Are "good" computer viruses still a bad idea? 13782

Are “ Good” Computer Viruses Still a Bad Idea?

Vesselin Bontchev

Research Associate

Virus Test Center

University of Hamburg

Vogt-Koelln-Str. 30, 22527 Hamburg, Germany

]

During the past six years, computer viruses have caused unaccountable amount of

damage – mostly due to loss of time and resources. For most users, the term

“ computer virus” is a synonym of the worst nightmares that can happen on their

system. Yet some well-known researchers keep insisting that it is possible to

use the replication mechanism of the viral programs for some useful and

beneficial purposes.

This paper is an attempt to summarize why exactly the general public appreciates

computer viruses as something inherently bad. It is also considering several of

the proposed models of “ beneficial” viruses and points out the problems in them.

A set of conditions is listed, which every virus that claims to be beneficial

must conform to. At last, a realistic model using replication techniques for

beneficial purposes is proposed and directions are given in which this technique

can be improved further.

The paper also demonstrates that the main reason for the conflict between those

supporting the idea of a “ beneficial virus” and those opposing it, is that the

two sides are assuming a different definition of what a computer virus is.

1. What Is a Computer Virus?

The general public usually associates the term “ computer virus” with a small,

nasty program, which aims to destroy the information on their machines. As usual,

the general public’s understanding of the term is incorrect. There are many

kinds of destructive or otherwise malicious computer programs and computer

viruses are only one of them. Such programs include backdoors, logic bombs,

trojan horses and so on [Bontchev94]. Furthermore, many computer viruses are not

intentionally destructive – they simply display a message, play a tune, or even

do nothing noticeable at all. The important thing, however, is that even those

not intentionally destructive viruses are not harmless – they are causing a lot

of damage in the sense of time, money and resources spent to remove them –

because they are generally unwanted and the user wishes to get rid of them.

A much more precise and scientific definition of the term “ computer virus” has

been proposed by Dr. Fred Cohen in his paper [Cohen84]. This definition is

mathematical – it defines the computer virus as a sequence of symbols on the

tape of a Turing Machine. The definition is rather difficult to express exactly

in a human language, but an approximate interpretation is that a computer virus

is a “ program that is able to infect other programs by modifying them to include

a possibly evolved copy of itself”.

Unfortunately, there are several problems with this definition. One of them is

that it does not mention the possibility of a virus to infect a program without

modifying it – by inserting itself in the execution path. Some typical examples

are the boot sector viruses and the companion viruses [Bontchev94]. However,

this is a flaw only of the human-language expression of the definition – the

mathematical expression defines the terms “ program” and “ modify” in a way that

clearly includes the kinds of viruses mentioned above.

A second problem with the above definition is its lack of recursiveness. That is,

it does not specify that after infecting a program, a virus should be able to

replicate further, using the infected program as a host.

Another, much more serious problem with Dr. Cohen’s definition is that it is too

broad to be useful for practical purposes. In fact, his definition classifies as

“ computer viruses” even such cases as a compiler which is compiling its own

source, a file manager which is used to copy itself, and even the program

DISKCOPY when it is on diskette containing the operating system – because it can

be used to produce an exact copy of the programs on this diskette.

In order to understand the reason of the above problem, we should pay attention

to the goal for which Dr. Cohen’s definition has been developed. His goal has

been to prove several interesting theorems about the computational aspects of

computer viruses [Cohen89]. In order to do this, he had to develop a

mathematical (formal) model of the computer virus. For this purpose, one needs a

mathematical model of the computer. One of the most commonly used models is the

Turing Machine (TM). Indeed, there are a few others (e. g., the Markoff chains,

the Post Machine, etc.), but they are not as convenient as the TM and all of

them are proven to be equivalent to it.

Unfortunately, in the environment of the TM model, we cannot speak about

“ programs” which modify “ other programs” – simply because a TM has only one,

single program – the contents of the tape of that TM. That’s why Cohen’s model

of a computer virus considers the history of the states of the tape of the TM.

If a sequence of symbols on this tape appears at a later moment somewhere else

on the tape, then this sequence of symbols is said to be a computer virus for

this particular TM. It is important to note that a computer virus should be

always considered as related to some given computing environment – a particular

TM. It can be proven ([Cohen89]) that for any particular TM there exists a

sequences of symbols which is a virus for that particular TM.

Finally, the technical computer experts usually use definitions for the term

“ computer virus”, which are less precise than Dr. Cohen’s model, while in the

same time being much more useful for practical reasons and still being much more

correct than the general public’s vague understanding of the term. One of the

best such definitions is ([Seborg]):

“ We define a computer ‘ virus’ as a self-replicating program that can

‘ infect’ other programs by modifying

them or their environment such that a call to an ‘ infected’ program implies

a call to a possibly evolved, and in

most cases, functionally similar copy of the ‘ virus’.”

