The Case for Beneficial Computer Viruses and Worms

- A Student's Perspective-

Type of Submission: Student Paper

Abstract: This paper reviews published material on the subject of beneficial computer viruses and worms, that is, self-replicating programs for useful purposes. The topics include Shoch and Hupp's worms at the Xerox Palo Alto Research Center (PARC), Cohen's proposals for uses for viruses, and Vesselin Bontchev's anti-virus sentiments. Finally a simulation of using viruses to destroy other viruses is included.

Author: Mr. Greg Moorer, Undergraduate Computer Science Student

Organizational Affiliation:
Department of Computer Science
Mississippi State University
PO Box 9637
Mississippi State, MS 39762
(662) 325-2756 (Voice)
(662) 325-8997 (Fax)

Email Address: gam3@ra.msstate.edu

Academic Endorsement:
Dr. Rayford B. Vaughn, Jr.
Department of Computer Science
PO Box 9637
Mississippi State, MS 39762
(662) 325-7450 (voice)
(662) 325-8997 (fax)

The Case for Beneficial Computer Viruses and Worms
- A Student's Perspective -

Abstract
This paper reviews published material on the subject of beneficial computer viruses and worms, that is, self-replicating programs for useful purposes. The topics include Shoch and Hupp's worms at the Xerox Palo Alto Research Center (PARC), Cohen's proposals for uses for viruses, and Vesselin Bontchev's anti-virus sentiments. Finally a simulation of using viruses to destroy other viruses is included.

Introduction
With the current media blitz of warnings concerning malicious viruses, little attention has been paid to the use of computer viruses for beneficial purposes. The fear of malicious code is well founded. The Robert Morris Internet Worm that was released in the fall of 1988 caused a great deal of damage to the Internet. Though the author did not intend to cause damage, a bug in the program resulted in uncontrolled replication that caused the worm to run rampant [7]. Because infected systems were heavily burdened with multiple copies of the worm, non-infected systems were removed from the network to prevent infection, resulting in an excessive loss of processing time and system availability [7]. More recently, widespread publicity concerning viruses such as the Chernobyl virus has led to a negative perception of self-replicating code. This negative connotation leads laypersons and professionals alike to ignore the possibility of using self-replicating code for positive purposes. The only mention of non-damaging viruses in mainstream media encountered is a single-page article appearing in *Newsweek*. This article refers to using virus programs as network searching tools and briefly introduces some research being performed [8]. Even the supposedly unbiased academic literature largely ignores beneficial computer viruses. Though the concept of using virus behavior for useful purposes was proposed in the earliest documentation of computer viruses, very little published research exists. One complication is defining what a beneficial virus means. Some viruses written to be malicious can become beneficial in certain applications. For example, a malicious virus was implanted on Iraqi computers in order to damage their air defense system before the start of the Gulf War [6]. Though written to be malicious, this virus was surely seen as beneficial by allied pilots flying over Iraq. Rob Rosenberger reports conflicting opinions regarding the validity of this story and his own stance is that the story originated as an April Fool's Day joke [9]. However, the point that a harmful virus can be used for beneficial purposes remains. For the purposes of this paper, a beneficial virus will be defined as a self-replicating program that performs some task without being harmful. Despite the distrust of self-replicating computer programs, some developments have been made.

The Xerox PARC Worm
An early attempt to use a self-replicating program in order to perform a useful task was the Xerox PARC worm. John F. Shoch and Jon A. Hupp conducted experiments with self-replicating programs that they likened to segmented worms [10]. These programs would be divided into several segments that would execute on separate hosts which the worm found to be idle [10]. Some stipulations that Shoch and Hupp had to consider were preventing the worms from writing to the disks of other users' machines and maintaining

a trust with the users whose machines would be occupied by the worms [10]. In order for their worms to work, Shoch and Hupp developed methods for communication between segments [10]. They did, however, run into some problems controlling the worms. After leaving one worm running overnight, they arrived the next morning to find that the worm had effectively crashed several hosts [10]. Even worse was the fact that some of the affected systems were physically inaccessible, and when systems were rebooted, they were quickly located by the worm and crashed again [10]. Luckily, Shoch and Hupp had placed code within the worm to have it shut down upon receiving a signal through the network [10]. This incident illustrated the necessity for careful control of such replicating programs. The worms with which Shoch and Hupp experimented performed such applications as displaying messages across the network, receiving messages to place wake-up calls for users through dial-out modems, disseminating animation computations through a network, and performing diagnostics on Ethernet systems [10]. These worms were programs that actually transmitted themselves through networks. Another replication technique was proposed in which a program would replicate by attaching itself to other programs. These programs were called computer viruses.

