



Unfections

infectionvectors.com
November 2004

Overview

The use of “anti-virus” techniques in many modern worms recalls two stories: the “war” between virus writers and the possibility of a beneficial virus. This article examines both in light of recent events in the virus world.

In early 2004, the release of Netsky began a rash of articles discussing “vigilante” coders trying to stop the spread of other worms. The war between Netsky’s author and the Beagle and MyDoom writers garnered widespread public attention; even articles in non-security sources questioned the impact of this fight.

The debate over using a computer virus to fight other viruses or to harden a system against potential attack has been around for decades¹. Although no virus has actually ever met an accepted definition of “beneficial,” many appear to have tried. Prior to Netsky, worms of all varieties (both real and theoretical) have been constructed to fit the role of “good virus.”

All For One

The concept of a beneficial virus has been discussed for years in the viral research arena². Historically, the debate has been framed by those that reject any self-replicating software that forces system changes as a legitimate aid and those that argue a “good” virus is theoretically possible. The latter group generally argues that past failures and a lack of tangible examples does not mean that the possibility of a good virus should be dismissed. This report begins with the general theory of a good virus and then looks at possible candidates that have been released.

To properly discuss the topic, there has to be a common definition of what makes up a “virus,” or at the least, some of its characteristics need be agreed upon. For the purposes of this analysis, “virus” will simply mean any software that replicates and makes system changes (whether these changes are perceived as positive or negative). The viral payload will be used to help discern the intention of the program/author.

Judging Good and Evil

There are intrinsic properties of viral code that make it dangerous for public release, no matter what the payload or intent. Even a tool that downloads and installs a single patch to every machine it is run on endangers applications or entire networks where fixes have

not been tested. Bypassing normal security configurations to allow individual users to install code on a company device is equally risky. Every systems administrator has likely had numerous boxes fail to work properly after installing a vendor-released patch. Any worm that bypasses normal testing and acceptance would cause widespread failures.

Anything that copies itself onto another machine (without permission) and makes changes (again, without the owner's consent) has great potential for doing harm. Even with the best intentions, the application would have to be very disciplined in order to prevent network traffic from reaching astronomical levels and hurting productivity. Arguing that a good virus could theoretically be coded to: limit the amount of network bandwidth or system resources, check for compatibility, and ask for permission is a synthetic attempt to meet the technical definition of virus while accounting for the inherent problems of releasing a worm onto the Internet. Anything that is produced to work only on a range of preconfigured machines,

When one evaluates all the theoretical restrictions on a good virus, such as limiting the bandwidth it uses, checking for compatibility, verifying acceptance, etc., the resultant is an application that artificially meets the term "virus" and fulfills a job better left to the existing patch/system management tools on the market.

Possible Payloads

The "benefits" of good viruses have taken many forms, both in practice and in theory. The following have been coded into viruses released onto the Internet:

Patch Installation

One incarnation of a good virus is one that infects a machine to install a patch, in theory protecting the box from future exploits. In August of 2003, Welchia (Nachi) was released to combat the rampant success of Blaster (LovSan). Blaster and Welchia both infected a machine by using RPC DCOM buffer overflows on Windows machines. Welchia, however, attempted to reach out to the Microsoft update site, download the patch (MS03-026), and install the new code. In addition, Welchia deleted copies of Blaster (msblast.exe) that it found on the local machine.

As mentioned above, installing a generic patch onto random machines invites disaster for some users, rendering certain applications and entire boxes unusable. Furthermore, Welchia crushed networks of all sizes; not just from download attempts but also from the very aggressive target scanning it undertook to find additional hosts.

Security Scanning

Certain worms do a very good job checking every host on a network for particular patches. They do this by enumerating hosts and sending specially crafted packets to each in hopes of discovering weaknesses. Those that are vulnerable get an exploit that opens a command shell, ftp server, etc. Some worms try the exploit against all hosts that appear to

be alive, skipping the step of checking for the vulnerability and blindly sending the exploit code everywhere (Sasser). In each case, an argument can be made that the worm is acting in the role of security scanner, actually providing useful data to the system administrators.

