

Virus Safety of Linux

By Collin Beck

Introduction

With computers and the Internet becoming more prevalent in the world, so are the amount and types of attacks that are being done on them. Identity thieves, resource thieves, Trojan horses, worms, viruses, Web site vandals, corporate spies, and stalkers all want access to your computer and to your data. If one of these can get access to your computer, it's likely that the other ones can as well. Viruses are probably the most common of the list, although not the most dangerous. Viruses look for the smallest vulnerabilities of a computer system and try to exploit them to accomplish their destructive tasks. Anti-virus software is all too common to us; the names McAfee, Symantec, Norton and AVG are programs that we've all heard of. Anti-virus software is so common that even some network administrators require us to have it in order to use their networks. Viruses attach themselves to a host file, and when executed, reproduce and attach to other files, with potential to affect the entire network. This is all too common for the operating system that we are accustomed to use.

Linux, a free and open-source operating system, claims that it doesn't have such problems. In fact, Linux doesn't even need an anti-virus protection program. Linux's popularity is increasing and it is becoming more widespread in its use. Has it not had virus problems just because it isn't popular enough for virus writers to focus on it yet? Linux claims stability and security, but what type of virus would be needed to take down a Linux system? In the rest of this paper I will discuss 1) some of the vulnerabilities of Linux, 2) conscious exploits done by malicious hackers, 3) why Linux is naturally immune, and 4) the evidences that Linux is secure.

Linux Vulnerabilities

The following sections are some of the most common ways that a Linux system can be compromised. This is not an all inclusive list, but includes the most common ways for a virus to start its vicious reproducing cycle. Although these are common ways to for a virus to get a start, they don't happen very often, and as will be described later, are easily fixed or avoided.

User Ignorance

Humans make mistakes. The majority of people that use computers are not experts at using them. We all lack knowledge on certain subjects and oft-times our ignorance can cause us to make mistakes that have consequences that we weren't even aware could happen. The easiest way for a virus to be put on a computer is for a user to place it there. This can be done by copying it from a jump drive/floppy disk, downloading it from a Web site, or even getting it as an attachment to an email. This is the easiest way for a virus to get on a computer and to infect it.

Another common way for someone to gain unauthorized access to your computer is by using password guessing tools. These tools first check to see if you have a password, then they try the most common default passwords (like admin, password, your user name, etc.), then they go through a list of words in a dictionary, and then many other random combinations are used to try to guess your

password. User ignorance on how to create a safe password is also a huge vulnerability. This is very easy to correct. Either the program can require the user to create a secure password, one that is a combination of numbers and letters and is more than six characters long, or require the user to change their password every 30 to 60 days. A combination of both techniques can be used if desired. (McClure, Scambray, Kurtz, 2005, p 217)

User ignorance is a vulnerability that will always be around; people without much computer knowledge will be using computers for years to come and they shouldn't be required to be computer savvy in order to do so. Linux comes in a wide range of varieties, but some of the more popular varieties have new helps for the common user's lack of knowledge.

Administrative (Root) Permissions

Administrative or root, as it is called in Linux, permissions are when the user is logged on as the system controller. The administrator can install, change, delete, or uninstall almost anything on the computer. If a person is logged onto the computer as the administrator and a virus is run on the computer, the virus will be able to do almost anything that it wants. Of the hundreds of versions of Linux, I only know of one that has root as the default account. Having root as the default account is not a good idea; this is the best way for a virus to take down your system. Ubuntu Linux, the most popular desktop version of Linux, has the root account disabled (Ubuntu.com). One can't even login as root. Ubuntu uses a different technique called sudo that requires the administrator's password anytime an administrator function is executed. The default accounts on Windows have administrator privileges and oddly enough, the Linux version that does the same is called Lindows (a Windows-like Linux version).

