is no longer exponential but has begun to flatten. The number of known viruses does not double every 9–10 months any more. This is not very reassuring, however, as there are still about 1,000 new viruses created every year, that is—about 2–3 new viruses per day.

This has already created problems to the authors of scanning software. First of all, it is already impossible for a newcomer to start from scratch. As of the date of this writing (April 1994) there are more than 4,300 different known viruses and their number increases literally every day.

Suppose somebody decides to enter the anti-virus business now. Regardless of what kind of anti-virus software he is going to push, it must include a known-virus scanner. There are

several reasons which make this necessary. *First*, the public is very accustomed to this kind of anti-virus program. *Second*, you must ensure that a system is virus-free before installing any other kind of anti-virus software on it. *Third*, all magazine reviewers are going to "test" only the scanner part of your product anyway—because it is the easiest one to test.

In order a scanner to "score" well during the tests, it must be able to detect most of the known viruses. Suppose it takes on average one hour to disassemble a virus, to understand it, to pick a good scan string for it, and to test that this scan string really detects all replicants of the virus and does not cause any false positives. Then, in order to be able to handle 4,300 different viruses, one must spend 4,300 hours on them. Assuming a 40-hour week, this is about 750 days, i.e. more than two years! And during those two years, about 2,000 new viruses are likely to appear...

Indeed, most anti-virus researchers work much harder—an 80-hour week is a rule, not an exception. But nevertheless, the above calculation is an underestimation—many viruses take much more time to analyze and "defeat." Some may take days, some—even months. Some are even more difficult. Two years after the infamous Mutating Engine (`MtE`) has appeared, the tests show that only a handful of scanners are able to detect it ([Bontchev93]). And there are still many scanners which are unable to detect `V2P6`—one of the first extremely polymorphic viruses.

We also should note that the ability to detect known viruses is not the only property that an author of a scanner must worry about. There are such things as designing the proper user interface, doing the actual coding of the program (just a collection of scan strings is still not a scanner), writing the documentation, marketing the product, and so on.

All this makes the game almost impossible for the small companies. The big companies, on the other side, are too clumsy and slow-reacting to be good players in the fast evolving anti-virus business. Thus, the rules of the game make it almost insupportable for a newcomer.

However, it is also not so easy to keep up with the game. Even those producers of anti-virus software who have been along since the beginning are facing serious problems. With the increase of the amount of known viruses, it becomes more difficult to maintain a scanner. If the scanner does not use one of the modern fast string search algorithms, it is going to become slower and slower when you add more scan strings to it. Switching to a faster algorithm might not be so easy, because it often requires a total re-design of the scanner and the way data is represented in it.

But even if the scanner's speed succeeds to remain constant when new scan strings are added to it, it is not so with the memory requirements. All those new scan strings and new virus names have to be stored somewhere. The fast scan algorithms are usually memory-hungry. All these problems are particularly acute for the memory-resident scanners, where the memory usage consideration is at premium.

One possible solution is to use optimized scan strings (i.e., a single scan string detects as many viruses as possible), short virus names, fuzzy detection (the viruses are not identified exactly), heuristic methods, etc.

But all of these are only temporary solutions. The process of virus creation is not going to stop or even to slow down significantly in the foreseeable future. After a few years of activity, the virus writers usually grow up and switch to other activities—but many new "wannabe" virus writers (usually adolescent kids) pop up in their place.

Because of all this we are likely to see some producers of anti-virus software pulling out of the game. The first major cases—XTree, Certus, Fifth Generation Systems, Central Point Software—have already happened.

What can be done against this trend? The governments of the countries where most viruses are created should be pressed to pass the appropriate legislative measures. Unfortunately, in many cases this is very difficult—Russia and some countries from the former Eastern Block have a notorious history of disregard of the concept of intellectual property. Other countries—the USA—consider virus writing as form of free speech and thus protected by the constitution. In general, the legislature is notoriously slow and backwards in dealing with computer-related problems.

From the technical point of view, the users must be educated to rely more on other anti-virus measures, not only on scanning. More research should be done into the generic virus detection methods like heuristic analysis—they are still too unreliable. The reviewers of anti-virus packages should be educated to pay more attention to the ability of the scanners to detect the viruses which are more likely to infect the users' computers and more attention to the non-scanning parts of the products.

## 2.2 Virus Authoring Packages.

We have already seen some of these—as simplistic as `VCS`, as sophisticated as the `MtE`, or as user-friendly as `VCL`. Those are programs which help the user to create viruses—even if s/he does not know well assembly language, or some of the modern virus writing techniques. Such packages lower significantly the barrier of initial knowledge that a "wannabe" virus writer must have. They make the most sophisticated virus writing techniques available to the masses.

The currently existing virus authoring packages are far from perfect. `VCS` is able to create only a single virus and allows the user only to change the text message in it. `VCL` is incredibly buggy. `MtE` and `TPE` are too sophisticated to use for the average kid. Even the `PS-MPCs` are not good enough. This does not make them less dangerous, however.

*First* of all, they make the ability to write viruses available to more people. This contributes to the glut problem mentioned above.

*Second*, many of them generate viruses in source—thus allowing the virus writers to learn some virus writing techniques, which they could use later in their own viruses.

*Third*, they are an initial, inciting tool, and as such are often used by newly formed virus writing groups to boost their efforts and to help them organizing themselves. The `ARCV` virus writing group is a typical example for this—they used `VCL` and `PS-MPC` for some time before they learned how to create their own viruses.

*Fourth*, while at least in some countries outlawing virus writing seems possible, forbidding the virus authoring packages is much more difficult, because most of them are not malicious and even do not generate viruses directly. Instead they generate viruses in source—the user is still required to take the deliberate action to assemble the source, in order to create a "live" virus. By the way, this is yet another danger—a virus in source is much easier to modify and to spawn new variants. It is often said that while a virus by itself is like a flying bullet, the source of this virus is like a loaded gun...