While the potential lessons of an infection should not be overlooked (see the article “Lessons Learned from Virus Infections” cited in the References), it is not a valid argument for constructing and releasing Internet worms. First, the information provided by a virus can be discovered and reported by true vulnerability scanners. These packages scan networks without the potential for destruction that worms do and provide organized documentation of their findings (without posting the data to an IRC server).

Applications such as Agobot (Gaobot, Phatbot, etc.) come the closest to straddling the scanner/malware line, although the inclusion of serial number theft functions is difficult for proponents of the “security tool” argument to overcome (not to mention inserting itself into the Windows startup routine, starting proxies, deleting/adding shares, etc.). The existence of free tools such as Nessus also calls the need for such applications into question, outside its propensity to knock vulnerable machines over (RPC DCOM and LSASS exploits tend to make systems reboot), there is no net benefit to a security administrator to using a Trojan like Agobot over a mainstream scanner. Agobot could be compiled in such a way as to make it a scanning tool, by removing many of the nefarious routines and propagation mechanisms that make it useful in its current role as bot-net application. These modifications would be clearly artificial however, rendering it a simple scanning tool that would be well behind sophisticated packages that have been developed for this purpose. Penetration testing tools like Metasploit Framework are good examples of tools that may include true exploit code, but do include replication routines or untamable scanning functions.

Trojans and rootkits often employ the services of these “building blocks” to accomplish a piece of their goal. Utilities that kill running processes, scan the local network, enumerate user accounts, etc. are packaged with more nefarious tools (or simply combined with other utilities in such a way to allow nefarious activity). Remote control applications such as the Dameware utility are good examples of programs that can be molded into an attackers toolkit. Dameware’s product can be installed remotely and completely takes over a network asset. Administrators use it to simplify many network management tasks. Fitted with a replication and reporting mechanism, however, the utility would resemble many of the Trojans currently in the wild. The reverse process of tempering a worm to fit the definition of “beneficial” would result in creating system utilities such as this.

Welchia and Hobbes³

The discussion of a “good virus” invariably winds its way towards vigilantism. The lack of centralized government on the Internet makes for one of the largest experiments in anarchy ever conducted.

Using a worm to patch a system is not an acceptable form of security, however⁴. Consider someone who tested the security of home door locks by smashing down the door and then installing a new lock. That type of “service” would not make anyone feel more secure in his or her homes. When a system is compromised by unknown sources, it is no longer trusted. There is no way to know what else was gathered, changed, or added to the system, whether that system is a house or a server.

As mentioned above, the random installation of patches can have detrimental effects on systems with proprietary or non-standard configurations, that’s the reason so many administrators wisely test any fix before deploying it to a large community.

The role of the vigilante on the Internet is especially important, as there is no worldwide (or more correctly Internet-wide) governing body that has jurisdiction to prosecute virus writers in any particular locality⁵. Some may argue that the Internet should not have any interference from an external body; that traditional government has no place inside this unique community that extends without national borders. This contention assumes that the Internet only exists in virtual space and does not affect tangible assets that are already covered by national/local laws. However, when a commerce server is taken down by a worm there is a very real cost involved in repairing and replacing the machine.

Finally, laws that target people that break into, steal information from, and damage machines cannot be applied only to virus writers. If classified as a crime, vigilantes that force code onto a device will be subject to the same laws and punishments as malicious coders. Intent is often considered when a case is before a court, and undoubtedly it would factor into sentences handed down to programmers truly seeking to promote Internet safety.

Robin Hood’s Gang

When considering what makes a “vigilante” coder, there are many examples to review. They range from exceptionally damaging to the entire Internet to those that make small changes only to infected machines.