Poorly Written Programs

Another vulnerability of Linux and software in general may arise due to poorly written programs. Most vulnerabilities are due to software that was not peer-reviewed well enough or was released, based on deadlines, when it wasn't completely finished. This is why it is so common in the software world to see patches and updates on a frequent basis. A good number of the updates are to fix vulnerabilities or cover up errors that are found in the code (these errors and vulnerabilities are often called bugs).

Virus writers search for these bugs and try to exploit or take advantage of them. The most common of these exploits come from race conditions and buffer-overflow attacks which will be discussed next.

Conscious Exploits

The previous vulnerabilities were brought about by human error and bad planning. They can be mostly avoided by using a well-known Linux variety that has error checking for user ignorance and that doesn't have default root/administrator permissions. Poorly written programs can still pose a problem for Linux if they are maliciously exploited. An exploit is a piece of software, data, or series of commands that takes advantage of a program's bugs to achieve unexpected results (Exploit, 2006).

Race Condition

In multi-user and multi-tasking computers, certain parts of programs are shared between all users. To avoid concurrency problems, certain security and safe programming techniques should be employed to avoid the race condition. Sometimes these techniques are not used and can allow for a virus to sneak into the computer. A race condition is accomplished by carefully planning a certain order

of execution that would allow for the program to be ran in the incorrect order (two processes racing to see who arrives first). For example, if you wanted to edit someone else's file on a computer, and you were allowed to open the file before security checked the ownership of it, by repeated opening attempts in hopes that you'd arrive before the security checker, then you would have access to modify the file and add a virus to it.

Programmers can avoid this problem by using good programming practices. The most common way to avoid this is by using the lock and key method. The lock and key method allows for a file to be locked and a key given to the user of the file, until that user unlocks the file and gives the key up. This way, execution must be done in the correct order and the race condition no longer exists. Another way for system to be compromised is by an exploit called a buffer-overflow attack.

Buffer-overflow Attack

A buffer-overflow attack takes advantage of a badly written program, and uses it to either crash the program or gain special control over a system. This is probably the most frequently used exploit on a Linux machine.

The buffer-overflow attack takes advantage of a program that doesn't do much security checking, by allowing one to input more data than is expected. For example, if a program expects you to input a file name that is less and 30 characters and you enter 200 characters, if the size is not checked, when that name is copied to the program it could overwrite other data, crashing the program. If that data contains malicious code, it can then be run and compromise the security of the computer. This can be easily fixed if the programmer does a check on the size of the inputted data. Sometime the programmer doesn't do security checking on purpose, to allow for this vulnerability, and other times, the programmer just isn't experienced enough to make the security checks. There are several solutions to this problem that don't involve the programmer. A solution that is currently becoming more widespread deals with the computer's hardware. The hardware checks to make sure that input is not longer that is should be by not allowing the user to run certain code on the stack. Once this becomes more widespread, we should be seeing less and less buffer-overflow attacks.

Although the race condition and the buffer-overflow exploits exist, and can be carried out on a Linux machine to run or install a virus, the virus is not likely to spread. Linux has a natural immunity to these types of things.

Natural Immunity

Viruses are not commonplace in Linux. I have yet to find evidence of a virus outbreak in Linux. Even if there was a somewhat effective virus for Linux, it would have a hard time spreading throughout the computer or even to other computers. The Linux environment is naturally immune to viruses because of to its origins, open-source nature, and its file permissions.

Origins

Linus Torvalds started Linux in 1991; he started it as a hobby to make a more fully function kernel (the main part of an operating system). Original development was done over the Internet. Linux was freely shared and people from all over the world pitched in to get Linux working. It was Internet/network aware almost from the beginning. Because of this, security measures were implemented from the ground up. It was a multiuser system from the beginning as well, so security and privacy have long been an integral part of Linux. This helps to make Linux as stable as it is; a strong foundation is needed to have a strong overall product.