*Fifth*, it is much more difficult for a producer of a scanner to handle a virus authoring package. While a single virus is relatively easy to disassemble and understand, such a package (usually written in a high-level language) presents much more difficulties... And the anti-virus researcher must update his scanner in such a way, that all possible viruses that a particular package is able to generate are detected—even if the virus writers never bother to actually generate them. This makes the glut problem even more acute...

Without doubt, we shall see a significant improvement in the virus authoring packages of the future. They will be able to create viruses more easily, to create more different viruses, and more sophisticated viruses. Just like the `MtE` and the `TPE` provide ready-to-link libraries of polymorphic routines, we shall see kits with libraries of stealth routines, tunnelling routines, infection strategy routines, damage routines, and so on. (In fact, a package providing a tunnelling engine—`KRTT`—is already available.) All this—coupled with programs with nice user interfaces, which will make the creation of thousands of viruses with the selected capabilities easy. Thousands of different viruses!

Maybe in the near future we shall witness even the appearance of *virus meta-authoring packages*—programs, able to create virus authoring packages. This is a rather difficult thing to do and the final outcome for the virus writers is quite doubtful, but nevertheless we must consider such a possibility.

Another kind of attack, involving virus authoring packages, is to use them secretly. Currently, the authors of such packages are so proud with their creations, that they quickly make them available to their peers and virus writers. Regardless of the fact that these packages are often horribly buggy and the viruses generated by them barely work. This way, the packages finally end in the hands of the anti-virus researchers, who disassemble them and eventually make their anti-virus programs able to handle all possible viruses that the particular package is able to generate.

However, let's suppose for a moment that somebody creates such a virus authoring package—but a well-made one, one which is able to generate many different and sophisticated viruses. The viruses themselves do not need to be polymorphic—it is enough if there is a polymorphic routine in the package that is able to produce a different decryptor for each newly created virus. Suppose also that the author of such package does not distribute it, but instead uses it secretly to create thousands of similar, but nevertheless different, machine-generated viruses. The next step is obvious—one cannot hope to spread a few thousands of viruses without being caught, so the way to cause the most damage is to send them to the anti-virus researchers [Skulason]. This will keep them busy for quite a while... And since the generator that has been used to create those viruses will not be available, they will need to spend a lot more efforts to guess the particular polymorphic routine that has been used to generate the decryptors...

What can be done about this? To prevent it—practically nothing. We must consider the appropriate action that could diminish the damage. One possible reaction is to simply refuse to play the game and not to detect these viruses, or at least those of them that have never been found "in the wild." Unfortunately, this is a rather unrealistic scenario. The constant market competition between the producers of anti-virus packages, the "my package can detect more viruses than yours" game, the numbers game, the media hype—all this will cause some producer to begin to detect some of the viruses, then others will follow, and the race will be unstoppable. This is at the expense of the users, of course. They will

have to pay for more updates, and to use bigger, slower, more memory-hungry scanners...

Another technical solution is to attack the polymorphism at its roots. Currently most scanners can "see" only the decryption routine of the polymorphic viruses. This routine can be made rather short, generic, or modifiable—which makes the task of the scanners harder. If the scanner could "peel" the encryption and look at the decrypted virus, it will be much easier to detect many similar viruses in a single step.

This can be achieved by several means. One of them is to use fast, on-the-fly cryptanalysis of the virus body—a technique described by the Russian anti-virus researcher Eugene Kaspersky [Kaspersky]. Another one is to use improved heuristic analysis techniques to detect decryption routines. Probably the most effective one is to use some kind of emulator, which interprets the decryption routine, until the virus body is decrypted, and then to apply some kind of usual scanning technique.

Only the future can tell which technique will be the most efficient one. But the threat of the virus authoring packages is real, the outcome of it will be in the very near future, therefore it is urgent to do some research in these generic virus detection techniques.

## 2.3 Virus Mutators.

This threat is connected with the "glut", the "virus authoring packages", and the "polymorphism" problems, mentioned elsewhere in this paper.

The idea is to create a program that takes an existing program (e.g., a known virus) and modifies it in such a way, as to create an equivalent program. For instance, adjacent instructions that are not position-dependent could be swapped, one instruction could be replaced by a sequence of instructions that do the same thing, or with a call to a subroutine that does the same thing, and so on. Essentially, the `MtE` routine does exactly that to the decryptors it generates. The trick is to apply this method to a whole program, e.g. a virus.

It is extremely difficult to create such a "mutating" program, but it is a far from impossible task. Therefore, in a worst-case scenario we should assume that sooner or later somebody will do it.

Once such a program is created, a virus writer could take it and "apply" it to a whole collection of known viruses. The result will be that hundreds of new viruses will be generated. They will be very similar to the original ones, many of them will probably be detected as variants by the existing scanners, but nevertheless there will be hundreds, if not thousands of new viruses. And, applying the program many times, will result in many new thousands of new viruses being generated. The viruses themselves will not be polymorphic—they will be just new, and it will be very easy to generate them.

Such a program (we shall call it a "virus mutator") will quickly make obsolete any known-virus scanners, virus classification schemes, or even virus authoring packages. It will be the ultimate virus authoring package. With it, everybody will be able to generate a practically unlimited number of new viruses, even without knowing what a virus is. And there will be no way a scanner could cope with the output of such a program, because the problem of equivalent program transformation is known to be NP-complete.

What could be done about it? Firstly, no such program exists yet, and it is extremely difficult to write one, so this is not an imminent danger. Besides, most virus writers do not demonstrate any significant knowledge in theory of algorithms. Secondly, one could use a

known-virus scanner to detect at least those of the viruses generated by a virus mutator, that have been found in the wild. Third, some algorithms for automatic scan string capturing could be implemented, like the ones used in the anti-virus product `Victor Charlie`. They work quite well for non-polymorphic viruses. Finally, the users should be educated not to rely on known-virus scanning techniques alone.

## 2.4 Targeted Attacks.

Another trend that we can see nowadays is the creation of viruses, designed to target particular anti-virus products. The more popular the anti-virus product is, the more likely that it will be attacked.