Already mentioned in the leagues of supposed “good viruses” is Netsky, a mass mailer released with messages such as the following embedded in the code (from Netsky.C):

```
<-<- we are the skynet - you can't hide yourself! - we kill malware
writers (they have no chance!) - [LaMeRz-->]MyDoom.F is a thief of our
idea! - -
< SkyNet AV vs. Malware >- -->>
```

These types of statements and the inclusion of routines that removed Beagle, MyDoom, and Mimail variants began the “war” of early 2004 between the writers of the respective worms. Netsky, however, did quite a bit of harm to the Internet and its recipients. The amount of traffic created by its mail engine continues to clog mail servers. Its backdoor routine provides unwanted access to thousands of machines. In the end, Netsky has done as much, possibly more, damage to networked systems than its “competitors.”

Another worm often written about as having “good intentions” is Welchia/Nachi. Welchia has some markings of a beneficial worm: it removes Blaster infections, patches the system against future compromises, and removes itself (albeit approximately 5 months after its release). However, other functions of the virus make it appear less benevolent: it hides its “beneficial” code by taking on Windows system file names, sets itself to restart with the machine until its expiration date, and leaves a port open (usually TCP 707) on the infected box. The worm was not always successful at patching a box, sometimes leaving it open to compromise while it continued to spray ICMP packets in search of additional hosts. The power of the Welchia search routine generated enough traffic to bring many large-scale networks to a halt, as well as ATM machines around the world. This “beneficial” worm was one of the most costly in 2003⁶.

Other worms simply claim to be innocuous, providing a comment upon the author’s intentions. A worm known as Midfin⁷, which infects VBS files (and overwrites the user’s IE home page), includes the following line:

```
Hi, this's a worm but not dangerous
```

Many other viruses are deemed “not dangerous” in virus catalogs. Examples of these worms are those that simply write additional characters and copies of themselves without damaging data or network bandwidth. However, the idea that any virus or worm is “not dangerous” begs the inevitable question of whether a virus could be “good.” Distinguishing a program that is classified as a virus as “not dangerous” must be seen as distinct from “not harmful” or “not malware.” Without this distinction (which seems extremely difficult to maintain), the possibility for good and beneficial worms is strengthened. Certainly, there are some worms that are not damaging to local systems, and those can be classified by “impact” ratings (as are used by most AV vendors), but to stipulate a virus as safe is to feed the “good worm” argument⁸.

In what may be the least nefarious of all examples, some IRC server operators placed code on their boxes that would disable Fizzer infections⁹. When a machine is infected with the Fizzer worm it attempts to register with one of multiple hard-coded IRC servers, alerting the authors and allowing for future exploitation of the compromised device. Some owners of the specified servers (who were not affiliated with the worm) decided to place code on the servers that intercepted the communication attempt and would then remove the Fizzer worm from the infected computer. The server operators later removed this function due to fears over the legality of the move. Indeed, they were forcing system changes to computers, but only devices that already had a virus. Although the IRC operators had the best intentions, they were making non-requested system modifications based on the assumption that the machine is infected with Fizzer (although the chance

that the request for a certain channel with a certain nick, etc. is legitimate was extremely unlikely, that possibility existed) and that the cure will not be worse than the sickness.

Benefiting Everyone

There are few people that would actually hope for any of the so-called beneficial viruses to infect their machines (possibly virus researchers and those with a particular penchant for computer viruses are the only exceptions). In practice, and as presented here, in theory, there is no way to actually construct a “good worm.” Past attempts to craft such a virus have resulted in some of the most destructive applications ever released onto the Internet¹⁰. Building a virus that spreads in a sane manner, responsibly patches a machine, or provides any benefit to a user requires such constraints on the code that the finished product would not meet the definition of a virus by most standards, but would resemble a systems administration utility.

Appendix A: Lessons Learned¹¹

A previous article, published on SecurityFocus.com, described the lessons one could take away from a virus infection. This article, while clearly stating that there was no good reason to intentionally infect a machine with a worm, was the starting point for this report.

The “Lessons Learned” article examined the extreme network testing that a network worm utilizes in an effort to find more hosts. The testing and the worm’s payload are both not worth the information gleaned, especially in light of the very capable vulnerability scanners that exist on the market today.