Cohen's Original Concept

Fred Cohen, who originally developed the concept of computer viruses, is the greatest proponent of beneficial virus research. In his first paper on viruses, Cohen proposed the use of viruses for compression of infected files [3]. To facilitate this, he suggests having the virus compress each executable and attach the decompression algorithm to each [3]. This would cause each program to be decompressed prior to execution of the original program [3]. Cohen mentions conducting successful tests with such a virus [3]. This example of a possible application of computer virus behavior is the most widely mentioned by other authors. However, it often proves to be the only example that other texts mention. Pfleeger repeats this concept [7] as does Bontchev [1]. However, Bontchev elaborates by attacking such use of viruses.

Bontchev's Thoughts on Beneficial Viruses

One of the few authors who addresses beneficial viruses, though in a negative light, is Vesselin Bontchev. Bontchev in his paper "Are 'Good' Computer Viruses Still a Bad Idea?" details twelve arguments against beneficial viruses altogether and uses these to attack several examples of beneficial viruses. Bontchev cites the following arguments against the use of viruses for beneficial purposes:

1. viruses are difficult to fully control,
2. viruses waste resources,
3. viruses are difficult to identify and remove if unwanted,
4. viruses often contain bugs,
5. viruses are not compatible with different platforms,
6. viruses cannot perform a task in a better manner than a normal program,
7. viruses alter data without user consent,
8. viruses infecting other programs can nullify technical support for those products,
9. good viruses may be used as a guise for an attacker to gain entry to a system,
10. malicious virus work may be presented as beneficial virus research to the public,
11. viruses utilize resources on users' systems without the users' knowledge or consent,

12. viruses carry with them a common negative connotation [1].

He strikes down the idea of a virus targeted at destroying malicious viruses stating that the anti-virus virus causes the same problems as the virus it's meant to attack [1]. This author disagrees with this assertion and will expound upon it later. Bontchev also makes a statement against the file-compression virus primarily based on the idea that the operating systems file system can perform the same function without having to append the decompression algorithm to each file [1]. He uses the same basis to discredit the idea of a virus that encrypts the files on a system [1]. Bontchev also takes a stand against one of Cohen's more recent propositions of a virus that performs various system administration tasks. Included in Bontchev's arguments against a maintenance virus are that its tasks could be performed by concurrent processes in memory, that its mechanisms to avoid unwanted replication (detection of a file indicating an invitation to infect) are insufficient, and that it lacks efficiency, wasting system resources [1]. In an interview, Cohen responded to Bontchev's arguments, stating that in his experiments maintenance viruses consumed few system resources [5]. He also states that the system resources required to implement the maintenance virus reflected a great decrease in the amount of human effort required [5]. Vesselin Bontchev generally attacks concepts of beneficial viruses, but he does provide his own model of a beneficial computer virus. Bontchev's model of "good viruses" is a complicated set of invitations and verifications that actually behave more like worms than viruses. This model controls the spread of viruses by requiring the system to actively invite the virus to infect and suggests establishing virus repositories to await such invitations [1]. Bontchev also recommends requiring the exchange of digital signatures between the two hosts in order to insure both that the invitation was not forged and that the virus (or worm) received is that which was requested [1]. With so many constraints, it is much easier to simply download the desired program from the host and run it. Amazingly, after completing this model of beneficial viruses, Bontchev never mentions them in a very thorough paper on future virus trends [2]. In an interview with an online magazine, Bontchev reiterates his twelve conditions that beneficial viruses must meet and states a general distrust of virus writers [11]. Despite the continuing stand by virus experts like Bontchev against development of beneficial viruses, research continues in this field.

Cohen's More Recent Work

Cohen published a book entitled *It's Alive: The New Breed of Living Computer Programs* in 1994 which provides more ideas for beneficial viruses. In order to disassociate the programs with the negative connotation of the term "virus," he refers to these programs, which behave like both viruses and worms, as "living programs." Among the new ideas he provides are living programs that perform such tasks as software distribution across networks, implementing distributed databases, and performing routine maintenance operations such as cleaning up garbage files [4]. He also provides examples of these as UNIX scripts [4]. Cohen's research is driving beneficial virus research in its current direction.