The attacks can range from benign ones—just avoiding infecting a program known to perform some kind of self-checking—to sophisticated ones, designed to fool the user that the protection is still in place and working, while actually disabling it.

Just a few examples. There are already many viruses which avoid infecting programs with names beginning with "SC" or something like that. The `Tequila` virus removes the checksums that the program `ViruScan` (from McAfee) adds to the files (when used with the `/AV` option). The `Peach` virus successfully attacks the integrity checking in Central Point Anti-Virus by simply deleting the databases of checksums that the product creates. The `Groove` virus does the same, but targets several anti-virus products known to use integrity checking, not only a single one. The `Tremor` virus successfully disables the anti-virus protection supplied with MS-DOS 6.0. The full list is much longer.

We must notice that the design of many of the existing anti-virus products is rather sloppy and does not offer any serious defense even against simple attacks. Other, much more sophisticated forms of attacks are possible, especially against integrity checking software (see [Bontchev92]), and the producers of such products must finally take the appropriate steps to improve their products. This is difficult, but by no means impossible, and the reference mentioned above provides advice in this aspect.

## 2.5 Polymorphism.

We have already briefly mentioned this when discussing "glut" and the "virus authoring packages" problems above. There are "easy" to handle viruses and "difficult" to handle viruses—from the point of view of the developer of a scanner. The polymorphic viruses are in the class of the most difficult ones.

Those are viruses which change their appearance each time they infect a new object. Usually, this is achieved by encrypting the virus body with a different key each time, and then prepending a small decryption routine which gets executed at run time. The fact that the routine is small, makes scanning difficult enough as they they could cause false positives. Even worse, the sophisticated polymorphic viruses use different decryption routines each time, usually by slightly varying the algorithm, or the registers used, or by inserting "do-nothing" instructions between the ones that actually perform the decryption [Solomon92]. Some of the most sophisticated and polymorphic viruses are the `V2Px` family and the `MtE`- and `TPE`-based viruses. Most of the currently existing scanners are still unable to detect them reliably.

All these problems are well known to the virus writers. Their reasoning is obvious—most people use scanners for virus protection, the polymorphic viruses are the most difficult ones for the scanners, so let's create more polymorphic viruses.

Fortunately, the design and the implementation of a well-made highly polymorphic virus is not a trivial task. But the existence of virus authoring tools like `MtE` and `TPE` makes it much easier and more available even to the inexperienced virus writers. Therefore, we are going to see much more polymorphic viruses in the future.

What can be done about it? The users must be educated not to rely on scanners alone—even the most polymorphic virus is easy to detect with an integrity checker. Of course, there are ways to attack the integrity checkers too, so the users must build a multi-level anti-virus defense. Unfortunately, many of them still don't know exactly what a virus is, let alone how to build a sound anti-virus defense. Hence the need for more user education about viruses and computer security in general.

Some technical ways to defeat polymorphism have been already mentioned in the section about the virus authoring packages.

## 2.6 False Positives.

With the existence of so many viruses, it might seem easier to detect the legitimate programs than the malicious ones. From time to time scanners detect one of the scan strings they are using for some viruses in a perfectly legitimate program and announce it to be a virus. This is called a *false positive alert* or simply a *false positive*.

False positives are usually very annoying both to the user and the producer of anti-virus software. The user gets scared that his/her computer is infected. Sometimes the effects of such a scare can be disastrous—users in panic are known to low-level format their disks and lose data that is worth months of work. Other effects of the false positive alerts are discussed in [Solomon93].

On the other hand, the producers of the scanner that has caused the false positive must react quickly and this can often be very costly. They need to inform urgently the users of their product about the false positive, to correct the scan string that causes it, to distribute updates (often at their own costs), and so on. Failing to do so may have unpleasant consequences for the anti-virus producer as the company that markets the program which is incorrectly accused of containing a virus could and sometimes does seek reparations, as in the case of Imageline vs. McAfee Associates. (This particular case has been settled out of the court.)

As the number of viruses continues to increase, we shall see more cases of false positives and maybe some court cases related to them. Regardless of the outcome of these court cases, the users are who are likely to lose. If the court rules that the anti-virus producer whose product is causing the false positive owes reparations to the company whose product is accused of containing a virus, then we could see cases in which companies deliberately create products that are likely to be flagged incorrectly as viruses—in order to get reparations from the anti-virus producer. As a defense, some anti-virus producers may decide not to detect some viruses, instead of running the risk to be sued for causing a false positive. As a result, either the users will be less protected, or the costs will be laden on them.

If the courts decide the opposite—that the anti-virus producer does not have any responsibility to provide a product that does not cause false positives, then this could reflect in the worsening of the quality of these products. Hence, the user is likely to lose again.

The virus writers don't make the game easier. One of the `Gotcha` viruses deliberately carries some scan strings of other viruses, used in a popular scanning product. In this case, the purpose is to cause misidentification, but a next step could be to insert these scan strings in legitimate programs (without actually infecting them).

What are the possible defenses? One possibility is to carefully select the scan strings used in a scanner, and to do a lot of testing and quality control before releasing the product. However, this is often too expensive and time-consuming. The fact that a scanner must be updated often, in order to keep up with the new viruses does not make the things easier.

Another, much better defense, is to use exact virus identification. This means that the scanner carries some kind of virus map—a list of the constant areas of code in the virus and their checksums. When the virus is detected, it is matched against this map and the program is declared as infected only if an exact match is found. This rules out the possibility for a false positive. Unfortunately, it also makes the development of the scanner much more expensive and time consuming. It also means that even a single-bit change in the virus is likely to leave it undetected—because no match will be found. Therefore, this technique has to be combined in some reasonable way with the conventional scan string technique.

## 2.7 LAN-aware Viruses.

Many of the currently existing viruses simply crash when executed on a computer with a LAN shell loaded. There are also, however, many that behave well enough and are able to successfully run and infect. If the security settings of the LAN allow them to modify the files, of course. At last, there are about a dozen (actually—variants of two main families), which are LAN-aware. Currently, this awareness consists in monitoring some of the undocumented interrupts used in Novell NetWare and using them to steal passwords that are sent in unencrypted packets. This is, however, still too primitive—we must expect significant sophistication in this area in the future.