Appendix B: Agobot as a Scanner

Admittedly Agobot could be constructed in such a way as to make it a legitimate, albeit dangerous, network vulnerability scanner. The code would have to be compiled without its propagation mechanism (preventing it from spreading once a scan was initiated) and without the startup hooks it plants on an infected machine. Furthermore, the theft routines (which lift keys for games and Microsoft Windows) would have to be left out.

The danger still exists in using the actual exploits to test vulnerability, however, there are many people who may prefer this method (as Nessus allows for such a scan).

Finally, the argument must be made that any worm can be modified as is described above. Every self-propagating piece of code must enumerate hosts in some way and then test whether or not the target is vulnerable to a specific exploit. There is no technical reason that Slammer could not be modified to only scan a specified subnet and to send a report to a specified email address instead of starting a replication process. At that point, however, the code is just a homegrown scanning tool, not a worm.

References

1. Many authors have chronicled the debate. See some of these classic virus references for details.

“Argument for a ‘Good’ Virus” MidNyte. April 1999.
<http://vx.netlux.org/lib/static/vdat/epmggvir.htm>

Vesselin Bontchev
<http://www.virusbtn.com/old/OtherPapers/GoodVir/>

“History of computer viruses”
<http://www.antivirusworld.com/articles/history.php>

“Darwin, a Game of Survival of the Fittest among Programs”
<http://www.cs.dartmouth.edu/~doug/darwin.pdf>

2. Two good references to check out in this arena are below:

“What About Good Viruses?”
<http://www.cknow.com/vtutor/vtgood.htm>

“The Case for Beneficial Computer Viruses and Worms” Greg Moorer.
<http://csrc.nist.gov/nissc/2000/proceedings/papers/601.pdf>

3. Quick high-level review of Thomas Hobbes.
“Thomas Hobbes” Professor John Rogers.
http://www.bbc.co.uk/history/state/monarchs_leaders/hobbes_01.shtml

4. The not so subtle position of Kaspersky Labs is presented in the following article.

“Good viruses don’t exist” Kaspersky Labs, August 26, 2003.
<http://www.viruslist.com/en/news?id=65946>

5. Vigilantism is hinted at or blatantly advocated in a fair amount of anti-virus literature.

The possibility of using a virus to fight MyDoom is offered:

“Fight Virus with Virus” Paul Bouton. July 27, 2004
<http://slate.msn.com/id/2104432/>

The ability of an Internet user to use a “good offense” to stop threats:

“Right to Defend” Tim Mullen, July 29, 2002.
<http://www.securityfocus.com/columnists/98>

6. Welchia ravaged a great number of networks, including two of the largest in the world.

Navy Marine Corp Network hit hard with Welchia

<http://www.nwfusion.com/news/2003/0819navy.html>

Nachi worm infects Diebold ATMs

http://www.theregister.co.uk/2003/11/25/nachi_worm_infected_diebold_atms/

Welchia Technical Reference

<http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html>

7. Midfin – says “not dangerous”

<http://www.symantec.com/avcenter/venc/data/vbs.midfin@mm.html>

8. Examples of “not dangerous” Classifications for Viruses

MKWorm.715

<http://www.viruslist.com/en/viruses/encyclopedia?virusid=7398>

Zorm.643.a

<http://www.viruslist.com/en/viruses/encyclopedia?virusid=12859>

9. IRC server owners using code that removes worm from infected box upon connection
“Counter-Hacking: Vigilante or Savior” Tony Bradley.

http://netsecurity.about.com/cs/generalsecurity/a/aa052103_p.htm

Fizzer Technical Reference

http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_FIZZER.A&Vsect=T

10. Additional arguments for the need to restrict virus writing:

“The Regulation of Virus Research and the prosecution for unlawful research?” Alistair Kelman, October 31, 1997.

http://elj.warwick.ac.uk/jilt/compcrim/97_3kelm/KELMAN.DOC

11. “Lessons Learned from Virus Infections” Jason Gordon, October 4, 2004.

<http://www.securityfocus.com/infocus/1804>

Copyright © 2004 infectionvectors.com. All rights reserved.