The ways of viruses in Linux

HOW SAFE?

Some say an attack is looming, and others say we don't have to worry. What's the real story on viruses in Linux?

BY TOMASZ KOJM

Linux may not be as vulnerable as Windows, but if you think Linux viruses don't exist, you'd better think again. Virus writers have any number of possibilities for passing viruses into Linux, although the damage will be limited if you're careful and follow a few simple rules. In this article, I'll describe some examples of how Linux viruses work, and I'll give you some tips for keeping your system safe.

A Theoretical Linux Virus

Most Linux distributions come with *gzexe*, a small utility that compresses executable files and automatically uncompresses them when they're started. For example, you can copy */bin/date* to */tmp* and run *gzexe /tmp/date* to compress the executable file. The size of */bin/date* and */tmp/date* should differ, and the latter should be noticeably smaller. Now try to run both files. Do you notice any difference?

Every executable compressed with gzexe includes a special stub at the beginning of the file. When you open */tmp/date* with your favorite editor, you will notice that the stub is just a plain shell script. Below the stub, there is binary data with the compressed executable. The shell code is responsible for decompressing the data into a temporary file and executing it. The whole process is transparent to the end user, and on fast computers, the delay in running compressed files is marginal.

Now think about a modified stub that does the following things before decompressing and running the original code:

- searches *$PATH* for a randomly writable executable file (or a file owned by a current user) that is not a shell script;
- compresses the executable file (it can include the code from the *gzexe* shell script) and inserts the same modified stub into it.

This stub fits Wikipedia's definition of a virus as "a self-replicating program that spreads by inserting copies of itself into other executable code or documents"[1]. Let's call it Linux.Gzipper.

As this example shows, it is not a big challenge to write a simple Linux virus. It shouldn't be a surprise that virus writers can also use more sophisticated methods. The ELF (Executable and Linking) format of Linux executable files is very similar to the PE (Portable Executable) format used on Microsoft Windows and provides almost the same functionality. That means, virus writers can use many advanced infection techniques of the executable files they developed on Windows during the last decade. Of course, a number of ELF-infecting viruses already exist. In fact, some viruses can even infect both PE and ELF files. However, even the most advanced Linux viruses will face the same problem that our simple Gzipper faces: it's not that easy to damage a Linux system.

Native Anti-virus Protection

Virus writers exploit the fact that most users of commercial operating systems are accustomed to working at a highly privileged level that allows direct manipulation of critical resources for the system.

In Linux, and UNIX in general, it is a fundamental principle that one should only use the root account for administrative actions and

Listing 1: Start with an Infected File

```
testuser@testsystem:~/testfiles$ ls -l
total 727
-rwxr-xr-x  1 testuser testuser  49084 Sep  4 03:32 cp
-rw-r--r--  1 testuser testuser    651 Jul 28  2004 crontab
-rwxr-xr-x  1 testuser testuser  88038 Nov  6 17:51 date.infected
-rw-r--r--  1 testuser testuser   1489 Feb 10  2004 fam.conf
-rw-r--r--  1 testuser testuser    292 Jun 18 02:05 hosts
-rwxr-xr-x  1 testuser testuser  71996 Sep  4 03:32 ls
-rw-r--r--  1 testuser testuser   1426 Nov  6 01:44 passwd
testuser@testsystem:~/testfiles$ clamscan --no-summary
/home/testuser/testfiles/cp: OK
/home/testuser/testfiles/ls: OK
/home/testuser/testfiles/crontab: OK
/home/testuser/testfiles/hosts: OK
/home/testuser/testfiles/fam.conf: OK
/home/testuser/testfiles/date.infected: Linux.Rst.A FOUND
/home/testuser/testfiles/passwd: OK
```

never for regular work. As long as this rule is obeyed, most viruses can't do any harm globally because the mechanism of file permissions protects the system and the individual users' files. Of course, a virus can try to exploit security bugs in a target system to escalate privileges, but then, unpatched systems are open to other kinds of attacks, not just viruses.

Our theoretical Gzipper searches *$PATH* for writable executable files it could infect. When Gzipper is executed by an unprivileged user, it will only be able to infect executables owned or writable by this user, and in most cases, it will fail to spread out of the user's environment. Unfortunately, there are some "user friendly" Linux distributions that promote the bad model of a highly privileged single user, which opens a wide gate for virus attacks and allows the virus to infect core parts of the operating system. Such distributions can be called "virus friendly" as well as "user friendly."

The Many Faces of Linux

Another thing that makes it hard for viruses to spread is the number of Linux distributions and supported architectures, and the many technical differences between them. Of course, a binary virus compiled for the x86 architecture will not run on SPARC and vice versa. Even "portable" viruses written in script languages like Perl or shell may fail to run if they depend on elements that are not available on the victim's system.

Spreading Problems

Plain viruses, in contrast to worms, have no mechanism to replicate between computer systems. They can only spread along with the host file, contaminating other files in the process. These days, most Linux users and sysadmins only install software from their distributions or from official sources that are considered reliable. Additionally, official packages are usually digitally signed and can be verified before installation.

Unfortunately, even the best mechanisms cannot prevent human error.

In September 2005, the official Korean versions of Mozilla Suite 1.7.6 and Thunderbird 1.0.2 for Linux were found to be infected with the Linux.RST.B virus. The incident was very serious because the web browser is most often installed globally from the root account to make it available to all users on a system. Execution of the infected suite from a privileged account could allow the virus to easily infect system files. The Mozilla Foundation published a security advisory, telling Korean users who installed affected products to scan their systems with an anti-virus scanner[2].