The LAN hackers have discovered some very serious security holes in Novell NetWare. They have created two programs that demonstrate these holes—`KNOCK` and `HACK` ([Gold]).

`KNOCK` works successfully on NetWare versions 3.10 and below. It is able to log into any given account, without knowing the password for this account, exploiting a bug in the NetWare's encryption algorithm. Of course, the supervisor account represents the greatest danger, because it has the highest privileges.

It is relatively trivial to construct a virus, which will be able to detect whether the NetWare shell is loaded (and which version of it), and which tries (in the background) to log in as a supervisor, using the `KNOCK` algorithm. Indeed, Novell has distributed security patches that close this particular hole. But, having in mind how widely used the software is, it is very likely that the number of machines without the security patch installed is relatively big. Therefore, the population of computers on which such a virus will be able to breach the security will be big enough for the virus to survive.

`HACK` is another program which, reportedly, is able to log in from a workstation as any user (the supervisor is the most interesting one, of course), which is already logged in—from

another workstation. The method used by `HACK` relies on spoofing `IPX` packets. It is based on a very serious design bug in the NetWare, which can be fixed only with a complete re–design of the system or with full public–key based encryption of the packets that pass through the line.

Novell has solved the problem in version 4.x of NetWare. Meanwhile, they have distributed security patches for the users of the older versions. Unfortunately, those patches seem to cause additional problems when installed—they occasionally log out perfectly legitimate users. Therefore, until version 4.x becomes widely used, this security hole is likely to be present on many networked computers. It is only a question of time until the viruses begin to use it. The main question is whether this time will be enough to replace the old versions of NetWare.

Another LAN–related virus problem has nothing to do with bugs and security holes and is connected to the ability of the viruses to spread transitively. Most LAN administrators are reasonable enough to set the protection settings in the directories that contain the executable files used by everybody in such a way, that the users are able only to list, read, and execute the files. However, viruses don't need to have direct access to the files in order to infect them. All they need is to have access to the files of somebody who has access to the protected files, or to somebody who has access to the files of somebody who has access to the protected files, or... In short, there must be a transitive path of information flow between somebody who has access to the protected files (at least the supervisor does) and some other, already infected account.

Additionally, viruses could use other LAN–oriented features. For instance, if the virus has `Create` rights in a directory, it is possible to use a companion–type attack to infect the `EXE` files in it, even if the files themselves are write–protected [Cohen]. Next, if a user does not have `Write` rights to the files but does have the right to `Modify` the rights, a virus could use this to temporary grant itself `Write` rights, in order to infect the files. It is important to note that a virus always runs with the effective rights of the user whose workstation is infected. For this reason, the users with supervisor privileges are particularly vulnerable and dangerous. Therefore, they should use extreme care when logging into their accounts with such privileges.

For some unknown reason, the virus writers are usually not very good at hacking. Their knowledge about LANs and mainframes is not significant. Therefore, it will probably pass a lot of time, until they begin to use the security holes mentioned above. Nevertheless, we must be prepared for it. LAN users must be aware of the security patches issued by their LAN software suppliers and must install them. We need more LAN–oriented anti–virus and general security products. We need products that can evaluate the security settings of a LAN and to suggest ways to improve the situation, display the possible routes of virus infection, and inform the LAN administrator about any security holes found in the system.

## 2.8 Viruses for Other Operating Systems and Computers.

Nowadays, the platform that is most attacked by viruses is the IBM PC compatible computer running MS–DOS. Viruses for the Macintosh platform are very widespread too, but there are not very many of them and the existing anti–virus software is able to handle all of them properly. We expect this situation to change in the future.

As alternative operating systems to MS-DOS become more widely used, viruses that attack them will appear. We already have several Windows-specific and OS/2-specific viruses. They are relatively simple, but much more sophisticated ones are possible.

The DR-DOS (and Novell DOS 7) operating system begins to gain popularity. The currently existing protections in it (passwords) are trivial to bypass by a knowledgeable attacker, but they are able to stop most of the currently existing viruses. Nevertheless, we expect to see viruses in the future that will be able to detect this operating system and to modify, disable, bypass, or even use its security features.

The successful discovery and prosecution of several Macintosh virus writers has probably discouraged those who were seeking fame in this field. Macintosh computers are also relatively expensive and those people who have afforded one are usually using it for work and not for hacking. The knowledge of low-level tricks about this computer is not as much widespread among the Mac users, as it is among the MS-DOS users.

Nevertheless, the prices are dropping down and more and more users are able to afford a Mac or a Unix box. Many of the tricks used in the MS-DOS virus world are directly applicable there, especially tunnelling, virus authoring packages, and polymorphism. The last two are even easier than in the MS-DOS environment—because the MacOS provides a much more user-friendly way to create nice user interfaces and because the 68x00 CPU has a much more orthogonal set of instructions than the 80x86. Several virus writing groups, currently active in the MS-DOS world, have expressed their intention to expand their activities to cover the Macintosh world as well.

Therefore, the producers of anti-virus software for non MS-DOS platforms should pay particular attention to the activities in "their" field. Currently, the anti-virus programs used there are almost exclusively of the scanner type. Their authors must be prepared for the same kind of anti-scanner attacks that are popular among the MS-DOS viruses and consider the development and usage of other kinds of anti-virus software.

## 2.9 Multi-Platform Viruses.

This problem is somehow related to the previous one, although we do not expect it to become a serious threat in the future.

The current viruses are limited to a particular platform. There is no way for an MS-DOS virus to infect a Macintosh computer—unless the latter is running some kind of MS-DOS emulator. And even then, the infection will be limited within the emulated environment.

It must not be necessarily so. It is perfectly possible to write a virus that will be able to work on more than one kind of CPU. In particular, a boot sector infector for both IBM PCs and Atari STs is particularly easy—because the two computers use almost the same file systems ([Ferbrache]). A Mac-IBM infector is more difficult to write, but still perfectly possible. One could even imagine a program, that spreads like a worm between Internet-connected Unix machines. Once it succeeds to install itself on a machine, it could use virus techniques specific to the particular hardware, in order to gain full control of the machine and spread further.

Nevertheless, we believe that the multi-platform viruses will not represent a considerable problem in the future. It is much easier to write two viruses for two different platforms, than a single virus that is able to spread on any of the two platforms. In the same time, such

multi-platform virus will not spread easily between the two platforms, because the software exchange between them is relatively low.

# 3 Dirty Tricks.

Above we presented some of the generally successful ideas in virus writing that will form the trends of the future. Here we shall mention a few tricks discovered by the virus writers on the MS-DOS platform that seem to be successful and will be probably used more widely in the future.

## 3.1 No Clean Reboot.

This was first discovered by the author of the `EXE_Bug` virus ([Davidson]). The idea consists of modifying the CMOS of the computer in such a way, as to indicate that the first floppy disk drive is not present. On some BIOSes this causes the computer to try to boot from the hard disk first. This way the virus (which has infected the hard disk) gets loaded in memory, checks for the presence of a diskette in the floppy disk drive, loads its boot sector and transfers control to it. From the point of view of the user, the process looks as if the computer has booted from a clean floppy. Yet the virus is active in memory and since it is also stealth, it is relatively difficult to discover its presence on the hard disk.

Fortunately, this trick works on relatively few computers. However, on those, on which it does work, it represents a significant hassle to the user. It also makes the life difficult for the producers of anti-virus software—now the action "boot from a clean diskette" is no longer so easy to explain and perform.

And the trick can be extended further. Many BIOSes have a special bit in the CMOS, the meaning of which is exactly "try to boot from the hard disk first." If the virus is able to detect the computers with such BIOSes, it will be able to implement the trick described above in a much more elegant and portable way. On the top of that, it could even install a password on the CMOS settings, so that even if the user happens to discover that something has gone wrong, s/he will not be able to correct the CMOS settings easily. S/he will have to disconnect the CMOS battery or to use a special program to patch the CMOS directly. The latter could be intercepted by a virus which constantly monitors the contents of the CMOS and restores it to a status that is favorable for the virus...

## 3.2 Hardware-level Stealth.

This is another new trick which has been first implemented in the Russian virus `Strange` [Kaspersky93]. The virus intercepts the "device ready" interrupts, issued by the hard disk controller after a hard disk access occurs (XTs) or when it is about to take place (ATs). The virus then modifies the contents of the buffer that has been read (or respectively modifies the access request that is about to take place) in a way to conceal its presence. Therefore, even if the anti-virus program has direct access to the `INT 13h` vector (via interrupt tracing or direct calls to the EPROM of the hard disk controller), the virus is still able to stealth the fact that it has infected the hard disk.

This one more time demonstrates how futile all anti-stealth techniques are and how important is for the user to boot from a clean diskette before attempting any virus hunting. Unfortunately, as mentioned above, tricks like the one used by the EXE_Bug virus could make this rather difficult to achieve.

Because the idea seems successful, it will probably be widely adopted by the viruses of the future. As measures against it, the users must be educated how to perform a clean reboot in a safe way (and why it is so important). Also, the anti-virus programs must watch for viruses that have intercepted the "device ready" interrupts and to disconnect them, if possible.

## 3.3 Polymorphic Viruses, Using the Commander Bomber Infection Technique.

About two years ago, the Bulgarian virus writer, known under the handle Dark Avenger, created a virus, called Commander Bomber. By itself, this fact is not so amazing—Dark Avenger is known to have created more than two dozens of different viruses. However, the Commander Bomber virus was different—it used a new infection technique.

The virus infects only COM files, but in a very special way. It inserts its body at a random place in the file. Then, it generates several small random pieces of code, which it puts in different places in the attacked file. One of them is always in the beginning of the file. Those pieces of code do nothing in particular—the instructions in them swap some values between some registers and transfer control to the next piece of code until eventually the main virus body receives control.

The outcome of all this is that a scanner has no way to determine where exactly in the file the virus is. All "smart" scanning techniques, like entry point tracing, top-and-tail scanning, etc. suddenly stop working. The simplest way to detect the virus is to scan the whole file.

This is not a big problem with this particular virus, except that it slows down the scanning process. Unfortunately, the next idea is obvious—to combine the infection technique of Commander Bomber with a polymorphic mechanism like the one used in the MtE. Indeed, all current algorithms to detect the MtE-based viruses rely on the fact that it is relatively easy to find the entry point to the decryptor generated by the MtE. If it is concealed in a way similar to the one used in Commander Bomber, the task suddenly becomes of an order of magnitude harder...

Probably the best counter-measure against such attack is to make the scanners contain some kind of CPU emulator. This emulator will be able to interpret the beginning of the virus, until it reaches the point at which the virus will have received control and decrypted itself. This could also be used as a generic weapon against the polymorphic viruses, as mentioned elsewhere in this paper.

## 3.4 Slow Multi-Partite Stealth Polymorphic Viruses.

We have seen all kinds of clever ideas implemented in the currently existing viruses. What has not been observed yet are viruses that cleverly combine all the clever ideas, in order to achieve as fast intra-computer spread as possible.

The fast infecting viruses try to achieve wide dissemination by infecting all accessible objects (e.g., files) in a system. However, this also usually leads to the quick detection of the virus. And once the virus is detected, it has almost no chance—the user could use a scanner to detect all infected objects, regardless how many they are, and to replace or disinfect them. At the same time, viruses that infect only a single, rarely accessed object (e.g., the master boot sector) often remain undetected for a longer time and achieve a wider dissemination in the long run. They are also very easy to remove, but as we have seen above, the same is true for the viruses which infect more than one object—once they are discovered, of course. In the same time, viruses which infect only single objects, especially if those objects are known to modify themselves, are likely to remain undetected for much longer—even by an integrity checking based defense.

Such viruses, which infect only the objects that are being modified, are known as slow viruses. While they are rather effective against integrity checking programs, they also spread very slowly, so the probability for a user to get infected by one of them is quite low. However, the slow infection strategy could be combined with other of the virus tricks that are known to be successful.

For instance, the virus could indeed infect only a single object on the attacked computer, but this object could be randomly selected from a wide range of candidates. Some obvious candidates are the master boot sector, the DOS boot sector, the two DOS hidden files, the command interpreter, the device drivers, the `CONFIG.SYS` and `AUTOEXEC.BAT` files, and programs loaded during the startup process, etc. Some of these programs are known to be self-modifiable (e.g. `SETVER.EXE`) and their modification by the virus is unlikely to raise the suspicion of the user. At the same time, advanced methods for infection could be used—the way the `StarShip` virus infects the master boot sector, or the DOS file fragmentation attack ([Bontchev92]) are just two of the possible examples.

Once loaded in memory, the virus could use floppy disk boot simulation like the `EXE_Bug` virus, and hardware level stealth like the `Strange` virus—those attacks were already explained above. It could also continue to work as a slow virus—that is, to infect anything that is modified. For instance, the files that are copied to a floppy, the files that are being archived ([Bontchev92]), and so on.

And "anything" above could really mean "anything executable"—the virus could be multi-partite and infect COM and EXE files, device drivers, OBJ files, libraries, boot sectors ([Bontchev92]). The stealth capabilities of the virus would prevent the infection from being discovered easily on the machine where the virus is active. In the same time, a user who gets a new, already infected file, will not know its original contents and thus will not be able to notice that the file is already infected.

Actually, the virus could be even polymorphic and apply `MtE/TPE`-level of polymorphism, in order to make its detection difficult by the scanners—even after the anti-virus researchers become aware of the existence of the virus. In fact, the virus could also mutate very slowly. That is, it could be able to potentially generate a huge amount of different variants, but generate a new variant only from time to time, rarely. This would make more difficult the creation of a large set of different replicants and thus the testing of the quality of the scanners will be more troublesome.

As a measure against such viruses, the users should consider a multi-level defense strategy. Neither a monitoring program nor a scanner, nor an integrity checker will be able to

provide enough protection, if used alone. The user should rely on a system that uses a combination of these methods, each of them implemented in a secure way. The integrity of the startup procedure (CMOS, MBR, DBR, DOS files, device drivers, INSTALLed programs, command interpreter, programs loaded from `AUTOEXEC.BAT`) must be watched very carefully. The system must be scanned in advance (i.e., before the installation of the anti-virus package) for known viruses. Each new software must be scanned for known viruses too, before any attempt is made to install it on the protected system. Some kind of decoy launching system could be used in an attempt to catch the slow infectors. Anti-stealth techniques should be used whenever possible, but the users should be educated not to rely only on them and how to boot from a clean system, if a virus is suspected.

## 3.5 Viruses Written in High-Level Languages.

It is possible to use almost any computer language to write a virus, including the high-level ones like C, Pascal, BASIC, etc. And indeed, these languages are sometimes used to create viruses. These viruses are usually extremely primitive (of the overwriting type), although it is perfectly possible to write a full-featured virus in such a language (the `Sentinel` virus is an example of this). However, while very unlikely to spread, these viruses are still a serious hassle for the producers of anti-virus programs of the scanning type.

The problem is connected with the "false positives" problem which we discussed above. Since the most part of a virus that is written in a high-level language consists of system libraries, it is extremely difficult to pick a characteristic scan string for it. That is, a scan string that does not cause false positives. And indeed, if the anti-virus researcher commits the mistake to select a scan string from a system library that can be found in the virus, his scanner is likely to "detect" the virus in any other program, written in the same high-level language and compiled with the same compiler (and containing the same library). There have been several such cases already—for viruses like `Kamikaze`, `Wonder`, etc.

We already discussed the problems that a false positive could cause. Here, we would like only to emphasize, that more high-level language viruses should be expected in the future. They are significantly easier to write than the viruses written in assembly language—which means that more people have the knowledge to write them. We don't think that such viruses will represent a major problem in the future—instead, the problems will be caused by the scanners with inappropriate scan strings for those viruses.

There are several ways to deal with this problem. The first is simply not to try to detect these viruses—they are not a serious threat anyway. Unfortunately, the numbers game and the competition ("my scanner detects more viruses than your scanner") force the anti-virus producers to include in their scanners the ability to detect even such viruses. The other solution is to very carefully select the scan strings. But this is already very difficult and there are some ways in which the virus author could make it even more difficult, e.g. by using short and "ordinary" operators to write to the attacked files, for instance. The third solution seems to be the most appropriate one—to perform exact identification of such viruses.

# 4 Social Dangers.

Above we discussed the major technical ideas that should be expected to be implemented in the viruses of the feature. In the next few sections, we shall try to present some social actions that could result in the increased impact of the viruses to the users.

## 4.1 Anti–Anti–Virus Researchers Attacks.

We have already witnessed several personal attacks on well known anti–virus researchers. Usually this consists of including some libel or offensive messages in the viruses. Unfortunately, there are many other ways to do it, and we should expect that they all will be used by the virus writers.

One way is to disseminate a well–prepared rumor, e.g. that a particular version of a particular anti–virus product releases a virus, or does not always detect a particular widespread virus, or causes some damage. Since the qualified testers of anti–virus programs are very few, and since many anti–virus programs do contain bugs, such rumor is relatively easy to prepare and difficult to disprove.

To combat this, a qualified body of testers of anti–virus products is needed, and possibly even an organization that provides some kind of external quality control and certification.

Another attack, which we can see even nowadays, is the constant trojanization of anti–virus programs (that is—the distribution of fake "new" versions, which contain a virus or perform some other damage). Also, scanners are often "cracked" and the scan strings used by them are made public. This presents a double danger—the virus writers can modify their viruses easily to avoid detection by a particular scanner, if they know the particular scan string, used to detect the virus. Also, many producers of scanners do not feel particularly happy if the result of their hard work—the virus scan strings—are released to be used by their competitors. Probably the scanner that is most attacked this way is `ViruScan`, produced by McAfee.