Open-Source Nature

There are basically two methods of program development: proprietary and open-source. Proprietary are the programs that you usually pay for. Some company creates a product and doesn't allow anyone else to see how it was created. The bugs in a proprietary program are initially hidden from the world, but with enough reverse engineering they are discovered and exploited. On the other hand, open-source programs are usually free. Open-source programs are peer-reviewed and as stated in Eric Raymond thesis, “with enough eyeballs, all bugs are shallow” (2002). If an open-source program is popular enough, potentially hundreds of people could review and make suggestions on a programs code.

Another one of the benefits of open-source is that if someone thinks they can improve a program, they are free to do so. They can make changes to almost any program and redistribute it as pleased. This creates competition and evolution of programs. Unlike proprietary programs, open-source programs need to be void of bugs or else they will be exploited really easily. The bugs can't just be hidden, they must be non-existent. In a four-year analysis done by Stanford University, Linux had an average of 0.17 errors per-thousand lines of code, whereas commercial software encountered 20–30 errors per-thousands lines of code (Delio 2004). No software is perfect, open-source programs still have problems and bugs, but most of the time they are fixed before they are exploited or allowed to spread. For example, Mozilla Firefox (an open-source project), on average, fixes bugs within 24 hours of the exploit, whereas Internet Explorer (a proprietary program), on average, fixes bugs within 9 days of the known exploit (Bangeman, 2006). As long as you use popular open source software and keep it up-to-date, it is very unlikely that you'll be affected by a virus.

As with viruses in real-life, it is difficult for them to spread between genetically diverse species of plants or animals. This can be paralleled with computer viruses. Because Linux is open-source, many different varieties of Linux exist. They are basically genetically different. Where one variety stores its applications, may be completely different from the location of the applications of another variety. Due to this diversity, viruses have even a harder time spreading in Linux. On the other hand, when a virus is written to exploit a proprietary application, it can infect the whole species. Not only can the whole species be infected, but the only people that are able to fix the application are the makers of it. This causes for the virus to have more time to spread and thus, more damage to be incurred. An open-source vulnerability can be looked at and fixed by any number of people, making its correction a much faster process.

Open-source is not perfect and still has errors. If you are using an open-source application that is not widely-used, it may not have been highly peer-reviewed and could contain many different vulnerabilities. Another issue with open-source software is that it is often not as high-tech as its proprietary competitors. For example, you are not going to find an equivalent to a several-thousand dollar AutoCAD program. Open-source programs are free and most people don't get paid full-time to make and develop them. For the normal computer user, open-source programs are more than enough and usually more stable.

File Permissions

One of the main reasons that Linux is immune to viruses and any other malicious software is due to its file permissions. File permissions keep track of who can run read, write, or execute a file/program. The permissions are divided up into three groups: owner, group, and universe. These three groups are what specify who can use or make changes to a file. Each different group can have different permissions. When a computer is not logged in as root, it cannot change or modify any of the

files that don't belong to it. This makes it so viruses can attach themselves to the files that they need to in order to spread. And a virus that doesn't spread dies. This is one of the greatest immunities to viruses. Without the correct permissions, the most damage they could do it to the users own files.

Another protection that permissions and Linux provide is that of file execution. For a virus to spread it must be what is called executable and then the user has to run the file. In Windows, an executable file normally ends with .exe, but in Linux it's part of the permissions. This is a protection because when a file is copied or emailed, it loses its executable permissions, and has to be manually changed back to executable for it to be run. This eliminates most virus emails from even having a chance at being run. Such is not the case when dealing with an .exe file, if it is emailed, it can be instantly executed and start the virus spreading because it is recognized as executable.

Security Evidences

Even though these things look good on paper, and Linux claims to be a secure and protected environment, we want to see how it does in real life. What are the evidences that Linux is really immune? What proof is there that its origins, open-source background, and its file permission really protect it from viruses?

As was stated earlier, due mostly to file permissions, viruses don't have much of a chance to spread in Linux. According to wildlist.org, which is an organization that receives reports on active viruses from 80 well-respected names in viruses, there is not a single active virus for Linux that is known. It's not that no one writes viruses for Linux because it's not very popular, but more due to the very nature of Linux immunity. When we think of Linux as not being very popular, we are mostly thinking of computers that we see on a daily basis: desktop computers. But when we talk about computers that we don't see very often, but we use every day, Linux is the most popular. The computers I'm talking about are called servers.

Servers are the computers that we use for the Internet and networks. They are where all of the Web sites in the world are stored. When we download something, we download it from a server. Our on-line email is all stored on a server, somewhere in the world. These computers need to be up all of the time and can't afford a shutdown or an attack from a virus. Imagine if your banks server got a virus and they lost all of your banking information. These are very important computers and need to be very secure as well. According to netcraft.com, 60 percent of the worlds Web sites are run on Apache (which is an open-source server that runs on Linux) (Netcraft 2006). This must mean that Linux works very well for them. Google uses Linux servers as well; they've had some run for more than a year without a restart (Netcraft 2006).

Even that National Security Agency has had a part in Linux security. The NSA developed a security module as an open-source program for highly confidential information called Security-Enhanced Linux. This module was then integrated into Linux in 2003 making Linux's security record even more credible (Nation Security Agency).

Conclusion

Want to take down a Linux system? You probably won't have much success. Viruses, Trojan horses, worms, identity thieves, and corporate spies haven't had much success either. True, Linux isn't perfect and neither are the programs for it. Vulnerabilities and exploits still exist in Linux but they are usually small and fixed by the open-source community before they are exploited. This is part of the

reason that Linux is immune. Although, Linux's immunity mostly comes from its origins as a multi-user-networked operating system, its open-source-viewed-by-many development scheme, and its strict file permissions. Evidences of this are found in many places, including, its lack-of-appearance on the wildlist's list of known-active viruses, it's prevalence in the server community, and the work done by the NSA. Theoretically, it's possible to take down a Linux system, but it's not very plausible. As Scott Granneman, a known author of computer security books said, "To mess up a Linux box, you need to work at it; to mess up your Windows box, you just need to work on it" (2003).

References:

- Bangeman, E. (2006, September 25). Number of browser vulnerabilities rising. Retrieved December 4, 2006, Web site: <http://arstechnica.com/news.ars/post/20060925-7818.html>
- Delio, M. (2004, December, 14). Linux: fewer bugs than rivals. *Wired Magazine*, 12:12, Retrieved November 10, 2006, from <http://www.wired.com/news/linux/0,1411,66022,00.html>
- Exploit, (2006, January 9). Wikipedia, the free encyclopedia. Retrieved November 20, 2006, from Exploit (Computer Security) - Wikipedia, the free encyclopedia Web site: http://en.wikipedia.org/wiki/Exploit_%28computer_security%29
- Granneman, S. (2003, October 2). Linux vs. windows viruses. Retrieved December 4, 2006, Web site: <http://www.securityfocus.com/columnists/188>
- McClure, S., Scambray, J., & Kurtz, G., (2005). *Hacking exposed 5th edition: Network security secrets and solutions*. Emeryville, CA: McGraw-Hill.
- Nation Security Agency, (n.d.). Security-enhanced linux. Retrieved November 15, 2006, Web site: <http://www.nsa.gov/selinux/>
- Netcraft, (2006, November). November 2006 web server survey. Retrieved December 1, 2006, Web site: http://news.netcraft.com/archives/2006/11/01/november_2006_web_server_survey.html
- Netcraft, (2006, December 1). Top sites for Google Inc.. Retrieved December 1, 2006, Web site: <http://uptime.netcraft.com/up/hosted?netname=GOOGLE,66.249.64.0,66.249.95.255>
- Raymond, E. (2002, September 02). The cathedral and the bazaar. Retrieved November 12, 2006, Web site: <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>
- Ubuntu, (2006, August 30). Ubuntu: linux for human beings. Retrieved November 10, 2006, Web site: <http://www.ubuntu.com>
- WildList (2006, September). The wild list organization international. Retrieved November 15, 2006, from WildList September 2006 Web site: <http://www.wildlist.org/WildList/200609.htm>