Another Look at the Anti-Virus Virus

As stated earlier, this author disagrees with Bontchev's statement against anti-virus viruses. In a rudimentary simulation, an anti-virus virus is shown to be able to disseminate through a discourse community and effectively remove viruses. This behavior has been exhibited in simulations run by this author. It would be useful for virus researchers to have an anti-virus virus prepared in the event that one of their programs escapes the confines of the original research space. Releasing such an anti-virus quickly after the virus escapes could prevent the spread of the actual virus and the hysteria that will accompany it. An anti-virus virus must be released in the same discourse community as its target. It must be released from a "clean" machine to insure that the released anti-virus does not contain the target virus. It must also be equipped with a function to remove itself after a period deemed sufficient to remove all instances of its target. The source code (which should be compatible with any ANSI C compiler) of the simulation is included as Appendix A. Sample output from this simulation is provided as Appendix B. The source and output are also available at http://www2.msstate.edu/~gam3/virussim.cpp and http://www2.msstate.edu/~gam3/output.txt, respectively.
The simulation operates by maintaining a list of users and their state of infection, whether or not they are infected by the malicious virus or the anti-virus virus. In this situation there are ten users labeled User A through User J that make up a hypothetical discourse community. The virus is introduced into the group (either maliciously or accidentally) by User A. As User A provides executable content to others within the group and further sharing occurs between the other users, the virus spreads throughout the community. After the virus has been discovered, another virus is written that will spread and destroy the target virus. The anti-virus virus remains resident on the infected machine to detect and prevent any further infection attempts by the malicious virus. The anti-virus virus is introduced into the discourse community from a machine, labeled User K, that has not been infected by the target virus. As User K begins interaction in the discourse community, the anti-virus virus spreads and eventually eliminates the malicious virus. Within the anti-virus virus is code that will cause it to destroy itself after the amount of time deemed necessary for success has passed. The time necessary for the successful destruction of a malicious virus is dependent upon the amount of time the malicious virus has been active, the size of the discourse community, and the rate at which the discourse community shares executable content.

Summary

The use of beneficial viruses is a highly controversial concept. The negative connotation associated with viruses greatly contributes to the sentiment against such research. As demonstrated by Shoch, Hupp, and Cohen, self-replicating programs can be useful. Shoch and Hupp developed programs that could distribute computations by the worm's process of replication. Cohen's research uses replicating programs to perform various functions such as distributing software, implementing distributed databases, and performing redundant network administration tasks. Though there are some valid arguments against self-replicating programs, including the need to prevent such programs from reproducing uncontrollably, research into beneficial uses should not be ignored. Hopefully, both sides of the argument can be satisfied and a useful virus can be developed that does not threaten to cause damaging side effects.

References

1. Bontchev, Vesselin. "Are 'Good' Computer Viruses Still a Bad Idea?" <http://www.commandcom.com/html/virus/res/goodvir.html> (18 June 1999).
2. Bontchev, Vesselin. "Future Trends in Virus Writing." *International Review of Law, Computers and Technology* 11.1 (1997): 129-146.
3. Cohen, Frederick B. "Computer Viruses." *Computer Security: A Global Challenge*, Elsevier Press 1984, pp. 143-158.
4. Cohen, F. *It's Alive: The New Breed of Living Computer Programs* (New York, NY: John Wiley and Sons, 1994).
5. Cohen, F. Personal interview. 28 June 1999.
6. "Gulf War." <http://www.cryan.com/war> (2 July 1999).
7. Pfleeger, Charles P. *Security in Computing*, 2nd ed. (Upper Saddle River, NJ: Prentice Hall, 1997) 176-195.
8. Rogers, Adam. "Is There a Case for Viruses?" *Newsweek* 27 Feb. 1995: 65.
9. Rosenberger, Rob. "Myth: The 'Gulf War' Virus." *Computer Virus Myths.* <http://www.kumite.com/myths/myths/myth016.htm> (23 June 2000).
10. Shoch, John F. and Hupp, Jon A. "The 'Worm' Programs: Early Experience with a Distributed Computation," *Communications of the ACM* 25.3 (1982): 172-180.
11. Stojakovic-Celustka, Suzana. "Alive Vol. 1 No. 1: April - July 1994." <http://www.bocklabs.wisc.edu/~janda/alive11.html> (18 June 1999).