In order to thwart this threat, the producers of anti–virus software must provide some kind of authentication for their products. Especially for those that are distributed as shareware, because they are more prone to trojanization. Some kind of scheme using public key cryptography seems to be the most appropriate method.

Finally, the virus authors could cause some trouble to the currently existing schemes for virus naming and classification. Each of the currently existing schemes contains drawbacks that could be exploited by the virus writers.

For instance, the CARO naming scheme contains several rules stating how virus names should *not* be selected (e.g. names of anti–virus researchers, offending words, geographic locations). However, it is very difficult to select a name for a virus which does nothing but replicate. If this virus additionally contains a single text word, it is very likely that this word will be selected as a name for the virus—because it is the most obvious choice. Regardless what the naming rules are, people who discover the virus will almost certainly use the obvious name.

Furthermore, it is very difficult to classify a virus, hidden behind a variable decryptor of a polymorphic scheme. For instance, the `Pogue` virus is clearly a variant from the `Gotcha` family, but nevertheless it is classified as an `MtE` variant—because the `MtE` encryption "hides"

the virus from most scanners and the decryptor is the only thing that can be "seen." However, there are already at least two polymorphic engines available—`MtE` and `TPE` (the latter even exists in several closely related variants). There will be certainly more of them in the future. If somebody takes all the `MtE` based viruses and links them with `TPE` instead, this will cause a significant mess in the classification schemes.

In order to deal with this problem, the virus classification must be improved significantly. It must be based entirely on the structure of the virus and not be related in any way with such concealing parts like the decryptor or the polymorphic engine. Unfortunately, this also means that if a scanner wants to be compatible with such a classification scheme, it has to be able to peel the decryptor of the virus and reach the pure code. Few scanners nowadays are able to achieve that, but as we are going to see more polymorphic viruses in the future, implementing means to "look" beyond the decryptor might be the only way the scanners must go anyway.

## 4.2 Wide Dissemination of Virus Information.

It all began with Ralf Burger's book about computer viruses, which published the source of several primitive viruses. Todor Todorov's `Virus eXchange` BBS in Sofia, Bulgaria, was the next logical move. Nowadays we have dozens of such BBSes around the world. Some of them are even connected in a FidoNet-based virus distribution network. We also have Mark Ludwig's infamous "*Little Black Book*" and even his virus writing quarterly magazine. Several other electronic publications exist (like *40-Hex*, *Crypt Newsletter*, *NukE InfoJournal*, etc.) which promote virus writing, contain detailed explanations about how to write different kinds of viruses and even ready-to-type virus sources or `DEBUG` scripts.

It is only a matter of time, and not a lot of time, until the virus writers begin to use the *Internet* to organize themselves, to distribute viruses, and knowledge about how to write them. Virus distribution newsgroups based on the `alt.*` hierarchy and virus exchange `anonymous ftp` sites will appear soon. All this will make the knowledge about how to write viruses even more popular and available. Therefore, even more people will be tempted to write their own virus. Also, criminals or disgruntled employees will have a ready source for viruses they would want to release in the world.

What can be done against this trend? It will happen sooner or later, but we must do our best to delay it as much as possible. Responsible system administrators must watch for the usage (and the misuse) of the anonymous ftp sites they are moderating. They must refuse to carry and re-distribute any newsgroup used for virus distribution. The appropriate *Usenet* authorities must be contacted in each case when network misuse it reported.

## 4.3 Pro-Virus Organizations.

There are already rather well organized groups of virus writers all over the world. In some countries (e.g., the USA) any kind of creativity and publishing is considered protected under the right of free speech—even if this is the creation of a virus or the publishing of a source code for a virus. One could easily imagine that in these countries we shall witness soon the creation of activist groups, protecting the rights of the virus writers.

Even if such organizations do not create viruses themselves, they could cause a lot of trouble, which will have as a net result a negative effect to the users. For instance, they could provide financial and legal help when some virus writer is caught and is being sued. They could lobby against any laws that would recede the "rights" of the virus writers. They could even sue the anti-virus researchers for copyright infringement—because they are including part of other people's viruses (the scan strings) in their scanners—without the permission of the authors of these viruses! As the English anti-virus expert Dr. Alan Solomon often says, in the computer virus world white is often black and black is often white.

The threat described above would seem funny, if it was not so serious. Of course, the right of free speech is too precious to make it a victim to any anti-virus laws. Virus writing per se does not need to be made illegal. Instead, the laws should concentrate on the damage caused by the virus. A virus writer should not be held responsible—unless his virus appears somewhere where it is not wanted. But if it does, then its creator must be prosecuted (if known, of course)—even if he is not directly responsible for spreading the virus. Of course, the person who has spread it intentionally is even more guilty and should be prosecuted more severely, but the original author should be held responsible too—for letting his creation "escape." Maybe "criminal negligence" is the proper legal term here.

## 4.4 Viruses For Sale.

In many countries the intentional infection of somebody's machine without the authorisation of the owner of the machine is a criminal act. However, providing a virus to somebody while informing him about the fact that this is a virus is usually not considered illegal. The problems here are closely related to the free virus writing and virus exchange mentioned above. And, what is not illegal, should be permitted, right? So, why not the selling of viruses?

From the economical point of view, there is only one main question—is there enough market for viruses? Unfortunately, the answer to this question is often "yes" [Solomon93a]. So, who needs to buy viruses?

It seems that the obvious answer would be criminals or disgruntled employees, who need a virus to attack a particular system. However, they could easily obtain a virus for free from many of the existing virus exchange BBSes. Actually, such a virus exchange BBS even used to be run by the US Government—the department of Treasury.