The important thing to note is that a computer virus is a program that is able to replicate by itself. The definition does not specify explicitly that it is a malicious program. Also, a program that does not replicate is not a virus, regardless of whether it is malicious or not. Therefore the maliciousness is neither a necessary, nor a sufficient property for a program to be a computer virus.

Nevertheless, in the past ten years a huge number of intentionally or non

intentionally destructive computer viruses have caused an unaccountable amount

of damage – mostly due to loss of time, money, and resources to eradicate them –

because in all cases they have been unwanted. Some damage has also been caused

by a direct loss of valuable information due to an intentionally destructive

payload of some viruses, but this loss is relatively minor when compared to the

main one. Lastly, a third, indirect kind of damage is caused to the society –

many users are forced to spend money on buying and time on installing and using

several kinds of anti-virus protection.

Does all this mean that computer viruses can be only harmful? Intuitively,

computer viruses are just a kind of technology. As with any other kind of

technology, they are ethically neutral – they are neither “ bad” nor “ good” – it

is the purposes that people use them for that can be “ bad” or “ good”. So far

they have been used mostly for bad purposes. It is therefore natural to ask the

question whether it is possible to use this kind of technology for good purposes.

Indeed, several people have asked this question – with Dr. Cohen being one of

the most active proponents of the idea [Cohen91]. Some less qualified people

have attempted even to implement the idea, but have failed miserably (see

section 3). It is natural to ask – why? Let’s consider the reasons why the idea

of a “ good” virus is usually rejected by the general public. In order to do this,

we shall consider why people think that a computer virus is always harmful and

cannot be used for beneficial purposes.

2. Why Are Computer Viruses Perceived as Harmful?

About a year ago, we asked the participants of the electronic forum Virus-

L/comp. virus, which is dedicated to discussions about computer viruses, to list

all reasons they could think about why do they perceive the idea of a

“ beneficial” virus as a bad one. What follows is a systematized and generalized

list of those reasons.

2. 1. Technical Reasons

This section lists the arguments against the “ beneficial virus” idea, which have

a technical character. They are usually the most objective ones.

2. 1. 1. Lack of Control

Once released, the person who has released a computer virus has no control on

how this virus will spread. It jumps from machine to machine, using the

unpredictable patterns of software sharing among the users. Clearly, it can

easily reach systems on which it is not wanted or on which it would be

incompatible with the environment and would cause unintentional damage. It is

not possible for the virus writer to predict on which systems the virus will run

and therefore it is impossible to test the virus on all those systems for

compatibility. Furthermore, during its spread, a computer virus could reach even

a system that had not existed when that virus has been created – and therefore

it had been impossible to test the virus for compatibility with this system.

The above is not always true – that is, it is possible to test the virus for

compatibility on a reasonably large number of systems that are supposed to run

it. However, it is the damaging potential of a program that is spreading out of

control which is scaring the users.

2. 1. 2. Recognition Difficulty

Currently a lot of computer viruses already exist, which are either

intentionally destructive or otherwise harmful. There are a lot of anti-virus

programs designed to detect and stop them. All those harmful viruses are not

going to disappear overnight. Therefore, if one develops a class of beneficial

viruses and people actually begin to use them, then the anti-virus programs will

have to be able to make the difference between the “ good” and the “ bad” viruses

– in order to let the former in and keep the latter out.

Unfortunately, in general it is theoretically impossible even to distinguish

between a virus and a non-viral program ([Cohen89]). There is no reason to think

that distinguishing between “ good” and “ bad” viruses will be much easier. While

it might be possible to distinguish between them using virus-specific anti-virus

software (e. g., scanners), we should not forget that many people are relying on

generic anti-virus defenses, for instance based on integrity checking. Such

systems are designed to detect modifications, not specific viruses, and

therefore will be triggered by the “ beneficial” virus too, thus causing an

unwanted alert. Experience shows that the cost of such false positives is the

same as of a real infection with a malicious virus – because the users waste a

lot of time and resources looking for a non-existing problem.

2. 1. 3. Resource Wasting

A computer virus would eat up disk space, CPU time, and memory resources during

its replication. A computer virus is a self-replicating resource eater. One typical example is the Internet Worm, accidentally released by a Carnegie-Mellon student. It was not designed to be intentionally destructive, but in the process of its replication, the multiple copies of it used so much resources, that they practically brought down a large portion of the Internet.