Such incidents are still very marginal, but it's likely they will intensify in the future when more and more software becomes available in binary form from third party sites.

The Real Stuff

Computer viruses very often carry a *payload*, which is a special action they take after spreading. The payload of the Linux.Gzipper virus was to compress target files before infection. While in some cases even such potentially inno-

Listing 2: Passing the Infection

```
testuser@testsystem:~/testfiles$ ./date.infected
Sun Nov  6 18:02:46 CET 2005
testuser@testsystem:~/testfiles$ ls -l
total 1010
-rwxr-xr-x  1 testuser testuser  97890 Nov  6 18:02 cp
-rw-r--r--  1 testuser testuser    651 Jul 28  2004 crontab
-rwxr-xr-x  1 testuser testuser  88038 Nov  6 17:51 date.infected
-rw-r--r--  1 testuser testuser   1489 Feb 10  2004 fam.conf
-rw-r--r--  1 testuser testuser    292 Jun 18 02:05 hosts
-rwxr-xr-x  1 testuser testuser 120802 Nov  6 18:02 ls
-rw-r--r--  1 testuser testuser   1426 Nov  6 01:44 passwd
testuser@testsystem:~/testfiles$ clamscan --no-summary
/home/testuser/testfiles/cp: Linux.Rst.A FOUND
/home/testuser/testfiles/ls: Linux.Rst.A FOUND
/home/testuser/testfiles/crontab: OK
/home/testuser/testfiles/hosts: OK
/home/testuser/testfiles/fam.conf: OK
/home/testuser/testfiles/date.infected: Linux.Rst.A FOUND
/home/testuser/testfiles/passwd: OK
```

Listing 3: Backdoor is Born

```
testuser@testsystem:~/testfiles$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY     STAT START   TIME COMMAND
[...]
testuser 28000  0.0  0.1   2116   876 ?       S    18:02   0:00 ./date.
infected
[...]
testuser@testsystem:~/testfiles$ netstat -upan
[...]
udp        0      0 0.0.0.0:5503           0.0.0.0:*
        28000/date.infected
```

Listing 5: Svat as root

```
root@testsystem:~/Svat#
clamscan --no-summary
/home/root/Svat/svat: Linux.
Svat.C FOUND
root@testsystem:~/Svat# ls -l
/usr/local/include
total 0
root@testsystem:~/Svat# ./
svat
Example file infected with
Svat.
root@testsystem:~/Svat# ls -l
/usr/local/include
total 16
-rw-r--r--  1 root root
14614 Nov  6 19:10 stdio.h
```

cent payloads may cause a serious system malfunction, most of the real world viruses are not as nice as Gzipper.

The RST virus that infected the Mozilla Suite is one of the few Linux viruses seen in the wild and probably the most effective one. The first version of this virus was discovered in late 2001. The name stands for "Remote Shell Trojan." RST will try to infect executable files in the current directory, and if it has enough privileges, it will also infect the system files in */bin*.

Listing 1 shows a directory with a number of clean files and one file infected with the virus. Running *date.infected* as an unprivileged user results in infection of executable files in the current directory (Listing 2). Additionally, the virus activates its payload and starts a backdoor server listening on the UDP socket (Listing 3). Now, an attacker with knowledge of a special protocol can control the backdoor and spawn a remote shell on the infected system.

There are not many Linux viruses, and most of them were not as successful as RST. However, some of the viruses that do exist implement very interesting infection techniques. Linux.Svat is an example of such a virus. Instead of direct infection, it attempts to modify the operating system to create infected files. Listing 4 shows the compilation process of a standard "Hello, World" program. Svat's idea is based on the design of the compiler. When GCC encounters an *#include <file.h>* macro, it first looks for the header file *file.h* in */usr/local/include* and later in */usr/include*, which is the directory where all important header files are installed. The *stdio.h* file is one of the most used header files.

As shown in Listing 5, when the Linux.Svat infected file is run by root, it installs a new header file in */usr/local/include*. The new *stdio.h* includes the original one and additionally redefines the system function *close()*, which now calls the virus routine *virfunc()* before closing file descriptors. The routine contains bugs and will cause a segmentation fault if it has no write access to the */usr/local/include* directory – the sloppy code limits the virus's chances to replicate.

The infection routine will be included in every new compiled file that uses *stdio.h*. Because our example *hello.c* file doesn't call the *close()* function, the virus code in *hello2* will never be activated.

"Safe Hex" in Linux

The rules for protecting Linux against viruses in are similar to the rules for other systems:

1. Never use the root account for regular work.
2. Avoid running binary files of unknown origin. Check them with rootkit and virus scanners first.
3. Carefully check every file before running it from the root account.
4. Keep your operating system up to date. Regularly install official security updates.
5. Secure your environment by using hard-to-guess passwords and other protections.
6. Track changes in the system using file system integrity tools. ■

Listing 4: Compiling Hello, World

```
testuser@testsystem:~/hello$ cat hello.c
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello world!\n");
    return 0;
}
testuser@testsystem:~/hello$ gcc hello.c -o hello1
testuser@testsystem:~/hello$ ./hello1
Hello world!
testuser@testsystem:~/hello$ ls -l
total 12
-rw-r--r--  1 testuser testuser  100 Nov  6 18:46 hello.c
-rwxr-xr-x  1 testuser testuser 7340 Nov  6 18:50 hello1
```

THE AUTHOR

Tomasz Kojm is a computer scientist and the project leader of Clam AntiVirus project. ClamAV is an Open Source (GPL) anti-virus toolkit for Unix and Linux that is widely used as a server-side email virus scanner.