However, the contents of these BBSes is usually a horrible mess ([Bontchev93a]). They contain viruses, corrupted or partially infected files, which somebody's scanner has declared to be viruses, virus construction tools, trojan horses, virus sources, virus disassemblies, raw outputs of a disassembler (usually `Sourcer` from V Communications) when run on an infected file, virus-related electronic newsletters, etc. There are many duplicated files, different viruses under one and the same name, one and the same viruses under different names, non-working viruses, programs written with the intent to write a virus, but so buggy that they could never replicate, etc. Often there are even perfectly legitimate programs like `FORMAT`, etc.

Very few virus collections are well-organized. At the same time, there are people, who feel to have the legitimate need for a well-organized virus collection. Those are companies who decide to enter the anti-virus business. For reasons explained elsewhere in this paper, it

is almost impossible for a new company to successfully establish itself in this business. But most people who are not well enough acquainted with the virus situation, do not know this fact. And since it is almost impossible to get a large virus collection from the self-respecting anti-virus researchers, newcomers in the anti-virus business are often tempted to obtain the viruses they need for their product via semi-legal means. If somebody appears, providing a well-organized (or even a not so well-organized) rich virus collection for sale, it is quite probable that he will find customers.

Other prospective customers could be evaluators of anti-virus products for the different computer magazines. They often feel the need of a large virus collection in order to verify the claims of the authors of anti-virus products to detect "all known and unknown viruses."

In fact, there have been virus collections on sale before. This will probably happen again. What can be done about it?

The main solution is human education and appropriate legislation. People must know that the possession of a large virus collection does not guarantee the creation of a successful anti-virus product. Only qualified and commercially unbiased anti-virus experts should be consulted to evaluate the anti-virus software. At last, people should be aware that according to the laws of some countries (e.g., the UK), selling viruses could be considered as an incitement to commit a crime (i.e., to spread the virus) and is therefore illegal. Perhaps more countries should pass similar laws.

## 4.5 Viruses Used as Weapons.

Several countries are reportedly researching into the possibilities to use viruses as a weapon against an enemy. However, it is unlikely that the outcome of such research will be positive—computer viruses are too difficult to aim towards a particular target. They could be used much more successfully in a terrorist attack—when the attacker does not know and does not care how much and which particular targets will be hit.

The countries which are more vulnerable to this kind of attack are the most developed ones—the ones which are widely relying on computers in their economics. A virus attack could be even more successful if performed on a cluster of highly networked computers, especially if the virus used knows and uses the security holes in the network to spread itself faster. Actually, this could be a combination of a worm and a multi-partite (or multi-platform) virus.

Probably the widest set of computers networked together is the Internet. Many of the computers there are using similar operating systems—usually a variation of Unix. The particular implementations often have widely known security holes and/or are maintained by people who are new to the system administrator's job—or even people for whom this is not their main job. These people are mainly interested in keeping the computer working—not in converting it into an electronic variant of Fort Knox. Hence, many of the computers on the Internet are believed to be insecure and vulnerable to hacker attacks.

We have already witnessed several attacks on a net-wide basis. The most famous of them is probably the notorious Internet worm. Others include the `WANK/OILZ` worms, the `Father Christmas` worm, the `CHRISTMA EXEC` chain letter and its variants, and so on.

Most computers on the Internet belong to educational institutions and are not very tempting as victims of a terrorist virus attack. However, more and more companies connect

their computers to the Internet too—in order to use its capabilities of electronic conferencing, electronic mail, anonymous ftp, and so on. Therefore, the number of victims suitable for attack steadily increases.

In order to diminish the danger, all system administrators of the machines that are attached to the Internet should be educated in maintaining security of their sites to an acceptable level. Whenever possible, the process of enhancing the security should be automated. Tools like `Cops` and `Tripwire` should be widely used. Whenever possible, encryption should be used to protect the communications between the sites and public-key authentication should be used to authenticate each site. `Kerberos` is one of the most suitable tools for this purpose, but due to some legal problems its full version is not exportable outside the USA.

# 5 Conclusion.

The computer virus problem is not going to disappear soon. It is going to be with us in the years to come and it is going to become even worse. Those people who have accepted the duty to fight it should carefully examine the possible methods of attack that are likely to be used by the virus authors in the future and take some steps to counter them. Take them now, while there is still time.

# References


*[Bontchev92]* *Vesselin Bontchev, "Possible Integrity Attacks Against Integrity Programs And How To Prevent Them," Proc. 2nd Int. Virus Bulletin Conf., September 1992, pp. 131–141.*

*[Bontchev93]* *Vesselin Bontchev, "MtE detection test," Virus News International, January 1993, pp. 26–34.*

*[Bontchev93a]* *Vesselin Bontchev, "Analysis and Maintnance of a Clean Virus Library," Proc. 3rd Int. Virus Bulletin Conf., September 1993, pp. 77–89.*

*[Cohen]* *Dr. Frederick B. Cohen & Sanjay Mishra, "Some Initial Results from the QUT Virus Research Network," Proc. 2nd Int. Virus Bulletin Conf., September 1992, pp. XV–XXV.*

*[Davidson]* *Iolo Davidson. "The Death of the Clean Boot?." Virus News International, April 1993, pp. 50–52.*

*[Ferbrache]* *David Ferbrache. "A Pathology of Computer Viruses," Springer-Verlag, 1992.*

*[Gold]* *Steve Gold. "Novell NetWare Security Breach," Virus News International, November 1992, pp. 84–87.*

*[Kaspersky]* *Eugene Kaspersky, personal communication, based on a paper in print.*

[Kaspersky93] *Eugene Kaspersky & Vadim Bogdanov. "Strange—A New Way to Hide." Virus Bulletin. April 1993. pp. 12–13.*

[Solomon92] *Alan Solomon. "Mechanisms of Stealth." Virus News and Reviews. June 1992, pp. 232–238.*

[Solomon93] *Alan Solomon. "False Alarms." Virus News International. February 1993, pp. 50–52.*

[Solomon93a] *Alan Solomon. "Viruses—Supply and Demand." Virus News International, April 1993, p. 47.*

[Skulason] *Fridrik Skulason. "Virus Trends". Proc. 2nd Int. Virus Bulletin Conf., September 1992.*