Even when the computer virus uses a limited amount of resources, it is considered as a bad thing by the owner of the machine on which the virus is doing it, if it happens without authorization.

2. 1. 4. Bug Containment

A computer virus can easily escape the controlled environment and this makes it very difficult to test such programs properly. And indeed – experience shows that almost all computer viruses released so far suffer from significant bugs, which would either prevent them from working in some environments, or even cause unintentional damage in those environments.

Of course, any program can (and usually does) contain bugs. This is especially

true for the large and complex software systems. However, a computer virus is

not just a normal buggy program. It is a self-spreading buggy program, which is

out of control. Even if the author of the virus discovers the bug at a later

time, there is the almost untreatable problem of revoking all existing copies of

the virus and replacing them with fixed new versions.

2. 1. 5. Compatibility Problems

A computer virus that can attach itself to any of the user’s programs would

disable the several programs on the market that perform a checksum on themselves

at runtime and refuse to run if modified. In a sense, the virus will perform a

denial-of-service attack and thus cause damage.

Another problem arises from some attempts to solve the “ lack of control” problem

by creating a virus that asks for permission before infecting. Unfortunately,

this causes an interruption of the task being currently executed until the user

provides the proper response. Besides of being annoying for the user, it could

be sometimes even dangerous. Consider the following example.

It is possible that a computer is used to control some kind of life-critical

equipment in a hospital. Suppose that such a computer gets infected by a

“ beneficial” computer virus, which asks for permission before infecting any

particular program. Then it is perfectly possible that a situation arises, when

a particular program has to be executed for the first time after the virus has

appeared on the computer, and that this program has to urgently perform some

task which is critical for the life of a patient. If at that time the virus

interrupts the process with the request for permission to infect this program,

then the caused delay (especially if there is no operator around to authorize or

deny the request) could easily result in the death of the patient.

2. 1. 6. Effectiveness

It is argued that any task that could be performed by a “ beneficial” virus could

also be performed by a non-replicating program. Since there are some risks following from the capability of self-replication, it would be therefore much better if a non-replicating program is used, instead of a computer virus.

2. 2. Ethical and Legal Reasons

The following section lists the arguments against the “ beneficial virus” idea, which are of ethical or legal kind. Since neither ethics, nor the legal systems are universal among the human society, it is likely that those arguments will have different strength in the different countries. Nevertheless, they have to be taken into account.

2. 2. 1. Unauthorized Data Modification

It is usually considered unethical to modify other people’s data without their authorization. In many countries this is also illegal. Therefore, a virus which performs such actions will be considered unethical and/or illegal, regardless of

any positive outcome it could bring to the infected machines. Sometimes this problem is perceived by the users as “ the virus writer claims to know better

than me what software should I run on my machine”.

2. 2. 2. Copyright and Ownership Problems

In many cases, modifying a particular program could mean that copyright,

ownership, or at least technical support rights for this program are voided.

We have witnessed such an example at the VTC-Hamburg. One of the users who

called us for help with a computer virus was a sight-impaired lawyer, who was

using special Windows software to display the documents he was working on with a

large font on the screen – so that he could read them. His system was infected

by a relatively non-damaging virus. However, when the producer of the software

learned that the machine was infected, they refused any technical support to the

user, until the infection was removed and their software – installed from clean

originals.

2. 2. 3. Possible Misuse

An attacker could use a “ good” virus as a means of transportation to penetrate a

system. For instance, a person with malicious intent could get a copy of a “ good” virus and modify it to include something malicious. Admittedly, an attacker could trojanize any program, but a “ good” virus will provide the attacker with means to transport his malicious code to a virtually unlimited population of computer systems. The potential to be easily modified to carry malicious code is one of the things that makes a virus “ bad”.

2. 2. 4. Responsibility

Declaring some viruses as “ good” and “ beneficial” would just provide an excuse

to the crowd of irresponsible virus writers to condone their activities and to claim that they are actually doing some kind of “ research”. In fact, this is already happening – the people mentioned above are often quoting Dr. Fred Cohen’s ideas for beneficial viruses as an excuse of what they are doing – often

without even bothering to understand what Dr. Cohen is talking about.

2. 3. Psychological Reasons

The arguments listed in this section are of psychological kind. They are usually

a result of some kind of misunderstanding and should be considered an obstacle

that has to be “ worked around”.

2. 3. 1. Trust Problems

The users like to think that they have full control on what is happening in

their machine. The computer is a very sophisticated device. Most computer users

do not understand very well how it works and what is happening inside. The lack

of knowledge and uncertainty creates fear. Only the feeling that the reactions

of the machine will be always known, controlled, and predictable could help the

users to overcome this fear.

However, a computer virus steals the control of the computer from the user. The

virus activity ruins the trust that the user has in his/her machine, because it

causes the user to lose his/her belief that s/he can control this machine. This

may be a source of permanent frustrations.

2. 3. 2. Negative Common Meaning

For most people, the word “ computer virus” is already loaded with negative

meaning. The media has already widely established the belief that a computer

virus is a synonym for a malicious program. In fact, many people call “ viruses”

many malicious programs that are unable to replicate – like trojan horses, or

even bugs in perfectly legitimate software. People will never accept a program

that is labelled as a computer virus, even if it claims to do something useful.

3. Some Bad Examples of “ Beneficial” Viruses

Regardless of all the objections listed in the previous section, several people

have asked themselves the question whether a computer virus could be used for

something useful, instead of only for destructive purposes.

And several people have tried to positively answer this question. Some of them

have even implemented their ideas in practice and have been experimenting with

them in the real world – unfortunately, without success. In this section we

shall present some of the unsuccessful attempts to create a beneficial virus so

far, and explain why they have been unsuccessful.

3. 1. The “ Anti-Virus” Virus

Some computer viruses are designed to work not only in a “ virgin” environment of

infectable programs, but also on systems that include anti-virus software and

even other computer viruses. In order to survive successfully in such

environments, those viruses contain mechanisms to disable and/or remove the said

anti-virus programs and “ competitor” viruses. Examples for such viruses in the

IBM PC environment are Den_Zuko (removes the Brain virus and replaces it with

itself), Yankee_Doodle (the newer versions are able to locate the older ones and

“ upgrade” the infected files by removing the older version of the virus and

replacing it with the newer one), Neuroquila (disables several anti-virus

programs), and several other viruses.

Several people have had the idea to develop the above behaviour further and to

create an “ anti-virus” virus – a virus which would be able to locate other

(presumably malicious) computer viruses and remove them. Such a self-replicating

anti-virus program would have the benefits to spread very fast and update itself

automatically.

Several viruses have been created as an implementation of the above idea. Some

of them locate a few known viruses and remove them from the infected files,

others attach themselves to the clean files and issue an error message if

another piece of code becomes attached after the virus (assuming that it has to

be an unwanted virus), and so on. However, all such pieces of “ self-replicating

anti-virus software” have been rejected by the users, who have considered the

“ anti-virus” viruses just as malicious and unwanted as any other real computer

virus. In order to understand why, it is enough to realize that the “ anti-virus

viruses” matches several of the rules that state why a replicating program is

considered malicious and/or unwanted. Here is a list of them for this particular

idea.

First, this idea violates the Control condition. Once the “ anti-virus” virus is

released, its author has no means to control it.

Second, it violates the Recognition condition. A virus that attaches itself to

executable files will definitely trigger the anti-virus programs based on

monitoring or integrity checking. There is no way for those programs to decide

whether they have been triggered by a “ beneficial” virus or not.

Third, it violates the Resource Wasting condition. Adding an almost identical

piece of code to every executable file on the system is definitely a waste – the

same purpose can be achieved with a single copy of the code and a single file,

containing the necessary data.

Fourth, it violates the Bug Containment condition. There is no easy way to locate and update or remove all instances of the virus.

Fifth, it causes several compatibility problems, especially to the selfchecking programs, thus violating the Compatibility condition.

Sixth, it is not as effective as a non-viral program, thus violating the Effectiveness condition. A virus-specific anti-virus program has to carry thousands of scan strings for the existing malicious viruses – it would be very ineffective to attach a copy of it to every executable file. Even a generic anti-virus (i. e., based on monitoring or integrity checking) would be more effective if it exists only in one example and is executed under the control of the user.

Seventh, such a virus modifies other people’s programs without their authorization, thus violating the Unauthorized Modification condition. In some

cases such viruses ask the user for permission before “ protecting” a file by infecting it. However, even in those cases they cause unwanted interruptions,

which, as we already demonstrated, in some situations can be fatal.

Eight, by modifying other programs such viruses violate the Copyright condition.

Ninth, at least with the current implementations of “ anti-virus” viruses, it is trivial to modify them to carry destructive code – thus violating the Misuse condition.

Tenth, such viruses are already widely being used as examples by the virus writers when they are trying to defend their irresponsible actions and to disguise them as legitimate research – thus the idea violates the responsibility

condition too.

As we can see from the above, the idea of a beneficial anti-virus virus is “ bad”

according to almost any of the criteria listed by the users.