Appendix A - Source for Anti-Virus Virus Simulation

```
// virussim.cpp
// virus simulator for anti-virus virus

// VIRUS NOTES
// The virus "V" is already on the system and infecting user A's
// machine. V infects users by infecting each user who shares
// programs with an infected user.

// ANTI-VIRUS NOTES
// The anti-virus "Z" is introduced after virus V has already
// begun to infect a discourse community.  Z will theoretically
// be distributed through the discourse community in a similar
// manner and at a similar rate to V.  Z will infect systems in
// the same manner as V and will destroy V upon encountering it
// and will prevent V from infecting again.  Z will destroy
// itself after a set date which is estimated by the author
// before release

// ASSUMPTION:
// 1.  The infection platform allows Z to remain resident and
//                 prevent V from infecting a system infected by Z

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>


void main (void)
{

        int users[11];          // an array that represents users 0 - 9
        // are users A - J and 10 is user K which is the user that
        // initially introduces the anti-virus virus into the
        // discourse community from a "clean" machine


        int randnum1, randnum2;         // just hold a random numbers

        char rootletter = 0x41;// base for ascii characters
                                                        // representing letters used for
                                                        // output of user letters

        // *NOTE*
        // members of the array and the variable memory will have 3
        // possible values:
        // 0 = not infected
        // 1 = infected by V
        // 2 = infected by Z

        // setup a file to mirror screen output for easier reading
        ofstream outFile;
        outFile.open("output.txt");

        // seed random number generator
        srand ((unsigned) time (NULL));

        // initialize array
        for (int index = 0; index < 11; index++)
                users[index] = 0;               // all users clean

        users[0] = 1;                                   // user A is infected by V
        users[10] = 2;                                  // user K is infected by Z

        // begin simulation
        // have 100 sharing of programs in random order with proper
```

```
		// virus activity modeled
		// * NOTE users[10] - is assumed to not be introduced yet
		for (index = 0; index < 25; index++)
		{
			randnum1 = rand() % 10; // random number between 0 and 9
			randnum2 = rand() % 10; // "                          "

			// report which users share programs ignoring identical
			// users
			if (randnum1 != randnum2)
			{
				cout << "User "
				 << (char)(rootletter + (char)randnum1)
				 << " shares with User "
				 << (char)(rootletter + (char)randnum2)
				 << endl;

				outFile << "User "
				 << (char)(rootletter + (char)randnum1)
				 << " shares with User "
				 << (char)(rootletter + (char)randnum2)
				 << endl;
			}

			// infect user if other user is infected
			if ((users[randnum1] == 1) && (users[randnum2] == 0))
			{
				users[randnum2] = 1;
				cout << "User ";
				cout << (char)(rootletter + (char)randnum2);
				cout << " is now infected by V" << endl;

				outFile << "User ";
				outFile << (char)(rootletter + (char)randnum2);
				outFile      << " is now infected by V" << endl;
			}

		}

// report infected programs
		cout << "Users infected by V:";
		outFile << "Users infected by V:";
		for (index = 0; index < 10; index++)
			if (users[index] == 1)
			{
				cout << " "
					<< (char)(rootletter + (char)index);
				outFile << " "
					<< (char)(rootletter + (char)index);
			}
		cout << endl;
		outFile << endl;

		cout << "Introducing anti-virus..." << endl;
		outFile << "Introducing anti-virus..." << endl;

// now introduce user with Z into mix
// simulate 40 sharing instances between users in random order
// with Z in the mix
		for (index = 0; index < 75; index++)
		{
			randnum1 = rand() % 11; // random number between 0 and 10
			randnum2 = rand() % 11; // "                          "


			// report which users share programs ignoring identical
			// users
			if (randnum1 != randnum2)
			{
				cout << "User "
```

```
                    << (char)(rootletter + (char)randnum1)
                    << " shares with User "
                    << (char)(rootletter + (char)randnum2)
                    << endl;

                    outFile << "User "
                    << (char)(rootletter + (char)randnum1)
                    << " shares with User "
                    << (char)(rootletter + (char)randnum2)
                    << endl;
            }

            // infect user if other user is infected
            if ((users[randnum1] == 1) && (users[randnum2] == 0))
            {
                    users[randnum2] = 1;
                    cout << "User ";
                    cout << (char)(rootletter + (char)randnum2);
                    cout << " is now infected by V" << endl;

                    outFile << "User ";
                    outFile << (char)(rootletter + (char)randnum2);
                    outFile      << " is now infected by V" << endl;
            }

            // infect with Z (if not previously infected), remove V
            if ((users[randnum1] == 2) && (users[randnum2] != 2))
            {
                    if (users[randnum2] == 1)
                    {
                            cout << "Virus removed from user "
                                    << (char)(rootletter + (char)randnum2)
                                    << endl;

                            outFile << "Virus removed from user "
                                      << (char)(rootletter + (char)randnum2)
                                      << endl;
                    }

                    users[randnum2] = 2;
                    cout << "User  ";
                    cout << (char)(rootletter + (char)randnum2);
                    cout << " is now infected by Z" << endl;

                    outFile << "User  ";
                    outFile << (char)(rootletter + (char)randnum2);
                    outFile << " is now infected by Z" << endl;
            }



}

// report infected users
cout << "Z destroys itself at a given time." << endl;
outFile << "Z destroys itself at a given time." << endl;
cout << "Users infected by V:";
outFile << "Users infected by V:";
for (index = 0; index < 11; index++)
            if (users[index] == 1)
            {
                    cout << " "
                            << (char)(rootletter + (char)index);
                    outFile << " "
                              << (char)(rootletter + (char)index);
            }
cout << endl;
outFile << endl;
```

```
        // close output file
        outFile.close();
}// end program
```