3. 2. The “ File Compressor” Virus

This is one of the oldest ideas for “ beneficial” viruses. It is first mentioned in Dr. Cohen’s original work [Cohen84]. The idea consists of creating a self-replicating program, which will compress the files it infects, before attaching

itself to them. Such a program is particularly easy to implement as a shell script for Unix, but it is perfectly doable for the PC too. And it has already been done – there is a family of MS-DOS viruses, called Cruncher, which appends itself to the executable files, then compresses the infected file using Lempel-Zev-Huffman compression, and then prepends a small decompressor which would decompress the file in memory at runtime.

Regardless of the supposed benefits, this idea also fails the test of the criteria listed in the previous section. Here is why.

First, the idea violates the Control condition. Once released, the author of the virus has no means to controls its spread. In the particular implementation of Cruncher, the virus writer has attempted to introduce some kind of control. The virus asks the user for permission before installing itself in memory, causing unwanted interruptions. It is also possible to tell the virus to install itself without asking any questions – by the means of setting an environment variable.

However, there are no means to tell the virus not to install itself and not to

ask any questions – which should be the default action.

Second, the idea violates the Recognition condition. Several virus scanners

detect and recognize Cruncher by name, the process of infecting an executable

triggers most monitoring programs, and the infected files are, of course,

modified, which triggers most integrity checkers.

Third, the idea violates the Resource condition. A copy of the decompressor is

present in every infected file, which is obviously unnecessary.

Fourth, the idea violates the Bug Containment condition. If bugs are found in

the virus, the author has no simple means to distribute the fix and to upgrade

all existing copies of the virus.

Fifth, the idea violates the Compatibility condition. There are many files which

stop working after being compressed. Examples include programs that perform a

self-check at runtime, self-modifying programs, programs with internal overlay

structure, Windows executables, and so on. Admitedly, those programs stop working even after being compressed with a stand-alone (i. e., non-viral) compression program. However, it is much more difficult to compress them by accident when using such a program – quite unlike the case when the user is running a compression virus.

Sixth, the idea violates the Effectiveness condition. It is perfectly possible to use a stand-alone, non-viral program to compress the executable files and prepend a short decompressor to them. This has the added advantage that the code for the compressor does not have to reside in every compressed file, and thus we don’t have to worry about its size or speed – because it has to be executed only once. True, the decompressor code still has to be present in each compressed file and many programs will still refuse to work after being compressed. The solution is to use not compression at a file level, but at a disk level. And

indeed, compressed file systems are available for many operating environments

(DOS, Novell, OS/2, Unix) and they are much more effective than a file-level

compressor that spreads like a virus.

Seventh, the idea still violates the Copyright condition. It could be argued

that it doesn’t violate the Data Modification condition, because the user is

asked to authorize the infection. We shall accept this, with the remark

mentioned above – that it still causes unwanted interruptions. It is also not

very trivial to modify the virus in order to make it malicious, so we’ll assume

that the Misuse condition is not violated too – although no serious attempts are

made to ensure that the integrity of the virus has not been compromised.

Eighth, the idea violates the responsibility condition. This particular virus –

Cruncher – has been written by the same person who has released many other

viruses – far from “ beneficial” ones – and Cruncher is clearly used as an

attempt to condone virus writing and to masquerade it as legitimate “ research”.

3. 3. The “ Disk Encryptor” Virus

This virus has been published by Mark Ludwig – author of two books and a newsletter on virus writing, and of several real viruses, variants of many of which are spreading in the real world, causing real damage.

The idea is to write a boot sector virus, which encrypts the disks it infects with a strong encryption algorithm (IDEA in this particular case) and a user-supplied password, thus ensuring the privacy of the user’s data. Unfortunately,

this idea is just as flawed as the previous ones.

First, it violates the Control condition. True, the virus author has attempted to introduce some means of control. The virus is supposed to ask the user for permission before installing itself in memory and before infecting a disk.

However, this still causes unwanted interruptions and reportedly in some cases

doesn’t work properly – that is, the virus installs itself even if the user has told it not to.

Second, it violates the Recognition condition. Several virus-specific scanners recognize this virus either by name or as a variant of Stealth_Boot, which it actually is. Due to the fact that it is a boot sector infector, it is unlikely

to trigger the monitoring programs. However, the modification that it causes to

the hard disk when infecting it, will trigger most integrity checkers. Those

that have the capability to automatically restore the boot sector, thus removing

any possibly present virus, will cause the encrypted disk to become inaccessible

and therefore cause serious damage.

Third, the idea violates the Compatibility condition. A boot sector virus that

is permanently resident in memory usually causes problems to Windows