Appendix B - Sample Output for Anti-Virus Virus Simulation

User C shares with User G
User F shares with User A
User A shares with User D
User D is now infected by V
User H shares with User C
User E shares with User D
User F shares with User H
User H shares with User F
User I shares with User F
User B shares with User A
User D shares with User H
User H is now infected by V
User I shares with User G
User E shares with User D
User F shares with User H
User D shares with User B
User B is now infected by V
User F shares with User G
User A shares with User B
User A shares with User F
User F is now infected by V
User G shares with User H
User C shares with User I
User H shares with User I
User I is now infected by V
User H shares with User A
User A shares with User J
User J is now infected by V
User D shares with User C
User C is now infected by V
Users infected by V: A B C D F H I J
Introducing anti-virus...
User I shares with User D
User A shares with User K
User K shares with User J
Virus removed from user J
User J is now infected by Z
User C shares with User F
User D shares with User G
User G is now infected by V
User F shares with User E
User E is now infected by V
User B shares with User E
User E shares with User I
User A shares with User F
User K shares with User A
Virus removed from user A
User A is now infected by Z
User J shares with User K
User J shares with User G
Virus removed from user G
User G is now infected by Z
User H shares with User D
User J shares with User A
User B shares with User C
User E shares with User A
User E shares with User C
User H shares with User J
User C shares with User D
User F shares with User B
User E shares with User G
User A shares with User B
Virus removed from user B
User B is now infected by Z
User H shares with User I
User G shares with User K
User F shares with User K

User A shares with User F
Virus removed from user F
User F is now infected by Z
User D shares with User E
User I shares with User C
User G shares with User F
User J shares with User G
User B shares with User H
Virus removed from user H
User H is now infected by Z
User C shares with User E
User A shares with User K
User H shares with User G
User B shares with User E
Virus removed from user E
User E is now infected by Z
User D shares with User I
User I shares with User E
User I shares with User J
User J shares with User C
Virus removed from user C
User C is now infected by Z
User C shares with User D
Virus removed from user D
User D is now infected by Z
User F shares with User B
User E shares with User F
User A shares with User F
User C shares with User F
User J shares with User I
Virus removed from user I
User I is now infected by Z
User K shares with User E
User A shares with User C
User E shares with User I
User G shares with User C
User K shares with User G
User J shares with User C
User D shares with User I
User G shares with User A
User J shares with User I
User C shares with User E
User F shares with User A
User H shares with User D
User F shares with User I
User B shares with User G
User H shares with User I
User I shares with User E
User H shares with User F
User C shares with User J
User B shares with User I
User G shares with User C
User K shares with User B
User G shares with User C
User A shares with User G
User J shares with User A
Z destroys itself at a given time.
Users infected by V: